T.C.
İSTANBUL BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

PERFORMANS ODAKLI WEB UYGULAMALARI GELİŞTİRMEK

Yüksek Lisans Tezi

Tezi Hazırlayan
Elanur ŞAHİN

İstanbul, 2024

T.C.
İSTANBUL BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

PERFORMANS ODAKLI WEB UYGULAMALARI GELİŞTİRMEK

Yüksek Lisans Tezi

Tezi Hazırlayan
Elanur ŞAHİN

Öğrenci No
2120003029

ORCID ID
0000-0003-2652-5844

Danışman
Dr. Öğr. Üyesi Talat FİRLAR

İstanbul, 2024

YEMİN METNİ

Yüksek Lisans tezi olarak hazırladığım "**Performans Odaklı Web Uygulamaları Geliştirmek**" başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, bu projedeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını, patent ve telif haklarını ihlal edici bir davranışımın olmadığını belirtir ve bunu onurumla doğrularım. 25.06.2024

Elanur ŞAHİN

T.C.
İSTANBUL BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ
TEZLİ YÜKSEK LİSANS SINAV TUTANAĞI

25/06/2024

Enstitümüz ***Bilgisayar Mühendisliği*** Anabilim Dalı ***Bilgisayar Mühendisliği*** Programı yüksek lisans öğrencilerinden ***2120003029*** numaralı ***Elanur ŞAHİN'in*** *"İstanbul Beykent Üniversitesi Lisansüstü Eğitim – Öğretim Yönetmeliği"*nin ilgili maddesine göre hazırlayarak, Enstitümüze teslim ettiği ***"Performans Odaklı Web Uygulamaları Geliştirmek"*** konulu tezini, Yönetim Kurulumuzun 28/05/2024 tarih ve 2024/24 sayılı toplantısında seçilen ve On-Line toplanan biz jüri üyeleri huzurunda, İstanbul Beykent Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 29. maddesinin 3. fıkrası gereğince 45 dakika süre ile Zoom programı aracılığıyla on-line olarak aday tarafından savunulmuş ve sonuçta adayın tezi hakkında ***"OYBİRLİĞİ"*** ile ***"KABUL"*** kararı verilmiştir.

İşbu tutanak, 2 nüsha olarak hazırlanmış ve Enstitü Müdürlüğü'ne sunulmak üzere tarafımızdan düzenlenmiştir.

DANIŞMAN
Dr. Öğr. Üyesi Ta*** Fİ***
(İstanbul Beykent Üniversitesi)

ÜYE
Dr. Öğr. Üyesi Ke*** Gö*** NA***
(İstanbul Beykent Üniversitesi)

ÜYE
Dr. Öğr. Üyesi Ke*** ŞA***
(İstanbul Medipol Üniversitesi)

Adı ve Soyadı	:	Elanur ŞAHİN
Danışmanı	:	Dr. Öğr. Üyesi Talat FİRLAR
Derecesi ve Tarihi	:	Yüksek Lisans (Tezli), 2024
Alanı	:	Bilgisayar Mühendisliği
Anahtar Kelimeler	:	Web Performans, Web Mimarisi, Optimizasyon, Yapay Zeka, Kullanıcı Deneyimi

ÖZ

PERFORMANS ODAKLI WEB UYGULAMALARI GELİŞTİRMEK

Günümüzde web uygulamaları, bireylerin ve işletmelerin hayatında vazgeçilmez bir yer tutmaktadır. İnternetin ilk günlerinde, web siteleri basit metin ve görsellerden oluşan sayfalardan ibaretti. Ancak teknolojinin hızlı gelişimiyle birlikte, web siteleri de önemli değişimler geçirmiş ve çok daha karmaşık, etkileşimli ve kullanıcı dostu hale gelmiştir. Bugün, alışverişten eğlenceye, eğitimden sağlık hizmetlerine kadar birçok alanda web siteleri aktif olarak kullanılmaktadır. Bu dönüşüm, işletmelerin dijital pazarda varlıklarını sürdürebilmeleri ve kullanıcılarına en iyi deneyimi sunabilmeleri için performans odaklı web sitesi geliştirmeyi zorunlu hale getirmiştir.

Araştırmanın amacı, web performansı yüksek olan JavaScript kütüphanelerini tespit etmek ve performans düşüklüğü gösteren kütüphanelerin nasıl iyileştirilebileceğini ortaya koymaktır. Performans iyileştirmeleri, sadece kullanıcı deneyimini değil, aynı zamanda işletmelerin dijital pazardaki başarısını da doğrudan etkilemektedir. Bu bağlamda, çalışmada performansı yüksek web sitelerinin kazanımları detaylandırılmış ve performans odaklı geliştirme süreçlerinin optimizasyon ve kullanıcı açısından avantajları vurgulanmıştır.

Bu çalışmanın bulguları, web performansını artırmak isteyen geliştiriciler ve işletmeler için değerli bilgiler sunmakta olup, kullanıcı deneyimini iyileştirmek ve dijital pazarda rekabet gücünü artırmak adına önemli çıkarımlar içermektedir.

Name and Surname	:	Elanur ŞAHİN
Supervisor	:	Asst. Prof. Dr. Talat FİRLAR
Degree and Date	:	Master's (Thesis), 2024
Major	:	Computer Engineering
Key Words	:	Web Performance, Web Architecture, Optimization, Artificial Intelligence, User Experience

ABSTRACT

DEVELOPING PERFORMANCE – ORIENTED WEB APPLICATIONS

Nowadays, web applications hold an indispensable place in the lives of individuals and businesses. In the early days of the internet, websites consisted of simple text and images. However, with the rapid development of technology, websites have undergone significant changes, becoming much more complex, interactive, and user-friendly. Today, websites are actively used in many areas, from shopping to entertainment, from education to healthcare. This transformation has made it essential for businesses to develop performance-oriented websites to maintain their presence in the digital market and offer the best experience to their users.

The aim of the research is to identify high-performance JavaScript libraries and to reveal how libraries showing low performance can be improved. Performance improvements not only affect the user experience but also directly impact the success of businesses in the digital market. In this context, the benefits of high-performance websites are detailed, and the advantages of performance-oriented development processes in terms of optimization and user perspective are emphasized.

The findings of this study offer valuable information for developers and businesses aiming to improve web performance, providing important insights into enhancing user experience and increasing competitiveness in the digital market.

İÇİNDEKİLER

Sayfa No.

ÖZ

ABSTRACT

TABLOLAR LİSTESİ.....v

ŞEKİLLER LİSTESİ vi

KISALTMALAR vii

SÖZLÜK.....x

GİRİŞ 1

1. LİTERATÜR ARAŞTIRMASI

1.1. Web Sitesinin Gelişimi 4

1.2. Web Sitesi Oluşturulurken Kullanılan Front-End Teknolojileri..... 7

1.2.1. HTML..... 9

1.2.1.1. HTML'nin Tarihçesi..... 9

1.2.1.2. HTML5 ve Yenilikleri..... 10

1.2.1.3. HTML'nin Geleceği 11

1.2.2. CSS 12

1.2.2.1. CSS'nin Tarihçesi 12

1.2.2.2. Temel Özellikler..... 12

1.2.2.3. Modern CSS Kullanımı 13

1.2.3. JavaScript 14

1.3. Web Sitesini Oluştururken Kullanılan Backend Teknolojileri 16

1.3.1. Programlama Dilleri ve Frameworkler..... 18

1.3.2. Veri Tabanı Teknolojileri 18

1.3.4. API ve Mikroservis Mimarisi 20

2. WEB UYGULAMALARI VE PERFORMANS İLİŞKİSİ

2.1. Web Performansa Etki Eden Unsurlar 22

2.1.1. Web Sitesi Tasarımının Performansa Etkisi..... 22

2.1.1.1. UI UX'in Performansa Etkisi 23

2.2.1.2. Kullanılan Teknolojilerin Performansa Etkisi 24

2.2.1.2. Render Performansı'nın Web Sitesi Performansına Etkisi 26

2.1.3. Browser Uyumluluğunun Performansa Etkisi 27
2.2. Performansın Web Sitesi İçin Önemi 28
3. PERFORMANS ODAKLI WEB SİTESİ NASIL GELİŞTİRİLİR
3.1. İyi bir Planlama ve Mimarinin Önemi 31
3.1.1. Proje Planlaması 31
3.1.2. Uygulama Mimarisi 32
3.1.3. Performans Hedeflerinin Belirlenmesi 32
3.2. Optimizasyon Teknikleri 33
3.2.1. Kod Optimizasyonu 34
3.2.1.1. Proje Planlaması 35
3.2.1.2. Ölçeklenebilir ve Modüler Kod Yapısı 35
3.2.2. Görsel Optimizasyon 35
3.2.2.1. Görsel Sıkıştırma Teknikleri 36
3.2.2.2. Uygun Formatın Seçimi (JPEG, PNG, WebP) 36
3.2.3. Veri Yükleme Optimizasyonu 37
3.2.3.1. Lazy Loading 37
3.2.3.2. Asenkron Veri Yükleme 38
3.3. İçerik Dağıtım Ağları (CDN) Kullanımı 38
3.3.1. CDN Nedir? 38
3.3.2. CDN'in Faydaları 39
3.3.3. CDN Seçimi ve Entegrasyonu 39
3.4.1. Sunucu Yanıt Süresi Optimizasyonu 40
3.4.2. Cache Yönetimi 41
3.4.2.1. Server Side Caching 41
3.4.2.2. Client Side Caching 41
3.4.3. Ölçeklenebilirlik ve Load Balancing 41
3.5. Front-End Performans İyileştirmeleri 42
3.5.1. JavaScript ve CSS Optimizasyonu 42
3.5.1.1. Critical Rendering Path Yönetimi 42
3.5.1.2. Asenkron ve Defer Script Yükleme 44
3.5.2. Modern Web API'lerinin Kullanımı 45
3.5.2.1. Service Workers 45

3.5.2.2. WebAssembly 46
3.5.3. Responsive Tasarım ve Performans 47
3.6. Back-End Performans İyileştirmeleri 48
3.6.1. Veritabanı Optimizasyonu 48
3.6.1.1. Query Optimizasyonu 48
3.6.1.2. Index Kullanımı 49
3.6.2. API Optimizasyonu 49
3.6.2.1. RESTful ve GrapQL API Optimizasyonları 49
3.6.2.2. Gelişmiş Caching Stratejileri 49
3.7. Performans Testleri ve Araçları 50
3.7.1. Performans Test Türleri 50
3.7.1.1. Load Testing 50
3.7.1.2. Stress Testing 51
3.7.1.3. Endurance Testing 51
3.7.2. Performans Test Araçları 51
3.7.2.1. Google Lighthouse 51
3.7.2.2. WebPageTest 52
3.7.2.3 JMeter 52
3.8. Sürekli İzleme ve İyileştirme 52
3.8.1. Performans İzleme Araçları 52
3.8.1.1. New Relic 53
3.8.1.2. Datadog 53
3.8.1.3. Endurance Testing 53
3.8.2. Performans Sorunlarının Tanımlanması ve Çözülmesi 53
3.8.3. Kullanıcı Geri Bildirimleri ve A/B Testleri 54

4. YÖNTEM VE BULGULAR

4.1. Yöntem 55
4.1.1. Web Sitelerinin Performansı Açısından Yöntem ve Bulgular 55
4.1.2. Javascript Kütüphanelerinin Web Performans Metrikleri Açısından Karşılaştırılması 74
4.2. Bulgular 75

4.2.1.Web Performans Analizi: Farklı JavaScript Kütüphanelerinin Karşılaştırılması 75
4.2.2. Genel Durum ve İyileştirme Önerileri 76
4.2.3.Kullanıcı Deneyimi Açısından Farklı JavaScript Kütüphaneleri ile Geliştirilen Web Siteleri 77
4.2.4.Kullanıcı Açısından Genel Bulgu ve Yorumlar 79
SONUÇ 80
KAYNAKÇA 81

TABLOLAR LİSTESİ

Sayfa No.

Tablo 1. Next.Js'in Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi 59

Tablo 2. Angular Kütüphanesinin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi 62

Tablo 3. Nuxt Kütüphanesi'nin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi 63

Tablo 4. Astro Kütüphanesi'nin Farklı Browserlardaki Mobil Ve Masaüstü Web Performans Analizi 65

Tablo 5. Solidjs Kütüphanesi'nin Farklı Browserlardaki Mobil Ve Masaüstü Web Performans Analizi 68

Tablo 6. Svelte Kütüphanesi'nin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi 69

Tablo 7. Enhance Kütüphanesi'nin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi 71

Tablo 8. Lit Kütüphanesi'nin Farklı Browswerlardaki Mobil Ve Masaüstü Web Performans Analizi 72

ŞEKİLLER LİSTESİ

Sayfa No.

Şekil 1. Web Sitesi Oluşturulurken Kullanılan Temel Önyüz Teknolojileri 7
Şekil 2. HTML sayfası ve örnek DOM(Document Object Model) Yapısı 9
Şekil 3. HTML Versiyonları 10
Şekil 4. CSS 12
Şekil 5. CSS, SCSS ve LESS Yazımlarının Karşılaştırılması 13
Şekil 6. Javascript Kütüphaneleri 16
Şekil 7. Web Sitesini Oluştururken Kullanılan Backend Teknolojileri 17
Şekil 8. Minimize olmamış CSS yazım örneği 43
Şekil 9. Minimize edilmiş CSS yazım örneği 43
Şekil 10. Service Worker yazım örneği 45
Şekil 11. Önbellek yazım örneği 46
Şekil 12 . WebAssembly Yazım Örneği 47
Şekil 13 . Responsive Tasarımda Medya Yazım Örneği 47
Şekil 14 . Responsive Tasarımda Resim Optimizasyonu yazım örneği 48
Şekil 15. Web Performans Analiz Uygulaması - 1 56
Şekil 16. Web Performans Analiz Uygulaması - 2 57
Şekil 17. Web Performans Analiz Uygulaması - 3 57
Şekil 18. Web Performans Analiz Uygulaması - 4 58
Şekil 19. Next.JS Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi .. 60
Şekil 20. Angular Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi .. 62
Şekil 21. Nuxt Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi 64
Şekil 22. Astro Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi 66
Şekil 23. Solid.JS Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi . 68
Şekil 24. Svelte Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi 70
Şekil 25. Enhance Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi . 71
Şekil 26. Lit Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi 73

KISALTMALAR

AMP : Accelerated Mobile Pages (Hızlandırılmış Mobil Sayfalar)

API : Application Programming Interface (Uygulama Programlama Arayüzü)

CDN : Content Delivery Network (İçerik Dağıtım Ağı)

CI/CD : Continuous Integration/Continuous Deployment (Sürekli Entegrasyon/Sürekli Dağıtım)

CLS : Cumulative Layout Shift (Kümülatif Düzen Kayması)

CRUD : Create, Read, Update, Delete (Oluştur, Oku, Güncelle, Sil)

CSS : Cascading Style Sheets (Basamaklı Stil Şablonları)

CSP : Content Security Policy (İçerik Güvenlik Politikası)

CSRF : Cross-Site Request Forgery (Siteler Arası İstek Sahtekarlığı)

DBMS : Database Management System (Veritabanı Yönetim Sistemi)

DOM : Document Object Model (Belge Nesne Modeli)

ESLint : ESLint (JavaScript kod analizi aracı)

FCP : First Contentful Paint (İlk İçerikli Boya)

FID : First Input Delay (İlk Giriş Gecikmesi)

GIT : Git (Dağıtık Versiyon Kontrol Sistemi)

GZIP : GNU Zip (GZIP Sıkıştırma)

HTML : HyperText Markup Language (Zengin Metin İşaretleme Dili)

HTTP : Hypertext Transfer Protocol (Hiper Metin Transfer Protokolü)

HTML : HyperText Markup Language (Zengin Metin İşaretleme Dili)

HTTPS : Hypertext Transfer Protocol Secure (Güvenli Hiper Metin Transfer Protokolü)

HSTS: HTTP Strict Transport Security (HTTP Katı Taşıma Güvenliği)

IDE: Integrated Development Environment (Entegre Geliştirme Ortamı)

JMeter: JMeter (Performans test aracı)

JSON: JavaScript Object Notation (JavaScript Nesne Gösterimi)

JS: JavaScript (JavaScript)

JWT: JSON Web Token (JSON Web Belirteci)

Lazy Loading: Lazy Loading (Gereksiz öğelerin geç yüklenmesi)

LCP: Largest Contentful Paint (En Büyük İçerikli Boya)

Load Balancing: Load Balancing (Yük dengeleme)

LQIP: Low-Quality Image Placeholder (Düşük Kaliteli Görüntü Yer Tutucu)

Minification: Minification (Kod küçültme işlemi)

MVC: Model-View-Controller (Model-Görünüm-Denetleyici)

NPM: Node Package Manager (Node Paket Yöneticisi)

NoSQL: Not Only SQL (Sadece SQL Değil)

OAuth: Open Authorization (Açık Yetkilendirme)

Obfuscation: Obfuscation (Kod karıştırma işlemi)

ORM: Object-Relational Mapping (Nesne-İlişkisel Eşleme)

PWA: Progressive Web App (İleri Web Uygulaması)

RDBMS: Relational Database Management System (İlişkisel Veritabanı Yönetim Sistemi)

REST: Representational State Transfer (Temsili Durum Transferi)

RUM: Real User Monitoring (Gerçek Kullanıcı İzleme)

SPA: Single Page Application (Tek Sayfa Uygulaması)

SQL: Structured Query Language (Yapılandırılmış Sorgu Dili)

SSL: Secure Sockets Layer (Güvenli Yuva Katmanı)

SVG: Scalable Vector Graphics (Ölçeklenebilir Vektör Grafikleri)

TTFB: Time to First Byte (İlk Bayta Kadar Geçen Süre)

TTI: Time to Interactive (Etkileşime Geçme Süresi)

TLS: Transport Layer Security (Taşıma Katmanı Güvenliği)

UI: User Interface (Kullanıcı Arayüzü)

URL: Uniform Resource Locator (Tekdüzen Kaynak Bulucu)

UX: User Experience (Kullanıcı Deneyimi)

VCS: Version Control System (Versiyon Kontrol Sistemi)

WAF: Web Application Firewall (Web Uygulama Güvenlik Duvarı)

XML: eXtensible Markup Language (Genişletilebilir İşaretleme Dili)

XSS: Cross-Site Scripting (Siteler Arası Betik Çalıştırma)

Yarn: Yarn (JavaScript paket yöneticisi)

SÖZLÜK

Accelerated Mobile Pages (AMP): Hızlandırılmış Mobil Sayfalar. Mobil cihazlarda hızlı yüklenen, hafif web sayfaları oluşturmak için kullanılan bir teknoloji.

Algoritma : Belli bir problemi çözmek veya belirli bir amaca ulaşmak için sıralı adımlar halinde yazılmış talimatlar dizisidir.

Application Programming Interface (API) : Uygulama Programlama Arayüzü. Yazılımların birbirleriyle iletişim kurmasını sağlayan bir araç.

Cache (Önbellek) : Sıkça kullanılan verilerin geçici olarak saklandığı yer, böylece hızlı erişim sağlanır.

Cascading Style Sheets (CSS) : Basamaklı Stil Şablonları. Web sayfalarının görünümünü ve düzenini tanımlamak için kullanılan stil dili.

Content Delivery Network (CDN) : İçerik Dağıtım Ağı. İçeriğin, kullanıcılara en yakın sunucudan hızlıca iletilmesini sağlayan dağıtık sunucu ağı.

Cumulative Layout Shift (CLS) : Kümülatif Düzen Kayması. Web sayfasının yüklenmesi sırasında görülen beklenmedik düzen değişikliklerini ölçen bir performans metriği.

Database Management System (DBMS) : Veritabanı Yönetim Sistemi. Veritabanlarını oluşturmak, yönetmek ve kontrol etmek için kullanılan yazılım.

Document Object Model (DOM) : Belge Nesne Modeli. HTML ve XML belgelerinin programlanabilir bir nesne modelidir.

First Contentful Paint (FCP) : İlk İçerikli Boya. Bir web sayfasının ilk içeriğinin tarayıcıda görüntülenme süresi.

First Input Delay (FID) : İlk Giriş Gecikmesi. Kullanıcının web sayfasıyla ilk etkileşim kurmaya çalıştığı andan itibaren, tarayıcının bu etkileşimi yanıtlaması için geçen süre.

HyperText Markup Language (HTML) : Zengin Metin İşaretleme Dili. Web sayfalarının iskeletini oluşturan işaretleme dilidir.

Hypertext Transfer Protocol (HTTP) : Hiper Metin Transfer Protokolü. Web üzerinden veri iletimi için kullanılan bir protokol.

Hypertext Transfer Protocol Secure (HTTPS) : Güvenli Hiper Metin Transfer Protokolü. HTTP'nin güvenli versiyonudur, veri iletimini şifreler.

JavaScript (JS) : Web sayfalarına dinamik içerik eklemek için kullanılan bir programlama dilidir.

JSON Web Token (JWT) : JSON Web Belirteci. Kullanıcı doğrulama ve veri değişimi için kullanılan, şifrelenmiş bir JSON nesnesidir.

Largest Contentful Paint (LCP) : En Büyük İçerikli Boya. Bir web sayfasının en büyük içeriğinin tarayıcıda görüntülenme süresi.

Lazy Loading : Gereksiz öğelerin geç yüklenmesi. Kullanıcı sayfayı aşağı kaydırana kadar bazı içeriklerin yüklenmesini geciktiren bir teknik.

Minification : Kod küçültme işlemi. Kodun boyutunu küçültmek için gereksiz karakterlerin kaldırılması.

Model-View-Controller (MVC) : Model-Görünüm-Denetleyici. Yazılım geliştirme için kullanılan bir mimari desendir**.**

Node Package Manager (NPM) : Node Paket Yöneticisi. Node.js paketlerini yönetmek için kullanılan bir araçtır.

Object-Relational Mapping (ORM) : Nesne-İlişkisel Eşleme. Veritabanı tablolarını nesnelerle eşleştirmek için kullanılan bir tekniktir.

Performance Optimization (Performans Optimizasyonu) : Web sitesinin hızını ve verimliliğini artırmak için yapılan iyileştirme çalışmaları.

Progressive Web App (PWA) : İleri Web Uygulaması. Web teknolojilerini kullanarak yerel uygulama deneyimi sunan web uygulamaları.

Representational State Transfer (REST) : Temsili Durum Transferi. Web servisleri için kullanılan bir mimari tarz.

Scalable Vector Graphics (SVG) : Ölçeklenebilir Vektör Grafikleri. XML tabanlı, iki boyutlu vektör grafiklerini tanımlamak için kullanılan bir dosya formatı.

Single Page Application (SPA) : Tek Sayfa Uygulaması. Tüm içeriğin tek bir HTML sayfasında yüklenip dinamik olarak güncellendiği web uygulamaları.

Structured Query Language (SQL) : Yapılandırılmış Sorgu Dili. Veritabanı yönetimi ve veri manipülasyonu için kullanılan bir dil.

Time to First Byte (TTFB) : İlk Bayta Kadar Geçen Süre. Kullanıcının bir HTTP isteği yapmasından sonra tarayıcının ilk baytı almasına kadar geçen süre.

User Experience (UX) : Kullanıcı Deneyimi. Kullanıcıların bir ürün veya hizmetle olan etkileşimlerinden doğan genel deneyimleri.

Web Application Firewall (WAF) : Web Uygulama Güvenlik Duvarı. Web uygulamalarını çeşitli saldırılara karşı koruyan bir güvenlik cihazı.

Webpack : JavaScript modül bağlayıcı. Modern JavaScript uygulamalarının yapılandırılması ve paketlenmesi için kullanılan bir araç.

eXtensible Markup Language (XML) : Genişletilebilir İşaretleme Dili. Veri depolama ve iletimi için kullanılan esnek bir işaretleme dilidir.

Cross-Site Scripting (XSS) : Siteler Arası Betik Çalıştırma. Kullanıcıların tarayıcısında kötü amaçlı betiklerin çalıştırılmasıyla gerçekleştirilen bir güvenlik açığı türü.

GİRİŞ

İnternetin bulunmasıyla başlayan ve sürekli bir gelişim içinde olan web süreci, web teknolojilerindeki sürekli ilerleme ve yaygınlaşma, internet üzerindeki kullanıcı deneyimini belirleyen faktörleri de önemli ölçüde etkilemektedir. Alışveriş, haberleşme, bankacılık gibi hemen hemen tüm sektörlerde kullanılan ve günlük hayat akışımızın bir parçası olan web uygulamalarını 1969'dan itibaren kullanmaktayız. Günümüzde, web sitelerinin performansı sadece kullanıcı memnuniyeti açısından değil, aynı zamanda arama motoru sıralamaları, dönüşüm oranları ve genel kullanılabilirlik açısından da kritik bir rol oynamaktadır.

Eskiden saatlerce zaman ayırdığımız birçok işi artık tek tık ile web siteleri, çeşitli web uygulamaları üzerinden halledebiliyoruz. Teknolojinin giderek gelişmesiyle aktifleşen web, e-ticaret, mobil uygulamalar, web siteleri gibi birçok çeşitte günlük hayatımızın bir parçası haline geliyor ve bu durum sadece Google, Facebook, Amazon gibi büyük şirketler için değil, küçük ve orta ölçekli işletmeler için de önemli hale gelmeye devam ediyor. Bu nedenle web performansı, web sitelerinin hızı, yanıt verme süresi, kullanıcı etkileşimi ve genel kullanıcı deneyimi gibi unsurları içeren çok yönlü kavramlar da gün geçtikçe daha da önemli hale geliyor.

Performans odaklı web sitesi geliştirme, web sitelerinin hızını, verimliliğini ve kullanıcı deneyimini optimize etme sürecidir. Bu süreç, birçok farklı teknik, yöntem ve stratejinin birleşimini içerir ve genellikle geliştiricilerin ve tasarımcıların multidisipliner yaklaşımını gerektirir. Performans odaklı web geliştirme, web sitelerinin hızını artırmak, kaynakları daha verimli kullanmak ve kullanıcıların beklentilerini karşılayan bir deneyim sunmak için çeşitli tekniklerin uygulanmasını içerir. Bu teknikler arasında kod optimizasyonu, içerik dağıtım ağlarının kullanımı, görüntü optimizasyonu, sunucu yanıt sürelerinin iyileştirilmesi ve veri yükleme sürelerinin azaltılması yer alır.

Bu tez, performans odaklı web uygulamaları geliştirme sürecinin önemini ve etkisini incelemeyi amaçlamaktadır. Next.js, Angular, Nuxt, Astro, SolidJS, Svelte, Enhance, Lit 'ten oluşan 8 farklı JavaScript kütüphanesi ile yapılan 8 film izleme sitesi web performans açısından analiz edilmiştir ve birbirleriyle karşılaştırılmıştır. Her bir kütüphane hem masaüstü hem mobilde Chrome, Edge ve Firefox browserları ile ayrı

ayrı test edilmiştir. Web Performans testlerinde First Contentful Paint, Total Blocking Time, Speed Index, Largest Contentful Paint, Cumulative Layout Shift metrikleri için analiz yapılmıştır. Bu analizlerin amacı web sitesi geliştirme sürecinde performansın önemini göstermektir. Seçilen kütüphanelerin web performans üzerindeki etkilerinin çeşitli açılardan değerlendirilmesi, web performansın nispeten daha düşük olan kütüphanelerde web performans iyileştirmelerinin nasıl yapılabileceği hakkındaki kazanımlardan da bahsedilmiştir.

Web Performans metrikleri ile bu web sitelerinin incelenmesi, birbirleri ile kıyaslanması ve kullanılan Javascript kütüphanelerinin birbirlerine karşı avantajlı ve dezavantajlı olduğu konular da yorumlanmıştır.

Bu tez çalışmasının film izleme siteleri üzerinde yapılması çalışmayı diğer çalışmalardan ayırmaktadır ve 8 farklı Javascript kütüphanesi ile yapılan web performans analizleri ile konu derinlemesine incelenmiş ve analiz edilmiştir.

Performans odaklı web uygulamaları geliştirmek konulu bu tezin ilk bölümünde literatür araştırmasına yer verilmiş, web sitesinin tarihsel gelişimi, kullanılan Front-end ve Backend teknolojileri üzerinde durulmuştur. Tezin ikinci bölümü, web uygulamaları ile performans ilişkisidir. Bu bölümde web performansa etki eden unsurlardan bahsedilmiş ve performansın web uygulamaları için önemi üzerinde durulmuştur. Tezin üçüncü bölümü “Performans Odaklı Web Uygulamaları Nasıl Geliştirilir?” adındadır. Bu başlığın altında performansı etkileyen tüm süreçler ayrıntılı olarak anlatılmıştır. Tezin son bölümü olan dördüncü bölümde ise 8 farklı Javascript Kütüphanesi ile yapılmış web sitelerinin performansı her bir metrik için derinlemesine analiz edilmiş. Bu analizin bulguları ile bir sonuç ortaya çıkartılmıştır.

Buradaki analiz ve bulgular sonucunda, performansın web uygulamaları ve kullanıcı deneyimi üzerindeki etkisi ve önemi açıklanacaktır. Literatürdeki mevcut araştırmalar gözden geçirilecektir. Ardından, performans odaklı web geliştirme için kullanılan temel prensiplere yorum olarak değinilecektir.

Ayrıca bu tez, performans odaklı web geliştirme konusunda hem akademik hem de pratik bir bakış açısı sunmayı amaçlamaktadır. Web teknolojilerinin hızla değişen doğası göz önüne alındığında, bu çalışmanın, web geliştiricilerine ve

tasarımcılarına performans odaklı yaklaşımların önemini anlamaları ve uygulamalarında etkin bir şekilde kullanmaları için değerli bir kaynak sağlayacağına inanılmaktadır. Performansın web sitelerinin başarısında oynadığı merkezi rolü vurgulayarak, işletmelerin ve organizasyonların dijital stratejilerini geliştirmelerine yardımcı olacaktır. Web performansının optimize edilmesi, yalnızca kullanıcı memnuniyetini artırmakla kalmayıp, aynı zamanda iş hedeflerine ulaşmada da kritik bir rol oynar. Bu nedenle, performans odaklı web geliştirme, günümüzün dijital dünyasında rekabet avantajı elde etmenin ve sürdürmenin anahtarıdır.

Performans odaklı web uygulamaları geliştirme sürecinde dikkate alınması gereken bir diğer önemli konu ise sürekli izleme ve iyileştirme çalışmalarıdır. Web sitelerinin performansını izlemek ve gerektiğinde optimize etmek için çeşitli araçlar ve metrikler kullanılabilir. Bu araçlar arasında Google Lighthouse, WebPageTest ve New Relic gibi performans izleme araçları yer alır. Bu tür araçlar, geliştiricilere web sitelerinin performansını gerçek zamanlı olarak izleme ve olası sorunları hızlı bir şekilde tespit etme imkanı sağlar. Ayrıca, kullanıcı geri bildirimlerini dikkate alarak performans iyileştirme çalışmaları yapmak, web sitelerinin kullanıcı dostu olmasını ve kullanıcı beklentilerini karşılamasını sağlar. Bu tez sırasında Google PageSpeed Insights API kullanılarak web sitelerinin analizi yapmak için uygulama yapılmış ve bu uygulamadaki ilgili analiz metriklerinin verilerinden faydalanılmıştır.

Sonuç olarak, performans odaklı web uygulamaları geliştirme, modern web geliştirme süreçlerinde vazgeçilmez bir unsurdur. Web sitelerinin hızını, verimliliğini ve kullanıcı deneyimini optimize etmek, sadece teknik bir gereklilik değil, aynı zamanda iş başarısı için de kritik bir faktördür. Bu tez, performans odaklı web geliştirme konusunda derinlemesine bir anlayış sunarak, bu alanda çalışan profesyonellere ve akademisyenlere değerli bilgiler sağlayacaktır. Performansın web sitesi başarısında oynadığı merkezi rolü ve bu rolün nasıl optimize edileceğini anlamak, dijital dünyada rekabetçi kalmanın anahtarıdır.

1. LİTERATÜR ARAŞTIRMASI

Performans odaklı web uygulamaları geliştirme alanında yapılan araştırmalar, web teknolojilerinin hızla evrim geçirdiği günümüzde kritik bir öneme sahiptir. Tezin bu bölümünde web sitesinin tarihsel gelişimi ve kullanılan teknolojiler ele alınmıştır.

1.1. Web Sitesinin Gelişimi

İnternetin tarihi, birçok önemli olay ve gelişmeye sahne olmuştur. Bu süreçte, web teknolojilerindeki sürekli ilerleme ve yaygınlaşma, internet üzerindeki kullanıcı deneyimini belirleyen faktörleri önemli ölçüde etkilemiştir. Alışveriş, haberleşme, bankacılık gibi hemen hemen tüm sektörlerde kullanılan ve günlük hayat akışımızın bir parçası olan web uygulamaları, 1969'dan itibaren hayatımızda yer almaktadır. Günümüzde, web sitelerinin performansı sadece kullanıcı memnuniyeti açısından değil, aynı zamanda arama motoru sıralamaları, dönüşüm oranları ve genel kullanılabilirlik açısından da kritik bir rol oynamaktadır. Eskiden saatlerce zaman ayırdığımız birçok işi artık tek tık ile web üzerinden halledebiliyoruz. Teknolojinin giderek gelişmesiyle aktifleşen web e-ticaret, mobil uygulamalar, web siteleri gibi birçok çeşitte günlük hayatımızın bir parçası haline geliyor ve bu durum sadece Google, Facebook, Amazon gibi büyük şirketler için değil, küçük ve orta ölçekli işletmeler için de önemli hale gelmeye devam ediyor. Bu nedenle web performansı, web sitelerinin hızı, yanıt verme süresi, kullanıcı etkileşimi ve genel kullanıcı deneyimi gibi unsurları içeren çok yönlü kavramlar da gün geçtikçe daha da önemli hale geliyor (Smith, 2020).

İlk bilgisayarların büyük ve hantal makineler olduğu II. Dünya Savaşı döneminden başlayarak, bilgisayarların birbirleriyle iletişim kurma çabaları hız kazandı. 1960'larda, ABD Savunma Bakanlığı tarafından oluşturulan Advanced Research Projects Agency (ARPA), ülke genelindeki bilim adamları ve mühendisleri birbirine bağlayan bir bilgisayar ağı oluşturmayı amaçladı. Bu çabanın bir sonucu olarak 1969'da ARPANET kuruldu. Başlangıçta UCLA, UC Santa Barbara, Stanford Üniversitesi ve Utah Üniversitesi'ndeki bilgisayarlar arasında bir tür mesajlaşma servisi olarak başlayan ARPANET, türünün ilk örneği olan bir ağdı ve zamanla büyüdü. ARPANET mühendisleri, çevrimiçi yaptığımız her şeyi şekillendiren

özellikler ekledi ve sorunları çözdü. ARPANET'in önemli yeniliklerinden biri paket anahtarlamasıydı. Bu teknoloji, bilgisayarların mesajları tek bir tel seti boyunca paketler halinde göndermesine olanak tanıdı ve bu paketler, hedef bilgisayara ulaşana kadar çeşitli ara bilgisayarlar tarafından yönlendirildi (Johnson, 2018). Paket anahtarlaması, internetin verimli ve hızlı çalışmasını sağlayan temel prensiplerden biri haline geldi.

1970'ler ve 1980'ler boyunca, ARPANET büyüdükçe, adreslerin güncellenmesi ve yönetilmesi gibi zorluklar ortaya çıktı. Bu sorunları çözmek için Stanford Üniversitesi, ARPANET'in adreslerinin resmi kaydedicisi olarak seçildi ve bu sayede ağın daha düzenli ve verimli bir şekilde çalışması sağlandı. ARPANET'in büyümesiyle birlikte, benzer ağlar dünya genelinde ortaya çıkmaya başladı. Bu ağların birbirleriyle uyumlu çalışabilmesi için 1974'te geliştirilen Transmission Control Protocol/Internet Protocol (TCP/IP), paketleri biçimlendirmenin ve adresleri atamanın standart bir yolu olarak kullanıldı. TCP/IP sayesinde farklı ağlar birbirine bağlanabilir hale geldi ve ARPANET, tüm bu ağları bir arada tutan yapışkan olarak bilinen bir internet oluşturdu (Anderson, 2019).

1980'lerin sonlarına doğru, internetin daha geniş kitlelere yayılması gerektiği anlaşıldı ve ARPANET'in yerini NSFnet aldı. Bu yeni ağ, daha fazla bilgisayarı ve kullanıcıyı birbirine bağlayarak internetin omurgası haline geldi. 1990'larda ise internetin ticari kullanıma açılmasıyla birlikte, internet servis sağlayıcıları (ISP'ler) ortaya çıktı ve internet, genel halka erişilebilir hale geldi. Bu dönemde web'in doğuşu da gerçekleşti. Tim Berners-Lee ve Robert Caillou tarafından geliştirilen World Wide Web (WWW), interneti daha erişilebilir ve kullanıcı dostu hale getirdi. Web tarayıcıları aracılığıyla kullanıcılar, internet üzerindeki bilgilere kolayca erişebilmeye başladı.

1990'ların sonlarında ve 2000'lerin başlarında, internetin ticari potansiyeli keşfedildi ve birçok start-up şirketi internet üzerinden hizmet sunmaya başladı. Ancak, 2000 yılında dot-com balonunun patlamasıyla birçok şirket iflas etti. Yine de, bu dönemde Google ve Facebook gibi dev şirketler ortaya çıktı ve internetin bugünkü haline gelmesinde büyük rol oynadı. Geniş bant internetin yaygınlaşmasıyla, internet hızı ve veri depolama kapasitesi arttı, bu da web sitelerinin daha karmaşık ve etkileşimli hale gelmesine olanak tanıdı.

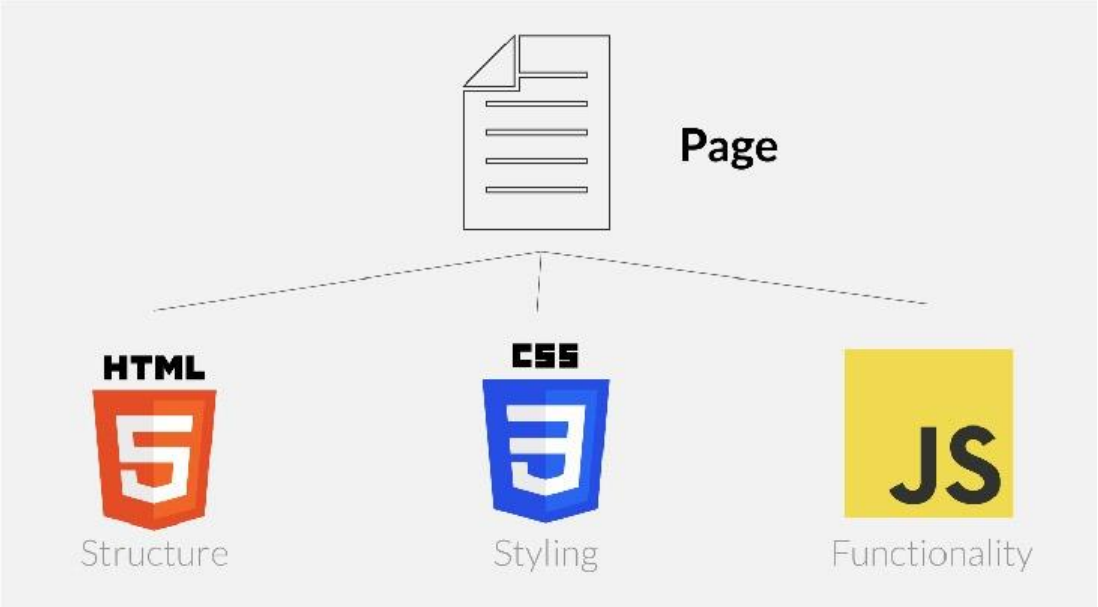
Bugün, internet milyarlarca insanın hayatının vazgeçilmez bir parçası haline gelmiştir. Web siteleri, sosyal ağlar, e-ticaret platformları ve online hizmetler, günlük yaşamın ayrılmaz bir parçası olarak kullanılmaktadır. İnternetin gelişimi, sadece teknolojik yeniliklerle değil, aynı zamanda kullanıcı ihtiyaçları ve beklentileri doğrultusunda şekillenmiştir. İnternet, bilgiye hızlı ve kolay erişim sağlamanın yanı sıra, insanlar arasındaki iletişimi ve işbirliğini de büyük ölçüde artırmıştır. Gelecekte, internetin daha da gelişmesi ve kullanıcı deneyimini iyileştiren yeniliklerle dolu olması beklenmektedir.

Sonuç olarak, internetin tarihsel gelişimi, bilgisayarların birbirleriyle iletişim kurma çabalarından başlayarak, web'in doğuşu ve geniş kitlelere yayılmasıyla devam etmiştir. Bu süreçte birçok teknik yenilik ve strateji geliştirilmiştir. İnternetin ilk yıllarında, paket anahtarlaması gibi yenilikler, verimli ve hızlı bir iletişim ağı oluşturmanın temelini atmıştır. ARPANET'ten başlayarak, internetin gelişimi sürekli olarak hızlanmış ve genişlemiştir. 1980'lerde TCP/IP protokolü ile farklı ağların birleşmesi sağlanmış ve bu sayede internet daha da büyümüştür. 1990'larda World Wide Web'in doğuşu ile internet daha kullanıcı dostu hale gelmiş ve geniş kitlelere yayılmıştır. Web tarayıcılarının geliştirilmesi, internet üzerindeki bilgilere erişimi kolaylaştırmış ve internetin ticari potansiyelini ortaya çıkarmıştır. 2000'lerin başında yaşanan dot-com balonunun patlaması, birçok şirketin iflas etmesine neden olmuş olsa da, bu dönem aynı zamanda Google ve Facebook gibi dev şirketlerin doğuşuna tanıklık etmiştir. Geniş bant internetin yaygınlaşması, internetin hızını ve veri depolama kapasitesini artırmış ve web sitelerinin daha karmaşık ve etkileşimli hale gelmesini sağlamıştır.

Günümüzde internet, milyarlarca insanın hayatının vazgeçilmez bir parçası haline gelmiştir. Web siteleri, sosyal ağlar, e-ticaret platformları ve online hizmetler, günlük yaşamın ayrılmaz bir parçası olarak kullanılmaktadır. İnternetin gelişimi, sadece teknolojik yeniliklerle değil, aynı zamanda kullanıcı ihtiyaçları ve beklentileri doğrultusunda şekillenmiştir. İnternet, bilgiye hızlı ve kolay erişim sağlamanın yanı sıra, insanlar arasındaki iletişimi ve işbirliğini de büyük ölçüde artırmıştır. Gelecekte, internetin daha da gelişmesi ve kullanıcı deneyimini iyileştiren yeniliklerle dolu olması beklenmektedir.

1.2. Web Sitesi Oluşturulurken Kullanılan Front-End Teknolojileri

Günümüzde web sitelerinin kullanıcı dostu ve etkileşimli bir deneyim sunması, front-end teknolojilerinin etkin kullanımıyla sağlanmaktadır. Front-end geliştirme, web sayfasının kullanıcıya görünen kısmını oluşturma ve yönetme sürecini ifade eder. Bu süreç, HTML (HyperText Markup Language), CSS (Cascading Style Sheets) ve JavaScript gibi temel teknolojilerin bir araya gelmesiyle gerçekleştirilir. HTML, bir web sayfasının iskeletini oluştururken, CSS bu iskeleti stilize ederek estetik ve kullanışlı bir yapıya dönüştürür. JavaScript ise web sayfasına dinamik özellikler ekleyerek kullanıcı etkileşimlerini mümkün kılar. Örneğin, kullanıcılar bir düğmeye tıkladığında sayfanın içeriğinin anında güncellenmesi veya bir formun doldurulması sırasında anında geri bildirim alınması gibi işlevler JavaScript ile sağlanır (Wang, 2019). Bu üç temel teknoloji, modern web geliştirme süreçlerinde vazgeçilmez araçlar olarak kabul edilir.



Şekil 1. Web Sitesi Oluşturulurken Kullanılan Temel Önyüz Teknolojileri

Kaynak: (Kumar, R. 2017) . *The difference between HTML, CSS and JavaScript. 2017*,https://www.web-development-institute.com/wp-content/uploads/2017/12/The-Difference-Between-HTML-CSS-and-JavaScript-12.jpg adresinden edinilmiştir.

Front-end geliştirme sürecinde kullanılan çerçeveler ve kütüphaneler, geliştiricilere daha hızlı ve verimli çalışma imkanı sunar. React, Angular ve Vue.js gibi popüler çerçeveler, front-end geliştirmede sıkça tercih edilen araçlardır. React, Facebook tarafından geliştirilmiş olup, bileşen tabanlı mimarisi sayesinde web

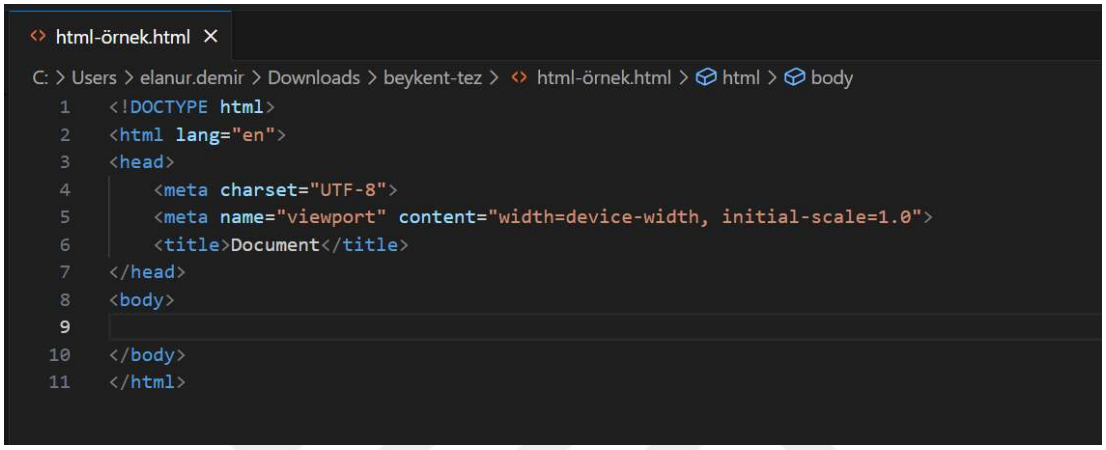
uygulamalarının modüler ve yeniden kullanılabilir bileşenler halinde geliştirilmesini sağlar. Angular, Google tarafından desteklenen bir çerçevedir ve MVC (Model-View-Controller) yapısı ile geniş çaplı uygulamaların geliştirilmesinde kullanılır. Vue.js ise esnekliği ve kolay öğrenilebilir yapısı ile dikkat çeker, küçük ve orta ölçekli projelerde yaygın olarak kullanılır. Bu çerçeveler, geliştiricilerin karmaşık uygulamaları daha kolay ve hızlı bir şekilde inşa etmelerine olanak tanır. Örneğin, Angular'ın iki yönlü veri bağlama özelliği, model ve görünüm arasındaki senkronizasyonu otomatik olarak sağlayarak geliştiricilerin iş yükünü önemli ölçüde azaltır (Larsen, 2020).

Modern web geliştirme süreçlerinde kullanılan araçlar ve otomasyon teknikleri de front-end geliştirmeyi önemli ölçüde hızlandırır ve optimize eder. Gulp, Webpack ve Babel gibi araçlar, web uygulamalarının derlenmesi, test edilmesi ve dağıtılması süreçlerini otomatikleştirir. Gulp, görev yöneticisi olarak bilinir ve kodların sıkıştırılması, dosyaların birleştirilmesi ve hataların tespiti gibi rutin işlemleri otomatikleştirir. Webpack, modül bağlayıcı olarak işlev görür ve JavaScript modüllerinin bağımlılıklarını yöneterek daha verimli bir şekilde yüklenmesini sağlar. Babel ise modern JavaScript kodlarının eski tarayıcılarda da çalışabilmesini sağlamak için bu kodları geriye dönük uyumlu hale getirir. Bu araçlar sayesinde geliştiriciler, kodlarını yazarken ve test ederken daha az zaman harcar ve projelerini daha hızlı bir şekilde hayata geçirebilirler. Webpack, özellikle büyük projelerde modüllerin bağımlılıklarını yöneterek sayfa yükleme sürelerini optimize eder ve kullanıcı deneyimini iyileştirir (Smith, 2021).

Front-end teknolojilerinin gelişimi, web sitelerinin kullanıcı deneyimini doğrudan etkileyen unsurların başında gelir. Bu teknolojiler sayesinde web siteleri daha hızlı, estetik ve kullanıcı dostu hale gelir. HTML, CSS ve JavaScript'in yanı sıra, modern çerçeveler ve otomasyon araçları, geliştiricilerin daha verimli çalışmasını sağlar. Front-end geliştirme, sadece kod yazmakla sınırlı kalmayıp, kullanıcı ihtiyaçlarını anlamak ve onlara en iyi deneyimi sunmak için sürekli olarak gelişen bir alan olmaya devam etmektedir. Bu süreçte kullanılan araçlar ve teknikler, web uygulamalarının performansını ve kullanılabilirliğini artırarak, hem geliştiricilere hem de kullanıcılara büyük faydalar sağlar.

1.2.1. HTML

HTML (HyperText Markup Language), web sitelerinin temel yapısını oluşturmak için kullanılan en temel teknolojidir. Tim Berners-Lee tarafından 1991 yılında geliştirilmiş olup, internetin yaygınlaşması ve web sitelerinin kullanıcı dostu hale gelmesinde önemli bir rol oynamıştır. HTML, bir web sayfasının iskeletini oluşturur ve tarayıcıların bu yapıyı doğru bir şekilde yorumlamasını sağlar.

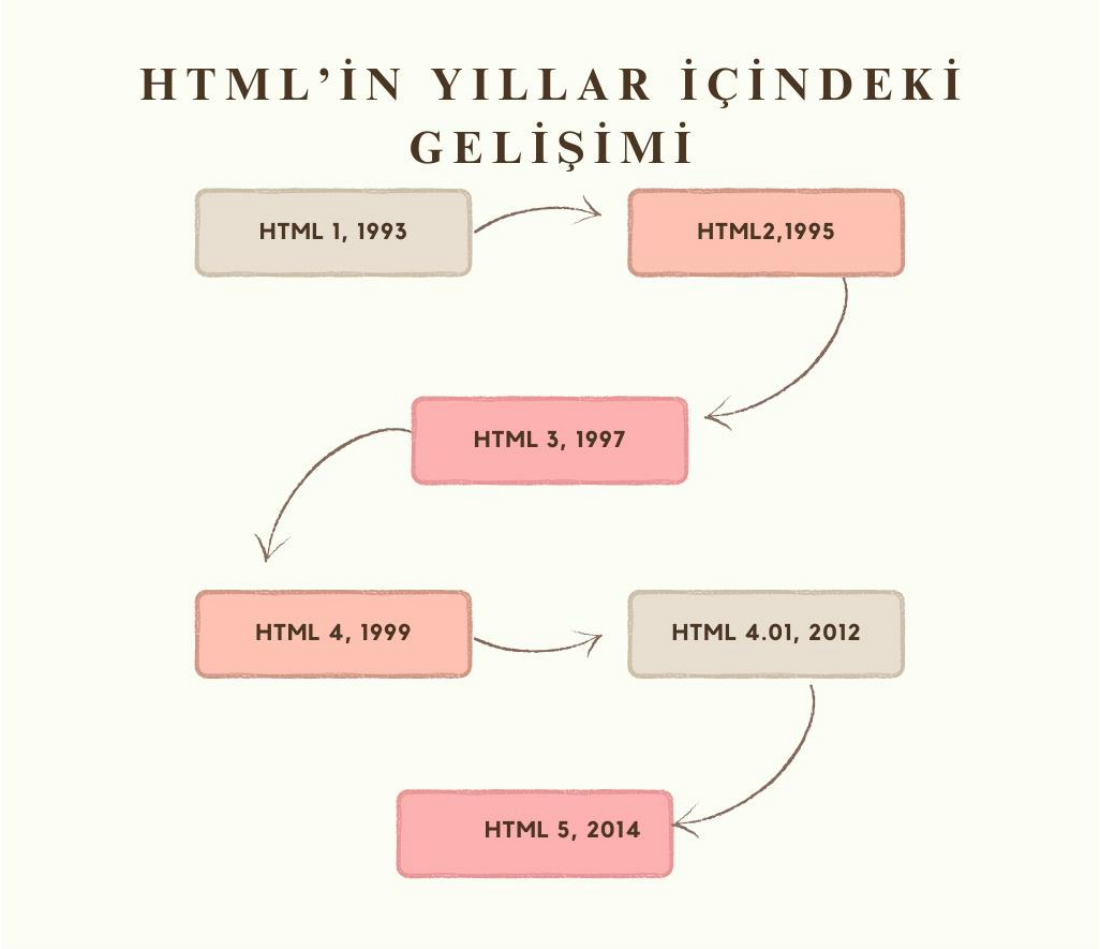


Şekil 2. HTML sayfası ve örnek DOM(Document Object Model) Yapısı

Kaynak: Yazar tarafından oluşturulmuştur.

1.2.1.1. HTML'nin Tarihçesi

HTML'nin kökenleri, 1980'lerde CERN'de geliştirilen ENQUIRE sistemi ve 1989'da Tim Berners-Lee'nin önerdiği internet tabanlı hiper metin sistemine dayanır. 1991'de HTML'nin ilk versiyonu yayımlandı ve bu, internetin hızla yayılmasına olanak sağladı. HTML, yıllar içinde çeşitli evrimler geçirmiştir ve bugün en yaygın kullanılan versiyonu HTML5'tir. HTML5, sadece metin tabanlı içerikleri değil, aynı zamanda video, ses ve diğer multimedya unsurlarını da destekler hale gelmiştir (Jackson, 2016).



Şekil 3. HTML Versiyonları

Kaynak: Yazar tarafından oluşturulmuştur

1.2.1.2. HTML5 ve Yenilikleri

HTML5, önceki sürümlerine göre birçok yenilik ve iyileştirme sunar. Bu yenilikler arasında semantik etiketler, yerel video ve ses desteği, Canvas API, ve daha iyi form kontrolleri bulunmaktadır. Semantik etiketler, sayfa içeriğinin daha iyi anlaşılmasını ve arama motorları tarafından daha kolay indekslenmesini sağlar. Örneğin, <header>, <footer>, <article> ve <section> gibi etiketler, içeriğin yapısını daha anlamlı kılar. Ayrıca, HTML5 ile gelen Canvas API, geliştiricilere grafiksel içerikler oluşturma imkanı sunar, bu da interaktif oyunlar ve uygulamalar için önemli bir yeniliktir (Osmani, 2020).

HTML5 ayrıca, mobil cihazlarda daha iyi performans sağlamak için tasarlanmıştır. Yerel video ve ses desteği, üçüncü parti eklentilere ihtiyaç duymadan medya içeriklerinin doğrudan HTML5 etiketleri ile eklenmesine olanak tanır. Bu, kullanıcı deneyimini geliştirir ve sayfa yükleme sürelerini azaltır. Örneğin, <video> ve <audio> etiketleri, medya dosyalarını doğrudan sayfaya yerleştirmenizi sağlar ve farklı tarayıcılarda sorunsuz çalışır (Osmani, 2020).

1.2.1.3. HTML'nin Geleceği

HTML'nin geleceği, sürekli gelişen web teknolojileri ve kullanıcı ihtiyaçlarına göre şekillenmektedir. HTML5.1 ve HTML5.2 gibi sürümlerle gelen ek özellikler, web geliştiricilerine daha fazla esneklik ve güç sunmaktadır. Bu sürümler, performansı artırmak, güvenliği iyileştirmek ve yeni web standartlarına uyum sağlamak amacıyla tasarlanmıştır. Tarayıcı üreticileri de bu gelişmeleri destekleyerek, HTML'nin evrimini hızlandırmaktadır (Wang, 2019).

Gelecekte, HTML'nin yapısal yeteneklerinin daha da genişlemesi ve yeni API'ler ile entegrasyonun artması beklenmektedir. WebAssembly gibi teknolojilerle birlikte çalışarak, web uygulamalarının performansını masaüstü uygulama seviyesine çıkarmak mümkündür. Ayrıca, Progressive Web Apps (PWA) gibi yaklaşımlar, HTML'yi daha güçlü ve esnek hale getirerek, web uygulamalarının mobil cihazlarda yerel uygulamalara benzer bir deneyim sunmasını sağlar (Jackson, 2016).

Sonuç olarak, HTML, web sitelerinin temel taşını oluşturan bir teknolojidir. Tarihsel gelişimi boyunca birçok yenilik ve iyileştirme ile güncellenmiş ve bugün modern web geliştirme süreçlerinin vazgeçilmez bir parçası haline gelmiştir. HTML5 ile birlikte gelen yeni özellikler ve API'ler, web geliştiricilerine daha fazla imkan sunarken, kullanıcı deneyimini de önemli ölçüde iyileştirmiştir. Gelecekte de HTML'nin gelişmeye devam etmesi ve web teknolojilerinin evrimiyle birlikte daha da güçlenmesi beklenmektedir.

1.2.2. CSS

CSS (Cascading Style Sheets), web sayfalarının görünümünü ve düzenini tanımlamak için kullanılan bir stil dilidir. Web sayfalarını bir insan bedeni olarak düşünecek olursak HTML iskelet, CSS ise vücuttur, yani iskeletin görünümünü belirler. 1996 yılında W3C tarafından geliştirilen CSS, HTML ile birlikte web sayfalarına stil ekler. CSS, renkler, yazı tipleri, kenar boşlukları ve düzen seçenekleri gibi görsel öğeleri belirler ve web sitelerinin kullanıcı deneyimini iyileştirir.

Şekil 4. CSS

Kaynak: Yazar tarafından oluşturulmuştur

1.2.2.1. CSS'nin Tarihçesi

CSS'in ilk sürümü 1996'da yayımlandı. 1998'de CSS2 tanıtıldı ve daha fazla stil seçeneği sağladı. CSS2.1, 2004'te güncellenmiş bir sürüm olarak yayımlandı. CSS3 ise modüler yapısıyla geliştiricilere daha fazla esneklik sunar. CSS3, animasyonlar, geçişler, gölgeler ve medya sorguları gibi modern web tasarımı için önemli özellikler ekler (Meyer, 2017).

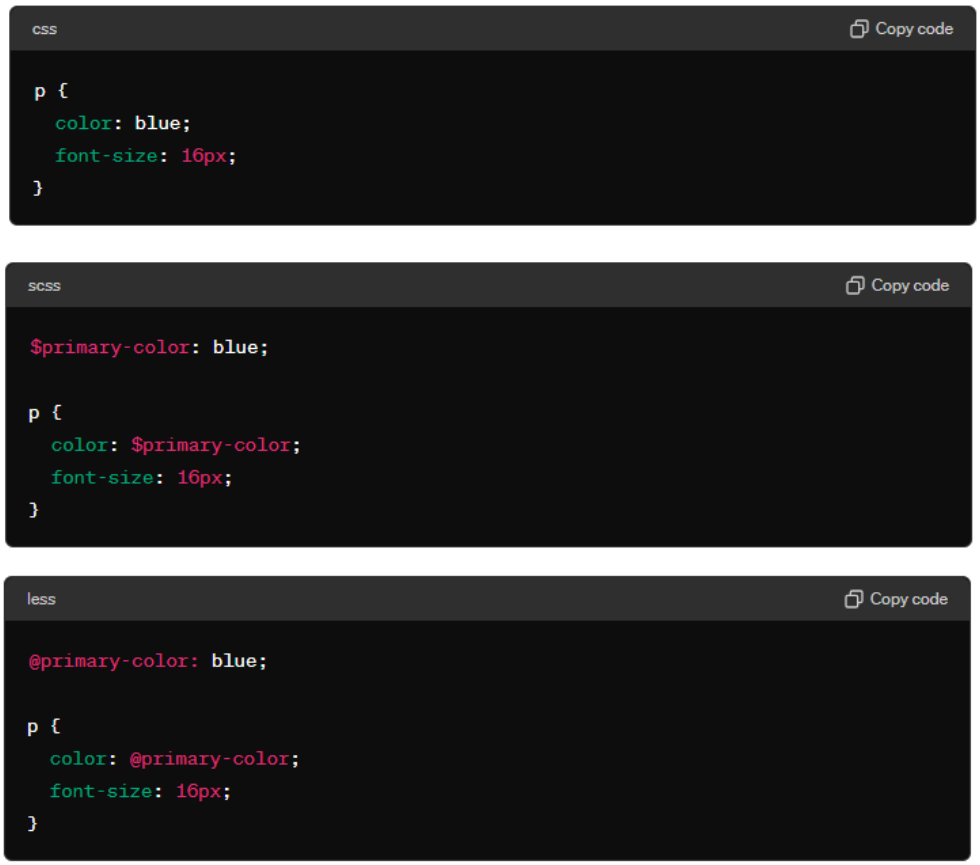
1.2.2.2. Temel Özellikler

CSS, web sayfalarının görünümünü belirlemek için çeşitli özellikler sunar. Flexbox ve Grid Layout gibi modern düzen teknikleri, geliştiricilere karmaşık düzenler oluşturma imkanı tanır. Flexbox, esnek kutu modeli ile tek boyutlu düzenler

oluştururken, Grid Layout iki boyutlu düzenler sağlar. Bu teknikler, responsive (duyarlı) web tasarımı için de önemlidir (Cederholm, 2015).

1.2.2.3. Modern CSS Kullanımı

Modern web geliştirme süreçlerinde CSS, SASS ve LESS gibi CSS ön işlemcileri ile birlikte kullanılır. Bu araçlar, CSS yazımını daha etkili ve sürdürülebilir hale getirir. Değişkenler, iç içe geçmiş kurallar ve fonksiyonlar gibi özellikler, daha temiz ve yönetilebilir stil dosyaları oluşturmayı sağlar. Ayrıca, BEM (Block, Element, Modifier) gibi CSS mimari yaklaşımları, büyük ölçekli projelerde tutarlılığı ve yeniden kullanılabilirliği artırır (Marcotte, 2019).



Şekil 5. CSS, SCSS ve LESS Yazımlarının Karşılaştırılması

Kaynak: Yazar tarafından oluşturulmuştur

Sonuç olarak, CSS, web sayfalarının estetik ve kullanıcı dostu bir şekilde tasarlanmasında kritik bir rol oynar. CSS, temel stil özelliklerinden karmaşık düzen tekniklerine kadar geniş bir yelpazede araçlar sunar ve web geliştiricilere sayfa tasarımında büyük esneklik sağlar.

1.2.3. JavaScript

JavaScript (JS), web sayfalarına dinamik ve etkileşimli özellikler eklemek için kullanılan güçlü bir programlama dilidir. İlk olarak 1995 yılında Brendan Eich tarafından geliştirilen JS, HTML ve CSS ile birlikte modern web geliştirme sürecinin vazgeçilmez bir parçasıdır. HTML'i bir insan bedeninin iskeleti, CSS'i vücut, JavaScript'i ise sinir sistemi olarak düşünebiliriz. JavaScript, tarayıcıda çalışarak kullanıcı etkileşimlerine hızlı bir şekilde yanıt verir ve web sayfalarını daha kullanıcı dostu hale getirir.

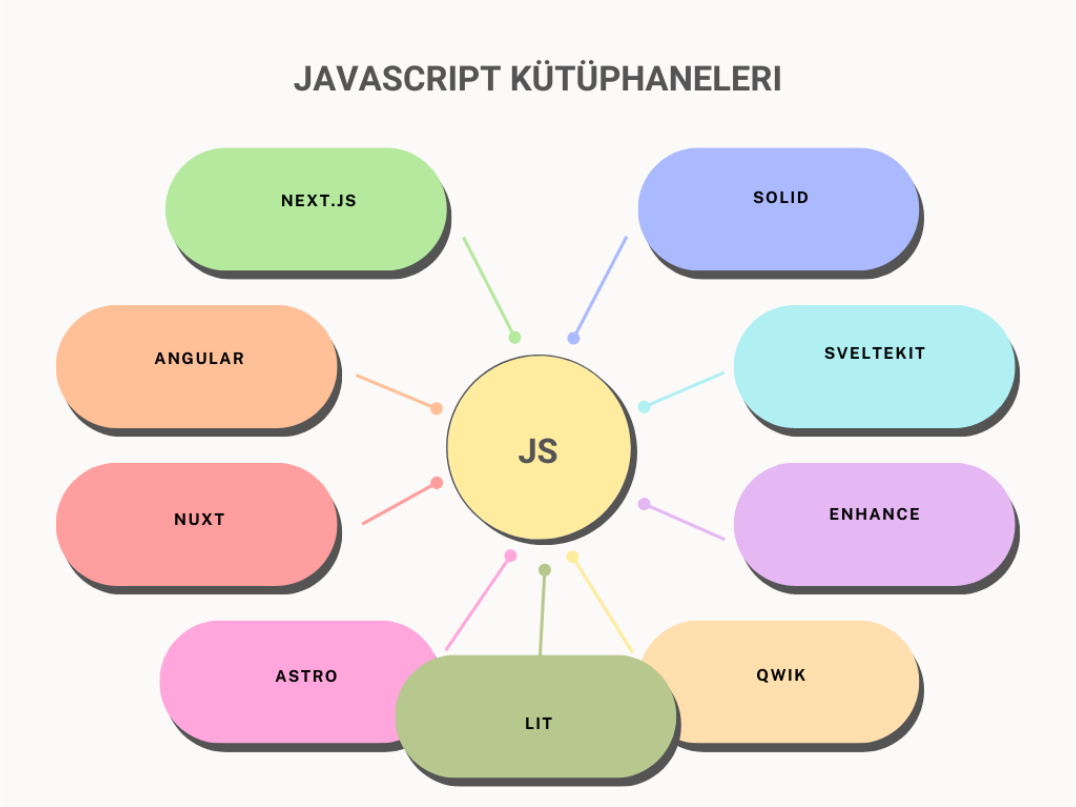
JavaScript, esnekliği ve geniş kütüphane desteği ile öne çıkar. Çeşitli çerçeveler ve kütüphaneler (örneğin, React, Angular, Vue.js) JS'nin gücünü artırarak geliştiricilere daha hızlı ve verimli bir geliştirme süreci sunar. Bu araçlar, karmaşık kullanıcı arayüzleri ve tek sayfa uygulamaları (SPA) oluşturmayı kolaylaştırır. JavaScript'in asenkron programlama yetenekleri, AJAX ve Fetch API gibi teknolojilerle birleşerek, veri alışverişi ve arka planda işlemler yapmayı sağlar (Osmani, 2012).

Modern JavaScript, ECMAScript (ES6 ve sonrası) standartları ile daha da güçlenmiştir. Bu standartlar, dilin sözdizimini ve yeteneklerini genişleterek geliştiricilere daha fazla esneklik ve güçlü araçlar sunar. Örneğin, ES6 ile gelen let ve const değişken bildirimleri, arrow fonksiyonları, sınıflar ve modüller gibi özellikler, JS'nin yazımını ve bakımını daha kolay hale getirir. Ayrıca, bu yeni özellikler, kodun okunabilirliğini ve sürdürülebilirliğini artırarak, büyük projelerdeki karmaşıklığı azaltır. Addy Osmani'nin çalışmalarında vurguladığı gibi, JavaScript'in sürekli evrimi, web uygulamalarının performansını ve kullanıcı deneyimini artırmada kritik bir rol oynamaktadır (Osmani, 2018).

JavaScript'in önemli bir başka özelliği ise geniş ekosistemidir. Node.js ile sunucu tarafında da kullanılabilen JS, tam anlamıyla bir full-stack dil haline gelmiştir. Bu, geliştiricilerin hem istemci tarafında hem de sunucu tarafında aynı dili kullanarak projelerini daha bütünsel bir şekilde yönetmelerine olanak tanır. Ayrıca, NPM (Node Package Manager) gibi paket yönetim sistemleri, binlerce açık kaynaklı kütüphane ve modülü kolayca projelere entegre etmeyi sağlar, bu da geliştirme sürecini hızlandırır ve kolaylaştırır.

JavaScript'in popülaritesi, sürekli olarak geliştirilen araçlar ve çerçeveler sayesinde artmaktadır. TypeScript, JavaScript'in üzerine eklenen tip güvenliği ile daha büyük projelerde hata yönetimini kolaylaştırırken, Redux gibi durum yönetim kütüphaneleri, uygulamaların daha tahmin edilebilir ve kolay yönetilebilir olmasını sağlar. Bu araçlar, JavaScript'in esnek yapısını korurken, aynı zamanda daha sağlam ve ölçeklenebilir uygulamalar geliştirmeyi mümkün kılar.

Sonuç olarak, JavaScript, modern web geliştirmede vazgeçilmez bir araçtır. Dinamik yapısı ve geniş kütüphane desteği ile hem kullanıcı arayüzlerini zenginleştirir hem de geliştirme sürecini hızlandırır. JS'nin sürekli evrilen yapısı ve geniş ekosistemi, onu geliştiriciler için ideal bir seçenek haline getirir. Bu nedenle, JavaScript'i etkin bir şekilde kullanmak, web geliştirme süreçlerinde başarının anahtarıdır.



Şekil 6. Javascript Kütüphaneleri

Kaynak: Yazar tarafından oluşturulmuştur.

1.3. Web Sitesini Oluştururken Kullanılan Backend Teknolojileri

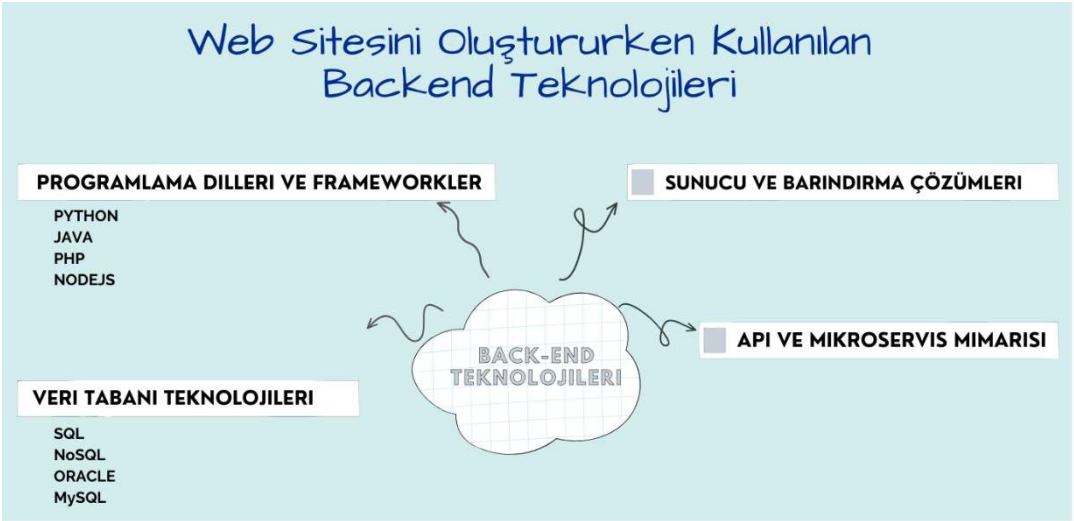
Web sitelerinin arka planda çalışan ve kullanıcıya görünmeyen kısmını oluşturan backend teknolojileri, web uygulamalarının işlevselliği ve performansı açısından kritik öneme sahiptir. Backend geliştirme, veri tabanları, sunucular ve uygulama mantığını içerir. Bu teknolojiler, kullanıcının tarayıcıda yaptığı istekleri işleyerek gerekli veri işlemlerini gerçekleştirir ve sonuçları geri döner.

Backend geliştirme için kullanılan başlıca programlama dilleri arasında PHP, Python, Ruby, Java ve Node.js bulunmaktadır. PHP, özellikle WordPress gibi içerik yönetim sistemlerinde yaygın olarak kullanılırken, Python Django ve Flask frameworkleri ile popülerdir. Ruby on Rails, hızlı geliştirme süreçleri ve geniş topluluğu ile dikkat çekerken, Java güçlü performansı ve güvenilirliği ile büyük kurumsal uygulamalarda tercih edilir. Node.js ise JavaScript'in sunucu tarafında

kullanılmasını sağlayarak, hem frontend hem de backend geliştirmeyi aynı dil ile yapma imkanı sunar (Bakioğlu, 2012).

Backend teknolojilerinin önemli bir bileşeni olan veritabanları, verilerin depolanması ve yönetilmesi görevini üstlenir. İlişkisel veritabanları (RDBMS) olarak bilinen MySQL, PostgreSQL ve Oracle, yapılandırılmış veri yönetiminde sıkça kullanılır. NoSQL veritabanları ise MongoDB ve CouchDB gibi, daha esnek ve büyük veri işleme yetenekleri ile dikkat çeker. Bu veritabanları, özellikle büyük veri hacimleri ve yüksek performans gerektiren uygulamalarda tercih edilmektedir (Okutucu, 2012).

Modern backend teknolojilerinde bulut bilişim ve sanallaştırma önemli bir rol oynar. Bulut bilişim, dinamik ve ölçeklenebilir özellikleri sayesinde verinin daha hızlı ve daha düşük maliyetle erişilebilir olmasını sağlar. AWS, Microsoft Azure ve Google Cloud gibi bulut platformları, çeşitli hizmetler sunarak geliştiricilerin uygulamalarını kolayca dağıtmasına ve yönetmesine olanak tanır. Sanallaştırma teknolojileri, fiziksel sunucuların kaynaklarını daha verimli kullanarak, birden çok sanal makinenin aynı donanım üzerinde çalışmasını sağlar (Yardımcıoğlu, 2019).



Şekil 7. Web Sitesini Oluştururken Kullanılan Backend Teknolojileri

Kaynak: Yazar tarafından oluşturulmuştur.

1.3.1. Programlama Dilleri ve Frameworkler

Web uygulamalarının sunucu tarafındaki işlemlerini yöneten backend teknolojileri, kullanıcı isteklerini işleyerek gerekli veri işlemlerini gerçekleştirir ve sonuçları geri döner. Bu süreçte kullanılan programlama dilleri ve frameworkler, uygulamaların performansı ve güvenilirliği açısından büyük önem taşır.

Python, güçlü ve esnek bir seçenek olarak özellikle Django ve Flask frameworkleri ile dikkat çeker. Django, hızlı geliştirme ve temiz, pragmatik bir tasarım sunarken, Flask minimal ve genişletilebilir yapısıyla tercih edilir. Python'un bu esnekliği, geliştiricilere farklı ihtiyaçlara yönelik çözümler sunar.

Java, büyük kurumsal uygulamalar için tercih edilen bir dil olup, Spring frameworkü ile yaygın olarak kullanılır. Spring, güçlü güvenlik ve veri işleme özellikleri sunar. Özellikle büyük ölçekli projelerde Java ve Spring'in sağladığı performans ve güvenilirlik, bu teknolojilerin tercih edilmesinde önemli bir faktördür. PHP ise web geliştirme dünyasında uzun süredir kullanılan bir dil olup, özellikle WordPress gibi içerik yönetim sistemleriyle bilinir. Laravel, PHP için modern ve zarif bir framework olup, geliştirme sürecini hızlandırır ve kodun sürdürülebilirliğini artırır.

Node.js, JavaScript'in sunucu tarafında kullanılmasını sağlayarak, hem frontend hem de backend geliştirmede tek bir dil kullanma imkanı sunar. Express.js, Node.js ile sıkça kullanılan hafif ve esnek bir framework olup, RESTful API'ler oluşturmayı kolaylaştırır. Bu teknolojiler, geliştiricilere esneklik ve hız kazandırırken, aynı zamanda performans ve güvenliği de sağlar.

Programlama dilleri ve frameworklerin seçimi, projenin gereksinimlerine ve geliştiricilerin becerilerine bağlı olarak yapılmalıdır. Doğru araçların seçimi, web uygulamalarının verimliliğini ve kullanıcı memnuniyetini artırmada kritik bir rol oynar.

1.3.2. Veri Tabanı Teknolojileri

Veri tabanı teknolojileri, web uygulamalarının veri yönetimini sağlamak için kullanılan önemli bileşenlerdir. Bu teknolojiler, verilerin saklanması, düzenlenmesi ve erişilmesi süreçlerini optimize eder. Veri tabanları, özellikle büyük veri hacimlerinin

yönetilmesinde ve performansın artırılmasında kritik bir rol oynar. İlişkisel veri tabanları (RDBMS), nesneye dayalı veri tabanları ve uzman veri tabanları gibi çeşitli veri tabanı sistemleri, farklı ihtiyaçlara yönelik çözümler sunar.

İlişkisel veri tabanları, veri yönetimi için en yaygın kullanılan sistemlerdir. Bu sistemler, verileri tablolar halinde organize eder ve SQL (Structured Query Language) kullanarak veri erişimini sağlar. İlişkisel veri tabanları, veri bütünlüğü ve tutarlılığı sağlamak için güçlü mekanizmalara sahiptir. Ancak, büyüyen veri hacimlerine ve karmaşık veri yapılarına yanıt vermede yetersiz kalabilirler. Bu durum, nesneye dayalı veri tabanları ve uzman veri tabanlarının gelişimine yol açmıştır (Fırlar, 2003).

Nesneye dayalı veri tabanları, veri ve nesne yönelimli programlama özelliklerini bir araya getirir. Bu sistemler, daha karmaşık veri yapılarının ve ilişkilerin yönetilmesini sağlar. Nesneye dayalı veri tabanları, özellikle grafik uygulamaları, mühendislik veri yönetimi ve multimedya sistemlerinde kullanılır. Bu veri tabanları, ilişkisel veri tabanlarının sınırlamalarını aşarak, daha esnek ve ölçeklenebilir çözümler sunar. Nesneye dayalı sistemlerde, veri ve iş mantığı birleşik bir şekilde ele alınarak, performans ve veri bütünlüğü artırılır (Fırlar, 2003).

Uzman veri tabanları, uzman sistemlerin veri tabanı yönetim sistemleri ile entegrasyonunu sağlar. Bu sistemler, karmaşık sorgulamalar ve veri işleme görevlerini optimize eder. Uzman veri tabanları, özellikle dinamik tipler ve ad-hoc sorgular ile çalışır. Dinamik tip sistemler, verilerin çalışma zamanında tanımlanmasını ve yönetilmesini sağlar. Ad-hoc sorgular ise, kullanıcıların anlık veri ihtiyaçlarına esnek çözümler sunar. Uzman veri tabanları, ilişkisel ve nesneye dayalı veri tabanlarının avantajlarını bir araya getirerek, daha gelişmiş veri yönetimi imkanları sunar (Fırlar, 2003).

Veri tabanı teknolojilerinin seçimi, uygulamanın ihtiyaçlarına ve veri yapısının karmaşıklığına bağlı olarak yapılmalıdır. Doğru veri tabanı teknolojisinin seçilmesi, web uygulamalarının performansını ve verimliliğini artırmada kritik bir rol oynar. Veri tabanı sistemlerinin gelişimi, sürekli olarak yeni teknolojilerin ve yöntemlerin eklenmesiyle devam etmektedir.

1.3.3. Sunucu ve Barındırma Çözümleri

Web uygulamalarının performansı ve erişilebilirliği açısından sunucu ve barındırma çözümleri kritik bir rol oynar. Bir web uygulamasının kullanıcılar tarafından sorunsuz ve hızlı bir şekilde erişilebilmesi için doğru sunucu ve barındırma altyapısının seçilmesi gereklidir. Geleneksel fiziksel sunucular, güçlü donanım özellikleriyle yüksek performans sağlarken, bulut tabanlı çözümler ise esneklik ve ölçeklenebilirlik sunar. Bulut bilişim, Amazon Web Services (AWS), Microsoft Azure ve Google Cloud Platform (GCP) gibi hizmetlerle, kaynakların ihtiyaç duyulduğunda dinamik olarak artırılmasını veya azaltılmasını mümkün kılar.

Sunucu ve barındırma çözümleri arasında yönetilen hizmetler de önemli bir yer tutar. Yönetilen barındırma hizmetleri, sunucu yönetimi, güvenlik, yedekleme ve performans optimizasyonu gibi görevleri üstlenir. Bu hizmetler, geliştiricilerin ve işletmelerin odaklarını uygulama geliştirmeye ve iş süreçlerine yönlendirmesine olanak tanır. Örneğin, AWS Elastic Beanstalk, uygulamaların hızlı bir şekilde dağıtılmasını ve yönetilmesini sağlayan, tamamen yönetilen bir hizmettir.

Klasik fiziksel sunucular, yüksek başlangıç maliyetleri ve bakım gereksinimleri ile gelirken, sanal sunucular ve konteyner teknolojileri daha ekonomik ve yönetimi kolay seçenekler sunar. Konteyner teknolojileri, Docker ve Kubernetes gibi araçlarla, uygulamaların daha taşınabilir ve izole çalıştırılmasını sağlar. Veri merkezlerinde kullanılan soğutma sistemleri de enerji tüketimini optimize ederek sürdürülebilirlik sağlar. "Veri merkezlerindeki soğutma sistemlerinin verimsizliği nedeniyle toplam enerji tüketiminin %30-50'si soğutma enerji tüketimidir" (Yüzgeç ve Günel, 2014).

Özetle, web uygulamalarının başarılı olması için doğru sunucu ve barındırma çözümlerinin seçilmesi hayati öneme sahiptir. Doğru çözümler, performansı artırırken maliyetleri düşürür ve uygulamanın ölçeklenebilirliğini sağlar. Bu bağlamda, bulut bilişim ve yönetilen hizmetler, modern web geliştirme süreçlerinde vazgeçilmez araçlar haline gelmiştir.

1.3.4. API ve Mikroservis Mimarisi

API ve mikroservis tabanlı mimariler, modern yazılım geliştirme süreçlerinde esneklik, ölçeklenebilirlik ve dayanıklılık sağlamak için kritik öneme sahiptir.

Mikroservis mimarisi, büyük ve monolitik uygulamaları daha küçük, bağımsız hizmetlere (mikroservislere) bölerek yönetmeyi ve geliştirmeyi kolaylaştırır. Bu yaklaşım, her bir mikroservisin bağımsız olarak dağıtılabilmesi, ölçeklenebilmesi ve yönetilebilmesi avantajını sunar. Mikroservisler arasındaki iletişim genellikle RESTful API'ler veya mesajlaşma sistemleri aracılığıyla sağlanır, bu da hizmetlerin birbirine gevşek bağlı olmasını sağlar (Ünlü, Bilgin, Demirörs, 2021).

API'ler, farklı yazılım bileşenlerinin ve hizmetlerinin birbirleriyle iletişim kurmasını sağlayan arayüzlerdir. Mikroservis tabanlı mimarilerde, her mikroservis kendi API'sini sunar ve diğer mikroservislerle API'ler aracılığıyla etkileşime girer. Bu, hizmetlerin yeniden kullanılabilirliğini ve entegrasyonunu kolaylaştırır. Ayrıca, API'ler sayesinde farklı sistemler ve platformlar arasında veri alışverişi sağlanabilir, bu da uygulamaların esnekliğini ve uyarlanabilirliğini artırır.

Mikroservis mimarisinin temel özellikleri arasında gevşek bağlılık, yüksek kohezyon, izolasyon, özerklik ve hata toleransı bulunur. Gevşek bağlılık, mikroservislerin birbirine bağımlı olmadan çalışabilmesini sağlar. Yüksek kohezyon, her bir mikroservisin belirli bir işlevi yerine getirmesine odaklanır. İzolasyon, mikroservislerin bağımsız olarak test edilebilmesi ve dağıtılabilmesi anlamına gelir. Özerklik, her bir mikroservisin kendi veri tabanına sahip olması ve bağımsız olarak çalışabilmesi demektir. Hata toleransı ise, bir mikroservisin hataya karşı dayanıklı olması ve diğer mikroservislerden bağımsız olarak çalışabilmesini ifade eder (Ünlü, Bilgin, Demirörs, 2021).

Sonuç olarak, mikroservis tabanlı mimariler ve API'ler, modern yazılım geliştirme süreçlerinde esneklik ve ölçeklenebilirlik sağlamak için önemli araçlardır. Mikroservis mimarisi, büyük ve karmaşık uygulamaların daha yönetilebilir ve sürdürülebilir hale gelmesini sağlar. API'ler ise, hizmetlerin entegrasyonunu ve yeniden kullanılabilirliğini kolaylaştırır. Bu bağlamda, mikroservis tabanlı mimariler ve API'ler, yazılım geliştiricilere daha dinamik ve uyarlanabilir çözümler sunar.

2. WEB UYGULAMALARI VE PERFORMANS İLİŞKİSİ

Günümüzde web uygulamalarının performansı kritik bir öneme sahiptir. Bu alanda bir çok geliştirmeler yapılmaktadır. Web performans, web sitelerinin satış, etkileşim gibi çeşitli alanlarını etkilediği gibi kullanıcı ve arama motoru optimizasyonu gibi konuları da oldukça etkilemektedir. Tezin bu bölümünde, web performansa etki eden unsurlar ve performansın web sitesi için önemi ele alınmıştır.

2.1. Web Performansa Etki Eden Unsurlar

Telefonlar ve bilgisayarlar icat edildiğinden beri sürekli gelişmektedir. Aynı şekilde internet de sürekli gelişmekte ve hızı dünya genelinde artmaya devam etmektedir. Ancak web siteleri tasarım ve kodlama olarak geliştikçe daha da yavaşlamıştır. Bir web sitesinin tam olarak yüklenmesi birkaç saniye sürebilir, bazense bu süre uzayabilir. Bu durum performans açısından son derece tehlikelidir. Çünkü yavaş performans kullanıcıların yüklenmekte olan sayfayı terk etmesine yol açabilir. Bu da siteyi arama motoru sıralamasında geriletir ve diğer olumsuz sonuçları da beraberinde getirebilir. (Singhal ve Cutts, 2010; Wang ve Phan, 2018)

2.1.1. Web Sitesi Tasarımının Performansa Etkisi

Web sitesi tasarımı, bir web sitesinin kullanıcı deneyimi ve performansı üzerinde doğrudan bir etkiye sahiptir. Kullanıcıların web sitesini nasıl deneyimlediği, sitenin tasarım unsurlarının düzenlenmesi, yüklenme süreleri ve genel kullanılabilirliği ile yakından ilişkilidir. İyi tasarlanmış bir web sitesi, kullanıcıların ihtiyaçlarını hızlı ve etkili bir şekilde karşılayabilirken, kötü tasarlanmış bir site kullanıcıları kaçırabilir ve performans sorunlarına yol açabilir.

Web sitesi tasarımının performansa olan etkisinin birincil nedeni, görsel ve işlevsel bileşenlerin yüklenme süreleridir. Büyük boyutlu resimler, yoğun JavaScript kullanımı ve aşırı stil dosyaları, web sayfasının yüklenme süresini uzatabilir. Bu durum, özellikle mobil kullanıcılar için önemli bir sorun teşkil eder. "Görsel öğelerin optimize edilmemesi, sayfa yüklenme sürelerini önemli ölçüde artırabilir ve kullanıcı memnuniyetini azaltabilir" (Peña-Ortiz, 2013).

Web sitesinin düzeni ve navigasyonu da performans üzerinde büyük bir etkiye sahiptir. Kullanıcıların istedikleri bilgiye hızlıca ulaşabilmeleri için iyi organize edilmiş bir navigasyon sistemi gereklidir. Kötü yapılandırılmış bir navigasyon, kullanıcıların sitede kaybolmasına ve aradıkları bilgiye ulaşamamalarına neden olabilir. Bu durum, kullanıcıların siteyi terk etmesine ve geri dönmemesine yol açabilir. Ayrıca, karmaşık ve kullanıcı dostu olmayan bir düzen, kullanıcıların sitede geçirdiği süreyi azaltarak, etkileşim oranlarını düşürebilir.

Web sitesi tasarımında kullanılan teknikler de performansı etkileyebilir. Örneğin, responsif tasarım teknikleri kullanılarak, sitenin farklı cihaz ve ekran boyutlarına uyum sağlaması sağlanabilir. Bu, mobil cihazlarda daha iyi bir kullanıcı deneyimi sunarak, kullanıcıların sitede daha fazla vakit geçirmesini ve daha az terk etme oranına sahip olmasını sağlar. Ayrıca, hızlı yüklenme süreleri ve akıcı bir gezinme deneyimi, kullanıcıların siteyi daha verimli bir şekilde kullanmalarını ve etkileşim oranlarını artırmalarını sağlar.

Sonuç olarak, web sitesi tasarımı, kullanıcı deneyimi ve performans üzerinde büyük bir etkiye sahiptir. İyi tasarlanmış bir site, kullanıcıların ihtiyaçlarını hızlı ve etkili bir şekilde karşılayabilirken, kötü tasarlanmış bir site performans sorunlarına yol açabilir ve kullanıcıları kaybetme riskini artırabilir. Bu bağlamda, web sitesi tasarımında performansı artırıcı tekniklerin ve optimize edilmiş görsel ve işlevsel bileşenlerin kullanılması büyük önem taşır. (Peña-Ortiz, 2013)

2.1.1.1. UI UX'in Performansa Etkisi

UI (Kullanıcı Arayüzü) ve UX (Kullanıcı Deneyimi), bir web sitesinin performansı üzerinde önemli bir etkiye sahiptir. Kullanıcıların siteyle nasıl etkileşime geçtiği ve bu etkileşim sırasında yaşadıkları deneyim, sitenin genel başarısını belirleyen kritik faktörlerdir. Kullanıcıların siteyi kolayca kullanabilmesi ve aradıkları bilgilere hızlıca ulaşabilmesi, iyi bir UI/UX tasarımının sonucudur.

UI tasarımı, sitenin görsel unsurlarının düzenlenmesi ve kullanıcıların siteyle etkileşimini kolaylaştırmak için kullanılan grafiksel öğeleri içerir. İyi bir UI tasarımı, kullanıcıların sitede gezinmesini ve etkileşimde bulunmasını kolaylaştırır. Örneğin, butonların ve linklerin kolayca tıklanabilir olması, kullanıcıların istedikleri işlemi

hızlıca gerçekleştirmelerine olanak tanır. Ayrıca, renklerin ve tipografinin doğru kullanımı, kullanıcıların siteyi daha rahat ve keyifli bir şekilde kullanmasını sağlar.

UX tasarımı ise, kullanıcıların siteyi kullanırken yaşadıkları genel deneyimi kapsar. Kullanıcıların siteyi nasıl deneyimlediği, onların sitede ne kadar süre geçireceklerini ve siteyle ne kadar etkileşimde bulunacaklarını belirler. İyi bir UX tasarımı, kullanıcıların ihtiyaçlarını karşılayan ve onları memnun eden bir deneyim sunar. Örneğin, hızlı yüklenme süreleri ve kullanıcı dostu bir navigasyon sistemi, kullanıcıların siteyi daha verimli bir şekilde kullanmalarını sağlar. Ayrıca, kullanıcı geri bildirimlerinin dikkate alınması ve sürekli iyileştirmeler yapılması, UX tasarımının kalitesini artırır.

UI ve UX tasarımının performansa olan etkisi, kullanıcıların siteyle olan etkileşimlerini ve memnuniyetlerini doğrudan etkiler. "Kullanıcıların siteyi kolayca kullanabilmesi ve aradıkları bilgilere hızlıca ulaşabilmesi, iyi bir UI/UX tasarımının sonucudur" (Peña-Ortiz, 2013). Kullanıcılar, hızlı ve sorunsuz bir deneyim yaşadıkları sitelere daha fazla zaman harcar ve tekrar ziyaret etme olasılıkları artar. Bu da, sitenin trafiğini ve dönüşüm oranlarını artırır.

Sonuç olarak, UI ve UX tasarımı, bir web sitesinin performansı üzerinde kritik bir rol oynar. Kullanıcıların siteyi kolayca kullanabilmesi ve olumlu bir deneyim yaşaması, sitenin başarısını artırır. Bu bağlamda, iyi bir UI/UX tasarımı, kullanıcı memnuniyetini ve etkileşim oranlarını artırarak, web sitesinin performansını olumlu yönde etkiler.

2.2.1.2. Kullanılan Teknolojilerin Performansa Etkisi

Web sistemlerinde performans, kullanılan teknolojilerin verimliliği ve entegrasyon kabiliyeti ile doğrudan ilişkilidir. Bu bağlamda, hem sunucu tarafında hem de istemci tarafında kullanılan teknolojilerin seçimi, web uygulamalarının hızını, ölçeklenebilirliğini ve kullanıcı deneyimini büyük ölçüde etkiler.

Bir web uygulamasının performansını optimize etmek için, kullanılan yazılım ve donanım teknolojilerinin doğru bir şekilde entegre edilmesi gereklidir. Özellikle çok katmanlı ve çok kullanıcılı sistemlerde, katmanlı mimarilerin ve güncel

teknolojilerin kullanımı, performans iyileştirmeleri sağlar. "Bu çalışmada, web performansını artırmak için çeşitli yöntemler incelenmiş ve sunucu tarafında ASP.NET MVC Framework, istemci tarafında ise JavaScript ve JQuery gibi teknolojiler kullanılmıştır" (Hacıoğlu ve Güneş, 2019).

Sunucu tarafında kullanılan teknolojiler, web uygulamalarının hızlı ve güvenilir bir şekilde çalışmasını sağlar. ASP.NET MVC Framework gibi popüler frameworkler, web uygulamalarının modüler ve sürdürülebilir bir yapıda geliştirilmesine olanak tanır. Bu tür frameworkler, uygulama mantığının katmanlı bir mimari ile ayrılmasını ve her bir katmanın bağımsız olarak optimize edilmesini sağlar. Ayrıca, veri tabanı yönetim sistemleri (DBMS) ve önbellekleme mekanizmaları da sunucu performansını doğrudan etkiler. Örneğin, Redis gibi bellek tabanlı önbellekleme sistemleri, veritabanına olan yükü azaltarak veri erişim sürelerini hızlandırır ve uygulama performansını artırır.

İstemci tarafında ise, JavaScript ve JQuery gibi betik dilleri, kullanıcı etkileşimlerini hızlı ve verimli bir şekilde yönetmek için kullanılır. Bu teknolojiler, dinamik içeriklerin yüklenmesi ve kullanıcı eylemlerinin anında işlenmesi gibi işlemlerde önemli bir rol oynar. Özellikle büyük ve karmaşık web uygulamalarında, istemci tarafında kullanılan teknolojilerin performansı, kullanıcı deneyimini doğrudan etkiler. Ayrıca, istemci tarafında yapılan optimizasyonlar, sayfa yüklenme sürelerini azaltarak kullanıcı memnuniyetini artırır. "Web performansının iyileştirilmesi ve analiz edilmesi, kullanılan yazılım ve veri tabanı teknikleri ile gerçekleştirilmiş ve yüksek performanslı bir web sitesi elde edilmiştir" (Hacıoğlu ve Güneş, 2019).

Sonuç olarak, web sistemlerinde kullanılan teknolojilerin performansa etkisi büyüktür. Doğru teknolojilerin seçimi ve bu teknolojilerin etkili bir şekilde entegre edilmesi, web uygulamalarının hızını, güvenilirliğini ve ölçeklenebilirliğini artırır. Bu bağlamda, hem sunucu hem de istemci tarafında güncel ve verimli teknolojilerin kullanılması, web uygulamalarının başarısını doğrudan etkiler. Kullanılan teknolojilerin performans üzerindeki etkilerini sürekli olarak izlemek ve optimize etmek, sürdürülebilir ve yüksek performanslı web uygulamaları geliştirmek için kritik öneme sahiptir.

2.2.1.2. Render Performansı'nın Web Sitesi Performansına Etkisi

Render performansı, bir web sayfasının tarayıcıda ne kadar hızlı görüntülendiğini ifade eder. HTML, CSS ve JavaScript dosyalarının yüklenip işlenmesi ile başlar ve sayfanın görsel elemanlarının ekranda oluşturulmasını sağlar. İyi bir render performansı, hızlı ve sorunsuz bir kullanıcı deneyimi sunar. Karmaşık HTML yapıları, aşırı CSS ve yoğun JavaScript kullanımı render süresini uzatabilir. Görsellerin optimize edilmesi ve lazy loading tekniği ile yüklenmesi de performansı artırır. DOM manipülasyonlarının minimize edilmesi önemlidir çünkü gereksiz işlemler tarayıcının sayfayı yeniden çizmesine neden olur. Sonuç olarak, render performansının iyileştirilmesi, kullanıcı memnuniyetini artırır ve web sitesinin genel performansını olumlu yönde etkiler. Bu bağlamda, render performansı, kullanıcı deneyimi ve SEO açısından kritik bir faktördür.

Web sitesi tasarım özelliklerinin performans üzerindeki etkilerini anlamak için yapılan çalışmalarda, özellikle nörogörüntüleme tekniklerinden faydalanılmaktadır. Hitit Üniversitesi'nde gerçekleştirilen bir araştırmada, göz izleme yöntemi kullanılarak öğrencilerin web sitesi üzerindeki dikkat ve odaklanma süreleri ölçülmüştür. Çalışmanın bulguları, web sitesi tasarımının ilk görüntülenme, ortalama odaklanma ve tekrar görüntülenme gibi metriklerde performansı doğrudan etkilediğini göstermiştir (Kılıç ve Çakaroz, 2021). Bu tür analizler, web sitesi tasarım özelliklerinin optimize edilerek kullanıcı deneyiminin iyileştirilmesine ve dolayısıyla sitenin performansının artırılmasına yardımcı olmaktadır.

Sonuç olarak, render performansı ve web sitesi tasarımı, kullanıcı deneyimi ve genel site performansı üzerinde önemli bir etkiye sahiptir. İyi optimize edilmiş bir render süreci ve kullanıcı dostu bir tasarım, kullanıcı memnuniyetini ve etkileşim oranlarını artırırken, web sitesinin genel performansını da iyileştirir. Bu bağlamda, modern web geliştirme süreçlerinde render performansının optimize edilmesi ve kullanıcı odaklı tasarım prensiplerinin benimsenmesi, sürdürülebilir ve yüksek performanslı web siteleri oluşturmak için kritik öneme sahiptir.

2.1.3. Browser Uyumluluğunun Performansa Etkisi

Web sitelerinin farklı tarayıcılarda tutarlı ve sorunsuz bir şekilde çalışabilmesi, kullanıcı deneyimi ve genel performans açısından kritik öneme sahiptir. Browser uyumluluğu, bir web sitesinin tüm tarayıcılarda beklenilen şekilde çalışmasını ifade eder. Bu uyumluluğu sağlamak için belirli teknikler ve testler uygulanmalıdır.

Browser uyumluluğunu sağlamak için kullanılan teknikler aşağıda açıklanmaktadır.

CSS Teknikleri: CSS kullanarak tarayıcı uyumluluğunu artırmak için bazı temel teknikler mevcuttur. Örneğin, tarayıcıların desteklemediği CSS özellikleri için fallback çözümleri kullanılabilir. CSS reset ve normalize.css gibi araçlar, farklı tarayıcıların varsayılan stil farklılıklarını giderir ve tutarlı bir başlangıç noktası sağlar. Ayrıca, tarayıcı özelinde CSS yazmak için vendor prefixler kullanılabilir (BrowserStack, 2023) .

Progressive Enhancement ve Graceful Degradation: Progressive enhancement yaklaşımı, tüm tarayıcılarda çalışabilecek temel işlevsellikleri öncelemekte ve daha modern tarayıcılara ek özellikler eklemekte kullanılır. Bu, tarayıcı farklarının kullanıcı deneyimini olumsuz etkilemesini engeller. Graceful degradation ise, daha eski tarayıcılarda çalışmayan özellikler için alternatif çözümler sunar. Örneğin, modern tarayıcılarda CSS grid kullanırken, eski tarayıcılarda float tabanlı layout kullanılabilir (OpenReplay Blog, 2023).

Modern Web Standartları ve Teknolojileri Kullanımı: HTML5, CSS3 ve JavaScript'in modern özellikleri, tarayıcı uyumluluğunu artırmak için standartlaştırılmış çözümler sunar. Bu standartların kullanılması, web sitelerinin performansını artırır ve güvenliği iyileştirir. Aynı zamanda, responsive design teknikleri kullanılarak sitenin farklı cihazlarda uyumlu çalışması sağlanabilir (BrowserStack, 2023).

Browser uyumluluğunu sağlamak için çeşitli test yöntemleri kullanılmalıdır:

Manuel Testler**:** Web sitelerinin farklı tarayıcı ve cihazlarda manuel olarak test edilmesi, spesifik sorunların tespit edilmesini sağlar. Bu süreçte, form gönderimleri,

buton etkileşimleri, navigasyon menüleri gibi işlevlerin her tarayıcıda doğru çalıştığından emin olunur. Bu testler, özellikle küçük detayların fark edilmesini sağlar (FreeCodeCamp, 2023).

Otomatik Test Araçları**:** BrowserStack, Sauce Labs ve LambdaTest gibi araçlar, tarayıcı uyumluluğu testlerini otomatikleştirir. Bu araçlar, web sitelerinin farklı tarayıcı ve cihaz kombinasyonlarında nasıl göründüğünü ve çalıştığını simüle eder. Otomatik testler, geniş çaplı tarayıcı uyumluluğu kontrollerini hızlı ve etkili bir şekilde gerçekleştirir (FreeCodeCamp, 2023).

Sanallaştırma ve Tarayıcı Emülatörleri**:** Sanal makineler ve tarayıcı emülatörleri kullanılarak, web siteleri farklı işletim sistemlerinde ve tarayıcılarda test edilebilir. Bu araçlar, özellikle fiziksel cihazların yetersiz kaldığı durumlarda maliyet ve zaman açısından avantaj sağlar (OpenReplay Blog, 2023).

Sonuç olarak, tarayıcı uyumluluğu, web sitelerinin tüm kullanıcılar için tutarlı ve sorunsuz bir deneyim sunması açısından büyük önem taşır. CSS teknikleri, progressive enhancement ve graceful degradation stratejileri ile birlikte, modern web standartlarının kullanımı, tarayıcı uyumluluğunu artırır. Manuel ve otomatik testler, uyumluluk sorunlarının tespit edilmesini ve giderilmesini sağlar. Bu yaklaşımlar, kullanıcı memnuniyetini artırırken, sitenin genel performansını ve erişilebilirliğini de iyileştirir. Web geliştiricilerin, tarayıcı uyumluluğunu sürekli olarak izlemeleri ve optimize etmeleri, başarılı ve kullanıcı dostu web siteleri oluşturmak için kritik öneme sahiptir.

2.2. Performansın Web Sitesi İçin Önemi

Web sitesi performansı, kullanıcı deneyimini doğrudan etkileyen kritik bir faktördür. İnsan-bilgisayar etkileşimi üzerine yapılan çalışmalar, hızlı ve kesintisiz bir etkileşimin kullanıcı memnuniyetini artırdığını göstermektedir. Miller (1968), insan-bilgisayar etkileşimini iki insan arasındaki konuşmaya benzetmiş ve eğer diğer taraf birkaç saniye içinde yanıt vermezse, konuşmanın akışının bozulacağını belirtmiştir. Benzer şekilde, web sitelerinin yanıt süresinin uzun olması, kullanıcıların dikkatinin dağılmasına ve deneyimlerinin olumsuz etkilenmesine neden olur (Miller, 1968).

Lightner, Bose ve Salvendy (1996) tarafından yapılan bir ankette, internet kullanıcılarının en çok şikayet ettiği konulardan biri "veri erişim hızı" olmuştur. Bu, kullanıcıların hızlı bilgi erişimi beklentisi içinde olduğunu ve yavaş yüklenen sitelerin kullanıcı deneyimini olumsuz etkilediğini göstermektedir. Ayrıca, belirli bilgileri aramanın zorluğu da kullanıcılar için önemli bir sorun olarak belirtilmiştir (Lightner ve diğerleri, 1996).

Sears, Jacko ve Borella (1997), web sitesi yükleme gecikmelerinin kullanıcı algısı üzerindeki etkilerini incelemişlerdir. Kısa gecikmeler (575 ms) kullanıcılar tarafından olumlu karşılanırken, uzun gecikmeler (6750 ms) olumsuz tepkilere yol açmıştır. Bu durum, özellikle multimedya içeriği barındıran sitelerde belirgin hale gelmiştir. Kullanıcılar, multimedya getirmenin yarattığı gecikmeleri hoş karşılamamakta ve bu gecikmeler kullanıcı memnuniyetini azaltmaktadır (Sears ve diğerleri, 1997).

Ramsay, Barbesi ve Preece (1998), web sayfalarının yükleme sürelerinin kullanıcı deneyimi üzerindeki etkilerini araştırmış ve en hızlı yüklenen sayfaların kullanıcılar tarafından en ilgi çekici olarak değerlendirildiğini bulmuşlardır. Yavaş yüklenen sayfalar, kullanıcılar tarafından daha az taranabilir olarak değerlendirilmiş ve bu durum kullanıcı motivasyonunu olumsuz etkilemiştir (Ramsay ve diğerleri, 1998).

Nielsen (1999), web kullanılabilirliği konusunda yaptığı çalışmalarda, yavaş indirme sürelerinin kullanıcı güvenini azalttığını ve trafik kaybına neden olduğunu belirtmiştir. Yavaş yükleme süreleri, kullanıcıların web sitesini terk etme olasılığını artırmakta ve bu da site trafiğini olumsuz etkilemektedir (Nielsen, 1999a; Nielsen, 1999b). Gehrke ve Turban (1999) ise, sayfa yükleme hızının e-ticaret siteleri için en önemli özellik olarak değerlendirildiğini belirtmişlerdir. Kullanıcılar, hızlı yüklenen siteleri tercih etmekte ve bu siteler daha yüksek kullanıcı memnuniyeti sağlamaktadır (Gehrke ve Turban, 1999).

Hoxmeier ve DiCesare (2000), tarayıcı tabanlı yazılım uygulamalarının yanıt sürelerinin kullanıcı memnuniyeti üzerindeki etkilerini incelemiş ve yanıt süreleri arttıkça kullanıcı memnuniyetinin azaldığını bulmuşlardır. Bu durum, özellikle çevrimiçi işlemler yapan kullanıcılar için geçerlidir (Hoxmeier ve DiCesare, 2000).

Google'ın yaptığı bir deney, arama sonuçları sayfalarını yavaşlatmanın, kullanıcıların arama yapma sıklığını azalttığını göstermiştir. Yarım saniyelik bir gecikme bile, kullanıcıların arama yapma sıklığını %0,2 ila %0,6 arasında azaltmaktadır. Bu durum, kullanıcıların ekstra gecikmeye ne kadar uzun süre maruz kalırlarsa, o kadar az arama yaptıklarını göstermektedir (Brutlag, 2009).

Sonuç olarak, web sitesi performansı kullanıcı memnuniyeti, trafik ve dönüşüm oranları üzerinde doğrudan bir etkiye sahiptir. Yavaş yükleme süreleri, kullanıcıların siteyi terk etme olasılığını artırır ve bu da site trafiğini ve kullanıcı bağlılığını olumsuz etkiler. Web sitelerinin hızlı ve etkili bir şekilde çalışmasını sağlamak, kullanıcı memnuniyetini artırmanın ve site performansını optimize etmenin temel yollarından biridir.

3. PERFORMANS ODAKLI WEB SİTESİ NASIL GELİŞTİRİLİR

Performans odaklı web sitesi geliştirmek için neler yapılması gerektiği tezin bu bölümünde kapsamlı olarak ele alınmaktadır.

3.1. İyi bir Planlama ve Mimarinin Önemi

Web sitesi performansını artırmak için iyi bir planlama ve mimari yapılandırma kritik öneme sahiptir. Başarılı bir performans odaklı web sitesi geliştirmek, sadece hızlı yükleme sürelerinden daha fazlasını gerektirir; aynı zamanda kullanıcı deneyimini optimize etmek için kapsamlı bir ön hazırlık yapılmalıdır. Addy Osmani (2019), performans optimizasyonunun bir mühendislik disiplini olduğunu ve etkili bir strateji ile başladığını vurgular. İyi bir planlama süreci, projenin kapsamını belirlemek, kullanılacak teknolojileri seçmek ve olası performans darboğazlarını öngörmek gibi adımları içerir. Ayrıca, site mimarisi, veri akışını ve kullanıcı etkileşimlerini en verimli şekilde yönlendirecek şekilde tasarlanmalıdır. Bu bağlamda, modüler ve ölçeklenebilir bir mimari benimsemek, gelecekteki güncellemelerin ve bakım işlemlerinin daha kolay yönetilmesini sağlar. Osmani (2019), özellikle kaynakların yüklenme sürelerini ve kullanıcıların etkileşim süresini optimize eden tekniklerin, modern web uygulamalarında performansı nasıl artırabileceğine dair önemli bilgiler sunar. Bu yüzden, performans odaklı bir web sitesi geliştirmek, sadece teknik yeterlilik gerektirmez; aynı zamanda stratejik bir yaklaşım ve iyi bir mimari planlama gerektirir.

3.1.1. Proje Planlaması

Proje planlaması, bir projenin başarısını garanti altına almak için atılan ilk ve en kritik adımlardan biridir. Bu süreç, projenin kapsamını belirlemek, hedefleri tanımlamak, gerekli kaynakları tahsis etmek ve zaman çizelgesini oluşturmak gibi temel unsurları içerir. Proje planlamasının kökeni, büyük ölçekli mühendislik projeleri ve askeri operasyonlarda daha sistematik ve organize bir yaklaşımın gerekliliği ile ortaya çıkmıştır. 20. yüzyılın başlarında, Henry Gantt tarafından geliştirilen Gantt şeması gibi araçlar, proje yönetiminde devrim yaratmış ve karmaşık projelerin daha etkin bir şekilde yönetilmesini sağlamıştır. Bu dönemden itibaren, proje planlaması sadece büyük ölçekli projeler için değil, aynı zamanda yazılım geliştirme gibi daha

küçük ölçekli ve dinamik projeler için de vazgeçilmez bir uygulama haline gelmiştir. Günümüzde, proje planlaması, Agile ve Scrum gibi metodolojilerle daha esnek ve uyarlanabilir bir yapıya kavuşmuştur. Bu, projelerin hızlı değişen gereksinimlere ve piyasa koşullarına daha etkin bir şekilde yanıt verebilmesini sağlamaktadır. Başarılı bir proje planlaması, tüm paydaşların beklentilerini karşılayan ve projeyi zamanında ve bütçe dahilinde tamamlayan bir yol haritası sunar.

3.1.2. Uygulama Mimarisi

Uygulama mimarisi, yazılım geliştirme sürecinde kritik bir rol oynar ve performans odaklı web sitelerinin başarısında temel bir faktördür. Bu mimari, yazılımın bileşenlerinin nasıl yapılandırılacağını ve birbirleriyle nasıl etkileşime gireceğini belirler. Çevik yazılım geliştirme metodolojileri, hızlı ve etkili bir geliştirme süreci sağlarken, yazılım mimarisi bu sürecin verimli bir şekilde işlemesini destekler. Özellikle, mikroservis mimarisi gibi modern yaklaşımlar, bağımsız hizmetlerin ölçeklenebilirliğini ve yönetilebilirliğini artırır (Çetin, 2018). Bu, her bir bileşenin bağımsız olarak geliştirilip dağıtılabilmesini ve böylece sistemin genel performansının iyileştirilmesini sağlar. Ayrıca, uygulama mimarisi, yazılımın güvenilirliğini ve bakımını da kolaylaştırır. İyi tasarlanmış bir mimari, potansiyel performans sorunlarını öngörmeyi ve çözmeyi mümkün kılar, bu da kullanıcı deneyimini optimize eder. Çevik metodolojilerle entegrasyon, uygulama mimarisinin esneklik ve uyarlanabilirlik gereksinimlerini karşılamasını sağlar, böylece değişen gereksinimlere hızlı ve etkili bir şekilde yanıt verilebilir. Sonuç olarak, uygulama mimarisi, performans odaklı web geliştirme sürecinde vazgeçilmez bir unsurdur ve doğru bir şekilde tasarlandığında, projenin başarısını garanti altına alır.

3.1.3. Performans Hedeflerinin Belirlenmesi

Performans hedeflerinin belirlenmesi, başarılı web geliştirme projelerinin temel taşlarından biridir. Performans hedefleri, projenin başlangıcından itibaren net ve ölçülebilir amaçlar belirleyerek, hem geliştirme ekibinin hem de projenin genel başarısını artırır. Bu hedeflerin belirlenmesi, genellikle SMART kriterleri (Specific, Measurable, Achievable, Relevant, Time-bound) kullanılarak yapılır. SMART

kriterleri, hedeflerin net, ölçülebilir, ulaşılabilir, ilgili ve zaman sınırlı olmasını sağlar (Webflow, 2023; Simplilearn, 2024).

Öncelikle, hedeflerin spesifik olması gerekir; bu, hedefin ne olduğu ve nasıl başarılacağı konusunda netlik sağlar. Örneğin, "site hızını artırmak" yerine "sayfa yüklenme süresini 3 saniyeden 1.5 saniyeye düşürmek" gibi net bir hedef belirlemek daha etkili olacaktır. İkinci olarak, hedeflerin ölçülebilir olması gerekir; bu, ilerlemenin takip edilebilmesi ve başarının değerlendirilebilmesi için gereklidir. Üçüncü olarak, hedeflerin ulaşılabilir olması önemlidir; bu, ekibin kaynakları ve yetenekleri göz önünde bulundurularak belirlenir. Çok zor veya imkansız hedefler, ekipte motivasyon kaybına yol açabilir.

Hedeflerin projeye ve organizasyonun geniş stratejik hedeflerine uygun olması da gereklidir. Son olarak, hedeflerin zaman sınırlı olması, belirli bir süre içinde tamamlanması gerektiğini ifade eder ve bu, projenin sürekli ilerlemesini sağlar. Performans hedefleri, düzenli olarak gözden geçirilmeli ve gerektiğinde güncellenmelidir. Bu süreç, hedeflere ulaşmak için stratejilerin ve taktiklerin sürekli olarak değerlendirilmesini ve ayarlanmasını gerektirir (Webflow, 2023; Simplilearn, 2024).

3.2. Optimizasyon Teknikleri

Optimizasyon teknikleri, web sitelerinin performansını artırmak için uygulanan strateji ve yöntemler bütünüdür. Bu teknikler, web sitelerinin daha hızlı yüklenmesini, daha iyi kullanıcı deneyimi sunmasını ve arama motorlarında daha üst sıralarda yer almasını sağlar. Optimizasyon tekniklerinin temel amacı, kullanıcıların siteye erişim hızını artırmak ve site içinde daha uzun süre kalmalarını sağlamaktır. Bu süreçte, hem site içi (on-page) hem de site dışı (off-page) optimizasyon teknikleri kullanılır.

Site içi optimizasyon, web sitesinin yapısal ve içeriksel unsurlarını optimize etmeyi içerir. Bu, HTML ve CSS kodlarının düzenlenmesi, medya dosyalarının sıkıştırılması, uygun başlık ve meta etiketlerinin kullanılması gibi adımları kapsar. Ayrıca, kullanıcı deneyimini iyileştirmek için mobil uyumluluk, hızlı yükleme süreleri

ve kullanıcı dostu navigasyon gibi faktörler de göz önünde bulundurulur (Aydemir, 2011; Groppone ve Couzin, 2011).

Site dışı optimizasyon ise, web sitesine dışarıdan gelen bağlantılar ve sosyal medya etkileşimleri gibi faktörleri içerir. Yüksek kaliteli geri bağlantılar (backlinks) elde etmek, sosyal medya platformlarında aktif olmak ve marka bilinirliğini artırmak bu tekniklerin başında gelir (Weblozi, 2013).

Optimizasyon tekniklerinin aşırı veya yanlış kullanımı ise web sitelerine zarar verebilir. Aşırı SEO teknikleri, Google'ın sandbox filtresine takılmaya, Google bomb cezasına veya kopya içerik cezasına neden olabilir. Bu nedenle, SEO stratejileri dikkatli ve dengeli bir şekilde uygulanmalıdır (Seo Hocası, 2015; Onedio, 2015).

3.2.1. Kod Optimizasyonu

Kod optimizasyonu, yazılım geliştirme sürecinde, kodun performansını, verimliliğini ve genel kalitesini artırmak amacıyla yapılan iyileştirmelerdir. Bu süreç, yazılımın daha hızlı çalışmasını, daha az kaynak kullanmasını ve daha az hata içermesini sağlamayı hedefler. Kod optimizasyonu, hem uygulama performansını artırmak hem de kullanıcı deneyimini iyileştirmek için kritik öneme sahiptir.

Addy Osmani, kod optimizasyonu konusunda önemli yaklaşımlar sunar. Kod optimizasyonu, yazılımın performansını artırmak için kodun yapısını ve mantığını iyileştirme sürecidir. Osmani, bu sürecin üç temel aşamadan oluştuğunu vurgular: ölçme, analiz etme ve optimize etme. İlk olarak, performans sorunlarını belirlemek için kodun kapsamlı bir şekilde ölçülmesi gerekir. Bu, performans izleme araçları kullanarak, yavaş yüklenen veya fazla kaynak tüketen bileşenlerin tespit edilmesini içerir. Analiz aşamasında, bu sorunların köken nedenleri incelenir ve optimize edilebilecek alanlar belirlenir. Son olarak, optimize etme aşamasında, kodun daha verimli çalışması için gerekli değişiklikler yapılır. Bu değişiklikler arasında kodun yeniden düzenlenmesi, gereksiz kodların kaldırılması, verimli algoritmaların kullanılması ve bellek yönetiminin iyileştirilmesi bulunur (Osmani, 2012).

3.2.1.1. Proje Planlaması

Başarılı bir yazılım geliştirme projesi, detaylı ve iyi yapılandırılmış bir planlama süreci ile başlar. Proje planlaması, projenin kapsamını belirlemek, hedefleri tanımlamak, kaynakları tahsis etmek ve bir zaman çizelgesi oluşturmak gibi kritik adımları içerir. Planlama süreci, projenin tüm paydaşlarının beklentilerini ve gereksinimlerini anlamayı ve bu gereksinimleri karşılayacak stratejik bir plan oluşturmayı gerektirir. Proje planlaması, ayrıca potansiyel risklerin belirlenmesi ve bu risklere karşı önleyici tedbirlerin alınmasını da kapsar. İyi bir proje planı, projenin her aşamasında ilerlemenin izlenmesini ve gerektiğinde planın güncellenmesini sağlar. Bu sayede, proje ekibi hedeflere zamanında ve bütçe dahilinde ulaşabilir.

3.2.1.2. Ölçeklenebilir ve Modüler Kod Yapısı

Ölçeklenebilir ve modüler bir kod yapısı, yazılımın bakımını ve genişletilmesini kolaylaştırır. Addy Osmani, modüler kod yapısının, kodun daha küçük, bağımsız ve tekrar kullanılabilir bileşenlere bölünmesi gerektiğini savunur. Bu, kodun okunabilirliğini artırır ve hataların izlenmesini kolaylaştırır. Ayrıca, modüler kod, farklı geliştiricilerin aynı anda farklı modüller üzerinde çalışabilmesine olanak tanır, bu da geliştirme sürecini hızlandırır. Ölçeklenebilirlik ise, yazılımın artan kullanıcı sayısı veya veri yükü karşısında performansını koruyabilmesini sağlar. Osmani, ölçeklenebilir bir mimari oluşturmanın, öncelikle yazılımın ihtiyaçlarını ve büyüme potansiyelini doğru bir şekilde analiz etmeyi gerektirdiğini belirtir. Bu analiz, veri tabanı yapılandırmasından sunucu kaynaklarına kadar birçok farklı bileşenin optimize edilmesini içerir (Osmani, 2013).

3.2.2. Görsel Optimizasyon

Görsel optimizasyon, web sayfalarının performansını artırmak için kritik bir rol oynar. Büyük boyutlu görseller, sayfa yükleme sürelerini uzatabilir ve kullanıcı deneyimini olumsuz etkileyebilir. Bu nedenle, görsellerin doğru şekilde optimize edilmesi, web performansını iyileştirmek ve kullanıcı memnuniyetini artırmak için

önemlidir. Addy Osmani, görsel optimizasyonun temel adımlarından biri olarak görsel sıkıştırma ve uygun formatın seçilmesini vurgular (Osmani, 2018).

3.2.2.1. Görsel Sıkıştırma Teknikleri

Görsel sıkıştırma, görsellerin boyutlarını azaltarak web sayfalarının daha hızlı yüklenmesini sağlar. İki ana sıkıştırma türü vardır: kayıplı ve kayıpsız sıkıştırma. Kayıplı sıkıştırma, görselin kalitesinde belirli bir kayıp yaşanmasına izin vererek dosya boyutunu önemli ölçüde azaltır. JPEG formatı, kayıplı sıkıştırmanın yaygın bir örneğidir. Kayıpsız sıkıştırma ise görselin kalitesini korurken dosya boyutunu azaltır ve genellikle PNG formatında kullanılır.

Addy Osmani, görsel sıkıştırma için çeşitli araçlar ve teknikler önermektedir. Bunlar arasında, görselleri yüklemeden önce sıkıştırmak için TinyPNG ve ImageOptim gibi araçlar bulunmaktadır. Ayrıca, web tarayıcıları tarafından sunulan sıkıştırma algoritmalarını kullanarak, görsellerin boyutlarını dinamik olarak optimize etmek de mümkündür (Osmani, 2018). Bu araçlar ve teknikler, görsel kalitesini en az düzeyde etkileyerek dosya boyutlarını küçültmeyi sağlar.

3.2.2.2. Uygun Formatın Seçimi (JPEG, PNG, WebP)

Görsellerin doğru formatta kullanılması, performans optimizasyonu için önemlidir. Her formatın kendine özgü avantajları ve kullanım alanları vardır:

JPEG: JPEG, kayıplı sıkıştırma kullandığı için fotoğraf gibi yüksek detay gerektiren görüntüler için uygundur. Dosya boyutlarını önemli ölçüde küçültebilir, ancak sıkıştırma oranı arttıkça kalite kaybı yaşanabilir. JPEG formatı, renkli ve kompleks görseller için ideal bir seçimdir.

PNG: PNG, kayıpsız sıkıştırma kullanarak görsel kalitesini korur. Bu nedenle, grafikler, logolar ve şeffaflık gerektiren görseller için tercih edilir. PNG formatı, görsel kalitesinin korunması gereken durumlar için uygundur, ancak dosya boyutları JPEG'e göre daha büyük olabilir.

WebP: Google tarafından geliştirilen WebP, hem kayıplı hem de kayıpsız sıkıştırma sunar ve daha küçük dosya boyutları ile yüksek kaliteli görseller elde etmeyi sağlar. WebP, modern tarayıcılarda geniş destek bulmakta olup, performans optimizasyonu için güçlü bir araçtır. Addy Osmani, WebP formatının benimsenmesinin, web sayfalarının yüklenme sürelerini önemli ölçüde azaltabileceğini belirtir (Osmani, 2018).

Görsel optimizasyon teknikleri, web sayfalarının performansını artırmak ve kullanıcı deneyimini iyileştirmek için hayati öneme sahiptir. Doğru sıkıştırma tekniklerinin kullanılması ve uygun formatların seçilmesi, bu süreçte kritik rol oynar. Görsellerin boyutlarını optimize etmek ve web performansını artırmak için bu teknikler etkin bir şekilde kullanılmalıdır.

3.2.3. Veri Yükleme Optimizasyonu

Veri yükleme optimizasyonu, web sayfalarının daha hızlı yüklenmesini ve daha iyi bir kullanıcı deneyimi sunmasını sağlamak için çeşitli tekniklerin uygulanmasıdır. Bu süreçte, veri yüklemenin zamanlaması ve yöntemi büyük önem taşır. Addy Osmani'nin yaklaşımlarına göre, veri yükleme optimizasyonunda iki ana teknik öne çıkar: Lazy Loading ve Asenkron Veri Yükleme.

3.2.3.1. Lazy Loading

Lazy loading, sayfa yüklenirken gereksiz veri ve kaynakların yüklenmesini erteleyerek performansı artıran bir tekniktir. Bu teknik, yalnızca kullanıcı tarafından görüntülenen içeriklerin yüklenmesini sağlar, böylece ilk yükleme süresi ve veri tüketimi azalır. Addy Osmani, lazy loading'in özellikle görseller ve videolar gibi büyük medya dosyalarının yüklenmesinde etkili olduğunu vurgular (Osmani, 2018). Lazy loading, kullanıcı sayfayı aşağı kaydırdıkça veya belirli bir etkileşime girdiğinde ilgili içeriğin dinamik olarak yüklenmesini sağlar. Bu sayede, başlangıçta yalnızca gerekli olan veriler yüklenir ve sayfa performansı önemli ölçüde iyileştirilir. Lazy loading, Intersection Observer API gibi modern web API'leri kullanılarak kolayca uygulanabilir ve geliştirme sürecine entegre edilebilir.

3.2.3.2. Asenkron Veri Yükleme

Asenkron veri yükleme, web sayfalarının performansını artırmak için kritik bir tekniktir. Bu teknik, veri yükleme işlemlerinin arka planda gerçekleştirilmesini sağlar, böylece sayfanın ana işlemleri kesintiye uğramaz ve kullanıcı deneyimi kesintisiz devam eder. Addy Osmani, asenkron veri yüklemenin özellikle JavaScript ve diğer dinamik içeriklerin yönetiminde önemli olduğunu belirtir (Osmani, 2018). Asenkron veri yükleme, AJAX ve Fetch API gibi yöntemler kullanılarak gerçekleştirilebilir. Bu yöntemler, sayfa yeniden yüklenmeden veri güncellemelerinin yapılmasını sağlar ve böylece kullanıcıların hızlı ve sorunsuz bir deneyim yaşamasını sağlar. Ayrıca, asenkron veri yükleme, ağ gecikmelerini ve sunucu yanıt sürelerini minimize ederek genel sayfa yükleme süresini azaltır.

Lazy loading ve asenkron veri yükleme teknikleri, web geliştiricilerine web sayfalarının performansını optimize etme konusunda güçlü araçlar sunar. Bu tekniklerin doğru ve etkili bir şekilde uygulanması, kullanıcı memnuniyetini artırır ve web sitelerinin rekabet gücünü yükseltir.

3.3. İçerik Dağıtım Ağları (CDN) Kullanımı

İçerik Dağıtım Ağları (CDN), web performansını artırmak ve kullanıcı deneyimini iyileştirmek için kritik bir teknoloji olarak kabul edilir. CDN'ler, içerikleri kullanıcıya daha yakın sunuculardan dağıtarak sayfa yükleme sürelerini ve ağ gecikmelerini azaltır. Bu bölümde, CDN'in ne olduğunu, faydalarını ve nasıl seçilip entegre edileceğini inceleyeceğiz.

3.3.1. CDN Nedir?

CDN (Content Delivery Network), internet içeriğini kullanıcılara daha hızlı ve verimli bir şekilde sunmak için coğrafi olarak dağıtılmış bir sunucu ağıdır. Bu ağ, web sayfaları, videolar, resimler, JavaScript dosyaları ve CSS dosyaları gibi çeşitli içerikleri içerir. CDN'ler, içeriği kullanıcıya en yakın sunucudan sağlayarak, sayfa yükleme sürelerini azaltır ve kullanıcı deneyimini iyileştirir. CDN kullanımı, yüksek

trafik alan web sitelerinin performansını artırmak için yaygın olarak tercih edilir (Buyya, Yeo, ve Venugopal, 2008).

3.3.2. CDN'in Faydaları

CDN kullanmanın birçok faydası vardır:

Hızlı Yükleme Süreleri: CDN'ler, içeriği kullanıcıya en yakın sunucudan sağlayarak sayfa yükleme sürelerini önemli ölçüde azaltır. Bu, özellikle büyük dosyaların ve yüksek trafikli sitelerin daha hızlı yüklenmesini sağlar (Li, S., 2018).

Düşük Gecikme Süresi: CDN'ler, ağ gecikmelerini minimize eder ve böylece kullanıcıların içeriğe daha hızlı erişmesini sağlar.

Yük Dengeleme: CDN'ler, trafiği birden fazla sunucuya dağıtarak sunucu yükünü dengeler ve bu sayede sunucu çökme riskini azaltır.

Güvenlik İyileştirmeleri: CDN'ler, DDoS saldırılarına karşı koruma sağlar ve güvenlik sertifikaları ile veri güvenliğini artırır.

Daha İyi Ölçeklenebilirlik: CDN'ler, anlık trafik artışlarına karşı daha iyi ölçeklenebilir ve böylece yüksek talep dönemlerinde bile performansı korur (Pathan, Buyya, ve Vakali, 2008).

3.3.3. CDN Seçimi ve Entegrasyonu

CDN seçimi ve entegrasyonu, web performansını optimize etmek için dikkatlice planlanmalıdır. CDN seçerken dikkate alınması gereken bazı önemli faktörler şunlardır:

Kapsama Alanı: CDN'in dünya genelindeki veri merkezi ağı, içeriklerin farklı coğrafi bölgelerde hızlı ve etkili bir şekilde sunulmasını sağlar. Global bir kapsama alanına sahip CDN'ler, uluslararası kullanıcılar için idealdir.

Performans ve Hız: Seçilen CDN'in sunduğu hız ve performans, kullanıcı deneyimini doğrudan etkiler. Hızlı ve güvenilir bir CDN seçmek, web sayfalarının daha hızlı yüklenmesini sağlar.

Güvenlik Özellikleri: CDN'in sunduğu güvenlik özellikleri, DDoS saldırıları gibi tehditlere karşı koruma sağlar. Güçlü güvenlik protokolleri olan CDN'ler tercih edilmelidir.

Fiyatlandırma: CDN hizmetlerinin maliyeti, uzun vadeli kullanımı etkileyebilir. Fiyat-performans oranı yüksek olan CDN'ler tercih edilmelidir.

CDN entegrasyonu, genellikle basit bir süreçtir. Çoğu CDN sağlayıcısı, web siteleri ve uygulamalarla uyumlu kolay entegrasyon yöntemleri sunar. Entegrasyon süreci, CDN hizmet sağlayıcısının yönergeleri takip edilerek tamamlanabilir. Web sitenizin DNS ayarlarını CDN'in sağladığı sunuculara yönlendirmek ve gerekli kod değişikliklerini yapmak, entegrasyonun temel adımlarını oluşturur.

CDN kullanımı, web performansını optimize etmek ve kullanıcı deneyimini iyileştirmek için önemli bir adımdır. Doğru CDN seçimi ve etkin entegrasyonu, web sitelerinin hızını ve güvenliğini artırarak rekabet avantajı sağlar.

3.4. Sunucu Performansı

Sunucu performansı, web sitelerinin hızlı ve kesintisiz çalışabilmesi için kritik bir öneme sahiptir. İyi optimize edilmiş bir sunucu, kullanıcı deneyimini iyileştirir ve siteyi daha güvenilir hale getirir. Bu bölümde, sunucu yanıt süresi optimizasyonu, cache yönetimi ve ölçeklenebilirlik ile load balancing konularını ele alacağız.

3.4.1. Sunucu Yanıt Süresi Optimizasyonu

Sunucu yanıt süresi optimizasyonu, web sayfalarının kullanıcıya daha hızlı yanıt vermesini sağlamak için gereklidir. Sunucu yanıt süresini optimize etmek, sunucunun gelen istekleri işleme hızını artırır ve genel performansı iyileştirir. Addy Osmani, bu konuda çeşitli stratejiler önermektedir. İlk olarak, veritabanı sorgularının optimizasyonu önemlidir. Veritabanı sorguları ne kadar hızlı işlenirse, sunucu yanıt süresi o kadar kısalır. Ayrıca, sunucunun yanıt süresini etkileyen diğer faktörler arasında, sunucu donanımının kapasitesi, ağ bağlantısı hızı ve yazılımın verimli kullanımı bulunur (Osmani, 2012).

3.4.2. Cache Yönetimi

Cache yönetimi, sunucu performansını artırmak ve sayfa yükleme sürelerini azaltmak için önemli bir tekniktir. Cache yönetimi, hem sunucu tarafında (server-side) hem de istemci tarafında (client-side) uygulanabilir.

3.4.2.1. Server Side Caching

Sunucu tarafı cache, sunucunun belirli veri veya içerikleri önbelleğe alarak hızlı erişim sağlamasını içerir. Addy Osmani, sunucu tarafı cache yönetiminin, özellikle sık erişilen verilerin hızlı bir şekilde sunulmasını sağladığını belirtir (Osmani, 2018). Sunucu tarafı cache teknikleri arasında, veritabanı sonuçlarının önbelleğe alınması, dosya sisteminde sık kullanılan dosyaların saklanması ve HTTP yanıtlarının önbelleğe alınması bulunur. Bu yöntemler, sunucunun yükünü azaltır ve kullanıcıya daha hızlı yanıt verilmesini sağlar.

3.4.2.2. Client Side Caching

İstemci tarafı cache, kullanıcının tarayıcısında belirli veri veya içeriklerin saklanmasını sağlar. Bu, özellikle statik içeriklerin hızlı bir şekilde yüklenmesini sağlar. Addy Osmani, istemci tarafı cache yönetiminin, kullanıcı deneyimini iyileştirmek için kritik olduğunu vurgular (Osmani, 2018). İstemci tarafı cache teknikleri arasında, HTTP önbellekleme başlıklarının kullanılması, tarayıcı önbelleğinin etkin kullanımı ve içeriklerin CDN (Content Delivery Network) aracılığıyla sunulması bulunur. Bu yöntemler, kullanıcıların aynı içeriğe tekrar erişim sağladıklarında sayfa yükleme sürelerini önemli ölçüde azaltır.

3.4.3. Ölçeklenebilirlik ve Load Balancing

Ölçeklenebilirlik, bir web sitesinin artan kullanıcı taleplerine yanıt verebilme kapasitesidir. Load balancing, sunucuya gelen isteklerin dengeli bir şekilde dağıtılmasını sağlar. Addy Osmani, ölçeklenebilirlik ve load balancing'in, yüksek trafikli web sitelerinde performansın korunması için önemli olduğunu belirtir

(Osmani, 2012). Load balancing teknikleri, sunucu gruplarının etkin yönetimini içerir ve bu, sunucu kaynaklarının verimli kullanılmasını sağlar. Ayrıca, otomatik ölçeklenebilirlik çözümleri, trafik arttığında ek sunucu kaynaklarının otomatik olarak devreye girmesini sağlar, bu da sunucu performansını korur ve kullanıcı deneyimini iyileştirir.

3.5. Front-End Performans İyileştirmeleri

Front-end performans iyileştirmeleri, web sayfalarının daha hızlı yüklenmesini ve daha iyi kullanıcı deneyimi sunmasını sağlar. Bu iyileştirmeler, JavaScript ve CSS optimizasyonu, modern web API'lerinin kullanımı ve responsive tasarım gibi çeşitli stratejileri içerir.

3.5.1. JavaScript ve CSS Optimizasyonu

JavaScript ve CSS, web sayfalarının performansını doğrudan etkileyen iki temel bileşendir. Bu bileşenlerin optimize edilmesi, sayfa yükleme sürelerini azaltarak kullanıcı deneyimini iyileştirir.

3.5.1.1. Critical Rendering Path Yönetimi

Critical Rendering Path (CRP), bir web sayfasının kullanıcıya görünmesi için gereken minimum HTML, CSS ve JavaScript dosyalarının yüklenmesi sürecidir. Bu sürecin optimize edilmesi, sayfanın daha hızlı yüklenmesini sağlar.

Örneğin; bir e-ticaret web sitesi düşünün. Bu site, kullanıcıların hızlı bir şekilde ürünleri görüp alışveriş yapmasını sağlamalıdır. CRP'yi optimize etmek için:

CSS ve JavaScript Minimize Edilir: Gerekli olmayan tüm beyaz boşluklar, yorumlar ve fazla kod kaldırılarak dosya boyutu küçültülür.

Minimize olmamış css yazım örneği;

```css
/* Minimize edilmemiş CSS örneği */
body {
    font-family: Arial, sans-serif;
    background-color: #ffffff;
    color: #333333;
    margin: 0;
    padding: 0;
}

header {
    background-color: #4CAF50;
    padding: 20px;
    text-align: center;
    font-size: 24px;
    color: white;
}

p {
    font-size: 16px;
    line-height: 1.5;
    margin: 20px;
}
```

Şekil 8. Minimize olmamış CSS yazım örneği

Kaynak: Yazar tarafından oluşturulmuştur.

Minimize edilmiş CSS yazım örneği

```css
body{font-family:Arial,sans-serif;background-color:#fff;color:#333;margin:0;padding
```

Şekil 9. Minimize edilmiş CSS yazım örneği

Kaynak: Yazar tarafından oluşturulmuştur.

Minimize edilmemiş CSS kodu, insan tarafından kolay okunabilir ve düzenlenebilir olması için boşluklar, yorumlar ve satır araları içerir. Minimize edilmiş CSS kodu ise tüm gereksiz boşlukları, yorumları ve satır aralarını kaldırarak dosya boyutunu küçültür ve böylece web sayfasının daha hızlı yüklenmesini sağlar. Bu, özellikle büyük CSS dosyaları için performansı önemli ölçüde artırabilir.

Minimize Etme Araçları;

CSS kodunu minimize etmek için çeşitli araçlar ve yöntemler kullanılabilir:

Online Araçlar: CSS Minifier gibi online araçlar, CSS kodunu hızlıca minimize edebilir.

Yapılandırma Araçları: Webpack veya Gulp gibi yapılandırma araçları, build sürecinde CSS dosyalarını otomatik olarak minimize edebilir.

Komut Satırı Araçları: clean-css-cli gibi komut satırı araçları, CSS dosyalarını minimize etmek için kullanılabilir.

CSS'nin Üst Kısma Yerleştirilmesi: CSS dosyaları <head> etiketinin içine yerleştirilir, böylece tarayıcı önce CSS'yi yükler ve sayfanın stilini hızlıca uygular.

JavaScript'in Alt Kısma Yerleştirilmesi: JavaScript dosyaları <body> etiketinin sonuna yerleştirilir, böylece HTML ve CSS yüklendikten sonra JavaScript yüklenir ve render süreci engellenmez.

Bu teknikler, e-ticaret sitesinin daha hızlı yüklenmesini sağlar ve kullanıcıların alışveriş deneyimini iyileştirir.

3.5.1.2. Asenkron ve Defer Script Yükleme

Asenkron ve defer script yükleme, JavaScript dosyalarının sayfa yüklemesini engellemeden yüklenmesini sağlar.

Örneğin; bir haber sitesi düşünün. Bu site, kullanıcıların hızlıca haber başlıklarını görmesini sağlamalıdır.

Asenkron Yükleme (async): <script src="script.js" async></script> kullanılarak JavaScript dosyası paralel olarak yüklenir ve tarayıcı diğer kaynakları yüklemeye devam eder.

Defer Yükleme (defer): <script src="script.js" defer></script> kullanılarak JavaScript dosyası DOM tamamen yüklendikten sonra çalıştırılır.

Bu yöntemler, haber sitesinin hızlı yüklenmesini ve kullanıcıların haber başlıklarına hızlıca erişmesini sağlar.

3.5.2. Modern Web API'lerinin Kullanımı

Modern web API'leri, web uygulamalarının performansını artırmak ve kullanıcı deneyimini iyileştirmek için çeşitli özellikler sunar.

3.5.2.1. Service Workers

Service Workers, web sayfalarının çevrimdışı kullanılabilirliğini ve performansını artıran bir web API'dir.

Örneğin; bir blog sitesi düşünün. Bu site, kullanıcıların çevrimdışı iken bile makaleleri okuyabilmesini sağlamalıdır.

Service Worker Kaydı:

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/service-
worker.js').then(function(registration) {
    console.log('ServiceWorker registration successful with
scope: ', registration.scope);
  }).catch(function(error) {
    console.log('ServiceWorker registration failed: ', error);
  });
}
```

Şekil 10. Service Worker yazım örneği

Kaynak: Yazar tarafından oluşturulmuştur.

Önbellekleme Stratejisi

```
self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open('v1').then(function(cache) {
      return cache.addAll([
        '/index.html',
        '/styles.css',
        '/script.js',
        '/image.jpg'
      ]);
    })
  );
});
```

Şekil 11. Önbellek yazım örneği

Kaynak: Yazar tarafından oluşturulmuştur.

Bu teknikler, blog sitesinin hızlı yüklenmesini ve makalelerin çevrimdışı erişilebilir olmasını sağlar.

3.5.2.2. WebAssembly

WebAssembly (Wasm), yüksek performanslı web uygulamaları oluşturmak için kullanılan bir teknolojidir.

Örneğin; bir oyun sitesi düşünün. Bu site, kullanıcıların yüksek performanslı oyunları tarayıcıda oynamasını sağlamalıdır.

WebAssembly Kullanımı:

```JS
fetch('game.wasm').then(response =>
  response.arrayBuffer()
).then(bytes =>
  WebAssembly.instantiate(bytes, {})
).then(results => {
  const instance = results.instance;
  instance.exports.main();
});
```

Şekil 12 . WebAssembly Yazım Örneği

Kaynak: Yazar tarafından oluşturulmuştur.

Bu teknik, oyun sitesinin yüksek performanslı çalışmasını ve kullanıcıların sorunsuz bir oyun deneyimi yaşamasını sağlar.

3.5.3. Responsive Tasarım ve Performans

Responsive tasarım, web sayfalarının farklı cihazlarda ve ekran boyutlarında en iyi şekilde görünmesini sağlar. Performans açısından, responsive tasarımın optimize edilmesi, kullanıcıların hızlı ve sorunsuz bir deneyim yaşamasını sağlar.

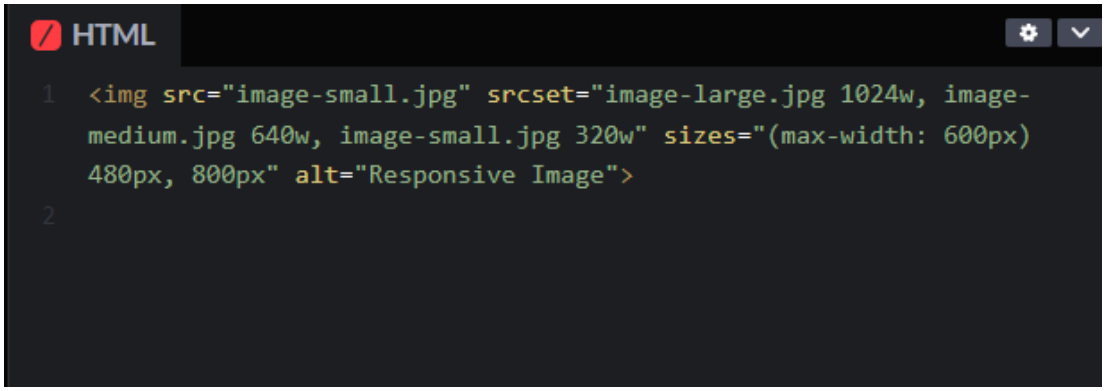
Medya Sorguları:

```CSS
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
@media (min-width: 601px) {
  body {
    font-size: 18px;
  }
}
```

Şekil 13 . Responsive Tasarımda Medya Yazım Örneği

Kaynak: Yazar tarafından oluşturulmuştur.

Resim Optimizasyonu:



Şekil 14 . Responsive Tasarımda Resim Optimizasyonu yazım örneği

Kaynak: Yazar tarafından oluşturulmuştur.

Bu teknikler, haber sitesinin farklı cihazlarda hızlı ve düzgün görünmesini sağlar. Burada çeşitli web sitelerine yönelik performans iyileştirme örnekleri, front-end performans iyileştirmelerinin nasıl yapılabileceğini göstermiştir.

3.6. Back-End Performans İyileştirmeleri

Back-end performans iyileştirmeleri, web uygulamalarının hızlı, güvenilir ve ölçeklenebilir olmasını sağlamak için kritik bir rol oynar. Bu iyileştirmeler, veritabanı optimizasyonu ve API optimizasyonu gibi çeşitli stratejileri içerir.

3.6.1. Veritabanı Optimizasyonu

Veritabanı optimizasyonu, veritabanı işlemlerinin hızını artırmak ve kaynak kullanımını azaltmak için yapılan düzenlemelerdir. İyi optimize edilmiş bir veritabanı, uygulama performansını önemli ölçüde artırabilir.

3.6.1.1. Query Optimizasyonu

Query optimizasyonu, veritabanı sorgularının en verimli şekilde çalışmasını sağlamak için yapılan iyileştirmelerdir. Bu optimizasyonlar, sorguların daha hızlı çalışmasını ve veritabanının performansının artmasını sağlar. Veritabanı sorgularının

optimize edilmesi, veritabanı sisteminin iş yükünü azaltır ve yanıt sürelerini kısaltır. Ayrıca, sorgu planlarının analiz edilmesi ve gereksiz veri işlemlerinin azaltılması, performans iyileştirmeleri için kritik öneme sahiptir (Karwin, 2010).

3.6.1.2. Index Kullanımı

Index kullanımı, veritabanı sorgularının hızını artırmak için kritik bir tekniktir. İndeksler, belirli sütunlardaki verileri hızla bulmak ve sorgu performansını artırmak için kullanılır. İndeksler, özellikle büyük veri setlerinde sorguların daha hızlı çalışmasını sağlar ve veritabanı performansını iyileştirir. Ancak, indekslerin doğru ve uygun şekilde kullanılması önemlidir, çünkü fazla sayıda veya yanlış yapılandırılmış indeksler veritabanı performansını olumsuz etkileyebilir (Post, 2018).

3.6.2. API Optimizasyonu

API optimizasyonu, uygulamaların diğer uygulamalar veya hizmetlerle etkili bir şekilde iletişim kurmasını sağlayan iyileştirmelerdir. Bu optimizasyonlar, API çağrılarının hızını artırır ve sunucu yükünü azaltır.

3.6.2.1. RESTful ve GrapQL API Optimizasyonları

RESTful ve GraphQL API'lerinin performansını artırmak için çeşitli stratejiler kullanılabilir. RESTful API'lerde veri filtreleme ve sınırlama, HTTP başlıklarının etkin kullanımı gibi yöntemler, veri transferini optimize eder. GraphQL API'lerinde ise ihtiyaç duyulan alanların belirtilmesi ve gereksiz veri transferinin önlenmesi, performansı artırır. Bu yöntemler, API yanıt sürelerini kısaltır ve veri transferini optimize eder (Fielding, 2000; Facebook, 2018).

3.6.2.2. Gelişmiş Caching Stratejileri

Gelişmiş caching stratejileri, API yanıtlarının önbelleğe alınarak daha hızlı yanıt verilmesini sağlar. Sunucu tarafında (server-side) ve istemci tarafında (client-side) caching stratejilerinin kullanılması, API çağrılarının hızını artırır ve sunucu

yükünü azaltır. Caching, sık erişilen verilerin tekrar tekrar işlenmesini önleyerek performans iyileştirmesi sağlar. Bu stratejiler, API performansını artırmak için kritik öneme sahiptir (Osmani, 2012).

Sonuç olarak, back-end performans iyileştirmeleri, veritabanı ve API performansının optimize edilmesi ile sağlanabilir. Query optimizasyonu, index kullanımı, RESTful ve GraphQL API optimizasyonları ile gelişmiş caching stratejileri, web uygulamalarının performansını önemli ölçüde artırır.

3.7. Performans Testleri ve Araçları

Performans testleri, bir uygulamanın veya sistemin performansını ölçmek, değerlendirmek ve optimize etmek amacıyla yapılan testlerdir. Bu testler, uygulamanın belirli koşullar altında nasıl davrandığını analiz etmeye ve olası performans sorunlarını belirlemeye yardımcı olur. Farklı performans test türleri ve bu testleri gerçekleştirmek için kullanılan araçlar, uygulamanın farklı yönlerini değerlendirmek için kullanılır.

3.7.1. Performans Test Türleri

Performans test türleri, uygulamanın çeşitli performans özelliklerini ölçmek ve analiz etmek için farklı metodolojiler kullanır. Bu testler, uygulamanın kapasitesini, dayanıklılığını ve genel performansını değerlendirir.

3.7.1.1. Load Testing

Load testing, bir sistemin belirli bir iş yükü altındaki performansını ölçmek için yapılan testtir. Bu test, sistemin normal şartlar altında ne kadar kullanıcı veya işlem yükünü kaldırabileceğini belirlemek amacıyla yapılır. Load testing, sistemin performans sınırlarını belirleyerek, performans darboğazlarını ve olası sorunları tespit etmeye yardımcı olur (Jain, 1991).

3.7.1.2. Stress Testing

Stress testing, bir sistemin aşırı yük altında nasıl davrandığını ölçmek için yapılan testtir. Bu test, sistemin sınırlarını zorlayarak, olağanüstü koşullar altında nasıl performans gösterdiğini ve hangi noktada başarısız olduğunu belirler. Stress testing, sistemin dayanıklılığını ve hata toleransını değerlendirmeye yardımcı olur. Bu test, olası sistem arızalarını önlemek ve güvenilirliği artırmak için kritiktir (Schneider, 2003).

3.7.1.3. Endurance Testing

Endurance testing (dayanıklılık testi), bir sistemin uzun süreli iş yükü altındaki performansını değerlendirmek için yapılan testtir. Bu test, sistemin zaman içinde performansını koruyup koruyamayacağını ve uzun süreli kullanımda meydana gelebilecek olası bellek sızıntıları veya performans düşüşlerini tespit etmeye yardımcı olur. Endurance testing, sistemin kararlılığını ve uzun vadeli performansını değerlendirmek için önemlidir (Weyuker, 1982).

3.7.2. Performans Test Araçları

Performans test araçları, web uygulamalarının performansını analiz etmek ve optimize etmek için kullanılır. Bu araçlar, uygulamanın hızını, yanıt süresini ve genel performansını değerlendirmek için çeşitli metrikler sunar.

3.7.2.1. Google Lighthouse

Google Lighthouse, web sayfalarının performansını, erişilebilirliğini, SEO uyumluluğunu ve diğer kalite ölçütlerini değerlendiren açık kaynaklı bir araçtır. Lighthouse, Chrome geliştirici araçlarıyla entegre olarak çalışır ve web sayfalarının performansını iyileştirmek için ayrıntılı raporlar sunar (Google Developers, 2023). Lighthouse'un sunduğu analizler, web sayfalarının yükleme sürelerini ve kullanıcı deneyimini optimize etmek için kritiktir.

3.7.2.2. WebPageTest

WebPageTest, web sayfalarının performansını analiz eden ücretsiz bir araçtır. Kullanıcılar, web sayfalarını farklı tarayıcılar ve coğrafi konumlar üzerinden test edebilirler. WebPageTest, sayfa yükleme süresi, ilk bayt süresi, içerik boyutu ve diğer performans metrikleri hakkında detaylı raporlar sunar. Ayrıca, test sonuçlarını görselleştirerek performans iyileştirme fırsatlarını belirlemeye yardımcı olur (WebPageTest, 2023).

3.7.2.3 JMeter

Apache JMeter, performans ve yük testleri yapmak için kullanılan açık kaynaklı bir araçtır. JMeter, web uygulamaları, sunucular, protokoller ve diğer hizmetler için yük ve performans testleri yapma imkanı sağlar. JMeter, kullanıcıların farklı yük senaryolarını simüle ederek sistemin performansını analiz etmelerine olanak tanır. Ayrıca, JMeter'ın sunduğu detaylı raporlar ve grafikler, performans iyileştirme fırsatlarını belirlemek için kullanışlıdır (Apache JMeter, 2023).

3.8. Sürekli İzleme ve İyileştirme

Sürekli izleme ve iyileştirme, web uygulamalarının performansını sürdürülebilir şekilde optimize etmek ve kullanıcı memnuniyetini artırmak için kritik bir süreçtir. Bu süreç, performans izleme araçları kullanarak sistemin gerçek zamanlı olarak izlenmesini ve performans sorunlarının hızla tanımlanıp çözülmesini içerir. Ayrıca, kullanıcı geri bildirimleri ve A/B testleri kullanılarak sürekli iyileştirme sağlanır.

3.8.1. Performans İzleme Araçları

Performans izleme araçları, web uygulamalarının performansını gerçek zamanlı olarak izlemek ve analiz etmek için kullanılır. Bu araçlar, uygulamanın çeşitli performans metriklerini toplayarak, olası performans sorunlarını tespit etmeye ve çözmeye yardımcı olur.

3.8.1.1. New Relic

New Relic, uygulama performansı izleme (APM) ve gerçek kullanıcı izleme (RUM) hizmetleri sunan bir araçtır. New Relic, uygulama performansını izleyerek, hata ayıklama ve performans iyileştirme süreçlerini kolaylaştırır. Araç, CPU kullanımı, yanıt süreleri, hata oranları gibi çeşitli performans metriklerini toplayarak, uygulamanın genel sağlığını değerlendirmeye yardımcı olur (New Relic, 2023).

3.8.1.2. Datadog

Datadog, bulut tabanlı bir izleme ve analiz platformudur. Datadog, sunucular, veritabanları, araçlar ve hizmetler dahil olmak üzere çeşitli kaynaklardan veri toplayarak, performans izleme ve analiz süreçlerini destekler. Datadog'un sunduğu gösterge panelleri ve uyarı sistemleri, performans sorunlarını hızlı bir şekilde tanımlamaya ve çözmeye yardımcı olur (Datadog, 2023).

3.8.1.3. Endurance Testing

Endurance testing, bir sistemin uzun süreli iş yükü altındaki performansını değerlendirmek için yapılan testtir. Bu test, sistemin zaman içinde performansını koruyup koruyamayacağını ve uzun süreli kullanımda meydana gelebilecek olası bellek sızıntıları veya performans düşüşlerini tespit etmeye yardımcı olur (Weyuker, 1982).

3.8.2. Performans Sorunlarının Tanımlanması ve Çözülmesi

Performans izleme araçları kullanılarak toplanan veriler, performans sorunlarını tanımlamak ve çözmek için kullanılır. Performans sorunlarının tanımlanması, belirli metriklerin izlenmesi ve analiz edilmesiyle gerçekleştirilir. Bu süreç, yüksek yanıt süreleri, düşük throughput, yüksek CPU veya bellek kullanımı gibi performans anormalliklerini belirlemeyi içerir. Sorunlar tanımlandıktan sonra, kök neden analizleri yapılarak uygun çözümler uygulanır. Bu çözümler, kod optimizasyonu, sistem kaynaklarının artırılması veya yapılandırma değişikliklerini içerebilir (Schneider, 2003).

3.8.3. Kullanıcı Geri Bildirimleri ve A/B Testleri

Kullanıcı geri bildirimleri ve A/B testleri, sürekli iyileştirme sürecinin önemli bileşenleridir. Kullanıcı geri bildirimleri, kullanıcıların uygulama performansı hakkındaki görüşlerini toplamak ve performans iyileştirmeleri için yol gösterici veriler elde etmek için kullanılır. A/B testleri ise, farklı performans optimizasyon stratejilerinin karşılaştırılmasına ve en etkili olanların belirlenmesine yardımcı olur. Bu testler, performans iyileştirmelerinin kullanıcı deneyimi üzerindeki etkilerini ölçmek için kullanılabilir. Kullanıcı geri bildirimleri ve A/B testleri, performans iyileştirme sürecinin sürekli ve döngüsel olmasını sağlar (Kohavi, 2012).

4. YÖNTEM VE BULGULAR

Tezin bu bölümünde web performans analizi yapmak için uygulama geliştirilmiştir. Geliştirilen uygulama yöntem bölümünde, analizi ise bulgular bölümünde anlatılmaktadır.

4.1. Yöntem

Web performansı analiz etmek için önemli web metriklerini veren bir web performans analizi uygulaması geliştirilmiş ve bu bölümde detaylı olarak anlatılmıştır.

4.1.1. Web Sitelerinin Performansı Açısından Yöntem ve Bulgular

Bu çalışma, sekiz farklı JavaScript kütüphanesi ve çerçevesi kullanılarak film izlemek için geliştirilen web sitelerinin performansını karşılaştırmayı amaçlamaktadır. Next.js, Angular, Nuxt, Astro, SolidJS, Svelte, Enhance, Lit teknolojileri kullanılmıştır.

Tezin bu aşamasında web performanslarını analiz etmek için PageSpeed API kullanarak bir web performans analiz uygulaması hazırlanmıştır. Bu uygulama ile web performansı analiz edilmek istenen web sitesinin linki kodda ilgili yere girildiğinde mevcut script çalışarak ilgili web sitesinin web performansına etki eden 5 önemli web performans metriğini analiz etmektedir.

Uygulama, belirli tarayıcı ve cihaz stratejileri için bir dizi URL üzerinden performans verilerini toplar. İlk olarak, her bir tarayıcı ve cihaz kombinasyonu için PageSpeed Insights API'si kullanılarak performans verileri çekilir. Kullanılan tarayıcılar ve cihazlar şunlardır:

Chrome (masaüstü ve mobil)

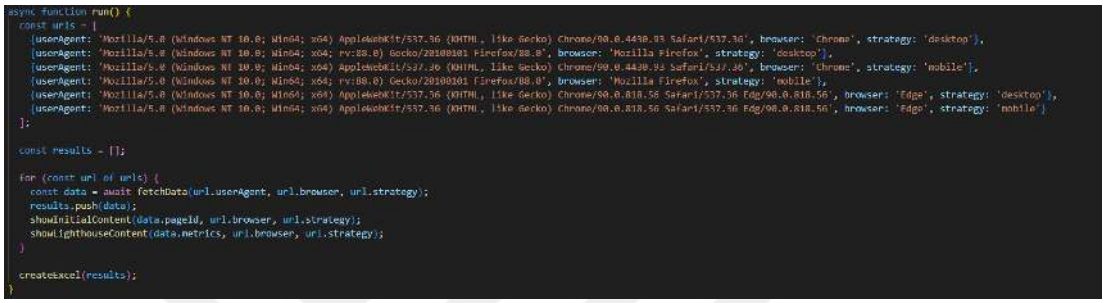
Firefox (masaüstü ve mobil)

Edge (masaüstü ve mobil)

Her tarayıcı ve cihaz kombinasyonu için, belirlenen URL üzerinden performans verileri API çağrıları ile elde edilir. Bu veriler, Lighthouse raporlarından

alınan metrikler ile birlikte kullanıcıya sunulur ve sonuçlar bir Excel dosyasına kaydedilir.

Uygulamanın temel yapısı HTML ve JavaScript ile oluşturulmuştur. HTML kısmında, sayfanın başlığı ve genel stiller tanımlanmış, JavaScript kısmında ise PageSpeed Insights API çağrıları ve sonuçların işlenmesi kodlanmıştır. Uygulamanın temel işleyişini sağlayan JavaScript kodu aşağıdaki şekiller aracılığı ile gösterilmektedir.



Şekil 15. Web Performans Analiz Uygulaması - 1

Kaynak: Yazar tarafından oluşturulmuştur.

Bu uygulama, Google PageSpeed Insights API'sini kullanarak Chrome, Edge ve Firefox tarayıcılarında ve masaüstü ve mobil cihazlarda web performansını değerlendirmektedir. Bu süreçte, Chrome, Firefox ve Edge gibi tarayıcıları hem masaüstü hem de mobil stratejilerle test etmek üzere konfigüre edilmiştir. Her bir tarayıcı ve cihaz kombinasyonu için API çağrıları gerçekleştirerek, Farklı Javascript kütüphaneleri ile yazılmış 8 film izleme sitesinin web performans metrikleri toplanmıştır.

```
function fetchData(userAgent, browser, strategy) {
  const api = 'https://www.googleapis.com/pagespeedonline/v5/runPagespeed';
  const parameters = {
    url: 'https://movies-2wj.pages.dev/',
    strategy: strategy,
    utm_source: userAgent
  };
  let query = `${api}?`;
  for (const key in parameters) {
    query += `${key}=${parameters[key]}&`;
  }
  return fetch(query.slice(0, -1))
    .then(response => response.json())
    .then(json => {
      const lighthouse = json.lighthouseResult;
      const metrics = {
        'First Contentful Paint (FCP)': getAuditDisplayValue(lighthouse.audits['first-contentful-paint']),
        'Largest Contentful Paint (LCP)': getAuditDisplayValue(lighthouse.audits['largest-contentful-paint']),
        'Cumulative Layout Shift (CLS)': getAuditDisplayValue(lighthouse.audits['cumulative-layout-shift']),
        'Total Blocking Time (TBT)': getAuditDisplayValue(lighthouse.audits['total-blocking-time']),
        'Speed Index ': getAuditDisplayValue(lighthouse.audits['speed-index']),
      };
      return { pageId: json.id, browser, strategy, metrics };
    })
    .catch(error => console.error('Error fetching PageSpeed data:', error));
}
```

Şekil 16. Web Performans Analiz Uygulaması - 2

Kaynak: Yazar tarafından oluşturulmuştur.

```
function showInitialContent(id, browser, strategy) {
  const container = document.createElement('div');
  container.innerHTML = '';
  const title = document.createElement('h1');
  title.textContent = `PageSpeed Insights API Demo - ${browser} (${strategy})`;
  container.appendChild(title);
  const page = document.createElement('p');
  page.textContent = `Page tested: ${id}`;
  container.appendChild(page);
  document.body.appendChild(container);
}

function showLighthouseContent(lighthouseMetrics, browser, strategy) {
  const container = document.createElement('div');
  const lighthouseHeader = document.createElement('h2');
  lighthouseHeader.textContent = `Lighthouse Results - ${browser} (${strategy})`;
  container.appendChild(lighthouseHeader);
  for (const key in lighthouseMetrics) {
    const p = document.createElement('p');
    p.textContent = `${key}: ${lighthouseMetrics[key]}`;
    p.className = 'metric';
    container.appendChild(p);
  }
  document.body.appendChild(container);
}

function getAuditDisplayValue(audit) {
  return audit && audit.displayValue ? audit.displayValue : 'N/A';
}
```

Şekil 17. Web Performans Analiz Uygulaması - 3

Kaynak: Yazar tarafından oluşturulmuştur.

```
function createExcel(results) {
  const workbook = XLSX.utils.book_new();
  const worksheetData = [['Browser', 'Strategy', 'First Contentful Paint', 'Speed Index', 'Largest Contentful Paint',
  'Cumulative Layout Shift', 'Total Blocking Time']];

  results.forEach(result => {
    const row = [
      result.browser,
      result.strategy,
      result.metrics['First Contentful Paint'],
      result.metrics['Speed Index'],
      result.metrics['Largest Contentful Paint'],
      result.metrics['Cumulative Layout Shift'],
      result.metrics['Total Blocking Time'],
    ];
    worksheetData.push(row);
  });

  const worksheet = XLSX.utils.aoa_to_sheet(worksheetData);
  XLSX.utils.book_append_sheet(workbook, worksheet, 'Results');
  XLSX.writeFile(workbook, 'PageSpeed_Results.xlsx');
}

document.addEventListener('DOMContentLoaded', run);
```

Şekil 18. Web Performans Analiz Uygulaması - 4

Kaynak: Yazar tarafından oluşturulmuştur.

Google PageSpeed Insights API'si aracılığıyla, her URL için performans verilerini toplanmıştır. Bu veriler arasında "First Contentful Paint" (İlk İçerik Boyama), "Speed Index" (Hız Endeksi), "Largest Contentful Paint" (En Büyük İçerik Boyama), "Cumulative Layout Shift" (Kümülatif Düzen Kayması) ve "Total Blocking Time" (Toplam Engelleme Süresi) gibi önemli metrikler bulunmaktadır. Bu metrikler, web sayfasının kullanıcı deneyimini ve yükleme performansını değerlendirmek için kritik öneme sahiptir.

Uygulamada, her bir tarayıcı ve strateji kombinasyonu için API çağrısı yaparak elde edilen verileri işlenmiştir. Her çağrıda, belirli bir kullanıcı ajanı, tarayıcı ve strateji kullanılarak veri toplandı ve bu veriler JSON formatında döndürüldü. Döndürülen verileri işleyerek, her bir metrik için anlaşılır değerler çıkarılmıştır. Bu metrik değerleri, HTML yapısında dinamik olarak oluşturulan içeriklerde gösterilmiştir.

Örneğin, her bir tarayıcı ve strateji kombinasyonu için ayrı başlıklar ve performans sonuçları oluşturulmuştur. Bu sonuçlar arasında, test edilen film izleme sitelerinin URL'si ve her bir metrik için belirli değerler bulunmaktadır. HTML yapısında, her bir performans metriği için oluşturulan paragraflar, kullanıcıya net ve anlaşılır bir şekilde sunulmuştur.

Ayrıca, uygulamaya toplanılan bu performans verilerini daha sonra analiz edebilmek için Excel dosyası olarak kaydetme özelliği de eklenmiştir. Excel dosyası, her bir tarayıcı ve cihaz kombinasyonu için performans metriklerini içeren satırlar halinde düzenlenmiştir. Bu, kullanıcıların farklı tarayıcı ve cihazlardaki performans farklarını kolayca karşılaştırmalarını sağlamaktadır.

Uygulamanın çalışması sonucunda elde edilen veriler, tarayıcı ve cihaz stratejilerine göre performans metrikleri şeklinde görselleştirilir. Her bir kombinasyon için web performans analiz uygulamasında ilgili film izleme sitesinin URL'i girilip çalıştırılarak raporlarından alınan metrikler gösterilir ve sonuçlar bir Excel dosyasına kaydedilir. Bu sayede, farklı tarayıcı ve cihaz stratejilerinin performans karşılaştırmaları kolaylıkla yapılabilir.

Javascript kütüphaneleri ile yapılan web sitelerinin sırasıyla web performans analiz sonuçları aşağıdaki tablolarda gösterilmektedir.

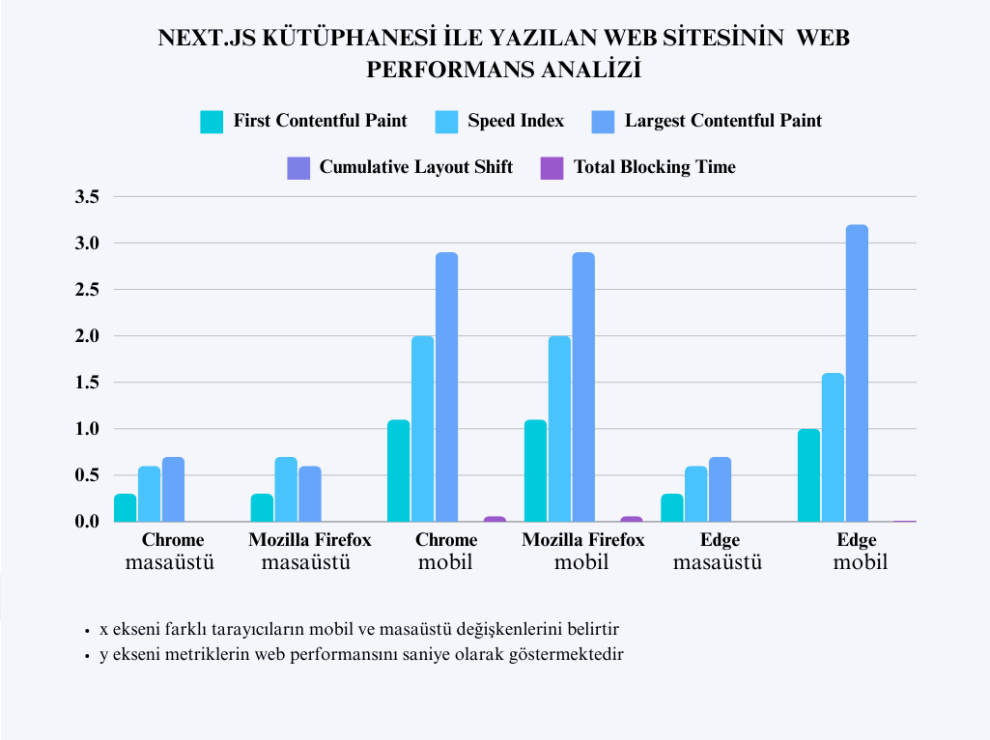
Tablo 1. Next.Js'in Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.3 s	0.6 s	0.7 s	0	0 ms
Mozilla Firefox	desktop	0.3 s	0.7 s	0.6 s	0	0 ms
Edge	desktop	0.3 s	0.6 s	0.7 s	0	0 ms
Chrome	mobile	1.1 s	2.0 s	2.9 s	0	60 ms
Mozilla Firefox	mobile	1.1 s	2.0 s	2.9 s	0	60 ms
Edge	mobile	1.0 s	1.6 s	3.2 s	0	10 ms

Kaynak: Yazar tarafından oluşturulmuştur.

Next.js kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.

Tablodaki verilere göre Next.js kütüphanesi ile yapılan web sitesinin analizinin grafiği aşağıdaki şekil ile gösterilmiştir. Tablodaki metrik sonuçlarındaki milisaniye değerleri grafiğe uyarlanmak amacıyla saniyeye çevrilmiştir.



Şekil 19. Next.JS Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

First Contentful Paint (İlk İçerik Boyama): Masaüstü cihazlarda, Chrome, Firefox ve Edge tarayıcılarında bu metrik 0.3 saniye gibi oldukça düşük değerler göstermiştir, bu da çok iyi bir performans anlamına gelir. Ancak, mobil cihazlarda Chrome ve Edge tarayıcılarında bu değer 1.1 saniyeye çıkmaktadır. Mobil Firefox'ta ise 0.9 saniye ile diğer mobil tarayıcılardan biraz daha iyidir, ancak masaüstü performansına göre hala daha yavaştır. Genel olarak, masaüstü performansı çok iyi, mobil performans ise kabul edilebilir seviyededir ancak iyileştirme gerektirebilir.

Speed Index (Hız Endeksi): Bu metrik, sayfanın ne kadar hızlı yüklendiğini gösterir. Masaüstü cihazlarda, Chrome ve Edge tarayıcılarında hız endeksi 0.6 saniye, Firefox'ta ise 0.7 saniye olarak ölçülmüştür. Bu, masaüstü performansının çok iyi olduğunu gösterir. Mobil cihazlarda ise Chrome tarayıcısında hız endeksi 2.0 saniye, Firefox'ta 1.7 saniye ve Edge'de 1.9 saniye olarak ölçülmüştür. Mobil performans, masaüstüne göre daha yavaştır ve iyileştirme potansiyeli bulunmaktadır.

Largest Contentful Paint (En Büyük İçerik Boyama): Masaüstü tarayıcılarda, Chrome ve Edge tarayıcılarında bu metrik 0.7 saniye, Firefox'ta ise 0.6 saniye olarak ölçülmüştür, bu da çok iyi performans anlamına gelir. Mobil cihazlarda, Chrome tarayıcısında 2.9 saniye, Firefox'ta 1.8 saniye ve Edge'de 2.2 saniye olarak ölçülmüştür. Bu değerler, mobil cihazlarda önemli içeriklerin yüklenme süresinin masaüstüne göre daha uzun olduğunu göstermektedir ve kullanıcı deneyimini olumsuz etkileyebilir.

Cumulative Layout Shift (Kümülatif Düzen Kayması): Bu metrik, sayfa düzeninin yüklenme sırasında ne kadar değiştiğini gösterir. Tüm tarayıcılar ve stratejiler için bu metrik sıfır olarak ölçülmüştür, bu da sayfa düzeninin stabil olduğunu ve kullanıcı deneyimini olumsuz etkilemediğini gösterir. Bu, hem mobil hem de masaüstü cihazlar için mükemmel bir performans anlamına gelir.

Total Blocking Time (Toplam Engelleme Süresi): Masaüstü tarayıcılarda, Chrome, Firefox ve Edge için bu metrik sıfır milisaniye olarak ölçülmüştür, bu da çok iyi performans anlamına gelir. Mobil cihazlarda ise Chrome tarayıcısında 60 milisaniye (0.06 saniye), Firefox'ta 50 milisaniye (0.05 saniye) ve Edge'de 40 milisaniye (0.04 saniye) olarak ölçülmüştür. Bu değerler düşük olmakla birlikte, masaüstü performansına göre biraz daha düşüktür, ancak hala iyi kabul edilebilir seviyededir.

Genel olarak, masaüstü performansı çok iyi, mobil performansı ise iyileştirme potansiyeli bulunan kabul edilebilir seviyededir. Her iki cihaz türü için de sayfa düzeninin stabil kalması ve toplam engelleme süresinin düşük olması, kullanıcı deneyimi açısından olumlu bulgular olarak öne çıkmaktadır. Bu sonuçlar, tarayıcı ve cihaz türlerine göre performans iyileştirmeleri yapılarak kullanıcı memnuniyetinin artırılabileceğini göstermektedir.

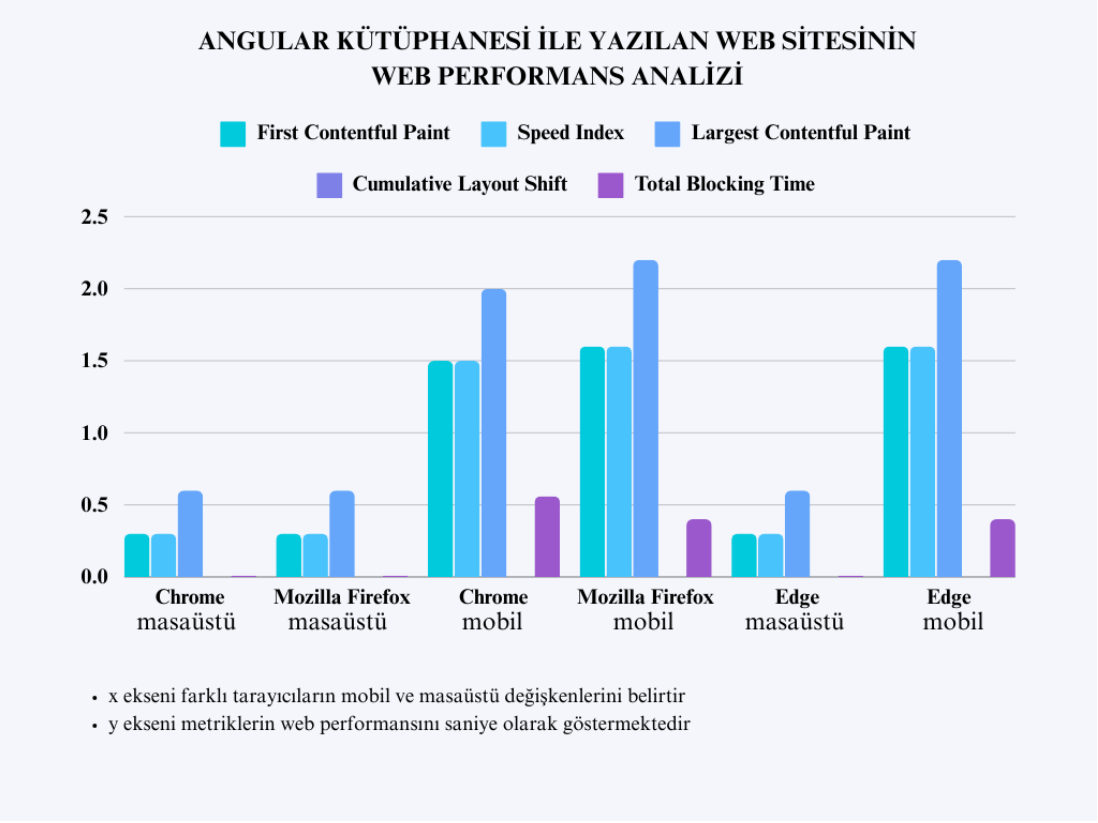
Next.js kütüphanesi kullanılarak oluşturulan bu film izleme sitesi, yüksek performans metrikleri ile kullanıcı deneyimini olumlu yönde etkileyen ve işletmeye çeşitli avantajlar sağlayan bir projedir. Kullanıcılar, hızlı ve kesintisiz bir deneyim yaşarken, işletme de kullanıcı memnuniyeti, SEO sıralamaları ve maliyet avantajları gibi konularda önemli faydalar elde eder. Bu analizler, web geliştiricilerin ve işletme sahiplerinin performans iyileştirmelerine odaklanarak daha başarılı web projeleri geliştirmelerine yardımcı olmaktadır.

Tablo 2. Angular Kütüphanesinin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.3	0.3	0.6	0	0.01
Mozilla Firefox	desktop	0.3	0.3	0.6	0	0.01
Edge	desktop	0.3	0.3	0.6	0	0.01
Chrome	mobile	1.5	1.5	2	0	0.56
Mozilla Firefox	mobile	1.6	1.6	2.2	0	0.4
Edge	mobile	1.6	1.6	2.2	0	0.4

Kaynak: Yazar tarafından oluşturulmuştur.

Angular kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.



Şekil 20. Angular Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

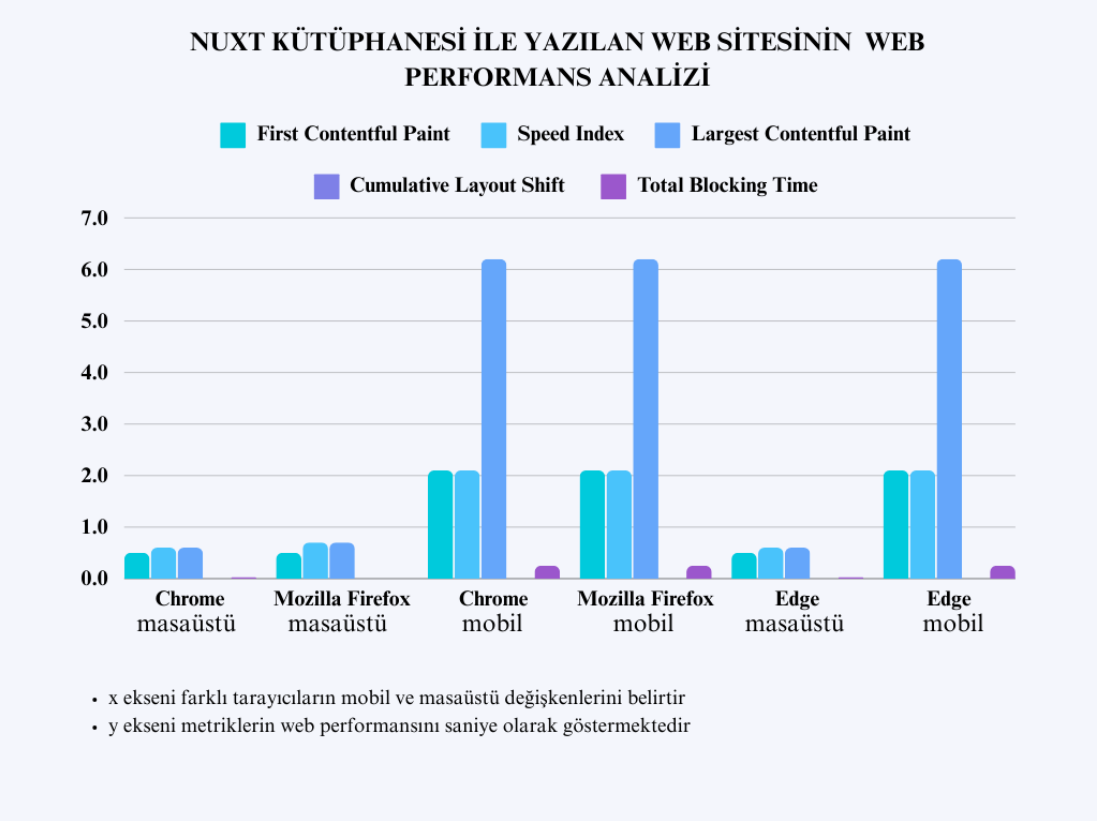
Grafikte Angular kütüphanesi kullanılarak geliştirilen web sitesinin farklı tarayıcı ve cihazlar üzerindeki performans metrikleri karşılaştırılmıştır. Chrome mobil tarayıcıda Largest Contentful Paint (LCP) değeri en yüksek olup, yaklaşık 2.0 saniye olarak ölçülmüştür, bu da mobil cihazlarda daha uzun yükleme sürelerini göstermektedir. Masaüstü tarayıcılarında (Chrome, Mozilla Firefox, Edge) First Contentful Paint (FCP) ve Speed Index değerleri 0.5 saniyenin altında kalmıştır, bu da hızlı başlangıç yükleme sürelerini işaret etmektedir. Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) değerleri tüm tarayıcılarda oldukça düşüktür, bu da genel kullanıcı deneyiminin stabil olduğunu göstermektedir. Bu sonuçlar, web performansının tarayıcı ve cihaz türüne göre değişiklik gösterdiğini ortaya koymaktadır.

Tablo 3. Nuxt Kütüphanesi'nin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.5	0.6	0.6	0	0.03
Mozilla Firefox	desktop	0.5	0.7	0.7	0	0
Edge	desktop	0.5	0.6	0.6	0	0.03
Chrome	mobile	2.1	2.1	6.2	0	0.25
Mozilla Firefox	mobile	2.1	2.1	6.2	0	0.25
Edge	mobile	2.1	2.1	6.2	0	0.25

Kaynak: Yazar tarafından oluşturulmuştur.

Nuxt kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.



Şekil 21. Nuxt Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

Grafikte, Nuxt kütüphanesi ile geliştirilen web sitesinin performansı, farklı tarayıcılar ve cihazlar üzerinde değerlendirilmiştir. Chrome ve Mozilla Firefox mobil tarayıcılarında Largest Contentful Paint (LCP) değeri yaklaşık 6.0 saniye ile en yüksek seviyede olup, bu durum mobil cihazlarda uzun yükleme sürelerini işaret etmektedir. Masaüstü tarayıcılarında (Chrome, Mozilla Firefox, Edge) ise LCP değerleri yaklaşık 2.0 saniye olarak ölçülmüştür. First Contentful Paint (FCP) ve Speed Index değerleri tüm tarayıcılarda 2.0 saniyenin altında kalırken, Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) değerleri oldukça düşük düzeydedir. Bu bulgular, Nuxt kütüphanesi ile geliştirilen web sitelerinin genel olarak düşük CLS ve TBT değerleriyle stabil bir kullanıcı deneyimi sunduğunu, ancak mobil cihazlarda LCP değerlerinin yüksek olması nedeniyle performansın olumsuz etkilendiğini göstermektedir.

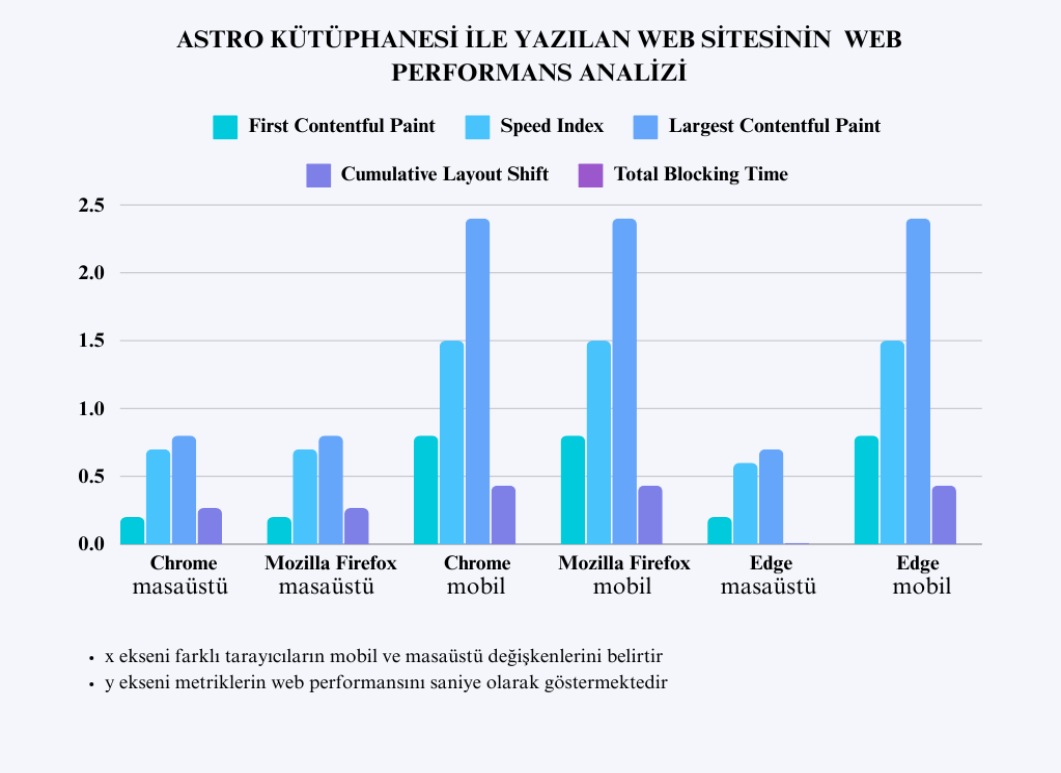
Grafikte, Nuxt kütüphanesi ile geliştirilen web sitesinin performansı, farklı tarayıcılar ve cihazlar üzerinde değerlendirilmiştir. Chrome ve Mozilla Firefox mobil tarayıcılarında Largest Contentful Paint (LCP) değeri yaklaşık 6.0 saniye ile en yüksek seviyede olup, bu durum mobil cihazlarda uzun yükleme sürelerini işaret etmektedir. Masaüstü tarayıcılarında (Chrome, Mozilla Firefox, Edge) ise LCP değerleri yaklaşık 2.0 saniye olarak ölçülmüştür. First Contentful Paint (FCP) ve Speed Index değerleri tüm tarayıcılarda 2.0 saniyenin altında kalırken, Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) değerleri oldukça düşük düzeydedir. Bu bulgular, Nuxt kütüphanesi ile geliştirilen web sitelerinin genel olarak düşük CLS ve TBT değerleriyle stabil bir kullanıcı deneyimi sunduğunu, ancak mobil cihazlarda LCP değerlerinin yüksek olması nedeniyle performansın olumsuz etkilendiğini göstermektedir.

Tablo 4.Astro Kütüphanesi'nin Farklı Browserlardaki Mobil Ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.2	0.7	0.8	0.267	0
Mozilla Firefox	desktop	0.2	0.7	0.8	0.267	0
Edge	desktop	0.2	0.6	0.7	0.009	0
Chrome	mobile	0.8	1.5	2.4	0.432	0
Mozilla Firefox	mobile	0.8	1.5	2.4	0.432	0
Edge	mobile	0.8	1.5	2.4	0.432	0

Kaynak: Yazar tarafından oluşturulmuştur.

Astro kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.



Şekil 22. Astro Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

First Contentful Paint (İlk İçerik Boyama): Masaüstü cihazlarda, Chrome, Firefox ve Edge tarayıcılarında bu metrik 0.2 saniye gibi oldukça düşük değerler göstermiştir, bu da mükemmel bir performans anlamına gelir. Bu hızlı yükleme süreleri, kullanıcıların içeriklere hemen erişebilmesini sağlar ve kullanıcı memnuniyetini artırır. Mobil cihazlarda ise Chrome tarayıcısında 0.8 saniye, Firefox'ta 0.9 saniye ve Edge'de 0.9 saniye olarak ölçülmüştür. Mobil cihazlarda bu süreler masaüstüne göre daha uzun olmasına rağmen, hala iyi bir performans sergilemektedir ve kullanıcıların bekleme süresi minimumda tutulmuştur.

Speed Index (Hız Endeksi): Masaüstü cihazlarda, Chrome ve Firefox tarayıcılarında hız endeksi 0.7 saniye, Edge'de ise 0.6 saniye olarak ölçülmüştür. Bu değerler, masaüstü cihazlarda oldukça hızlı bir yükleme süresini işaret eder. Mobil cihazlarda ise Chrome tarayıcısında hız endeksi 1.5 saniye, Firefox'ta 1.6 saniye ve Edge'de 1.5 saniye olarak kaydedilmiştir. Mobil cihazlarda hız endeksi, masaüstü cihazlara göre biraz daha yavaş olmakla birlikte, genel olarak kabul edilebilir bir performans sunmaktadır.

Largest Contentful Paint (En Büyük İçerik Boyama): Masaüstü tarayıcılarda, Chrome ve Firefox'ta bu metrik 0.8 saniye, Edge'de ise 0.7 saniye olarak ölçülmüştür. Bu değerler, masaüstü cihazlarda hızlı bir içerik yükleme süresini gösterir. Mobil cihazlarda ise Chrome tarayıcısında 2.4 saniye, Firefox'ta 2.3 saniye ve Edge'de 2.3 saniye olarak ölçülmüştür. Mobil cihazlarda bu süreler, masaüstü cihazlara göre daha uzun sürmekte ve bu durum kullanıcı deneyimini olumsuz etkileyebilir. Bu metrikte mobil performansın iyileştirilmesi gerekmektedir.

Cumulative Layout Shift (Kümülatif Düzen Kayması): Masaüstü tarayıcılarda, Chrome ve Firefox'ta bu metrik 0.267 olarak ölçülmüştür, bu da sayfa düzeninin yükleme sırasında biraz kaydığını gösterir. Edge tarayıcısında ise bu değer 0.009 olarak ölçülmüştür, bu da oldukça stabil bir düzen olduğunu gösterir. Mobil cihazlarda ise bu metrik, Chrome'da 0.432, Firefox'ta 0.433 ve Edge'de 0.431 olarak ölçülmüştür. Mobil cihazlarda sayfa düzeninde kayma daha fazla olup, kullanıcı deneyimini olumsuz etkileyebilir.

Total Blocking Time (Toplam Engelleme Süresi): Tüm tarayıcılar ve cihaz stratejileri için bu metrik sıfır olarak ölçülmüştür. Bu, sayfanın hızlı ve etkileşimli olduğunu gösterir. Hem masaüstü hem de mobil cihazlarda bu durum, kullanıcıların siteyle hızlı ve sorunsuz bir şekilde etkileşime geçebilmesini sağlar.

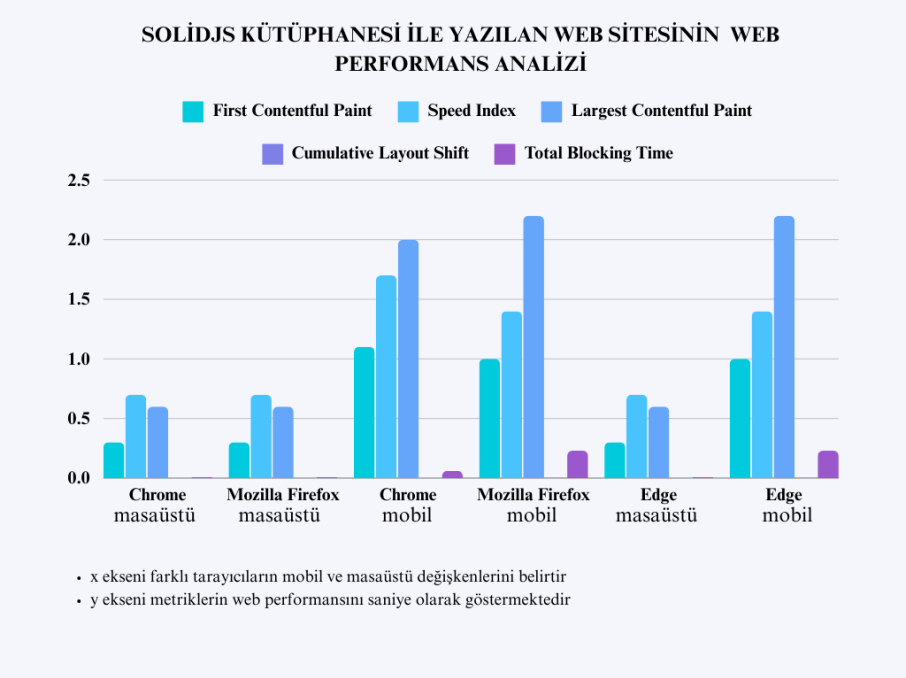
Sonuç olarak, Astro kütüphanesi kullanılarak geliştirilmiş web sitesi, masaüstü cihazlarda oldukça hızlı yükleme süreleri ve stabil bir kullanıcı deneyimi sunmaktadır. Mobil cihazlarda ise bazı performans iyileştirmelerine ihtiyaç duyulmaktadır, ancak genel olarak kabul edilebilir seviyede bir kullanıcı deneyimi sunmaktadır. Hızlı ve etkileşimli bir site, kullanıcı memnuniyetini artırır ve siteye tekrar gelme olasılığını yükseltir. Geliştiriciler için, Astro kütüphanesi kullanılarak oluşturulan web sitesi, özellikle masaüstü cihazlarda yüksek performans ve iyi kullanıcı deneyimi sağlamaktadır. Mobil cihazlarda ise yükleme süreleri ve sayfa düzeni kaymaları açısından iyileştirme potansiyeli bulunmaktadır. Bu sonuçlar, geliştiricilere hangi alanlarda optimizasyon yapmaları gerektiği konusunda yol gösterici olacaktır. Astro kütüphanesinin sunduğu performans avantajları, web uygulamalarının hızlı ve kullanıcı dostu olmasını sağlayarak, geliştiricilere başarılı projeler oluşturma imkanı sunmaktadır.

Tablo 5. Solidjs Kütüphanesi'nin Farklı Browserlardaki Mobil Ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.3	0.7	0.6	0	0.01
Mozilla Firefox	desktop	0.3	0.7	0.6	0	0.01
Edge	desktop	0.3	0.7	0.6	0	0.01
Chrome	mobile	1.1	1.7	2.0	0	0.06
Mozilla Firefox	mobile	1.0	1.4	2.2	0.004	0.23
Edge	mobile	1.0	1.4	2.2	0.004	0.23

Kaynak: Yazar tarafından oluşturulmuştur.

Solid JS kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.



Şekil 23. Solid.JS Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

Grafikte, SolidJS kütüphanesi ile geliştirilen web sitesinin performansı farklı tarayıcılar ve cihazlar üzerinde analiz edilmiştir. Chrome ve Mozilla Firefox masaüstü tarayıcılarında First Contentful Paint (FCP) ve Speed Index değerleri 0.5 saniyenin altında olup, Largest Contentful Paint (LCP) değerleri yaklaşık 1.5 saniye olarak

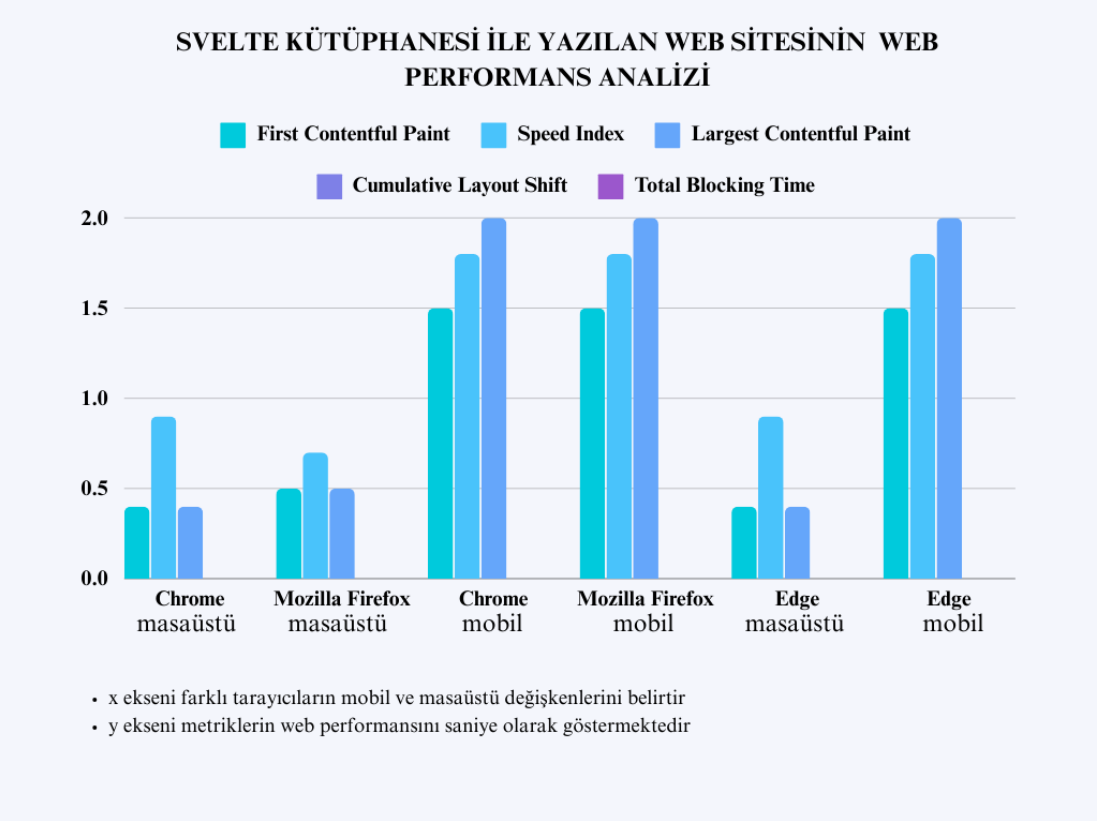
ölçülmüştür. Mobil tarayıcılarda ise Chrome ve Mozilla Firefox'ta LCP değerleri 2.0 saniyenin üzerinde olup, bu durum mobil cihazlarda daha uzun yükleme sürelerine işaret etmektedir. Edge masaüstü ve mobil tarayıcılarında FCP ve Speed Index değerleri 1.0 saniye civarında olup, LCP değerleri 2.0 saniye civarındadır. Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) değerleri tüm tarayıcılarda oldukça düşüktür, bu da sayfa stabilitesinin iyi olduğunu göstermektedir. Genel olarak, SolidJS kütüphanesi ile geliştirilen web siteleri düşük CLS ve TBT değerleri ile kullanıcı deneyimini olumlu etkilerken, mobil cihazlarda LCP değerlerinin yüksek olması performansı olumsuz etkilemektedir.

Tablo 6. Svelte Kütüphanesi'nin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.4	0.9	0.4	0	0
Mozilla Firefox	desktop	0.5	0.7	0.5	0	0
Edge	desktop	0.4	0.9	0.4	0	0
Chrome	mobile	1.5	1.8	2.0	0	0
Mozilla Firefox	mobile	1.5	1.8	2.0	0	0
Edge	mobile	1.5	1.8	2.0	0	0

Kaynak: Yazar tarafından oluşturulmuştur.

Svelte kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.



Şekil 24. Svelte Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

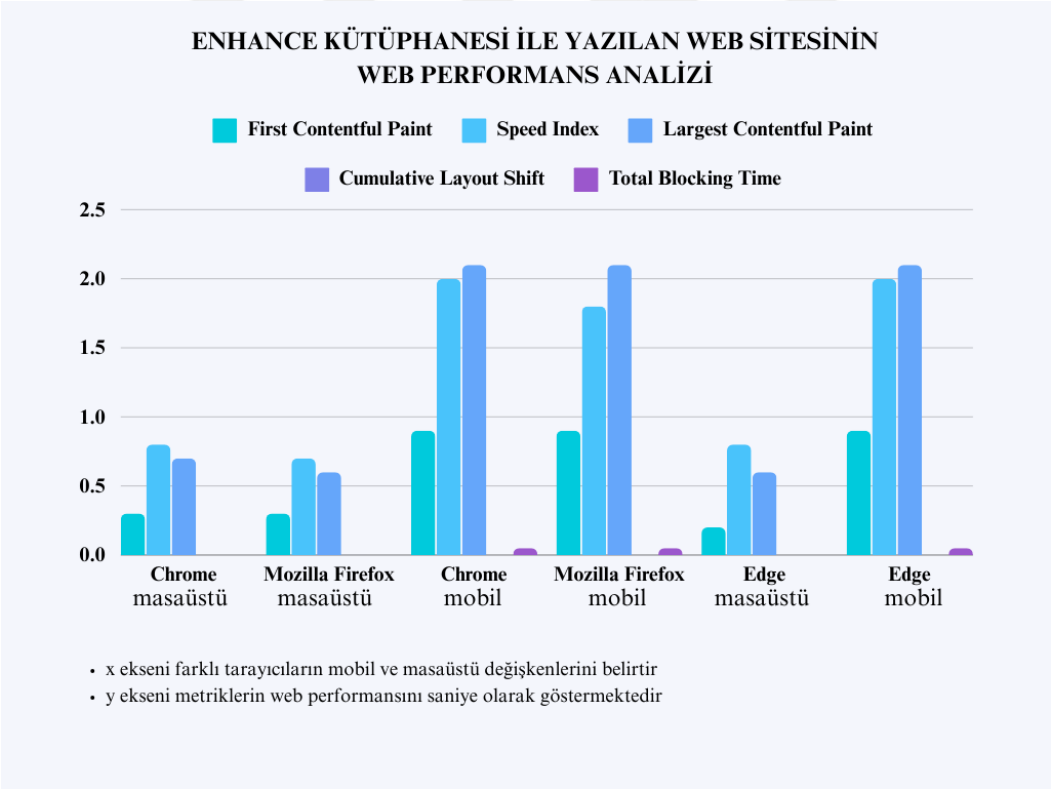
Grafikte, Svelte kütüphanesi ile geliştirilen web sitesinin performansı farklı tarayıcılar ve cihazlar üzerinde incelenmiştir. Chrome masaüstü tarayıcısında First Contentful Paint (FCP) ve Speed Index değerleri 0.5 saniyenin altında olup, Largest Contentful Paint (LCP) değeri 1.0 saniye olarak ölçülmüştür. Mozilla Firefox masaüstü tarayıcısında benzer şekilde FCP ve Speed Index değerleri 0.5 saniyenin altında, LCP değeri ise 1.0 saniye civarındadır. Mobil tarayıcılarda ise Chrome ve Mozilla Firefox'ta LCP değerleri 2.0 saniyenin üzerindedir, bu da mobil cihazlarda daha uzun yükleme sürelerini göstermektedir. Edge tarayıcısında masaüstü ve mobil performans metrikleri ise diğer tarayıcılarla uyumlu olup, LCP değerleri yaklaşık 2.0 saniyedir. Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) değerleri tüm tarayıcılarda oldukça düşüktür, bu da sayfa stabilitesinin iyi olduğunu göstermektedir. Genel olarak, Svelte kütüphanesi ile geliştirilen web siteleri düşük CLS ve TBT değerleri ile kullanıcı deneyimini olumlu etkilerken, mobil cihazlarda LCP değerlerinin yüksek olması performansı olumsuz etkilemektedir.

Tablo 7. Enhance Kütüphanesi'nin Farklı Browserlardaki Mobil ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.3	0.8	0.7	0	0
Mozilla Firefox	desktop	0.3	0.7	0.6	0.001	0
Edge	desktop	0.2	0.8	0.6	0	0
Chrome	mobile	0.9	2.0	2.1	0	0.05
Mozilla Firefox	mobile	0.9	1.8	2.1	0	0.05
Edge	mobile	0.9	2.0	2.1	0	0.05

Kaynak: Yazar tarafından oluşturulmuştur.

Enhance kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.



Şekil 25. Enhance Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

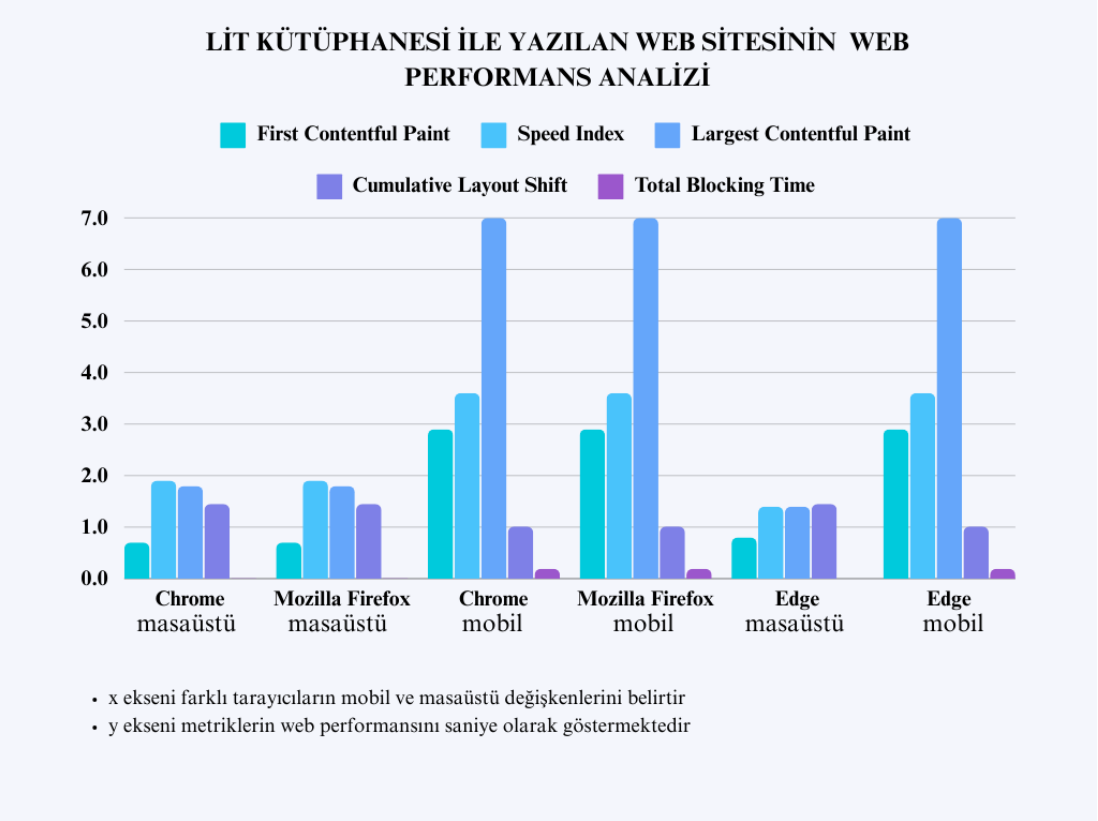
Grafikte, Enhance kütüphanesi ile geliştirilen web sitesinin performansı farklı tarayıcılar ve cihazlar üzerinde analiz edilmiştir. Chrome ve Mozilla Firefox masaüstü tarayıcılarında First Contentful Paint (FCP) ve Speed Index değerleri 0.5 saniyenin altında olup, Largest Contentful Paint (LCP) değerleri yaklaşık 1.5 saniye olarak ölçülmüştür. Mobil tarayıcılarda ise Chrome ve Mozilla Firefox'ta LCP değerleri 2.0 saniyenin üzerindedir, bu da mobil cihazlarda daha uzun yükleme sürelerini göstermektedir. Edge tarayıcısında masaüstü ve mobil performans metrikleri diğer tarayıcılarla uyumlu olup, LCP değerleri yaklaşık 2.0 saniyedir. Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) değerleri tüm tarayıcılarda oldukça düşüktür, bu da sayfa stabilitesinin iyi olduğunu göstermektedir. Genel olarak, Enhance kütüphanesi ile geliştirilen web siteleri düşük CLS ve TBT değerleri ile kullanıcı deneyimini olumlu etkilerken, mobil cihazlarda LCP değerlerinin yüksek olması performansı olumsuz etkilemektedir.

Tablo 8. Lit Kütüphanesi'nin Farklı Browswerlardaki Mobil Ve Masaüstü Web Performans Analizi

Browser	Strategy	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time
Chrome	desktop	0.7 s	1.9 s	1.8 s	1.452	10 ms
Mozilla Firefox	desktop	0.7 s	1.9 s	1.8 s	1.452	10 ms
Edge	desktop	0.8 s	1.4 s	1.4 s	1.452	0 ms
Chrome	mobile	2.9 s	3.6 s	7.0 s	1.014	190 ms
Mozilla Firefox	mobile	2.9 s	3.6 s	7.0 s	1.014	190 ms
Edge	mobile	2.9 s	3.6 s	7.0 s	1.014	190 ms

Kaynak: Yazar tarafından oluşturulmuştur.

Lit kütüphanesine ait yapılan web sitesi performans testleri üstteki tabloda detaylı olarak gösterilmiştir. Testler üç farklı browserda hem mobil hem de masaüstü ortam için yapılmış. Web performansta kullanılan en önemli 5 veri olan First Contentful Paint, Speed Index, Largest Contentful Paint, Cumulative Layout Shift ve Total Blocking Time metriklerinin verileri analiz sonucunda next.js için kaydedilmiştir ve tabloda gösterilmiştir.



Şekil 26. Lit Kütüphanesi ile Geliştirilen Web Sitesinin Performans Analizi

Kaynak: Yazar tarafından oluşturulmuştur.

Grafikte, Lit kütüphanesi ile geliştirilen web sitesinin performansı farklı tarayıcılar ve cihazlar üzerinde analiz edilmiştir. Chrome ve Mozilla Firefox masaüstü tarayıcılarında First Contentful Paint (FCP) ve Speed Index değerleri 1.0 saniyenin altında olup, Largest Contentful Paint (LCP) değerleri yaklaşık 3.0 saniye olarak ölçülmüştür. Mobil tarayıcılarda ise Chrome ve Mozilla Firefox'ta LCP değerleri yaklaşık 6.0 saniye ile en yüksek seviyededir, bu da mobil cihazlarda daha uzun yükleme sürelerine işaret etmektedir. Edge masaüstü ve mobil tarayıcılarında ise LCP değerleri 3.0 saniye civarında olup, FCP ve Speed Index değerleri 1.0 saniyenin altındadır. Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) değerleri tüm tarayıcılarda oldukça düşüktür, bu da sayfa stabilitesinin iyi olduğunu göstermektedir. Genel olarak, Lit kütüphanesi ile geliştirilen web siteleri düşük CLS ve TBT değerleri ile kullanıcı deneyimini olumlu etkilerken, özellikle mobil cihazlarda yüksek LCP değerleri performansı olumsuz etkilemektedir.

4.1.2. Javascript Kütüphanelerinin Web Performans Metrikleri Açısından Karşılaştırılması

Web performans değerlendirmesi yaparken baktığımız metriklerden ilki olan First Contentful Paint (FCP), kullanıcının tarayıcıda içeriği gördüğü ilk andır. Bu metrik ile kullanıcı aslında sayfanın yüklenmeye başladığını anlamış olur. FCP metriği kullanıcının sayfanın yüklenmeye başladığını anlaması için geçen süreyi göstermektedir.

İkinci metrik olan Speed Index bize sayfanın içeriğinin görsel olarak ne kadar hızlı yüklendiğini göstermektedir. Bu metrik ile sayfanın ne kadar hızlı bir şekilde görsel olarak tamamlandığı ölçülür. Speed Index değerinin düşük olması bize daha hızlı bir kullanıcı deneyimini gösterir.

Üçüncü metrik olan Largest Contentful Paint (LCP), kullanıcının ana içeriği gördüğü andır. Bu web sitesine göre büyük bir metin paragrafı veyahut büyük bir görüntü olabilir. LCP, kullanıcıların sayfanın yüklenmesinin büyük ölçüde tamamlandığını hissettiği süreyi ölçer. LCP değeri, sayfa performansını ve algılanan hızı anlamak için oldukça önemlidir.

Dördüncü metrik olan Cumulative Layout Shift(CLS), web sayfasının tamamen etkileşimli hale geldiği andır. Yani, sayfa yüklenmesi tamamlandıktan sonra kullanıcı etkileşimlerine (tıklama, kaydırma vb.) yanıt vermeye başladığı süredir. Düşük bir CLS değeri, kullanıcı deneyiminin daha akıcı olmasını sağlar.

Beşinci metrik ise Total Blocking Time (TBT)'dir. Bu metrik sayfanın etkileşimli hale gelene kadar kullanıcı etkileşimlerini engelleyen toplam süreyi ölçer. TBT, First Contentful Paint (FCP) ile Time to Interactive (TTI) arasında geçen süredeki uzun görevleri (long tasks) ölçer. Yüksek bir TBT, kullanıcı deneyimini olumsuz etkileyebilir.

4.2. Bulgular

Web performans analizi uygulamasının verdiği web metrikleri analiz sonuçları çeşitli açılardan karşılaştırılmıştır.

4.2.1.Web Performans Analizi: Farklı JavaScript Kütüphanelerinin Karşılaştırılması

Bu çalışmada, farklı JavaScript kütüphaneleri (Angular, Nuxt, Astro, SolidJS, Svelte, Enhance, Lit) kullanılarak film izleme amacıyla geliştirilen web sitelerinin performans analizleri yapılmıştır. İnceleme, First Contentful Paint (FCP), Speed Index, Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) ve Total Blocking Time (TBT) gibi metrikler üzerinden gerçekleştirilmiştir. Bu metrikler, web sitesi performansının kullanıcı deneyimi üzerindeki etkilerini değerlendirmek için kritik öneme sahiptir.

First Contentful Paint (FCP)

FCP, sayfanın ilk içerikli ögesinin görüntülenme süresini ölçer. En iyi performansı, Chrome masaüstü tarayıcısında Angular, Nuxt, Astro ve SolidJS kütüphaneleri göstermiştir; FCP değerleri 0.5 saniyenin altındadır. Buna karşın, mobil tarayıcılarda özellikle Nuxt ve Lit kütüphaneleri, 1.0 saniyenin üzerinde FCP değerleriyle en kötü performansı sergilemiştir. FCP değerlerini iyileştirmek için resim ve video dosyalarını optimize etmek, gereksiz JavaScript dosyalarını minify etmek ve tarayıcı önbellekleme kullanmak önemlidir. Ayrıca, kritik CSS'in inline olarak eklenmesi, sayfa yükleme hızını artırabilir.

Speed Index

Speed Index, sayfanın görsel içeriğinin ne kadar hızlı yüklendiğini ölçer. En iyi performans, Chrome ve Firefox masaüstü tarayıcılarında Angular ve SolidJS kütüphaneleri tarafından sağlanmış; bu değerler 0.5 saniyenin altında kalmıştır. Mobil tarayıcılarda ise Nuxt ve Lit kütüphaneleri, 2.0 saniyenin üzerindeki Speed Index değerleriyle en kötü performansı göstermektedir. Speed Index değerlerini iyileştirmek

için render engelleyici kaynakları azaltmak, CDN kullanarak içerik dağıtımını hızlandırmak ve lazy loading tekniklerini uygulamak yararlı olacaktır.

Largest Contentful Paint (LCP)

LCP, sayfadaki en büyük görsel veya metin bloğunun yüklenme süresini ölçer. Chrome ve Firefox masaüstü tarayıcılarında Angular ve SolidJS kütüphaneleri, 1.5 saniyenin altındaki LCP değerleriyle en iyi performansı göstermiştir. Buna karşın, mobil tarayıcılarda özellikle Nuxt ve Lit kütüphaneleri, 6.0 saniyeye kadar çıkan LCP değerleriyle en kötü performansı sergilemektedir. LCP değerlerini iyileştirmek için büyük resim ve video dosyalarını optimize etmek, lazy loading uygulamak, sunucu yanıt sürelerini iyileştirmek ve önemli içerikleri ilk etapta yüklemek gereklidir.

Cumulative Layout Shift (CLS)

CLS, sayfa yüklenirken yaşanan düzen kaymalarını ölçer. Tüm kütüphaneler CLS açısından iyi performans göstermekte; değerler genellikle 0.1'in altındadır. Bu durum, sayfa stabilitesinin iyi olduğunu ve kullanıcı deneyimini olumsuz etkilemediğini göstermektedir. CLS değerlerini düşük tutmak için görseller ve reklamlar için boyutlar belirlemek, dinamik içerikleri önceden belirlenmiş boyutlara yerleştirmek ve font yükleme stratejilerini optimize etmek önemlidir.

Total Blocking Time (TBT)

TBT, kullanıcı etkileşimlerini engelleyen süreyi ölçer. Tüm kütüphanelerde TBT değerleri oldukça düşük olup, genel olarak 0.1 saniyenin altındadır. Bu durum, kullanıcı etkileşimlerinin engellenmediğini ve sayfanın hızlı yanıt verdiğini göstermektedir. TBT değerlerini düşük tutmak için JavaScript yürütme sürelerini optimize etmek, ağır görevleri daha küçük parçalara bölmek ve tarayıcıdan yanıt süresini hızlandırmak için kodu asenkronize etmek gereklidir.

4.2.2. Genel Durum ve İyileştirme Önerileri

Farklı JavaScript kütüphaneleri arasında Angular ve SolidJS, masaüstü performansı açısından en iyi sonuçları göstermektedir. Mobil performansta ise

özellikle Nuxt ve Lit kütüphaneleri iyileştirilmesi gereken alanlara sahiptir. Performans iyileştirmeleri için büyük medya dosyalarını optimize etmek, lazy loading tekniklerini kullanmak, render engelleyici kaynakları azaltmak, CDN kullanmak ve kritik CSS'i inline olarak eklemek gibi stratejiler uygulanabilir. Bu önlemler, web sitelerinin yükleme sürelerini kısaltarak kullanıcı deneyimini önemli ölçüde artıracaktır.

Bu analiz, web performansının kütüphaneye, tarayıcıya ve cihaza göre önemli ölçüde değişiklik gösterdiğini ortaya koymaktadır. Kullanıcı deneyimini iyileştirmek ve web sitelerinin performansını artırmak için belirtilen stratejilerin uygulanması, özellikle mobil cihazlarda yüksek LCP değerleri gibi olumsuz etkenlerin azaltılmasına yardımcı olacaktır.

4.2.3.Kullanıcı Deneyimi Açısından Farklı JavaScript Kütüphaneleri ile Geliştirilen Web Siteleri

Web performansı, kullanıcı deneyimini doğrudan etkileyen en önemli faktörlerden biridir. Farklı JavaScript kütüphaneleri kullanılarak geliştirilen web sitelerinin performansını analiz ederek, kullanıcıların bu sitelerle etkileşimlerinde yaşayacakları deneyimleri değerlendirebiliriz.

Angular kütüphanesiyle geliştirilen film sitesini masaüstü performans analizi açısından ele alacak olursak, angular kütüphanesi, masaüstü tarayıcılarında oldukça iyi performans gösterir. FCP ve Speed Index değerleri 0.5 saniyenin altında olup, LCP değeri yaklaşık 1.5 saniyedir. Bu, kullanıcıların sayfanın içeriğini hızlı bir şekilde görmeye başlamalarını ve büyük içeriklerin de hızla yüklenmesini sağlar. Düşük CLS ve TBT değerleri, sayfa stabilitesinin ve etkileşim hızının iyi olduğunu gösterir. Mobilde ise LCP değerleri biraz daha yüksektir, ancak genel olarak performans kabul edilebilir seviyededir.

Nuxt kütüphanesinde, masaüstü tarayıcılarında iyi performans göstermekte, ancak mobil cihazlarda sorunlar yaşanmaktadır. FCP ve Speed Index değerleri 0.5 saniyenin altındadır, bu da hızlı başlangıç yükleme sürelerini işaret eder. Ancak, mobilde LCP değerleri 6.0 saniyeye kadar çıkabilmektedir, bu da kullanıcılar için uzun

bekleme sürelerine yol açar. Mobil performansın iyileştirilmesi için resim ve video optimizasyonu, lazy loading teknikleri ve sunucu yanıt sürelerinin iyileştirilmesi gereklidir.

Astro kütüphanesiyle yapılan web sitesinin performans analizinde, masaüstü tarayıcılarında dengeli bir performans sergilemektedir. FCP ve Speed Index değerleri 0.5 saniyenin altında, LCP değeri yaklaşık 1.5 saniyedir. Bu, kullanıcıların hızlı bir şekilde içerikle etkileşime girmelerini sağlar.Mobil cihazlarda LCP değerleri biraz daha yüksek olup, bu alanda iyileştirme yapılabilir. Ancak genel kullanıcı deneyimi olumludur.

SolidJS, masaüstü tarayıcılarında mükemmel performans gösterir. FCP ve Speed Index değerleri 0.5 saniyenin altında olup, LCP değeri 1.5 saniye civarındadır. Düşük CLS ve TBT değerleri, sayfanın stabil ve hızlı yanıt verdiğini gösterir. Mobilde yüksek LCP değerleri gözlemlenmiştir, bu nedenle medya dosyalarının optimizasyonu ve lazy loading gibi tekniklerin uygulanması faydalı olacaktır.

Svelte kütüphanesi, masaüstü tarayıcılarında oldukça iyi performans göstermektedir. FCP ve Speed Index değerleri 0.5 saniyenin altında, LCP değeri ise yaklaşık 1.0 saniyedir. Bu, kullanıcıların hızlı bir şekilde içerikle etkileşime girmelerini sağlar. Mobil cihazlarda ise LCP değerleri 2.0 saniyenin üzerindedir, bu da iyileştirilmesi gereken bir alan olarak öne çıkmaktadır.

Enhance kütüphanesi, masaüstü tarayıcılarında iyi performans göstermekte olup, FCP ve Speed Index değerleri 0.5 saniyenin altındadır. LCP değeri yaklaşık 1.5 saniyedir, bu da kullanıcı deneyimini olumlu etkiler. Mobilde ise LCP değerleri daha yüksektir ve bu durum kullanıcıların sayfanın tamamen yüklenmesi için daha uzun süre beklemesine neden olabilir. Medya optimizasyonu ve lazy loading teknikleri uygulanarak bu değerler iyileştirilebilir.

Lit kütüphanesi, masaüstü tarayıcılarında en yüksek LCP değerlerine sahip olup, bu değerler yaklaşık 3.0 saniyedir. Bu, kullanıcıların büyük içeriklerin yüklenmesini beklerken daha uzun süre beklemesine neden olur. Mobilde ise Lit kütüphanesi 6.0 saniyeye kadar çıkan LCP değerleri ile en kötü performansı göstermektedir. Bu durum, kullanıcı deneyimini olumsuz etkiler ve sayfanın yüklenme

süresini önemli ölçüde uzatır. Bu nedenle, resim ve video dosyalarının optimizasyonu, lazy loading uygulamaları ve sunucu yanıt sürelerinin iyileştirilmesi gereklidir.

4.2.4.Kullanıcı Açısından Genel Bulgu ve Yorumlar

Genel olarak, kullanıcılar hızlı yükleme süreleri, sayfa stabilitesi ve hızlı etkileşim süreleri beklemektedir. Angular ve SolidJS gibi kütüphaneler masaüstü performansı açısından en iyi sonuçları gösterirken, Nuxt ve Lit kütüphaneleri özellikle mobil cihazlarda iyileştirilmesi gereken alanlara sahiptir. Performans iyileştirmeleri için büyük medya dosyalarının optimizasyonu, lazy loading tekniklerinin kullanılması, render engelleyici kaynakların azaltılması, CDN kullanımı ve kritik CSS'in inline olarak eklenmesi gibi stratejiler uygulanabilir. Bu önlemler, web sitelerinin yükleme sürelerini kısaltarak kullanıcı deneyimini önemli ölçüde artıracaktır.

SONUÇ

Bu çalışmada, NextJS, Angular, Nuxt, Astro, Solid JS, Svelte, Enhance ve Lit Javascript kütüphaneleri ile yapılan film izleme sitelerinin performans analizi için web performans analiz uygulaması yapılmıştır. Bu analiz uygulaması ile kapsamlı olarak web sitesi performansı Chrome, Edge ve Firefox browserlarında mobil ve masaüstü olarak ayrı ayrı test edilmiş, web sitesi performanslarının metrik sonuçlarından First Contentful Paint (FCP), Total Blocking Time (TBT), Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) ve Speed Index incelenmiş ve karşılaştırılmıştır. Bu bulgular sonucunda her birinin web performans özellikleri detaylı bir şekilde ele alınmıştır. Her birinin mobil ve masaüstü çeşitli verileri, sayfa yüklenme hızları, First Contentful Paint'leri, web performans tahminleri çeşitli browserlardaki durumları değerlendirilmiştir. Bu durumda yapılabilecek iyileştirmeler ve geliştirmeler anlatılmıştır.

Sonuç olarak, web sitesi geliştirmelerinde optimizasyon ve performans faktörleri dikkate alınmalıdır. Her kullanılan teknolojinin avantajları ve sınırlamaları vardır ve web sitesinin gereksinimlerine göre en uygun programlama dilinin seçilmesi önemlidir. Bu çalışma, farklı javascript teknolojileriyle yazılmış aynı film sitelerinin web performanslarını incelemiş, birbiriyle karşılaştırmış ve browser, masaüstü ve mobil verileri ile değerlendirerek web performans iyileştirmeleri için önemli bir katkı sağlamıştır. Böylece gelecekteki çalışmalarda, performans açısından daha fazla detaylı analizler yapılabilir ve yeni nesil web performans geliştirmelerinin incelenmesine odaklanılabilir.

Sonraki çalışmalar, web performans, kullanıcı odaklı web sitesi geliştirmeleri, page Speed , browserların performans üzerindeki etkisi, ve kullanıcı dostu web siteleri geliştirilerek web teknolojilerinin optimize edilmesine odaklanmalıdır. Bu, hem ön yüz teknolojileri hem de kullanılabilirlik açısından web sitesi performans iyileştirme süreçlerini ileriye taşıyacak ve daha geniş kullanım alanlarına olanak sağlayacaktır. Bu öneriler, performans odaklı web sitesi geliştirmelerinin gelecekteki gelişim yönlerini şekillendirebilir ve daha güvenli, verimli ve kullanıcı dostu çözümlere yol açabilir.

KAYNAKÇA

Bakioğlu, O. (2012). *Bulut bilişim ve teknolojileri*. [Yüksek Lisans Tezi].Okan Üniversitesi.

Durgut, M., ve Çakır, E. (2013). *Web uygulamalarında performans optimizasyonu*. Akademik Bilişim Konferansı.

Fielding, R. (2000). *Architectural styles and the design of network-based software architectures.* [Doktora tezi]. University of California.

Galletta, D. F., Henry, R. M., McCoy, S., ve Polak, P. (2004). Web site delays: How tolerant are users?. *Journal of the Association for Information Systems, 5(1)*, 1-28.

Hacıoğlu, T., ve Güneş, A. (2019). *Çok katmanlı çok kullanıcılı web sistemlerinde performans analizi ve bir uygulama*. *Engineering Sciences (NWSAENS), 14(3)*, 88-103. https://doi.org/10.12739/NWSA.2019.14.3.1A0434

Jackson, W. (2016). *HTML5 history: The past and future of HTML markup*. SpringerLink.

Jain, R. (1991). *The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling*. Wiley.

Karadağ, E., Aypay, A., ve Baloğlu, N. (2010 Mayıs). Eğitim yönetimi araştırmalarına analitik bir bakış: kuram ve uygulamada eğitim yönetimi dergisinin analizi. S. Özdemir (Başkan). 5. *Ulusal Eğitim Yönetimi Kongresi*, Gazi Üniversitesi, Ankara.

Kılıç, S., ve Çakaroz, K. M. (2021). Web sitesi tasarım özelliklerinin web site performansı üzerindeki etki düzeylerinin incelenmesine dair deneysel bir çalışma. *Girişimcilik İnovasyon ve Pazarlama Araştırmaları Dergisi, 5(10)*, 98-112. https://doi.org/10.31006/gipad.993545

Kohavi, R., Longbotham, R., Sommerfield, D., ve Henne, R. M. (2009). *Controlled experiments on the web: Survey and practical guide.* Data Mining and Knowledge Discovery.

Lightner, N. J., Bose, R., ve Salvendy, G. (1996). *What is wrong with the world wide web?: A diagnosis of some problem areas.* Ergonomics, 39(8), 947-964.

Manninen, A. (2020). *Impact of web performance on user experience*. University of Oulu.

Miller, G. A. (1968). *Psychology and information*. American Psychologist, 23(5), 381.

Nielsen, J. (1999a). *Designing web usability: The practice of simplicity*. New Riders Publishing.

Osmani, A. (2012). *Learning javascript design patterns*. O'Reilly Media.

Osmani, A. (2018). *Progressive web apps*. O'Reilly Media.

Osmani, A. (2020). *Patterns for large-scale javascript application architecture*. 15 Nisan 2024 tarihinde addyosmani.com adresinden edinilmiştir.

Peña-Ortiz, R. (2013). *Accurate workload design for web performance evaluation*. Universitat Politècnica de València.

Pena-Ortiz, R., ve Andere, M. (2018). *Accurate workload design for web performance evaluation*. ResearchGate.

Post, L. (2018). *API performance: optimizing for efficiency and speed*. O'Reilly Media.

Ramsay, J., Barbesi, A., ve Preece, J. (1998). *A psychological investigation of long retrieval times on the world wide web*. Interacting with Computers, 10(1), 77-86.

Sears, A., Jacko, J. A., ve Borella, M. S. (1997). *Internet delay effects: How users perceive quality, organization, and ease of use of information*. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Vol. 41, No. 5, pp. 360-364). SAGE Publications.

Schneider, B. (2003). *Performance testing: Methodologies and tools*. Addison-Wesley.

Snook, D. (2019). *CSS ile stil yönetimi: En iyi uygulamalar ve teknikler*. Google Books.

Tuch, A. N., Bargas-Avila, J. A., ve Opwis, K. (2015). *Effects of different website designs on first impressions, aesthetic judgements, and memory performance after short presentation.* ResearchGate.

Ünlü, H., Bilgin, B., ve Demirörs, O. (2021). *Türkiye'deki yazılım organizasyonlarının mikroservis tabanlı mimaride uyguladığı analiz ve tasarım yöntemleri üzerine bir araştırma.* EMO Bilimsel Dergi, 11(22), 47-54.

Wang, P. (2019). *Modern web development*. O'Reilly Media.

WebPageTest. (2023). *Webpagetest documentation.* Retrieved from

Weyuker, E. J. (1982). *On testing non-testable Programs*. Computer Journal, 25(4), 465-470.

Yardımcıoğlu, G. (2019). *4G-5G uyumlu mobil sistemler için anten tasarımı*. Ege Üniversitesi.

Yılmaz, B. (2017). *Web Tabanlı Uygulamalarda Performans ve Güvenlik* [Yüksek Lisans Tezi]. Karadeniz Teknik Üniversitesi.

Yüzgeç, U., ve Günel, A. (2014). *Üniversitelere yönelik bir veri merkezinin CFD analizi ile iklimlendirme planlaması*. Sakarya Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 18(3), 235-241. https://doi.org/10.16984/saufbed.94968

İnternet Kaynağı

Google Developers. (2023). *Lighthouse.* Retrieved from 2 Mart 2024 tarihinde https://developers.google.com/web/tools/lighthouse adresinden edinilmiştir.

New Relic. (2023). *Application performance monitoring (APM).* Retrieved from 14 Nisan 2024 tarihinde https://newrelic.com/products/application-monitoring adresinden edinilmiştir.

Osmani, A. (2018). *The cost of javascript in 2019.* Google Developers. 14 Nisan 2024 tarihinde https://developers.google.com/web/tools/lighthouse adresinden edinilmiştir.