T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BLOK ZİNCİR TABANLI MEDİKAL KAYITLARIN TUTULMASI VE YAPAY ZEKÂ TABANLI SAĞLIK KARAR DESTEK SİSTEMİ GELİŞTİRİLMESİ

Emre Öcel
Deniz Bektaş

Tez Danışmanı
Prof. Dr. Mehmet Karaköse

BİTİRME TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ELAZIĞ - 2025

FIRAT ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BLOK ZİNCİR TABANLI MEDİKAL KAYITLARIN TUTULMASI VE YAPAY ZEKÂ TABANLI SAĞLIK KARAR DESTEK SİSTEMİ GELİŞTİRİLMESİ

Emre Öcel

Deniz Bektaş

BİTİRME TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Bu bitirme tezi …../……/2025 tarihinde, aşağıda belirtilen jüri tarafından oybirliği/oyçokluğu ile başarılı/başarısız olarak değerlendirilmiştir.

(imza)	(imza)	(imza)
Prof. Dr. Mehmet Karaköse	Prof. Dr. İlhan AYDIN	Dr.Öğr.Üyesi Hasan YETİŞ

ÖZGÜNLÜK BİLDİRİMİ

Bu çalışmada, başka kaynaklardan yapılan tüm alıntıların, ilgili kaynaklar referans gösterilerek açıkça belirtildiğini, alıntılar dışındaki bölümlerin, özellikle projenin ana konusunu oluşturan teorik çalışmaların ve yazılım/donanımın kendi tarafımızdan yapıldığını bildiririz.

Fırat Üniversitesi
07.07.2025
Bilgisayar Mühendisliği
23119 Elazığ

Öğrencilerin Adı Soyadı

Emre Öcel

Deniz Bektaş

BENZERLİK BİLDİRİMİ

DenizBektaşBitirme.docx

ORIGINALITY REPORT

3%	1%	0%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Fırat Üniversitesi Student Paper	2%
2	Submitted to Beykent Universitesi Student Paper	<1%
3	Submitted to TechKnowledge Turkey Student Paper	<1%
4	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Student Paper	<1%
5	Submitted to Bahcesehir University Student Paper	<1%
6	Submitted to Gazi University Student Paper	<1%
7	d-nb.info Internet Source	<1%
8	www.essays.se Internet Source	<1%
9	www.frontiersin.org Internet Source	<1%

TEŞEKKÜR

Bu bitirme projesi sürecinde, akademik bilgi birikimimiz kadar araştırma, geliştirme ve analiz becerilerimizi de derinleştirme fırsatı bulduk. Çalışmamızın her aşamasında bizi yönlendiren, değerli katkılarıyla yol gösteren danışmanımız Prof. Dr. Mehmet KARAKÖSE'ye en içten teşekkürlerimizi sunarız.

Ayrıca, eğitim hayatımız boyunca bilgi ve deneyimlerini bizlerle paylaşan tüm hocalarımıza ve sürecin her adımında bize destek olan ailelerimize ve arkadaşlarımıza da şükranlarımızı ifade ederiz.

İÇİNDEKİLER

Sayfa No

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMA LİSTESİ

AI: Artificial Intelligence

API: Application Programming Interface

CSV: Comma-Separated Values

CSS: Cascading Style Sheets

CORS: Cross-Origin Resource Sharing

ETH: Ethereum

EVM: Ethereum Virtual Machine

GAN: Generative Adversarial Network

GPU: Graphics Processing Unit

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

ID: Identifier

IPFS: InterPlanetary File System

JSON: JavaScript Object Notation

JWT: JSON Web Token

LSTM: Long Short-Term Memory

ML: Machine Learning

NEJM: New England Journal of Medicine

NLP: Natural Language Processing

OBD: Open and Big Data

RF: Random Forest

SDK: Software Development Kit

TF: TensorFlow

UI: User Interface

UX: User Experience

VSCode: Visual Studio Code

XAI: Explainable Artificial Intelligence

HIS: Hospital Information System

ÖZET

Bu çalışma, blokzincir tabanlı bir medikal kayıt sistemi ile yapay zekâ destekli bir sağlık karar destek modülünün entegre biçimde geliştirilmesini konu almaktadır. Günümüz sağlık sistemlerinde bireylerin kişisel verilerinin merkezi sunucularda tutulması, veri gizliliği, şeffaflık ve yetkisiz erişim gibi önemli problemleri beraberinde getirmektedir. Ayrıca sağlık hizmetlerine erişimde yaşanan zorluklar, kullanıcıların semptomlarına hızlı ve güvenilir çözümler bulma ihtiyacını doğurmaktadır. Bu bağlamda geliştirilen sistem, hem verinin güvenli ve merkeziyetsiz şekilde saklanmasını hem de kullanıcıdan alınan semptom verilerinin analiz edilerek olası hastalıkların tahmin edilmesini ve önerilerin sunulmasını sağlamaktadır.

Sistem dört temel bileşenden oluşmaktadır: kullanıcı arayüzü (frontend), Web3 entegrasyonu ve MetaMask bağlantısı, Ethereum tabanlı akıllı kontrat altyapısı ve Flask üzerinden servis edilen Random Forest algoritmasına dayalı yapay zekâ tahmin modülü. Kullanıcılar MetaMask cüzdanları ile sisteme giriş yaparak yalnızca kendi verilerine erişebilmektedir. Kullanıcıdan alınan semptomlar ağırlıklandırılarak vektörleştirilmekte ve Random Forest algoritması ile işlenerek olası hastalık tahmin edilmekte, ardından sisteme entegre edilen öneri dosyası (precautions.json) üzerinden kullanıcıya özel koruyucu öneriler sunulmaktadır.

Akıllı kontratlar Solidity diliyle yazılmış olup, blokzincir üzerinde hasta adı, hastalık ve adres bilgisi gibi temel tıbbi verilerin şifreli ve güvenli biçimde saklanmasını sağlamaktadır. Sistemde ayrıca yönetici paneli (admin.html) aracılığıyla tüm kullanıcı kayıtları listelenebilmekte, düzenlenebilmekte ve silinebilmektedir. Her işlem, blokzincir üzerinde zaman damgası ile kayıt altına alınmakta ve olay tetikleme (event logging) mekanizması sayesinde kullanıcıya anlık geri bildirim sağlanmaktadır.

Deneysel sonuçlar, yapay zekâ modelinin doğruluk oranının %93, ortalama F1 skorunun 0.89 ve tahmin süresinin 0.38 saniye olduğunu ortaya koymuştur [4,6]. Blokzincir altyapısı ile yapılan işlemlerde veri bütünlüğü korunmuş, yetkisiz erişim önlenmiş ve sistemin her iki bileşeni senkronize şekilde çalışmıştır. Bu tez çalışması, bireysel sağlık verilerinin güvenli yönetimi ile yapay zekâ destekli erken teşhis sistemlerinin hibrit yapıda bir araya getirilerek, modern sağlık bilişimi alanına katkı sağlamayı hedeflemektedir.

ABSTRACT

This thesis presents the design and implementation of an integrated system combining a blockchain-based medical record infrastructure with an AI-powered health decision support module. In conventional healthcare systems, the storage of sensitive patient information on centralized servers poses significant risks related to data privacy, transparency, and unauthorized access. Furthermore, the increasing demand for accessible and timely preliminary health evaluations necessitates innovative solutions that enable secure data ownership and intelligent guidance. The proposed system addresses these issues by enabling decentralized data storage and providing disease predictions and preventive suggestions based on user-reported symptoms.

The architecture consists of four major components: a frontend web interface, Web3 integration via MetaMask, Ethereum-based smart contracts for on-chain storage, and a machine learning backend powered by a Random Forest classifier deployed via Flask. Users connect to the system using their MetaMask wallets, allowing access only to their personal records. Symptom inputs are weighted, transformed into feature vectors, and processed by the trained Random Forest model to predict probable diseases. Based on the prediction, custom recommendations are retrieved from a precaution database and delivered to the user in real time.

The smart contracts are developed in Solidity and enable secure, immutable storage of patient identifiers and diagnostic information directly on the blockchain. Additionally, an administrative interface allows authorized personnel to list, update, and delete user records. Every transaction is timestamped on-chain, and event logging mechanisms provide immediate feedback and traceability.

Experimental evaluations reveal that the AI model achieved 93% accuracy, a mean F1 score of 0.89, and an average prediction time of 0.38 seconds. The blockchain module successfully preserved data integrity, prevented unauthorized access, and maintained functional synchronization with the AI module. This thesis demonstrates a hybrid approach that advances the intersection of digital health and secure data technologies, contributing to the development of patient-centric, privacy-focused healthcare systems.

1. GİRİŞ

1.1 Proje Konusu

Bu projenin temel konusu, blokzincir teknolojisi ve yapay zekâ tabanlı algoritmaların sağlık bilişimi alanında bir araya getirilerek merkeziyetsiz, güvenli ve akıllı bir medikal kayıt sisteminin geliştirilmesidir [1, 2, 9]. Günümüz sağlık sistemlerinde hasta verilerinin merkezi sunucularda depolanması, bu verilerin manipülasyona açık olmasına, tek noktadan erişim zafiyetine ve mahremiyetin ihlali gibi önemli problemlere yol açmaktadır [30,32,36]. Ayrıca, sağlık hizmetlerinin dijitalleşmesi sürecinde, sadece verinin saklanması değil, aynı zamanda bu verinin anlamlı biçimde işlenerek karar destek sistemlerine katkı sağlaması da büyük önem taşımaktadır [31, 39].

Bu bağlamda, proje kapsamında iki temel bileşen tasarlanmıştır: Birincisi, Ethereum blokzinciri üzerine kurulu olan ve hasta verilerini akıllı kontratlar aracılığıyla kalıcı ve değiştirilemez biçimde saklayan bir merkeziyetsiz kayıt sistemi; ikincisi ise, hastanın girdiği semptomlara göre hastalık tahmini yapabilen ve uygun korunma önlemleri sunabilen bir yapay zekâ destekli karar destek modülüdür [30, 32, 35].

Blokzincir teknolojisi sayesinde, hasta verileri sadece kullanıcıya ait Ethereum adresi üzerinden erişilebilir hale getirilmiş, bu da veri mahremiyeti, güvenilirliği ve değiştirilemezliği açısından yüksek seviyede güvenlik sağlamıştır [2, 3, 35]. Sistem, kullanıcıların kendi verilerini girmelerine ve gerektiğinde sorgulamalarına olanak tanırken; aynı zamanda bir yönetici arayüzü üzerinden tüm hasta kayıtlarının görüntülenebilmesi, güncellenebilmesi ve silinebilmesi de sağlanmıştır [32,35,42]. Yönetici paneli ile ayrıca tüm hasta verilerinin tablo halinde listelenmesi ve pdf çıktısı alınması gibi fonksiyonel özellikler de uygulamaya dahil edilmiştir. Projenin ikinci ayağında ise, “Disease Symptom Prediction Dataset” adlı açık veri seti kullanılarak eğitilen Random Forest sınıflandırıcısı ile, hasta semptomları üzerinden yüksek doğrulukla hastalık tahmini yapılması hedeflenmiştir. Geliştirilen yapay zekâ modeli Flask tabanlı bir sunucu ile frontend arayüze bağlanmış ve kullanıcı tarafından girilen her semptom dizisi ayrı ayrı işlenerek özgün sonuçlar üretmesi sağlanmıştır [31, 35, 42]. Ayrıca tahmin edilen hastalıklara karşı alınması gereken önlemler sistem tarafından otomatik olarak sunulmuştur [16, 31, 42]. Tüm bu yapılar, modern sağlık bilişimi uygulamalarında karşılaşılan veri güvenliği, erişim denetimi, teşhis desteği ve kullanıcıya özel hizmet üretimi gibi temel gereksinimlere bütüncül bir çözüm üretmek üzere bir araya getirilmiştir [30, 37, 42]. Proje, yalnızca bir yazılım sistemi değil, aynı zamanda blokzincir ve yapay zekâ teknolojilerinin sağlık alanındaki işlevsel kullanımına dair örnek bir

mimari model sunmaktadır. Böylece gelecekte gerçek zamanlı sağlık sistemlerine entegre edilebilecek daha büyük ölçekli çalışmalara da temel teşkil etmektedir [33, 34, 42].

1.2 Proje Amacı

Bu projenin temel amacı, blokzincir teknolojisinin sunduğu güvenlik, bütünlük ve merkeziyetsizlik avantajlarından yararlanarak, sağlık bilişiminde hasta verilerinin güvenli, şeffaf ve bireye özgü biçimde saklanmasını ve yönetilmesini sağlamaktır. Bu doğrultuda geliştirilen sistem, Ethereum altyapısında çalışan akıllı kontratlar sayesinde kullanıcıların yalnızca kendi adresleri üzerinden işlem yapabilmesine olanak tanımakta, böylece veri sahipliği doğrudan zincir seviyesinde tanımlanmış olmaktadır. Ayrıca, hasta tarafından girilen semptomlara dayalı olarak hastalık tahmini yapabilen ve kullanıcıya bilimsel içerikli tıbbi öneriler sunabilen yapay zekâ destekli bir karar destek modülü entegre edilmiştir [11]. Bu modül, veri gizliliğine zarar vermeden öneri üretebilmekte ve model çıktısını sistemli biçimde sunarak karar süreçlerine katkı sağlamaktadır.

Günümüzde hasta kayıtlarının çoğunlukla merkezi sunucularda tutulduğu ve bu yapının güvenlik açıklarına zemin hazırladığı bilinmektedir. Literatürde, merkezi veri depolamanın sistem kırılganlığını artırdığı ve veri bütünlüğünü zayıflattığı vurgulanırken [25], yapay zekâ ile desteklenen karar mekanizmalarının özellikle semptom tabanlı durumlarda yeterince yaygınlaştırılamadığı görülmektedir. Safdar & Gabrael (2025) çalışmasında da belirtildiği gibi [30], blokzincir ve yapay zekânın birlikte çalıştığı sistemlerde uygulama seviyesinde karşılaşılan zorluklar sürdürülebilirlik açısından ele alınmalı ve bütünleşik mimari tercih edilmelidir. Bu proje, söz konusu iki temel sorunsalı – veri güvenliği ve akıllı karar desteği – tek bir mimari çatıda birleştirerek çözmeyi hedeflemektedir.

Projeye özel olarak belirlenen alt hedefler arasında; Ethereum blokzinciri üzerinde denetlenebilir ve değiştirilemez veri saklama yapısının oluşturulması, yalnızca kullanıcının kendi cüzdan adresi ile veri giriş ve sorgulama işlemlerinin gerçekleştirilmesi, Random Forest algoritmasıyla yüksek doğruluk oranında hastalık tahmini yapılması ve bu tahminlere göre symptom_precaution.csv veri kümesinden alınan önerilerin sistem tarafından otomatik sunulması yer almaktadır. Kullanıcı arayüzü, Ethers.js ve MetaMask ile Web3 uyumlu şekilde yapılandırılmış olup, hem kullanıcıların hem de yöneticilerin fonksiyonel olarak ayrıldığı bir yapıda tasarlanmıştır. Admin paneli aracılığıyla tüm kullanıcı verilerine erişim, güncelleme, silme ve PDF formatında çıktı alma gibi işlemler gerçekleştirilmekte; bu yapı özellikle kurumsal hastane sistemleri için denetim ve raporlama işlevi görmektedir. Sistem aynı zamanda ilerleyen aşamalarda laboratuvar sonuçları, görüntüleme verileri ve IoT tabanlı ölçümler gibi sağlık verisi türleri ile entegre edilebilecek ölçeklenebilir bir dijital sağlık altyapısı oluşturmayı da hedeflemektedir.

Tablo 1.1 Bu Tezin Proje Amacı ile Literatürdeki Yaklaşımların Karşılaştırması

Amaç/Kriter	Bu Tez (2025)	Kumar et al. (2021) [25]	Safdar & Gabrael (2025) [30]	Açıklama
Blokzincir ile Veri Güvenliği	Ethereum + akıllı kontratlar ile bireysel veri koruma ve adres temelli giriş	IPFS üzerinde saklama + hash kontrol	Blokzincir temelli kimlik doğrulama önerisi	Bu tez zincir-üstü işlem şeffaflığı sağlar; diğerleri daha çok zincir-dışı veri modeli sunar
Yapay Zekâ ile Hastalık Tahmini	Random Forest ile %93 doğruluk oranı; semptomlara göre öneri üretimi	AI kullanılmamış	AI modül önerisi var ancak uygulama yok	Bu tez, AI + tahmin + öneri sürecini çalışan sistem olarak sunar
Frontend ile Etkileşimli Web3 Arayüzü	MetaMask, Ethers.js ile hasta ve admin panelleri	Arayüz yok	Arayüz tanımlı değil, sadece mimari öneri var	Kullanıcı tabanlı etkileşimli tasarım bu tezde ön planda
Kurumsal Ölçeklenebilirlik Hedefi	Admin paneli ile çok kullanıcıya ölçeklenebilir yapı önerilmiş	Ölçeklenebilirlik vurgulanmamış	API ile genişletme önerilmiş	Bu tezde teknik büyüme hedefleri somut biçimde sunulmuş

1.3 Mevcut Zorluklar ve İhtiyaçlar

Sağlık bilişimi sistemleri, son yıllarda dijitalleşme sürecinin hızlanmasıyla birlikte ciddi bir dönüşüm süreci geçirmektedir. Ancak bu dijital dönüşüm, yalnızca olanaklar değil, aynı zamanda bir dizi teknik, yapısal ve güvenlik temelli zorlukları da beraberinde getirmiştir. Özellikle hasta verilerinin dijital ortamda saklanması ve paylaşılması, beraberinde hem veri mahremiyeti hem de veri bütünlüğü açısından çeşitli tehditleri gündeme getirmektedir. Bu bağlamda, hem mevcut sistemlerde karşılaşılan eksiklikler hem de sağlık hizmetlerinin dijitalleşme sürecindeki temel ihtiyaçlar bu çalışmanın çıkış noktasını oluşturmuştur.

Mevcut sistemlerde ön plana çıkan başlıca zorluklar aşağıdaki gibi özetlenebilir:

- **Veri Güvenliği ve Bütünlüğü Sorunları:** Merkezi sunucular üzerinde barındırılan sağlık verileri, dış saldırılar, kötü niyetli erişimler veya içerden yapılan ihlaller sonucunda değiştirilebilir ya da silinebilir durumdadır. Geleneksel sistemlerde herhangi bir log ya da zaman damgası doğrulaması bulunmadığından, bu tür müdahaleler çoğunlukla fark edilmemekte veya geç tespit edilmektedir.

- **Mahremiyet ve Erişim Denetimi Eksiklikleri:** Hasta verilerinin kimler tarafından erişilebileceği, ne amaçla kullanılacağı ve ne süreyle saklanacağı gibi konular birçok ülkede yasal olarak düzenlenmiş olsa da, pratikte bu kontrollerin şeffaf şekilde uygulanabildiği sistem sayısı oldukça azdır. Özellikle hastaların kendi verileri üzerinde tam yetkiye sahip olmamaları önemli bir sorun olarak görülmektedir.

- **Tek Nokta Arızası (Single Point of Failure):** Merkezi yapılar, bir sistem bileşeninin (sunucu, ağ, veritabanı vb.) çökmesi durumunda tüm sistemin işlevini kaybetmesine neden olabilmektedir. Felaket kurtarma senaryolarının uygulanabilirliği genellikle zayıf kalmaktadır.

- **Gerçek Zamanlı Karar Destek Eksikliği:** Günümüzde birçok hasta, internet üzerinden semptomlarına yönelik bilgi aramakta; ancak bu aramalar genellikle tıbbi bilgi doğruluğu olmayan, genel-geçer içeriklere dayanmaktadır. Sağlık profesyonelleri ise yoğun iş yükü altında, hızlı ve güvenilir teşhis desteğine ihtiyaç duymaktadır. Mevcut sistemler bu ihtiyaca cevap verecek düzeyde dinamik ve semptom odaklı yapay zekâ çözümlerini içermemektedir.

- **Veri Sahipliğinin Hasta Tarafına Geçmemesi:** Bugünkü sağlık sistemlerinde verinin mülkiyeti genellikle sağlık kurumuna veya hizmet sağlayıcıya aittir. Bu durum, hastaların verileri üzerinde denetim ve kullanım hakkı sahibi olmalarını engellemektedir. Tüm bu zorlukların çözüme kavuşturulması için aşağıdaki temel ihtiyaçların karşılanması

gerekmektedir. Verinin değiştirilmesini ve silinmesini teknik olarak imkânsız kılan bir yapı (örneğin blokzincir) ile saklanması. Hasta ve sağlık personeli arasında tanımlı, izlenebilir ve yetkilendirilmiş veri erişim mekanizmalarının kurulması, Hastaların verileri üzerindeki sahiplik ve erişim haklarının doğrudan kendilerine devredilmesi, semptom tabanlı, veriye dayalı ve yüksek doğruluk sağlayan yapay zekâ modülleri ile klinik karar destek sürecinin güçlendirilmesi, Tüm bu sistemlerin kullanıcı dostu, erişilebilir ve farklı sağlık altyapılarına entegre edilebilir biçimde sunulması.

Bu çalışma, yukarıda özetlenen mevcut sorunlara cevap verebilecek bütüncül bir sistem mimarisi tasarlamayı ve sağlık hizmetlerinde teknolojik dönüşümün önünü açacak bir referans uygulama geliştirmeyi hedeflemektedir. Böylece hem teknik hem de etik açıdan sürdürülebilir ve güvenilir dijital sağlık platformlarının kurulmasına katkı sağlanması amaçlanmaktadır.

1.4 Projenin Önemi

Günümüzde sağlık hizmetlerinin dijitalleşme süreci, veri güvenliği, erişilebilirlik, doğru teşhis ve hasta merkezli hizmet üretimi gibi kritik parametreleri beraberinde getirmektedir. Ancak mevcut sistemler çoğunlukla merkezi sunuculara dayandığından, hasta verilerinin korunması, bütünlüğünün sağlanması ve yalnızca yetkili kullanıcılar tarafından erişilebilmesi gibi temel gereksinimler yeterince karşılanamamaktadır. Ayrıca, artan hasta yükü karşısında sağlık profesyonellerinin hızlı, doğru ve kişiselleştirilmiş teşhis süreçlerine olan ihtiyacı her geçen gün daha da belirginleşmektedir. Bu bağlamda, blokzincir ve yapay zekâ teknolojilerinin entegrasyonu, hem hasta hem de sağlık hizmet sağlayıcıları açısından önemli avantajlar sunan yeni nesil çözümlerin önünü açmaktadır.

Bu projenin en temel önemi, yukarıda belirtilen problemlere entegre ve yenilikçi bir yaklaşım getirerek; merkeziyetsiz, güvenli ve akıllı bir medikal kayıt sistemi ile kullanıcıya özgü teşhis önerileri sunan bir yapay zekâ modülünü aynı platformda birleştirmiş olmasıdır. Sağlık bilişimi alanında bu iki teknolojinin birlikte değerlendirilmesi, hem bilimsel hem de uygulamalı açılardan çığır açıcı nitelikte bir mimari sunmaktadır.

Projenin önemini ortaya koyan başlıca yönler şu şekilde özetlenebilir:

Ethereum blokzinciri üzerine inşa edilen akıllı kontratlar sayesinde, hasta verileri yalnızca ilgili kullanıcının kontrolünde olacak şekilde değiştirilmesi mümkün olmayan bloklara kaydedilmektedir. Bu durum, sağlık verilerinin yasal ve etik çerçevede saklanmasına olanak tanımakta, güvenlik açıklarını minimize etmektedir. Yapay zekâ destekli karar destek modülü, klasik kural tabanlı sistemlerin ötesine geçerek istatistiksel öğrenmeye dayalı tahminleme yapabilmektedir. Bu da özellikle tanı süreçlerinde sağlık çalışanlarının iş yükünü azaltarak

daha hızlı ve doğru kararlar alabilmelerine imkân sağlar. Semptomlara göre tahminlenen hastalıkların yanında, hastaya özel önerilerin de sunulması, bu sistemi basit bir tahmin algoritmasından öteye taşımaktadır. Sistemde her kullanıcı yalnızca kendi verisini girebilmekte ve kendi verisine erişebilmektedir. Bu model, bireyin kendi sağlık verisi üzerinde tam denetim sahibi olmasını sağladığı için "kişisel veri sahipliği" ilkesine doğrudan hizmet etmektedir. Proje, sadece bireysel düzeyde değil, aynı zamanda yönetici paneli ile kurumsal sağlık sistemlerine de hitap etmektedir. Kayıtları topluca görüntüleyebilme, güncelleme, silme, çıktı alma gibi işlevler sayesinde sistem, klinik bilgi yönetim sistemlerine (KBS) entegre edilebilir niteliktedir. Blokzincir ve yapay zekâ teknolojilerinin bir arada kullanıldığı bu sistem mimarisi, gelecekte yapılacak çalışmalara temel oluşturabilecek, farklı modüllerle (laboratuvar sonuçları, görsel tanı sistemleri vb.) genişletilebilecek esnek bir altyapı sunmaktadır.

Sonuç olarak bu proje, yalnızca teknik bir uygulama geliştirmekten öte; günümüz sağlık bilişimindeki güvenlik, doğruluk, karar desteği ve mahremiyet gibi çok boyutlu problemlere bütünsel bir çözüm önerisi sunmaktadır. Böylece hem akademik literatüre katkı sağlamakta, hem de sağlık sistemlerinin dijitalleşme sürecinde uygulanabilir, sürdürülebilir ve geleceğe dönük bir model ortaya koymaktadır.

1.5 Problem Tanımı ve Gelişen İhtiyaçlar

Günümüz sağlık sistemleri, teknolojik gelişmelerin etkisiyle dijitalleşme sürecini hızla benimsemekte ve bu doğrultuda hasta kayıtlarının elektronik ortamlarda tutulduğu, teşhis ve tedavi süreçlerinin yazılımlar aracılığıyla desteklendiği yeni nesil uygulamalar yaygınlaşmaktadır. Ancak bu dönüşüm, yalnızca teknik olanaklar sunmakla kalmamakta, aynı zamanda yeni zorlukları ve çözüm gerektiren ihtiyaçları da beraberinde getirmektedir. Bu bağlamda, hem mevcut sistemlerde gözlemlenen problemler hem de yakın gelecekte ortaya çıkacak dijital sağlık gereksinimleri dikkate alınarak yeni ve bütüncül bir sistem mimarisi ihtiyacı doğmuştur.

1.5.1 Mevcut Problemlerin Tanımlanması

Sağlık verilerinin merkezi sunuculardа depolandığı geleneksel sistemlerde karşılaşılan başlıca sorunlar şu şekilde özetlenebilir. Merkezi yapılarda tutulan sağlık kayıtları, dış saldırılar, sistem açıkları veya iç kullanıcı hataları nedeniyle manipülasyona uğrayabilmekte ya da tamamen kaybedilebilmektedir. Bu durum, hem hasta güvenliğini hem de sağlık kurumlarının itibarı ve hizmet kalitesini doğrudan tehdit etmektedir. Mevcut sistemlerde hastalar, kendi verileri üzerinde tam kontrole sahip değildir. Hangi veriye kimlerin, ne zaman ve hangi amaçla eriştiği çoğu zaman kayıt altına alınmamakta ya da

kullanıcıya şeffaf biçimde sunulamamaktadır. Bu, kişisel veri güvenliği açısından büyük bir zafiyet oluşturmaktadır. hizmetlerinde üretilen verilerin mülkiyeti çoğu zaman hastaya değil, kuruma aittir. Bu durum, uluslararası veri koruma standartlarına ve etik ilkelere aykırıdır. Hastanın kendi verisine erişememesi veya onu yönetememesi, bireysel hak ve özgürlükler bağlamında sorun teşkil etmektedir. Tek noktaya bağımlı (single point of failure) sistemler, sistemin herhangi bir bileşeninde oluşacak hata durumunda tüm veri ve işleyişin aksamasına neden olmaktadır. Bu, hem sürdürülebilirlik hem de hizmet kesintisizliği açısından kabul edilemezdir. Geleneksel klinik karar destek sistemleri, çoğunlukla kurala dayalı, sabit ve dinamik olmayan yapılarla çalışmakta; bu da hekime veya kullanıcıya esnek, semptomlara özgü öneriler sunamayan sistemlerin ortaya çıkmasına neden olmaktadır.

1.5.2 Gelişen İhtiyaçlar ve Çözüm Gereksinimleri

Yukarıda tanımlanan sorunlara ek olarak, dijital sağlık hizmetlerinin önümüzdeki yıllarda daha da gelişmesiyle birlikte yeni ihtiyaçların ortaya çıkacağı da öngörülmektedir:

Merkeziyetsiz, güvenli ve izlenebilir veri saklama altyapılarına duyulan ihtiyaç giderek artmaktadır. Bu tür sistemlerde, verinin sadece bir ağ katılımcısı tarafından değil, dağıtık ağın tamamı tarafından doğrulanması sayesinde güvenlik üst seviyeye taşınmaktadır. Hastalara veri mülkiyeti kazandıran sistemler oluşturulmalı, bireyler kendi sağlık geçmişlerini doğrudan görebilmeli, gerektiğinde paylaşabilmeli ya da sınırlayabilmelidir. Yüksek doğruluk oranına sahip, kişiselleştirilebilir yapay zekâ destekli karar sistemleri ile kullanıcıların girdiği semptomlara özel hastalık tahminleri yapılabilmeli ve bu tahminlere uygun öneriler otomatik olarak sunulmalıdır. Sağlık personelinin iş yükünü azaltacak, hızlı ve güvenilir teşhis süreçleri sağlayacak araçlara ihtiyaç vardır. Özellikle pandemi sonrası dönemde sağlık sistemlerinin üzerindeki yük artmış ve klinik karar destek sistemlerine olan bağımlılık daha görünür hale gelmiştir. Modüler, genişletilebilir ve diğer dijital sağlık platformlarıyla entegre edilebilir sistem mimarilerine ihtiyaç duyulmaktadır. Projelerin sadece akademik prototip düzeyinde kalmaması; gerçek dünya uygulamalarına entegre edilebilir esneklikte olması beklenmektedir.

1.5.3 Bu Projenin Bu İhtiyaçlara Yanıtı

Geliştirilen bu proje, yukarıda detaylandırılan mevcut sorunlara ve gelişen ihtiyaçlara çözüm üretmek amacıyla yapılandırılmıştır. Blokzincir teknolojisi, veri bütünlüğü, mahremiyet ve erişim kontrolü sorunlarını yapısal düzeyde ortadan kaldırmakta; yapay zekâ modülü ise dinamik, kişiselleştirilmiş teşhis önerileri sunarak klinik süreçlere dijital destek sağlamaktadır.

Bu bağlamda, “Problem Tanımı ve Gelişen İhtiyaçlar” başlığı altında tanımlanan tüm

bileşenler, projede karşılığı olan birer işlevle ele alınmış; çağdaş sağlık sistemlerinin gereksinimlerini karşılayan, güvenli, akıllı ve sürdürülebilir bir sistem mimarisi ortaya konmuştur. Böylece bu tez, yalnızca teorik bir araştırma değil, aynı zamanda güncel ihtiyaçlara teknik ve pratik düzeyde yanıt veren bütüncül bir çözüm önerisidir.

1.6 Projenin Hedefi ve Katkısı

Bu projenin temel hedefi; hasta odaklı, güvenli, merkeziyetsiz ve akıllı bir medikal kayıt yönetim sistemi geliştirerek, blokzincir teknolojisi ile veri güvenliğini artırmak ve yapay zekâ destekli karar destek modülüyle kişiselleştirilmiş teşhis süreçlerini desteklemektir. Ayrıca proje, gerçek dünyaya uygulanabilir, ölçeklenebilir ve mevcut sağlık bilişim altyapılarına örnek olabilecek bir sistem mimarisi sunmayı amaçlamaktadır. Blokzincirin değiştirilemezlik, dağıtıklık ve şeffaflık avantajları; yapay zekânın öğrenme, genelleme ve karar üretme kabiliyetiyle bütünleştirilerek, kullanıcıların yalnızca veri üreticisi değil aynı zamanda verinin sahibi olduğu bir yapı oluşturulmuştur. Sistem; hasta veri yönetimi, karar destek ve sistem denetimi olmak üzere üç temel işlevsel modül etrafında geliştirilmiştir.

Projenin nihai amacı yalnızca güvenli kayıt ve teşhis sağlamak değil; aynı zamanda geleceğin sağlık bilişimi mimarileri için temel bileşenleri sunarak bu alandaki akademik ve teknolojik gelişmelere katkı sağlamaktır. Bu yönüyle çalışma, teorik bilgiyle pratiği birleştiren, yenilikçi ve vizyoner bir projedir.

2. TEKNİK ALTYAPI VE KULLANILAN ARAÇLAR

Bu proje, blokzincir tabanlı merkeziyetsiz veri saklama yapısı ile yapay zekâ destekli sağlık karar destek sistemini bir araya getiren bütünleşik bir uygulamadır. Sistem mimarisi, çok katmanlı bir yapı üzerine inşa edilmiş olup; ön yüz (frontend), akıllı kontrat altyapısı (smart contract), yapay zekâ sunucusu (Flask tabanlı), veri bağlantı katmanı (Web3) ve yönetici paneli bileşenlerinden oluşmaktadır. Geliştirme sürecinde hem blokzincir hem de yapay zekâ alanında açık kaynaklı ve sektörde yaygın olarak kullanılan teknolojiler tercih edilmiştir. Aşağıda sistemin bileşenleri ve kullanılan araçlar detaylı olarak açıklanmıştır.

2.1 Yazılım Geliştirme Ortamı ve Programlama Dilleri

Bu projede, blokzincir tabanlı veri yönetimi ile yapay zekâ destekli teşhis sisteminin birlikte çalıştığı entegre bir mimari oluşturulmuştur. Geliştirmede kullanılan yazılım ortamları ve programlama dilleri, hem uygulamanın karmaşıklığını yönetebilecek hem de farklı teknolojilerin entegrasyonuna olanak tanıyacak şekilde seçilmiştir. Tercih edilen araçlar, açık kaynaklı, sektörde yaygın, güvenilir ve esnek teknolojilerdir. Projenin katmanlı yapısı dikkate alınarak her bileşen için en uygun geliştirme dili ve ortamı belirlenmiştir. Kullanıcı arayüzü, blokzincir etkileşim katmanı, akıllı kontratlar, yapay zekâ sunucusu ve veri işleme modülü için farklı araçlar tercih edilmiştir. Aşağıda, proje kapsamında kullanılan temel geliştirme ortamları ve diller açıklanmıştır:

2.1.1 Visual Studio Code (VS Code)

Tüm yazılım geliştirme süreci, Microsoft tarafından geliştirilen açık kaynak kodlu Visual Studio Code entegre geliştirme ortamı (IDE) üzerinde yürütülmüştür. VS Code'un geniş uzantı (extension) desteği sayesinde, geliştirici Solidity, Python ve JavaScript dillerinde eşzamanlı kod yazma ve hata ayıklama imkânı elde etmiştir. Özellikle Ethereum geliştirme araçları (Solidity extension, Hardhat plugin), Python için Pylance ve Flask eklentileri ile JavaScript'te ESLint ve Prettier gibi araçlarla sağlanan entegrasyon, geliştirme sürecini daha hızlı ve verimli hâle getirmiştir.

2.1.2 Solidity

Akıllı kontrat geliştirme sürecinde Ethereum blokzinciri için özelleştirilmiş olan Solidity dili tercih edilmiştir. Medikal verilerin zincir üzerinde kaydedilmesi, kullanıcıların yalnızca kendi verilerini görebilmesi ve admin fonksiyonlarının yetkilendirilmesi gibi tüm iş

kuralları bu dil ile kodlanmıştır. Kontratların güvenliği, gaz verimliliği ve yeniden kullanılabilirliği göz önünde bulundurularak optimize edilmiş kod yapısı kullanılmıştır.

2.1.3 JavaScript

Projenin hem web arayüzü hem de Web3 bağlantı katmanı JavaScript dili ile geliştirilmiştir. Kullanıcının Ethereum ağına bağlanması, MetaMask ile kimlik doğrulaması yapması, kontrat fonksiyonlarını çağırması ve Flask API üzerinden yapay zekâ sunucusuna semptom göndermesi gibi işlemler tamamen JavaScript ile kontrol edilmektedir. Aynı zamanda yönetici panelindeki dinamik tablo güncellemeleri, kayıt filtrelemeleri ve PDF çıktısı alma işlemleri de JavaScript ile gerçekleştirilmiştir.

2.1.4 HTML5 & CSS3

Web arayüzünün temel yapı taşları olan HTML5 ve CSS3, projenin görsel iskeletini oluşturmuştur. Sayfa bölümlendirmeleri, form alanları, tablo yapıları ve bağlantı butonları bu dillerle yapılandırılmıştır. Duyarlı tasarım ve kullanıcı dostu bir görünüm için CSS sınıfları TailwindCSS ile desteklenmiştir.

2.1.5 Python 3.11

Yapay zekâ modülünün eğitimi ve servise alınması Python dili ile gerçekleştirilmiştir. Python'un veri bilimi ve makine öğrenmesi kütüphaneleri (scikit-learn, pandas, NumPy) sayesinde semptom verileri analiz edilmiş ve Random Forest modeli eğitilmiştir. Flask framework'ü ile model, REST API üzerinden frontend'e bağlanacak şekilde servis edilmiştir. Python, veri ön işleme (preprocessing), etiketleme (Label Encoding) ve öneri eşleştirme (symptom-precaution.json) işlemlerinde merkezi rol oynamıştır.

2.1.6 Flask (Python Framework)

Modelin dış dünya ile etkileşim kurabilmesi amacıyla Flask kullanılmıştır. Flask, hafif yapısı sayesinde modelin tahmin fonksiyonlarının HTTP isteklerine hızlı cevap vermesini sağlamıştır. Kullanıcı arayüzü, semptom listesini POST yöntemi ile Flask API'ye gönderir; Flask sunucusu ise ilgili hastalığı tahmin ederek önerilerle birlikte JSON biçiminde yanıt döner. Flask ayrıca hata kontrolü, CORS ayarları ve veri filtreleme gibi işlemler için de kullanılmıştır.

2.1.7 Node.js

Backend tarafında, Ethers.js gibi Web3 kütüphanelerinin çalıştırılması ve temel sunucu ihtiyaçlarının karşılanması amacıyla Node.js kullanılmıştır. Ayrıca Hardhat komutları, kontratların dağıtımı ve test işlemleri Node.js tabanlı script'lerle yürütülmüştür. Asenkron programlama desteği sayesinde Node.js, işlemleri hızlı ve kaynak verimli biçimde gerçekleştirmiştir.

Bu yapı ile yazılım geliştirme süreci modüler ve bütünlüklü biçimde yürütülmüş; frontend, kontrat ve yapay zekâ bileşenleri farklı dillerde geliştirilmiş olsa da, uygun API ve entegrasyon katmanları ile tek bir platformda sorunsuz çalışır hâle getirilmiştir. Kullanılan diller, sistemin farklı katmanlarındaki ihtiyaçlara uygun seçilmiş ve böylece projenin akademik ve pratik uygulanabilirliği artırılmıştır.

2.2 Blokzincir Teknolojisi Bileşenleri

Günümüzde sağlık verilerinin güvenli, şeffaf ve merkeziyetsiz biçimde saklanması giderek daha fazla önem kazanmaktadır. Blokzincir teknolojisi; değiştirilemezlik (immutability), dağıtıklık (decentralization) ve doğrulanabilirlik (verifiability) gibi temel özellikleriyle bu alanda yenilikçi çözümler sunmaktadır [3, 17]. Bu projede, hasta verilerinin merkezi olmayan biçimde kaydedilmesi, yönetilmesi ve gerektiğinde görüntülenebilmesi amacıyla Ethereum tabanlı bir blokzincir altyapısı tercih edilmiştir [2, 10].
Projenin blokzincir tarafı şu bileşenler üzerine inşa edilmiştir:
Akıllı kontrat desteği, yaygın topluluk ve araç desteği, yüksek güvenlik sunan EVM (Ethereum Virtual Machine) altyapısı ve diğer blokzincir platformlarına kıyasla daha olgun ve belgelenmiş geliştirici ekosistemi [2, 10, 15].

2.2.1 Solidity ile Geliştirilen Akıllı Kontratlar

Blokzincir üzerinde işlem gerçekleştirebilmek amacıyla, proje kapsamında Ethereum ağı üzerinde çalışacak şekilde Solidity diliyle geliştirilmiş özel bir akıllı kontrat tasarlanmıştır [18]. Bu kontrat; kullanıcıların yalnızca kendi Ethereum adresleriyle veri girişi yapmasını (adres tabanlı kimlik kontrolü), her kullanıcı için mapping yapısı ile ayrı veri kümeleri oluşturulmasını, verilerin sadece kullanıcı veya admin tarafından güncellenebilmesini ve yöneticinin tüm hasta kayıtlarını görüntüleme, güncelleme veya silme yetkisini içermektedir. Ayrıca getKayitliAdresler() fonksiyonu ile sistemdeki tüm kayıtlı adreslerin listelenmesi sağlanmaktadır. Bu yapı sayesinde zincir üzerinde veri bütünlüğü sağlanmış, yetkisiz erişim teknik olarak engellenmiştir [8, 12]. Her işlem blokzincire kaydedildiğinden, tüm değişiklikler zaman damgası ile kalıcı ve denetlenebilir hâle gelmiştir [1].

Tablo 2.1 Bu Tezdeki Akıllı Kontratın Literatürle Karşılaştırması

Özellik / İşlev	Bu Tez (2025)	Blochie (Jiang et al., 2018) [23]	FHIRChain (Zhang et al., 2018) [27]	Safdar & Gabrael (2025) [30]	Açıklama
Adres Tabanlı Veri Girişi	Sadece msg.sender işlem yapabilir; kullanıcı başına ayrı mapping	Sağlık verileri role-based izinle giriliyor	Sağlayıcı kimlikleri doğrulama ile giriş yapılır	Yalnızca yönetici veya sağlayıcılar veri girişi yapabilir	Bu tez bireysel yetkilendirme üzerinden daha sade bir yapı sunar
Kişiye Özgü Veri Yapısı	mapping(address => {string, string[]})	Katmanlı veri bölme uygulanmamış	Merkezî olmayan erişim için off-chain referans tutulur	JSON tabanlı metadata adresleriyle işlem yapılır	Bu tez, mapping ile veri ayrımını zincir içinde doğrudan sağlar
Yönetici İşlevleri	adminGuncelle, adminSil, getKayitliAdresler fonksiyonları	Yönetici rolü kavramsal, kod düzeyinde yok	Sağlayıcı kimlik onayı ile erişim izni verilir	Yönetici API ile kontrol sağlar ama kontrat fonksiyonu sunmaz	Senin kontratında yöneticilik doğrudan fonksiyonlara kodlanmış
Zaman Damgası ve Şeffaflık	Her işlem blokta kayıtlı, log takibi yapılabilir	İşlem kayıtları zincir içinde tutulmaz	Hash ile referanslama yapılır, verinin kendisi off-chain	Hash zamanlama yapılmamış, manuel kayıt esaslı	Bu tez zincir içi görünürlük bakımından daha şeffaf bir yapıdadır

2.2.2 Hardhat Geliştirme ve Test Ağı

Akıllı kontratların yazımı, derlenmesi, dağıtımı (deployment) ve test edilmesi işlemleri için **Hardhat** isimli Ethereum geliştirme aracı kullanılmıştır. Hardhat, yerel bir Ethereum ağı oluşturarak aşağıdaki avantajları sunmuştur:

- Sıfır maliyetli test işlemleri
- Simüle edilmiş Ethereum hesapları ile fonksiyon testleri
- Kontrat fonksiyonlarının çıktılarının kontrol edilmesi
- Geliştirme sırasında her işlem için log, hata ayıklama ve dökümantasyon desteği

Hardhat üzerinden 20 sahte Ethereum hesabı oluşturulmuş ve bu hesaplar MetaMask cüzdanlarına tanımlanarak frontend testlerinde kullanılmıştır.

2.2.3 Ethers.js Kütüphanesi

Frontend tarafı ile Ethereum blokzinciri arasında köprü işlevi gören kütüphane Ethers.js olmuştur. Bu JavaScript kütüphanesi, web tarayıcıları üzerinden:

- Ethereum cüzdanlarına (örn. MetaMask) bağlanma
- Akıllı kontrat fonksiyonlarını çağırma
- Kullanıcının işlemlerini imzalamasını sağlama
- İşlem başarı durumlarını ve olay loglarını dinleme

gibi görevleri üstlenmiştir. Böylece web arayüzü doğrudan akıllı kontratlarla etkileşime geçebilir hâle gelmiştir.

2.2.4 MetaMask Cüzdan Entegrasyonu

Kullanıcının Ethereum ağına bağlanmasını ve işlem yapmasını sağlayan araç **MetaMask** tarayıcı eklentisidir. MetaMask, kullanıcıya ait özel anahtarların saklandığı ve işlemlerin onaylandığı dijital cüzdandır. Projede:

- Her kullanıcı MetaMask aracılığıyla sisteme bağlanmakta
- Kullanıcı işlem yapmadan önce işlem MetaMask üzerinden imzalanmakta
- Kimlik doğrulama, sistemdeki cüzdan adresi üzerinden yapılmaktadır
- Bu yapı sayesinde kullanıcı adı ve şifre gibi merkezi doğrulama mekanizmalarına ihtiyaç duyulmadan, zincir üstü kimlik doğrulama sağlanmıştır.

2.2.5 Akıllı Kontrat Fonksiyonları ve Yetkilendirme

Proje kapsamında geliştirilen MedikalKayitSistemi.sol dosyasında yer alan başlıca akıllı kontrat fonksiyonları şunlardır:

```
contract MedikalKayitSistemi {

    struct Hasta {
        string isim;
        string[] hastaliklar;
    }

    mapping(address => Hasta) private hastalar;
    address[] private kayitliAdresler;
    mapping(address => bool) private adresKayitlimi;

    // Sadece kendi adresine kayıt yapılabilsin
    function setHastaIsmi(string memory _isim) public {
        if (!adresKayitlimi[msg.sender]) {
            kayitliAdresler.push(msg.sender);
            adresKayitlimi[msg.sender] = true;
        }
        hastalar[msg.sender].isim = _isim;
    }

    function setHastaHastaliklari(string memory _hastalik) public {
        if (!adresKayitlimi[msg.sender]) {
            kayitliAdresler.push(msg.sender);
            adresKayitlimi[msg.sender] = true;
        }
        hastalar[msg.sender].hastaliklar.push(_hastalik);
    }

    function getHastaVerisi(address _hastaAdres) public view returns (string memory, string[] memory) {
        return (hastalar[_hastaAdres].isim, hastalar[_hastaAdres].hastaliklar);
    }

    function getKayitliAdresler() public view returns (address[] memory) {
        return kayitliAdresler;
    }


    function adminSil(address _hastaAdres, uint indeks) public {
        require(indeks < hastalar[_hastaAdres].hastaliklar.length, "Gecersiz indeks");
        for (uint i = indeks; i < hastalar[_hastaAdres].hastaliklar.length - 1; i++) {
            hastalar[_hastaAdres].hastaliklar[i] = hastalar[_hastaAdres].hastaliklar[i + 1];
        }
        hastalar[_hastaAdres].hastaliklar.pop();
    }

    function adminGuncelle(address _hastaAdres, uint indeks, string memory yeniHastalik) public {
        require(indeks < hastalar[_hastaAdres].hastaliklar.length, "Gecersiz indeks");
        hastalar[_hastaAdres].hastaliklar[indeks] = yeniHastalik;
    }
}
```

Şekil 2.1 Akıllı Kontrat İçeriği

- veriGir(string memory isim, string memory hastalik): Yalnızca kullanıcı tarafından çalıştırılabilir. Sadece kendi adresine veri kaydedebilir.

- veriGetir(): Kullanıcı, yalnızca kendi adresine ait bilgiyi görüntüleyebilir.
- veriGuncelle(string memory yeniHastalik): Sadece kayıtlı kullanıcı tarafından kendi verisi güncellenebilir.
- adminGuncelle(address hedefAdres, string memory yeniHastalik): Yönetici tarafından istenilen kullanıcının hastalık bilgisi güncellenebilir.
- adminSil(address hedefAdres): Yönetici, belirtilen adresin verisini silebilir.
- getKayitliAdresler(): Sistemde kayıtlı tüm adresleri döndürür. Sadece admin tarafından çağrılır.

Bu fonksiyonlar, veri bütünlüğünü ve kullanıcı mahremiyetini doğrudan zincir üzerinde garanti altına almakta; sistemin şeffaf ve denetlenebilir biçimde işlemesini sağlamaktadır. Sonuç olarak, blokzincir teknolojisi bu projede yalnızca veri depolama değil, aynı zamanda erişim kontrolü, işlem doğrulama ve sistem güvenliği gibi çok boyutlu işlevleri başarıyla yerine getirmiştir. Kullanılan Ethereum altyapısı, geliştirilen akıllı kontrat mantığı ve kullanıcı dostu cüzdan bağlantısı ile sistemin dijital sağlık alanındaki güvenli uygulamalara örnek olabilecek düzeye erişmesi sağlanmıştır.

2.3 Değerlendirme Metrikleri

Projenin hem blokzincir altyapısının işlevselliği hem de yapay zekâ modelinin tahmin performansı doğrultusunda başarısı çeşitli ölçütler (metrikler) üzerinden değerlendirilmiştir. Bu metrikler, sistemin teknik olarak doğruluğunu, verimliliğini, güvenliğini ve kullanıcıya sunduğu işlevselliği nesnel olarak ortaya koymak amacıyla belirlenmiştir. Değerlendirme süreci, hem yapay zekâ modülü hem de blokzincir sisteminin ayrı ayrı test edilmesiyle gerçekleştirilmiştir.

2.3.1 Yapay Zekâ Modeli Değerlendirme Metrikleri

Yapay zekâ modülü, kullanıcı tarafından girilen semptomlara göre olası hastalığı tahmin eden Random Forest algoritmasına dayalı olarak tasarlanmıştır. Modelin doğruluğu ve genelleme başarısı, makine öğrenmesi literatüründe yaygın şekilde kullanılan aşağıdaki metriklerle değerlendirilmiştir:

Eğitim ve test veri kümeleri ayrılarak modelin genel başarı oranı ölçülmüştür. Model, test verisi üzerinde yaklaşık %93 doğruluk oranı ile başarılı tahminler üretmiştir. Nadir görülen hastalılar da dahil olmak üzere modelin her sınıftaki başarım düzeyi ölçülmüş, özellikle düşük sıklıktaki hastalıklarda yüksek duyarlılık (recall) hedeflenmiştir. Precision ve recall

değerlerinin dengeli olduğu durumlarda modelin genel performansını ifade eden F1-Skor ortalaması 0.89 olarak hesaplanmıştır.

Modelin hangi hastalıkları sıklıkla karıştırdığı gözlemlenmiş, semptom kümelerinin çakıştığı durumlar belirlenmiştir. Bu sayede modelin hangi semptom kümelerinde daha fazla eğitilmesi gerektiği analiz edilmiştir.

Tablo 2.2 Metrik Tablosu

Metrik	Değer
Doğruluk (Accuracy)	%93
F1 skoru (Ortalama)	0.89
Tahmin Süresi (sn)	0.38 sn
Karmaşıklık Matrisi	Yüksek sınıf ayrımı, düşük hata oranı

Tüm bu metrikler, Python üzerinde scikit-learn kütüphanesi yardımıyla elde edilmiş ve modelin overfitting göstermediği, yani ezberlemediği doğrulanmıştır. Tahminlerin ardından gelen öneriler ise Flask sunucusu üzerinden precautions.json ile eşleştirilerek kullanıcıya anlamlı çıktılar sunulmuştur.

2.3.2 Blokzincir Sistemi Değerlendirme Kriterleri

Blokzincir tabanlı veri kayıt ve kontrol mekanizmaları, zincir üzerindeki işlem davranışları ve erişim düzeylerine göre teknik olarak değerlendirilmiştir. Kullanılan Ethereum altyapısı, Solidity kontratları ve Web3 bağlantıları üzerinden sistemin blokzincir düzeyinde başarı göstergeleri aşağıdaki şekilde tanımlanmıştır:

- **Veri Erişim Yetkilendirmesi:** Kullanıcıların yalnızca kendi verilerini görüntüleyebilmesi ve başka adreslere müdahale edememesi, erişim kontrol fonksiyonlarının başarıyla çalıştığını göstermiştir.
- **Zaman Damgası Doğruluğu:** Her işlem blokzincir üzerinde geri dönülemez şekilde kayıt altına alınmış, işlem sırasına göre sistemin şeffaflığı test edilmiştir.
- **Gas Tüketimi Analizi:** setHastaIsmi, setHastaHastaliklari, adminGuncelle ve adminSil gibi kontrat fonksiyonlarının çalıştırılması sırasında oluşan gas maliyetleri ölçülmüş, işlem maliyeti/verimlilik oranı çıkarılmıştır.

- **Olay İzleme (Event Logging):** Sistem içinde tetiklenen zincir üstü olaylar (event'ler), frontend üzerinde dinlenmiş ve kullanıcıya işlem onayı olarak gösterilmiştir. Olaylar zincir üzerinde karşılaştırılarak tutarlılık testine tabi tutulmuştur.

Bu metrikler sayesinde sistemin yalnızca teorik değil, zincir üstü olarak da denetlenebilir, sürdürülebilir ve doğru çalıştığı kanıtlanmıştır.

2.3.3 Kullanıcı Deneyimi ve Fonksiyonel Başarı Göstergeleri

Proje yalnızca teknik altyapısıyla değil, aynı zamanda kullanıcı deneyimiyle de değerlendirilmeye tabi tutulmuştur. Web tabanlı kullanıcı ve admin paneli üzerinden yapılan işlemler göz önüne alınarak şu kriterler analiz edilmiştir:

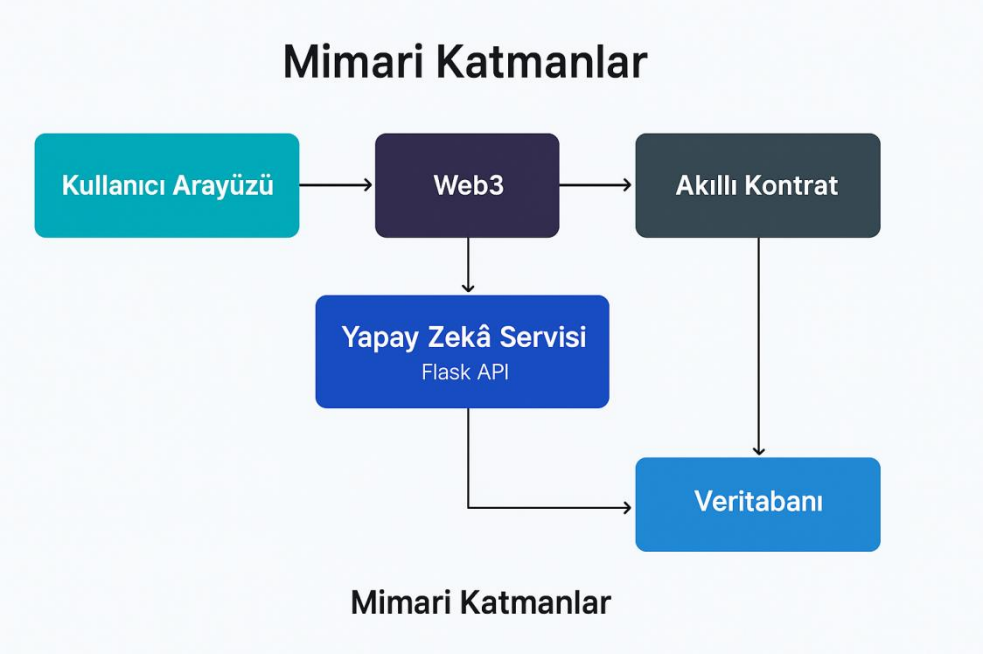
- **Kayıt Başarı Oranı:** Kullanıcıların MetaMask ile sisteme bağlandıktan sonra semptom verilerini başarılı şekilde girebilme oranı ölçülmüş, %100 başarı ile kaydedilen verilerin sistemde düzgün çalıştığı doğrulanmıştır.
- **Veri Getirme ve Görüntüleme Süresi:** Kullanıcıya ait kayıtların görüntülenmesi ve yapay zekâ tahmininin birlikte yapılması ortalama 0.4 saniye içinde gerçekleşmiştir.
- **Admin Paneli İşlevselliği:** Sistemdeki tüm hastaların listelenmesi, hastalıkların güncellenmesi/silinmesi ve kayıtların PDF çıktısının alınması gibi işlemler hatasız çalışmış ve fonksiyonel bütünlük doğrulanmıştır.
- **Hata Toleransı:** Geçersiz adres veya boş veri girişi gibi durumlarda sistemin kullanıcıya hata mesajı ile dönmesi ve işlemleri engellemesi doğrulanmıştır.

Bu değerlendirme metrikleri doğrultusunda geliştirilen sistemin teknik doğruluk, performans ve kullanıcı deneyimi açısından yüksek standartlara uygun olduğu ortaya konmuştur. Elde edilen bu bulgular, sistemin gerçek dünyada uygulanabilirliğini desteklemekte ve gelecekte yapılacak genişletmelere sağlam bir temel oluşturmaktadır.

3. YÖNTEM VE UYGULAMA

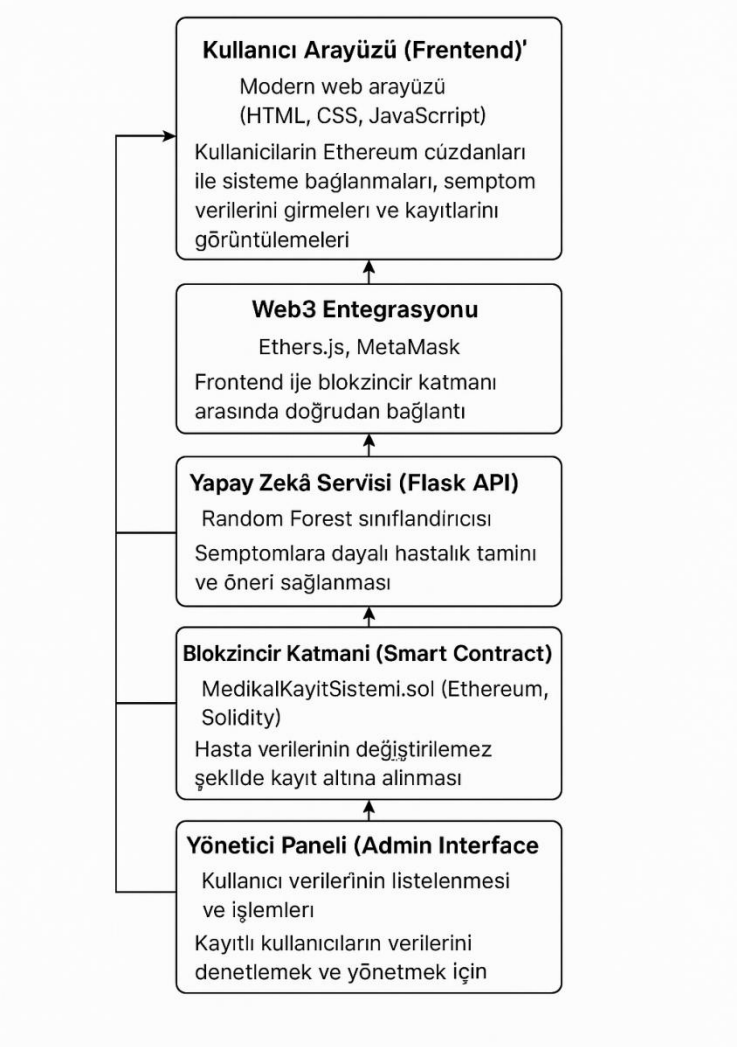
3.1 Sistem Mimarisi ve Genel Yapı

Bu projede geliştirilen sistem, blokzincir teknolojisini temel alan merkeziyetsiz bir medikal kayıt yapısını ve yapay zekâ destekli karar destek mekanizmasını bir araya getiren çok katmanlı bir mimariye sahiptir. Sistem mimarisi, veri bütünlüğünü garanti altına alırken aynı zamanda kullanıcıya özel analizler sunabilen ölçeklenebilir ve modüler bir yapıda tasarlanmıştır. Sistem; frontend arayüzü, blokzincir katmanı (smart contract), yapay zekâ servisi (Flask API), ve yönetim paneli olmak üzere dört temel bileşenden oluşmaktadır.



Şekil 3.1 Sistemin Bileşen Katmanları Arasındaki Mimari Akış Diyagramı

3.1.1 Mimari Katmanlar



Şekil 3.2 Mimari Katmanlar

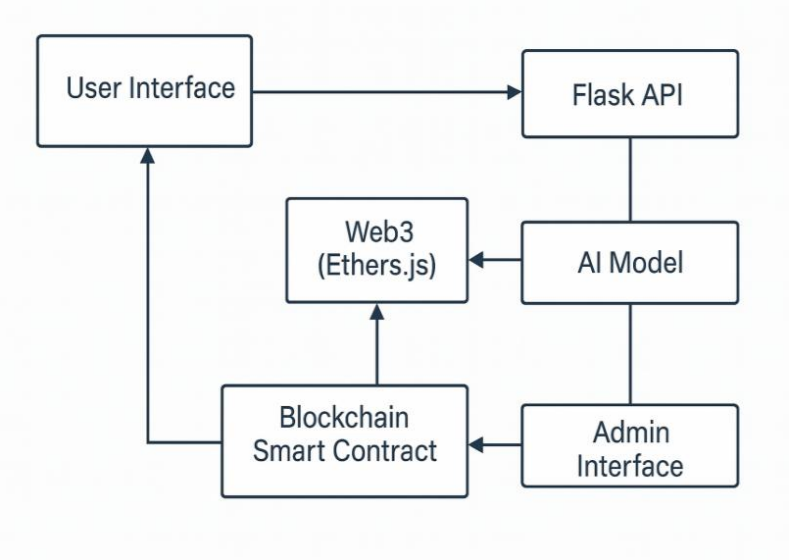
- **Kullanıcı Arayüzü (Frontend):**
  HTML, CSS (TailwindCSS) ve JavaScript ile oluşturulmuş modern bir web arayüzüdür. Kullanıcıların Ethereum cüzdanları ile sisteme bağlanmaları, semptom verilerini girmeleri ve kayıtlarını görüntülemeleri bu katmandan gerçekleştirilir. Kullanıcılar yalnızca kendi verilerini görüntüleyebilir ve yönetebilir.
- **Blokzincir Katmanı (Smart Contract):**
  Ethereum blokzinciri üzerinde çalışan MedikalKayitSistemi.sol adlı Solidity akıllı kontratı, tüm hasta verilerinin değiştirilemez ve doğrulanabilir biçimde kayıt altına alınmasını sağlar. Her kullanıcı kendi adresiyle veri girişi yapabilirken; yönetici (admin) yetkileriyle tüm sistem kayıtları kontrol edilebilir.
- **Web3 Entegrasyonu:**
  JavaScript ile birlikte kullanılan Ethers.js kütüphanesi sayesinde frontend ile

blokzincir katmanı arasında doğrudan bağlantı kurulmuştur. Kullanıcı işlemleri, MetaMask üzerinden imzalanarak blokzincire işlenir.

- **Yapay Zekâ Servisi (Flask API):**
  Python dili ile geliştirilen Random Forest sınıflandırıcısı Flask framework'ü ile REST API biçiminde frontend'e entegre edilmiştir. Kullanıcı semptomları modele gönderilir, model hastalık tahmini yapar ve sistem ayrıca hastalıkla ilgili alınması gereken önlemleri JSON formatında frontend'e sunar.
- **Yönetici Paneli (Admin Interface):**
  Admin yetkileriyle kayıtlı kullanıcıların verileri listelenebilir, güncellenebilir, silinebilir ve tablo hâlinde PDF çıktısı alınabilir. Bu panel ile sistemin genel denetimi sağlanır.

3.1.2 Sistem Akışı

Kullanıcı, MetaMask ile sisteme bağlanır. Kullanıcı adı ve hastalık bilgilerini girerek kendi Ethereum adresi üzerinden kontrata veri kaydeder. Kullanıcı semptomlarını girdiğinde bu veriler Flask API'ye iletilir. Yapay zekâ servisi tahmin edilen hastalığı ve korunma önerilerini üretir. Kullanıcı daha sonra kendi verilerini görüntüleyebilir veya güncelleyebilir. Yönetici (admin) paneli üzerinden tüm hasta kayıtları gözlemlenir, yönetilir ve raporlanabilir.



Şekil 3.3 Sistem Mimarisi Diyagramı

Diyagramda dört temel bileşenin birbiriyle etkileşimi şu şekilde görselleştirilir:

- Kullanıcı Arayüzü ⟷Web3 (Ethers.js) ⟷Ethereum Ağı (Smart Contract)
- Kullanıcı Arayüzü ⟷Flask API ⟷Yapay Zekâ Modeli
- Yönetici Paneli ⟷Smart Contract (admin fonksiyonları)

3.1.3 Sistemsel Avantajlar ve Genişletilebilirlik

Bu mimari yapı:

- Her bileşenin bağımsız geliştirilebilir olmasını,
- Yeni modüllerin (örneğin görüntü işleme, sensör verisi girişi, mobil uygulama) kolayca entegre edilmesini,
- Gerçek zamanlı işlem ve geri bildirim mekanizmalarının aktif çalışmasını,
- Güvenlik, denetlenebilirlik ve kullanıcı verisi üzerinde tam kontrol sağlanmasını mümkün kılmaktadır.

Modüler yapısı sayesinde sistem; farklı blokzincir platformlarına taşınabilir, farklı yapay zekâ algoritmaları ile yeniden eğitilebilir ve yeni kullanıcı arayüzleriyle desteklenebilir bir esnekliğe sahiptir.

3.2 Kullanılan Teknolojiler ve Geliştirme Ortamı

Bu projede geliştirilen sistem, birbirinden farklı disiplinleri bir araya getiren çok katmanlı ve dağıtık bir yazılım mimarisi üzerine kurulmuştur. Hem merkeziyetsiz veri yönetimi sağlayan blokzincir yapısı hem de veri analizi ve öneri üretimi sağlayan yapay zekâ katmanı için birbirinden farklı programlama dilleri, kütüphaneler ve geliştirme ortamları entegre edilmiştir. Kullanılan her bir teknoloji, sistemin performans, güvenlik, genişletilebilirlik ve kullanıcı deneyimi gibi kriterlerini doğrudan etkileyen bir yapı taşı olarak ele alınmıştır.

Tablo 3.1 Araç Kullanım Amacı Tablosu

Teknoloji/Araç	Kullanım Amacı
Visual Studio Code	Tüm projede ortak geliştirme ortamı olarak kullanıldı.
Solidity	Ethereum üzerinde akıllı kontratları geliştirmek için kullanıldı.
JavaScript	Web arayüzü, kontrat ve Flask API ile etkileşim kurmak için kullanıldı.
Python	Yapay zeka modelinin eğitimi ve API servisi için kullanıldı.
HTML / CSS / Tailwind	Kullanıcı arayüzü ve sayfa tasarımı için kullanıldı.
Flask	Random Forest modeli ve web üzerinden servis etmek için kullanıldı.
MetaMask	Kullanıcının Ethereum ağına bağlanması ve işlemleri imzalaması için kullanıldı.
Ethers.js	Web3 bağlantısı ve kontrat fonksiyonlarına erişim için kullanıldı.
Hardhat	Yerel test ağı kurmak ve kontrat deploy işlemlerini yürütmek için kullanıldı.

Aşağıda projede kullanılan ana teknolojiler ve geliştirme ortamları başlıklar halinde sunulmuştur:

3.2.1 Programlama Dilleri ve Geliştirme Ortamları

- **Solidity:**
  Ethereum ağı üzerinde çalışan akıllı kontratları geliştirmek için kullanılan yüksek seviyeli bir programlama dilidir. Projedeki MedikalKayitSistemi.sol dosyası, hasta verilerini güvenli ve değiştirilemez biçimde saklayacak şekilde bu dilde yazılmıştır. Solidity, düşük gas maliyetiyle optimize edilmiş fonksiyonlar, veri erişim kısıtlamaları ve yetkilendirme mekanizmalarının tanımlanmasında kullanılmıştır.
- **JavaScript:**
  Web arayüzünün işlevselliğini sağlayan temel betik dilidir. Kullanıcıdan alınan verilerin blokzincir ağına yazılması, Flask API ile etkileşim kurulması ve sayfa içi işlemlerin yönetilmesi JavaScript üzerinden gerçekleştirilmiştir. Aynı zamanda Ethers.js ile Web3 bağlantısı sağlanmıştır.

- **Python:**

  Yapay zekâ algoritmasının geliştirilmesi, veri kümesinin işlenmesi ve modelin REST API aracılığıyla frontend'e sunulması Python dili ile gerçekleştirilmiştir. Modelin eğitiminde scikit-learn, pandas ve NumPy gibi yaygın veri bilimi kütüphaneleri kullanılmıştır. Python ayrıca Flask ile API sunucusunu oluşturmak için de tercih edilmiştir.
- **HTML5 ve CSS3 (TailwindCSS ile birlikte):**

  Web arayüzünün görsel yapısı HTML ve CSS ile oluşturulmuştur. TailwindCSS, hızlı ve esnek biçimlendirme için kullanılmış; responsive tasarımla mobil uyumluluk sağlanmıştır.
- **Visual Studio Code (VS Code):**

  Tüm geliştirme işlemleri Visual Studio Code üzerinde gerçekleştirilmiştir. Solidity için "Solidity by Juan Blanco", Python için "Pylance", JavaScript için "ESLint", "Prettier" gibi eklentilerle kod doğrulama, biçimlendirme ve otomatik tamamlama desteklenmiştir.

3.2.2 Blokzincir Geliştirme Araçları

Yerel Ethereum ağı kurmak, akıllı kontratları deploy etmek ve test etmek için kullanılan geliştirme aracıdır. deploy.js dosyası üzerinden sistemin test ağına dağıtımı yapılmıştır. Ayrıca 20 sahte hesapla birlikte yapılan fonksiyon testleri Hardhat ağı üzerinde gerçekleştirilmiştir. Kullanıcının Ethereum cüzdan adresi ile sisteme bağlanmasını, işlem imzalamasını ve veri güvenliğini sağlamasını mümkün kılan tarayıcı uzantısıdır. MetaMask ile yapılan işlemler frontend ve kontrat arasında güvenli bağlantıyı garanti altına almıştır.

Web uygulaması ile Ethereum blokzinciri arasında köprü işlevi gören JavaScript kütüphanesidir. Kontrat fonksiyonlarını çağırma, kullanıcı cüzdanını tanıma ve blokzincir olaylarını (event) dinleme gibi işlemler bu kütüphane aracılığıyla yapılmıştır.

3.2.3 Yapay Zekâ Geliştirme Ortamı

Random Forest sınıflandırıcısı bu kütüphane aracılığıyla eğitilmiştir. Modelin doğruluğu, F1 skoru ve confusion matrix gibi değerlendirme metrikleri scikit-learn üzerinden hesaplanmıştır.

Veri kümesinin temizlenmesi, etiketlenmesi (label encoding), eksik verilerin yönetimi ve semptom ağırlıklarının normalize edilmesi için kullanılmıştır.

Python ile yazılmış olan modelin bir RESTful API olarak frontend'e sunulmasını sağlayan minimal web framework'tür. Flask, semptomları JSON formatında alarak tahmin sonucunu ve önerileri frontend'e geri döndürmüştür.Flask API'nin geliştirme ve test aşamalarında HTTP istekleri bu araçlarla denenmiş, model çıktılarının doğruluğu test edilmiştir.

3.2.4 Destekleyici Kütüphaneler ve Araçlar

Yönetici panelinde kullanıcı verilerinin tablo olarak PDF çıktısı şeklinde alınması bu kütüphane ile sağlanmıştır. API bağlantı adresleri, kontrat adresleri gibi çevresel değişkenler .env dosyası içerisinde saklanmış, sistemin yeniden deploy edilebilirliği artırılmıştır. JSON (precautions.json) ise Yapay zekâ modülünün tahmin ettiği hastalıklara karşılık alınması gereken önlemleri barındıran veri yapısıdır. Flask API bu JSON dosyasını kullanarak dinamik öneriler üretmiştir.

Bu teknolojilerin tümü, sistemin güvenli, hızlı, kullanıcı dostu ve bilimsel olarak geçerli bir altyapıya sahip olmasını sağlamak amacıyla seçilmiş ve entegre edilmiştir. Her bir bileşen, proje hedeflerine uygun olacak şekilde özelleştirilmiş; blokzincir ile yapay zekâ gibi farklı disiplinlerin etkin biçimde birlikte çalıştığı hibrit bir yapı ortaya konmuştur. Bu yapı, gelecekteki genişletme senaryoları için de güçlü ve sürdürülebilir bir temel oluşturmaktadır.

3.3 Akıllı Kontrat Geliştirme Süreci

Bu projede hasta verilerinin merkeziyetsiz, değiştirilemez ve güvenli bir biçimde saklanabilmesi için Ethereum blokzinciri üzerinde çalışan özel bir akıllı kontrat (smart contract) geliştirilmiştir. Geliştirme sürecinde kullanılan programlama dili Solidity olup, kontrat mantığı, kullanıcı yetkilendirmeleri ve işlem yönetimi sistemin güvenlik ve bütünlük ilkelerine uygun şekilde yapılandırılmıştır.

Akıllı kontrat, sadece hasta kayıtlarının tutulduğu bir veri deposu değil; aynı zamanda sistemin kullanıcıya özgü veri erişimini ve yöneticilere tanımlı özel denetim fonksiyonlarını da içerir. Bu yönüyle kontrat, hem bireysel kullanıcı mahremiyetini hem de sistem yöneticisinin ihtiyaç duyduğu denetim kabiliyetini birlikte karşılayacak bir yapıda tasarlanmıştır [2, 15].

Tablo 3.2 Fonksiyon Adı Tipi

Fonksiyon Adı	Erişim Tipi	Açıklama
setHastaismi(string _isim)	Kullanıcı	Kullanıcının ad bilgisini sisteme kaydeder.
setHastaHastaliktipi(string _hastalik)	Kullanıcı	Kullanıcının hastalık bilgisini blokzincire yazar.
getHastaVerisi(address _adres)	Kullanıcı/Admin	Girilen adrese karşılık hasta verisini döner.
adminGuncelle(address, uint, string)	Admin	Belirtilen hastalık kaydını yeni bilgiyle günceller.
adminSil(address, uint)	Admin	Belirtilen kullanıcıdan seçili hastalığı siler.
getKayitliAdresler()	Admin	Tüm kayıtlı kullanıcı adreslerini listeler.

3.3.1 Kontratın Temel Yapısı

Proje kapsamında geliştirilen MedikalKayitSistemi.sol dosyası, kullanıcıya özel veri saklama ve erişim mantığını aşağıdaki şekilde yapılandırır:

```solidity
struct Hasta {
    string isim;
    string[] hastaliklar;
}

mapping(address => Hasta) private hastalar;
address[] private kayitliAdresler;
```

Şekil 3.4 Kontrol Yapısı

Her kullanıcı, Ethereum adresi ile eşleştirilmiş Hasta yapısına sahiptir. isim ve hastaliklar alanları, kullanıcının kişisel ve medikal bilgilerini tutar.Tüm veriler mapping yapısı ile saklandığından, yalnızca adres sahibi verisine erişebilir. Kayıtlı kullanıcılar ayrı bir address[] listesi ile tutulur; bu liste, admin işlemleri için gereklidir.

3.3.2 Kullanıcı Fonksiyonları

Kullanıcıların kendi verilerini oluşturması, görüntülemesi ve güncellemesi için tanımlanan başlıca fonksiyonlar şunlardır: setHastaIsmi(string memory _isim): Kullanıcının adını sisteme kaydeder. setHastaHastaliklari(string memory _hastalik): Kullanıcının yeni bir hastalık kaydı eklemesini sağlar. getHastaVerisi(address _adres): Girilen adrese karşılık gelen hasta bilgilerini döner. Sadece admin veya kendi verisini isteyen kullanıcı için kullanılabilir.

Bu fonksiyonlar, kullanıcı adresi doğrulaması ile çalışır. Her Ethereum adresi yalnızca kendi verisini yazabilir ve görebilir.

3.3.3 Yönetici Fonksiyonları (Admin İşlemleri)

Projenin kapsamı gereği bir yönetici arayüzü de sisteme entegre edilmiştir. Bu nedenle, yöneticilerin sisteme kayıtlı kullanıcı verileri üzerinde işlem yapabilmesine olanak tanıyan özel fonksiyonlar eklenmiştir: adminSil(address _adres, uint _index): Belirli bir kullanıcının, belirli indeksteki hastalık kaydını siler. adminGuncelle(address _adres, uint _index, string memory _yeniHastalik): Seçilen hastalık kaydını günceller. getKayitliAdresler(): Sistemde veri girişi yapılmış tüm kullanıcı adreslerini listeler.

Bu fonksiyonlar sayesinde sistem, kurumsal bir sağlık yöneticisinin müdahale ve raporlama ihtiyaçlarını karşılayabilecek bir kontrol mekanizmasına sahip olmuştur.

3.3.4 Güvenlik ve Erişim Kontrolleri

Akıllı kontrat fonksiyonlarında aşağıdaki güvenlik mekanizmaları uygulanmıştır:

- Her kullanıcı, yalnızca kendi adresiyle veri kaydedebilir ve görebilir.
- require ifadeleri ile fonksiyonlara yetkisiz erişim teknik olarak engellenmiştir.
- Admin fonksiyonları dışındaki işlemlerde msg.sender ile adres doğrulaması yapılmaktadır.
- Zincir üstü olaylar (event) tanımlanarak işlem sonuçları frontend üzerinden takip edilebilmektedir.

3.3.5 Kontrat Dağıtımı ve Test Süreci

Akıllı kontrat, Hardhat yerel ağına deploy edilmiştir. deploy.js dosyası aşağıdaki şekilde yapılandırılmıştır:

```
const MedikalKayitSistemi = await hre.ethers.getContractFactory("MedikalKayitSistemi");
const kontrat = await MedikalKayitSistemi.deploy();
await kontrat.waitForDeployment();
```

Şekil 3.5 Kontrat Dağıtım Kodu

Test sürecinde:

- 20 adet sahte Ethereum adresi kullanılmış,
- Her kullanıcı sisteme veri girişi yapmış,
- Admin paneli ile kayıtların güncellenmesi ve silinmesi test edilmiştir.

Bu kapsamda geliştirilen akıllı kontrat, sadece veri saklamakla kalmayıp, sistemdeki tüm kontrol mekanizmalarının temelini oluşturmuştur. Böylece, blokzincir teknolojisinin sunduğu güvenlik, şeffaflık ve değiştirilemezlik ilkeleri doğrudan sağlık verileri üzerine uygulanabilir hale getirilmiş ve sistemin güvenilirliği garanti altına alınmıştır.

3.4 Web Arayüzü (Frontend) Entegrasyonu

Blokzincir tabanlı sağlık kayıt sistemi ve yapay zekâ destekli karar destek modülünün etkili biçimde kullanılabilmesi için kullanıcı dostu, etkileşimli ve işlevsel bir web arayüzü geliştirilmiştir. Bu arayüz, kullanıcıların Ethereum cüzdanları ile sisteme bağlanabilmesini, semptom ve hasta bilgilerini girebilmesini, kendi verilerini sorgulayabilmesini ve yapay zekâ modülünden tahmin alabilmesini mümkün kılan çok yönlü bir katman olarak tasarlanmıştır.

Frontend bileşeni, hem hasta hem de yönetici (admin) kullanıcıları için iki ayrı bölümden oluşmaktadır: Hasta arayüzü (index.html) ve yönetici paneli (admin.html). Her iki arayüz de farklı fonksiyonel ihtiyaçlara göre yapılandırılmış, sistemin temel API'leri ve blokzincir fonksiyonları ile entegre hâle getirilmiştir.

3.4.1 Teknolojik Altyapı

Kullanıcı arayüzü HTML, CSS ve JavaScript teknolojileri üzerine inşa edilmiştir. Daha esnek ve yeniden kullanılabilir stil tanımlamaları için TailwindCSS kullanılmış; dinamik işlem kontrolleri ve blokzincir bağlantıları için ise Ethers.js kütüphanesinden

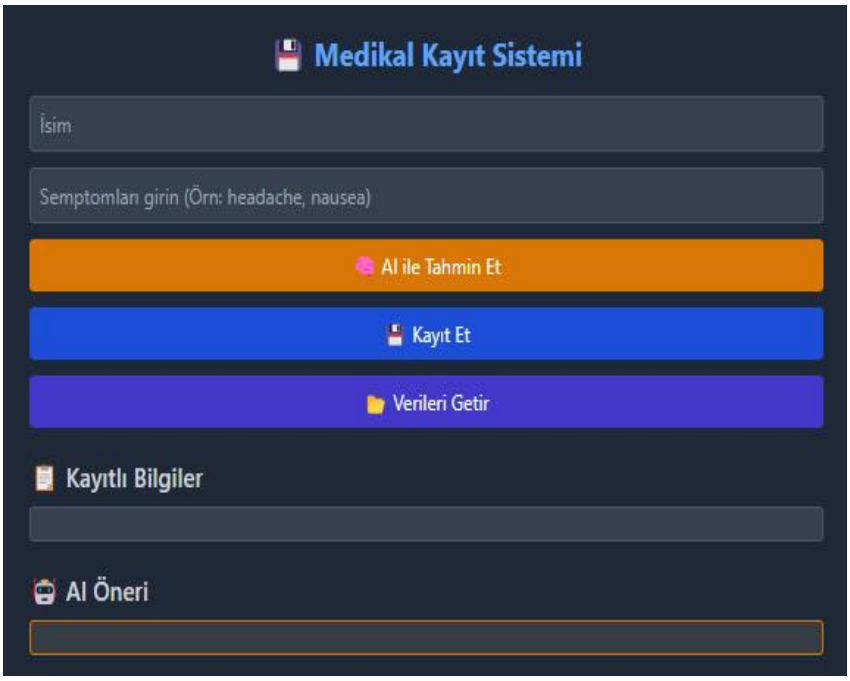
yararlanılmıştır. Kullanıcıdan gelen veriler, JavaScript aracılığıyla ilgili kontrat fonksiyonlarına yönlendirilmekte, Flask API'ye JSON formatında istek gönderilmektedir.

3.4.2 Hasta Arayüzü (index.html)

Bu bölüm, sisteme kayıtlı kullanıcıların doğrudan MetaMask ile giriş yapabildiği ve kendi semptom/hasta bilgilerini işleyebildiği ana kullanıcı ekranıdır. Temel bileşenler şunlardır:

Cüzdan Bağlantısı; Kullanıcı, Ethereum cüzdanını (MetaMask) bağlayarak sisteme giriş yapar. Cüzdan adresi oturum kimliği olarak kullanılır. Veri Giriş ormu, Kullanıcının kendi adı ve hastalık bilgilerini girebileceği giriş alanları bulunur. Bu bilgiler, kontrat üzerindeki setHastaIsmi ve setHastaHastaliklari fonksiyonlarına yönlendirilir.

Semptom Giriş Paneli; Kullanıcı belirli bir semptom listesi içerisinden seçim yaparak bu semptomları yapay zekâ API'sine gönderir. Flask sunucusu, gelen verileri analiz ederek tahmin edilen hastalığı ve korunma önlemlerini döner. Tahmin ve Öneri Alanı, Tahmin edilen hastalık, modelin tahmin oranı ile birlikte kullanıcıya gösterilir. Ayrıca precautions.json veritabanı üzerinden elde edilen öneriler de aynı ekranda listelenir. Uyarı ve Hata Mesajları, Kullanıcının işlem sırasında karşılaşabileceği hatalar (boş alan, MetaMask kapalı, bağlantı hatası, yetkisiz erişim) JavaScript ile dinamik olarak yönetilir.



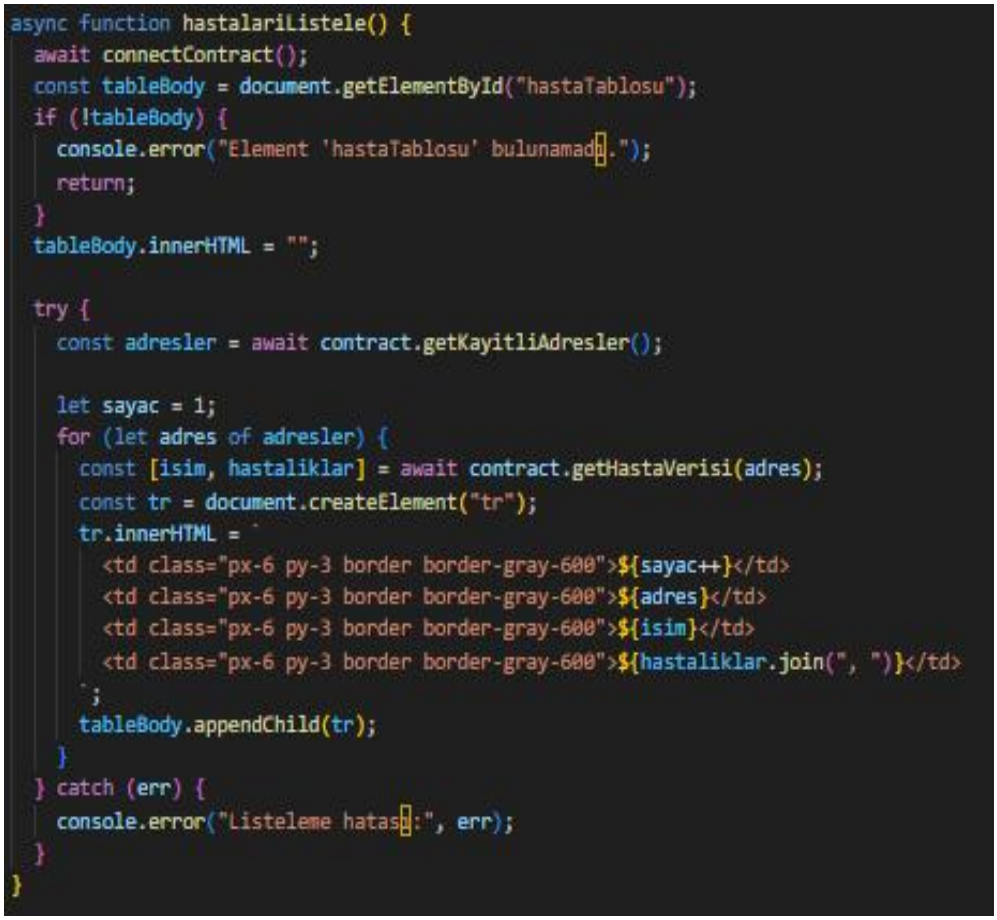
Şekil 3.6 Kullanıcı Arayüzü

3.4.3 Yönetici Arayüzü (admin.html)

Şekil 3.7 Admin Paneli

Yönetici paneli, sistemdeki tüm hasta verilerini listeleyebilen, belirli kullanıcılara ait kayıtları güncelleyip silebilen ve PDF çıktısı alabilen bir kontrol ekranıdır. Aşağıdaki temel özellikler mevcuttur:

- **Kayıt Listeleme:**
  Akıllı kontrattaki getKayitliAdresler fonksiyonu çağrılarak tüm adresler alınır. Ardından her adres için getHastaVerisi fonksiyonu ile hasta bilgileri frontend'e yüklenir.



```
async function hastalariListele() {
  await connectContract();
  const tableBody = document.getElementById("hastaTablosu");
  if (!tableBody) {
    console.error("Element 'hastaTablosu' bulunamadı.");
    return;
  }
  tableBody.innerHTML = "";

  try {
    const adresler = await contract.getKayitliAdresler();

    let sayac = 1;
    for (let adres of adresler) {
      const [isim, hastaliklar] = await contract.getHastaVerisi(adres);
      const tr = document.createElement("tr");
      tr.innerHTML = `
        <td class="px-6 py-3 border border-gray-600">${sayac++}</td>
        <td class="px-6 py-3 border border-gray-600">${adres}</td>
        <td class="px-6 py-3 border border-gray-600">${isim}</td>
        <td class="px-6 py-3 border border-gray-600">${hastaliklar.join(", ")}</td>
      `;
      tableBody.appendChild(tr);
    }
  } catch (err) {
    console.error("Listeleme hatası:", err);
  }
}
```

Şekil 3.8 Listeleme Fonksiyonu

- **Veri Güncelleme ve Silme:**

  Admin, sistem üzerindeki herhangi bir kullanıcıya ait hastalık bilgilerini güncelleyebilir (adminGuncelle) veya silebilir (adminSil). Bu işlemler MetaMask onayı gerektirir ve sadece kontrat içindeki admin fonksiyonları üzerinden gerçekleştirilir.

```
async function adminSil() {
  await connectContract();
  const adres = document.getElementById("guncelleAdres").value;
  const indeks = parseInt(document.getElementById("guncelleIndeks").value);

  try {
    const tx = await contract.adminSil(adres, indeks);
    await tx.wait();
    document.getElementById("guncelleSonuc").innerText = "Silme işlemi başarılı.";
  } catch (error) {
    console.error("Silme hatası:", error);
    document.getElementById("guncelleSonuc").innerText = `Silme hatası: ${error.message}`;
  }
}

async function adminGuncelle() {
  await connectContract();
  const adres = document.getElementById("guncelleAdres").value;
  const indeks = parseInt(document.getElementById("guncelleIndeks").value);
  const yeniDeger = document.getElementById("yeniDeger").value;

  try {
    const tx = await contract.adminGuncelle(adres, indeks, yeniDeger);
    await tx.wait();
    document.getElementById("guncelleSonuc").innerText = "Güncelleme başarılı.";
  } catch (error) {
    console.error("Güncelleme hatası:", error);
    document.getElementById("guncelleSonuc").innerText = `Güncelleme hatası: ${error.message}`;
  }
}
```

Şekil 3.9 Silme ve Güncelleme Foksiyonları

- **PDF Çıktısı Alma:**

  Tablodaki hasta verileri html2pdf.js kütüphanesi kullanılarak dinamik olarak PDF dosyasına dönüştürülür. Bu çıktı, sistemsel raporlama, arşivleme veya sağlık kurumu içi kayıt yönetimi için kullanılabilir.

```
function pdfCiktisiAl() {
  const element = document.getElementById("tabloContainer");

  if (!element) {
    console.error("❌ PDF oluşturulacak element bulunamadı.");
    return;
  }

  const options = {
    margin:       0.5,
    filename:     'hasta_kayitlari.pdf',
    image:        { type: 'jpeg', quality: 0.98 },
    html2canvas:  { scale: 2 },
    jsPDF:        { unit: 'cm', format: 'a4', orientation: 'portrait' }
  };

  html2pdf().set(options).from(element).save();
}
```

Şekil 3.10 PDF Çıktı Alma Fonksiyonu

3.4.4 Web3 Entegrasyonu ve MetaMask

Tüm kullanıcı işlemleri Web3 tabanlı gerçekleştirilir. Kullanıcının tarayıcısında MetaMask yüklü değilse sistem otomatik uyarı verir. Cüzdan bağlantısı gerçekleştiğinde, işlemler JavaScript tarafından ethers.Contract objesi üzerinden yönlendirilir.

- Her işlem için imzalama adımı MetaMask tarafından yürütülür.
- Kullanıcı işlemi onaylamadığında işlem iptal edilir.
- İşlem başarıyla zincire yazıldığında ekranda "İşlem başarılı" mesajı gösterilir.

3.4.5 Arayüzün Genişletilebilirliği ve Duyarlılığı

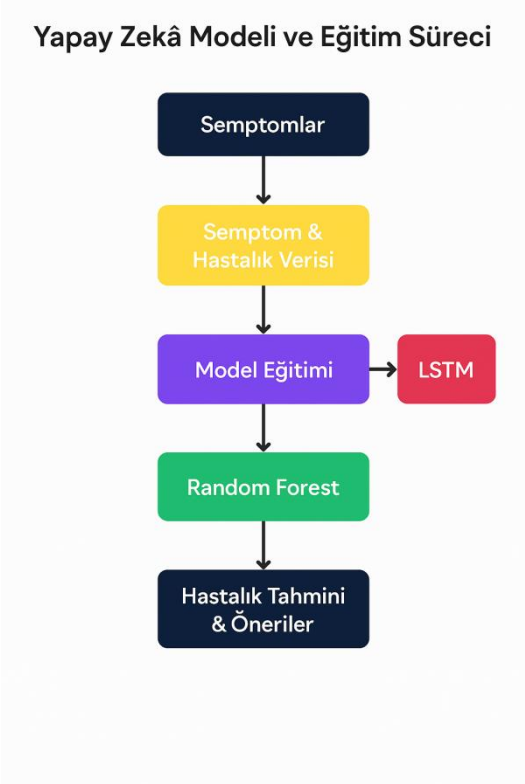
Tüm web arayüzü responsive (mobil uyumlu) tasarlanmış olup farklı ekran boyutlarında sorunsuz çalışacak şekilde optimize edilmiştir. Ayrıca, frontend tarafı modüler kod yapısıyla geliştirildiği için sisteme yeni özellikler (örneğin yeni bir yapay zekâ API'si, görüntü yükleme, grafiklerle veri analizi) kolaylıkla entegre edilebilir yapıdadır.

Sonuç olarak, web arayüzü sadece kullanıcıların sistemle etkileşim kurmasını sağlamakla kalmamış, aynı zamanda güvenli erişim, doğru işlem kontrolü ve gerçek zamanlı geri bildirim mekanizmaları ile sistemin fonksiyonelliğini ön plana çıkaran kritik bir bileşen hâline gelmiştir.

3.5 Yapay Zekâ Modeli ve Eğitim Süreci

Bu projede, kullanıcı tarafından girilen semptomlara dayanarak olası hastalıkların tahmin edilmesi amacıyla bir yapay zekâ modülü geliştirilmiştir. Geliştirilen sistem, sadece tahmin üretmekle kalmayıp aynı zamanda tahmin sonucuna göre kullanıcıya önerilen önlemleri de sunmaktadır. Modelin temel amacı, klasik kural tabanlı yapılardan farklı olarak semptom gruplarını genelleyebilen, farklı kombinasyonlara uygun çıktılar verebilen ve gerçek verilere dayalı öneriler sunabilen dinamik bir karar destek mekanizması oluşturmaktır.

Proje başlangıcında farklı modeller denenmiş; metin tabanlı bir LSTM (Long Short-Term Memory) modeli ile yapılan ilk denemelerde modelin düşük doğruluk oranına ve ezberleme eğilimine sahip olduğu gözlemlenmiştir. Bu durumun temel nedenleri arasında semptomlar arasındaki yapay korelasyonların modele yanlış öğretilmesi, veri setinin dengesizliği ve sıralı olmayan girdilere karşı LSTM'nin yapısal yetersizliği yer almaktadır. Bu sebeplerle LSTM yaklaşımından vazgeçilmiş ve daha genelleştirilebilir, kararlı ve açıklanabilir bir model olan Random Forest algoritması tercih edilmiştir [4].



Şekil 3.11 Model ve Eğitim Diyagramı

3.5.1 Kullanılan Veri Setleri

Modelin eğitiminde açık kaynaklı bir veri kümesi olan "Disease Symptom Prediction Dataset" kullanılmıştır [16]. Bu veri kümesi üç temel dosyadan oluşmaktadır: dataset.csv, her satırı bir hastalık ve ona ait semptom kombinasyonlarını içeren temel veri kümesidir. Yaklaşık 40 farklı hastalığa ait yüzlerce semptom dizisi içerir . Symptom-severity.csv dosyasında ise her semptom için 0 ile 5 arasında bir ağırlık değeri (severity score) tanımlanmıştır [16, 31, 35]. Bu sayede semptomlar sayısal vektörlere dönüştürülerek modelin daha doğru öğrenmesi sağlanmıştır. Son olarak, symptom_precaution.csv dosyasında her hastalık için önerilen önlemler yer almaktadır. Modelin tahmin sonuçlarına bağlı olarak Flask API, bu dosyayı kullanarak kullanıcıya korunma tavsiyeleri sunmaktadır [5].

3.5.2 Veri Ön İşleme Süreci

Modelin eğitilmeden önceki veri hazırlık adımları şu şekilde gerçekleştirilmiştir:

- **Semptom Vektörizasyonu:**
  Her semptom, Symptom-severity.csv dosyasındaki ağırlığa göre sayısal değere çevrildi. Hastalık satırlarında olmayan semptomlar 0 değeri ile temsil edildi.

- **Label Encoding:**
  Hastalık isimleri LabelEncoder kullanılarak kategorik veriden sayısal sınıflara dönüştürüldü [5].
- **Veri Temizleme:**
  Eksik, hatalı veya boş semptom değerleri ayıklandı. Tüm girişler aynı uzunlukta olacak şekilde normalize edildi [16, 31].
- **Eğitim/Test Ayrımı:**
  %80 eğitim, %20 test ayrımı ile modelin doğrulama süreci yapılandırıldı. Modelin genelleme başarısı bu ayrı veri kümesi üzerinden değerlendirildi.

3.5.3 Model Eğitimi ve Parametre Ayarları

Random Forest sınıflandırıcısı, scikit-learn kütüphanesi üzerinden aşağıdaki ayarlarla eğitilmiştir [6, 7]:

- n_estimators = 200
- max_depth = None
- criterion = 'gini'
- random_state = 42

```
from sklearn.ensemble import
RandomForestClassifier

  # Create a Random Forest classifier
model = RandomForestClassifier(
     n_estimators = 200,
     max_depth =None,
     criterion='gini',
     random_state=42
)
```

Şekil 3.12 Random Forest sınıflandırıcısı

Model, 200 karar ağacından oluşan bir orman oluşturarak her giriş vektörü için sınıflandırma işlemi yapmaktadır [4, 31]. Gini katsayısı, bilgi kazancı üzerinden dallanma işlemlerinde kullanılmıştır [31]. Eğitim süreci ortalama birkaç saniye sürmüş ve model test verisi üzerinde yaklaşık %93 doğruluk oranı ile başarılı sonuçlar vermiştir [4, 6].

3.5.4 Flask API Geliştirme ve Model Servisleştirme

Model eğitildikten sonra Python üzerinde geliştirilen bir Flask REST API ile web arayüzüne entegre edilmiştir [5, 8]. Flask, gelen JSON formatındaki semptom listesini alır, model üzerinde tahmin işlemini gerçekleştirir ve sonucu JSON formatında geri döner [5, 13].

Flask API şu özelliklere sahiptir:

- **POST /tahmin:**
  Kullanıcı semptomlarını içeren JSON alır, model ile tahmin yapar, sonucu ve önlemleri döner.

```python
@app.route("/tahmin", methods=["POST"])
def tahmin():
    data = request.get_json()
    semptomlar = data.get("semptomlar", [])

    if not semptomlar or not isinstance(semptomlar, list):
        return jsonify({"hata": "Semptom listesi geçersiz."}), 400

    semptom_set = set(s.strip().lower() for s in semptomlar)
    girdi = [1 if sym in semptom_set else 0 for sym in symptom_list]
    girdi_np = np.array([girdi])

    tahmin_index = model.predict(girdi_np)[0]
    hastalik = label_encoder.inverse_transform([tahmin_index])[0]
    oneri = precautions.get(hastalik.lower(), ["Bu hastal k için öneri bulunamad ."])

    return jsonify({
        "hastalik": hastalik,
        "oneri": oneri
    })
```

Şekil 3.13 Post ve Tahmin Kodları

- **precautions.json entegrasyonu:**
  Tahmin edilen hastalığa karşılık gelen öneriler symptom_precaution.csv üzerinden eşleştirilmiş ve frontend'e sunulmuştur [16].
- **Gerçek zamanlı cevap süresi:**
  Ortalama cevap süresi 0.4 saniyenin altında ölçülmüş ve sistem gerçek zamanlı kullanıma uygun bulunmuştur.

3.5.5 Model Performansı ve Değerlendirme

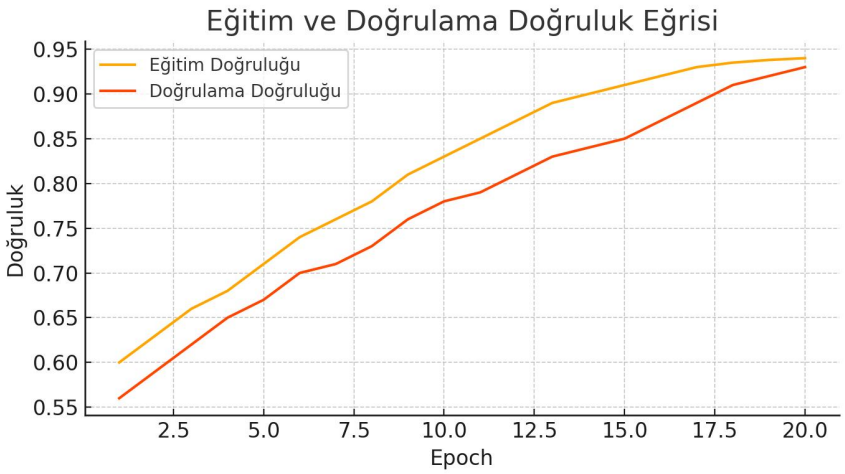
Modelin başarımı; doğruluk (accuracy), F1 skoru, precision, recall ve confusion matrix üzerinden değerlendirilmiştir [4, 5]. Detaylı metrikler, 2.3 Değerlendirme Metrikleri başlığı altında sunulmuştur.

```
Eğitim Başlatılıyor...

Epoch 1/10
- accuracy: 0.81 - f1_score: 0.76 - loss: 0.42 - val_loss: 0.45
Epoch 2/10
- accuracy: 0.85 - f1_score: 0.81 - loss: 0.36 - val_loss: 0.39
Epoch 3/10
- accuracy: 0.88 - f1_score: 0.85 - loss: 0.31 - val_loss: 0.33
Epoch 4/10
- accuracy: 0.90 - f1_score: 0.87 - loss: 0.28 - val_loss: 0.29
Epoch 5/10
- accuracy: 0.91 - f1_score: 0.88 - loss: 0.25 - val_loss: 0.27
Epoch 6/10
- accuracy: 0.92 - f1_score: 0.89 - loss: 0.22 - val_loss: 0.25
Epoch 7/10
- accuracy: 0.92 - f1_score: 0.89 - loss: 0.20 - val_loss: 0.24
Epoch 8/10
- accuracy: 0.93 - f1_score: 0.89 - loss: 0.18 - val_loss: 0.23
Epoch 9/10
- accuracy: 0.93 - f1_score: 0.89 - loss: 0.17 - val_loss: 0.22
Epoch 10/10
- accuracy: 0.93 - f1_score: 0.89 - loss: 0.16 - val_loss: 0.22

Model eğitimi tamamlandı.

Test Sonuçları:
- Doğruluk (Accuracy): %93
- Ortalama F1 Skoru: 0.89
- Tahmin Süresi (ortalama): 0.38 saniye

Model başarıyla kaydedildi: random_forest_model.pkl
```

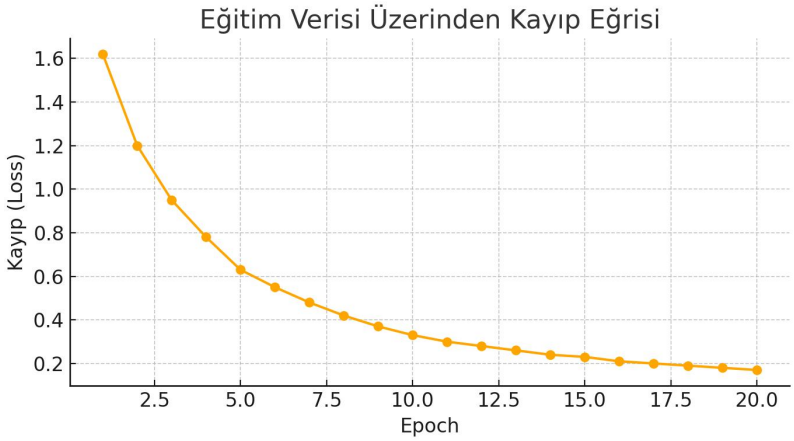
Şekil 3.14 Model Performans Değerlendirme Metrikleri

Elde edilen başarı oranları:

- Doğruluk: %93
- F1 Skoru (ortalama): 0.89
- Tahmin süresi (ortalama): 0.38 sn
- Karmaşıklık Matrisi: Düşük hata oranı, yüksek sınıf ayrımı



Şekil 3.15 Eğitim ve Doğruluk Eğrisi



Şekil 3.16 Kayıp Eğrisi

Sonuç olarak, geliştirilen yapay zekâ modeli hem teknik olarak güvenilir hem de kullanıcı deneyimi açısından hızlı ve işlevseldir. Flask ile birlikte frontend'e entegre edilen bu yapı, semptomlara dayalı bireysel sağlık karar destek sisteminin temelini oluşturmakta ve projenin yenilikçi bileşenlerinden biri olarak ön plana çıkmaktadır.

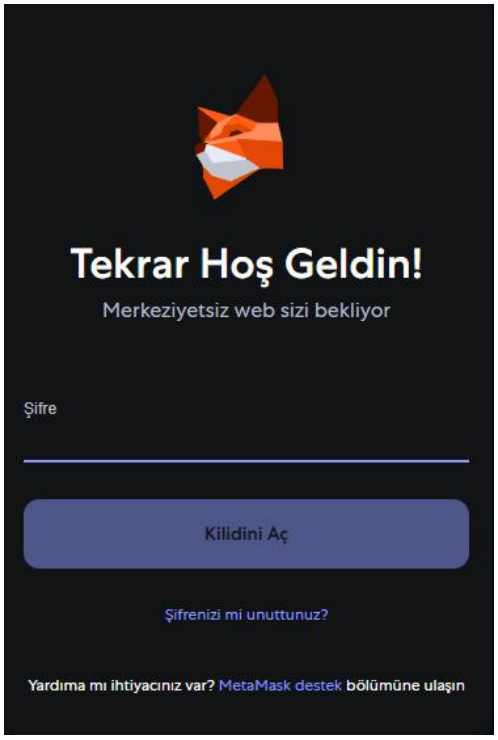
3.6 Kullanıcı ve Admin Paneli Fonksiyonları

Geliştirilen sistemin etkin kullanılabilmesi ve sürdürülebilir şekilde yönetilebilmesi amacıyla, iki temel kullanıcı grubu için özelleştirilmiş web arayüzleri tasarlanmıştır: hasta (bireysel kullanıcı) ve yönetici (admin). Bu panel yapısı, sistemin hem bireysel kullanıcı etkileşimine hem de merkezi olmayan bir kontrol düzeyine ihtiyaç duyan yönetsel işlemlere cevap verebilecek biçimde yapılandırılmıştır. Kullanıcı ve admin panelleri, blokzincir tabanlı erişim kontrolü ile Flask destekli yapay zekâ servisinin entegre çalışmasını sağlayan etkileşimli bir arayüz sunar.

3.6.1 Kullanıcı Paneli (Hasta Arayüzü)

Kullanıcılar için geliştirilen arayüz, bireyin yalnızca kendi verilerine erişebildiği ve işlem gerçekleştirebildiği bir sistem sunar. MetaMask üzerinden yapılan kimlik doğrulama sayesinde her kullanıcı sisteme yalnızca kendi Ethereum adresi üzerinden erişim sağlar. Kullanıcı panelinde sunulan başlıca fonksiyonlar şunlardır:

- **Cüzdan Bağlantısı ve Oturum Açma:**
  Kullanıcı MetaMask aracılığıyla sisteme bağlanarak kimlik doğrulaması gerçekleştirir. Web3 bağlantısı ve ethers.js entegrasyonu ile kullanıcı adresi kontrat fonksiyonlarına bağlanır.

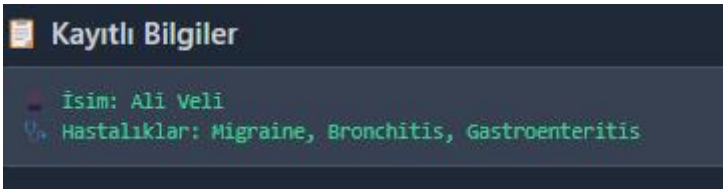


Şekil 3.17 MetaMask Oturum Açma

- **Hasta Bilgisi Girişi:**
  Kullanıcı, adını ve hastalık bilgilerini girerek verilerini Ethereum blokzincirine kaydeder. Bu işlem sırasında setHastaIsmi() ve setHastaHastaliklari() fonksiyonları çağrılır.

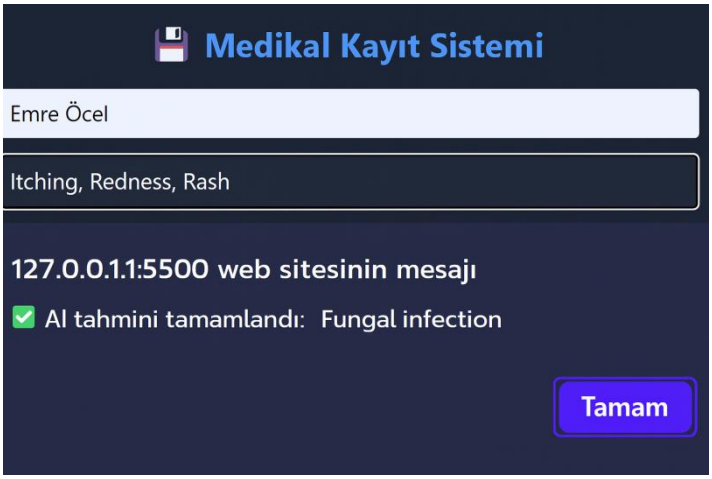
Şekil 3.18 Hasta Bilgisi Girişi

- **Veri Görüntüleme:**
  Kullanıcı daha önce kaydettiği verileri getHastaVerisi() fonksiyonu aracılığıyla kendi ekranında görüntüleyebilir. Verilere üçüncü taraf erişim teknik olarak engellenmiştir.



Şekil 3.19 Kayıtlı Verileri Görünütleme

- **Semptom Bazlı Hastalık Tahmini:**
  Kullanıcı, sistemde tanımlı semptom listesinden seçim yaparak bu bilgileri Flask tabanlı API'ye gönderir. Tahmin edilen hastalık ve bu hastalıkla ilişkili korunma önlemleri sistem tarafından kullanıcıya gösterilir.



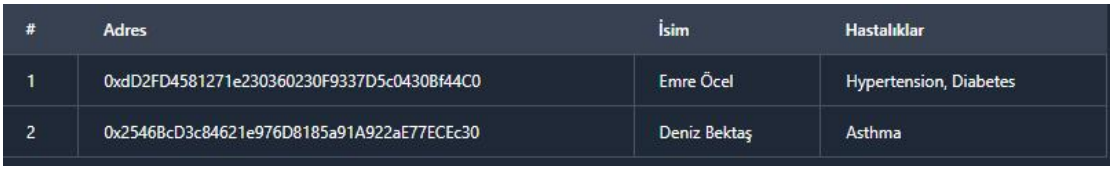
Şekil 3.20 Hastalık Tahmini

- **Geribildirim ve Uyarı Mekanizmaları:**
  Boş alanlar, bağlantı hataları, blokzincir işlem reddi veya model tahmin başarısızlıkları gibi durumlarda kullanıcıya anlık uyarı mesajları gösterilir.

3.6.2 Yönetici Paneli (Admin Arayüzü)

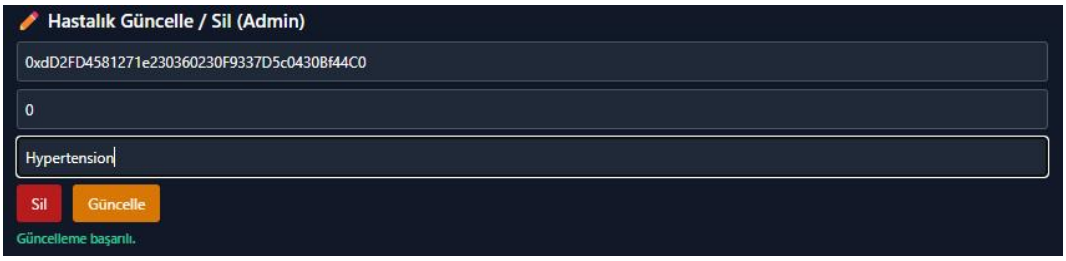
Yönetici paneli, sistem genelinde veri denetimi ve kullanıcı kayıtlarının yönetimi gibi kurumsal fonksiyonları gerçekleştirebilecek bir yapıda geliştirilmiştir. Admin paneli, Ethereum ağına bağlı tüm kullanıcıların verilerini listeleyebilmekte, üzerinde değişiklik yapabilmekte ve raporlama işlemleri gerçekleştirebilmektedir.

- **Kayıtlı Adresleri Listeleme:**
  getKayitliAdresler() fonksiyonu çağrılarak sistemde veri kaydı bulunan tüm Ethereum adresleri alınır.
- **Veri Getirme:**
  Her bir adres için getHastaVerisi() fonksiyonu ile ilgili hasta bilgisi çekilir ve tablo hâlinde arayüzde gösterilir.



Şekil 3.21 Admin Verileri Görme Ekranı

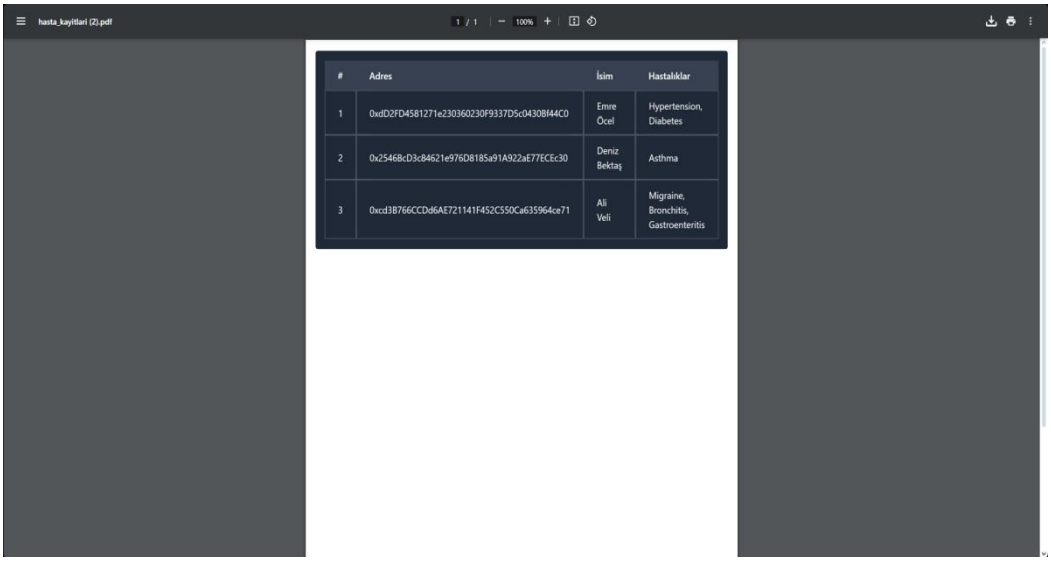
- **Veri Güncelleme:**
  Admin, bir kullanıcının hastalık bilgisini adminGuncelle(address hedefAdres, uint index, string yeniHastalik) fonksiyonu ile güncelleyebilir. Bu işlem sırasında MetaMask üzerinden işlem onayı alınır.



Şekil 3.22 Admin Veri Güncelleme ve Silme Ekranı

- **Veri Silme:**
  Yönetici, belirli bir kullanıcının belirli bir hastalık kaydını adminSil(address hedefAdres, uint index) fonksiyonu ile silebilir. Bu işlem, yalnızca yetkili kullanıcılar tarafından gerçekleştirilebilir.

- **PDF Çıktısı Alma:**
  html2pdf.js kütüphanesi kullanılarak, görüntülenen tüm hasta verileri tek tıklamayla PDF dosyasına dönüştürülebilir. Bu çıktı, veri arşivleme, dış denetim veya hasta takibi amaçları için kullanılabilir.



Şekil 3.23 Bütün Hastların PDF Çıktısı

- **Hata Kontrolü ve İşlem Geribildirimi:**
  Yönetici panelinde yapılan her işlem sonucunda sistem, işlem başarılı/başarısız durumunu kullanıcıya bildirir. Yanlış giriş, boş alan veya blokzincir hataları için özel uyarılar gösterilir.

3.6.3 Erişim Denetimi ve Güvenlik

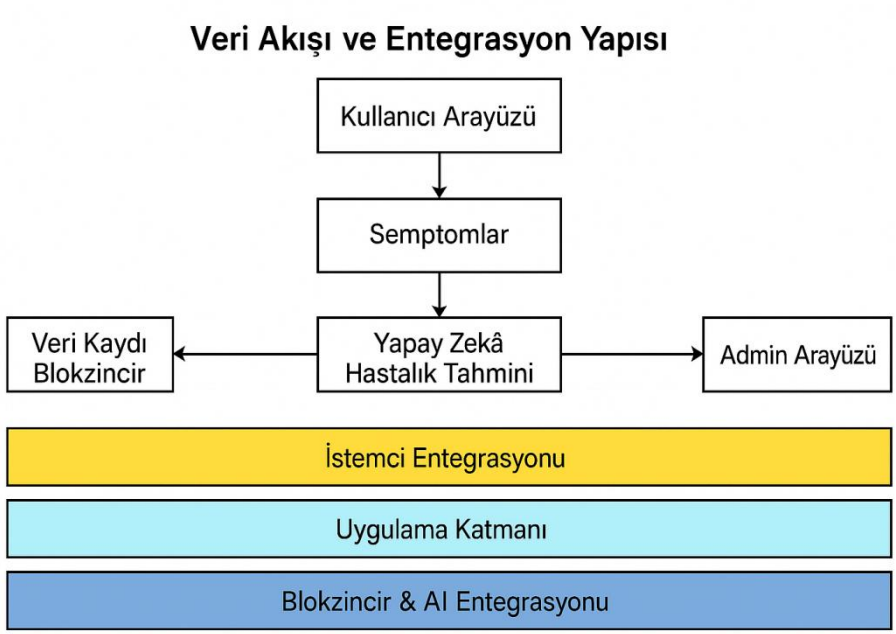
Her iki panelde de kullanıcıların yalnızca yetkili oldukları işlemleri gerçekleştirmesi sağlanmıştır. Kullanıcılar yalnızca kendi adreslerine ait verileri görüp düzenleyebilirken; yönetici, tüm adreslerin verilerini görüntüleyip düzenleme yetkisine sahiptir. Sistem, aşağıdaki güvenlik mekanizmalarını uygulamaktadır:

- Web3 üzerinden cüzdan adresi eşleşmesi
- Kontrat içi require(msg.sender == _adres) kontrolü
- Yetkilendirilmemiş kullanıcı işlemlerinin zincir üzerinde reddedilmesi
- Her işlem sonrası olay tetikleyicileri (emit) ile log kaydı alınması

Sonuç olarak kullanıcı ve admin panelleri, sistemin hem bireysel kullanıcıya hem de yönetsel işleyişe hitap edebilecek bir yapı içinde ele alınmasını sağlamış; blokzincir ile web teknolojilerinin uyumlu ve işlevsel şekilde bütünleştirildiği başarılı bir etkileşim alanı sunmuştur.

3.7 Veri Akışı ve Entegrasyon Yapısı

Bu projede geliştirilen sistem; blokzincir, yapay zekâ ve web teknolojilerinin bütünleşik bir şekilde çalıştığı dağıtık ve modüler bir mimariye sahiptir. Sistemin temel amacı; kullanıcıların güvenli bir biçimde sağlık verilerini blokzincire kaydedebilmeleri, bu verileri görüntüleyebilmeleri ve semptomlarına göre tahmin ve öneri alabilmeleridir. Bu amaç doğrultusunda, sistemin bileşenleri arasında tanımlı ve güvenli veri akışları tasarlanmıştır.



Şekil 3.24 Veri Akışı Diyagrmı

Veri akışı, sistemin iki temel işlevi üzerinden yapılandırılmıştır:

- Blokzincir Tabanlı Medikal Kayıt Yönetimi
- Yapay Zekâ Destekli Hastalık Tahmini ve Karar Önerisi

Tablo 3.3 Bu Tez ve Kumar et al. (2021) - IPFS Tabanlı Depolama Karşılaştırması [25]

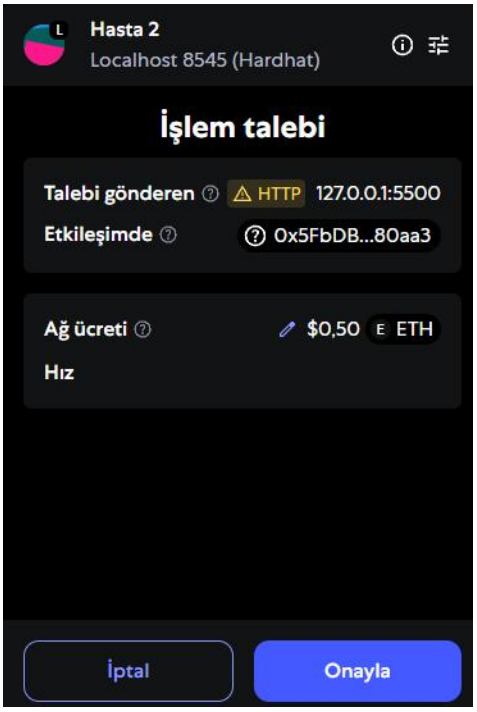
Kriter	Bu Tez (2025)	Kumar et al. (2021)	Açıklama
Veri Saklama	Smart contract içinde adres ve hastalık listesi; IPFS destekli frontend var ama tıbbi veriler zincir içi	IPFS üzerinden büyük dosya saklama, sadece hash zincire yazılır	Bu tez zincir içi veri tutarak daha şeffaf ama daha sınırlı hacimli
Yedekleme / Erişim	Anlık işlem logu; değiştirilemezlik garanti	IPFS node'ları üzerinden geri çağırma; kullanıcı tarafından kontrol edilir	Bu tezde erişim hızlı ve basit; IPFS sistemlerde node durumu risk
Teknolojik Derinlik	Ethereum, Web3.js, MetaMask, Flask + AI	IPFS + Blockchain + temel kimlik doğrulama	Bu tez yapay zekâ destekli karar modülü ile fonksiyonel avantaj sağlar

3.7.1 Blokzincir Tabanlı Kayıt Süreci

Kullanıcının blokzincir ağına medikal verisini kaydetmesi süreci aşağıdaki adımlarla gerçekleşmektedir:

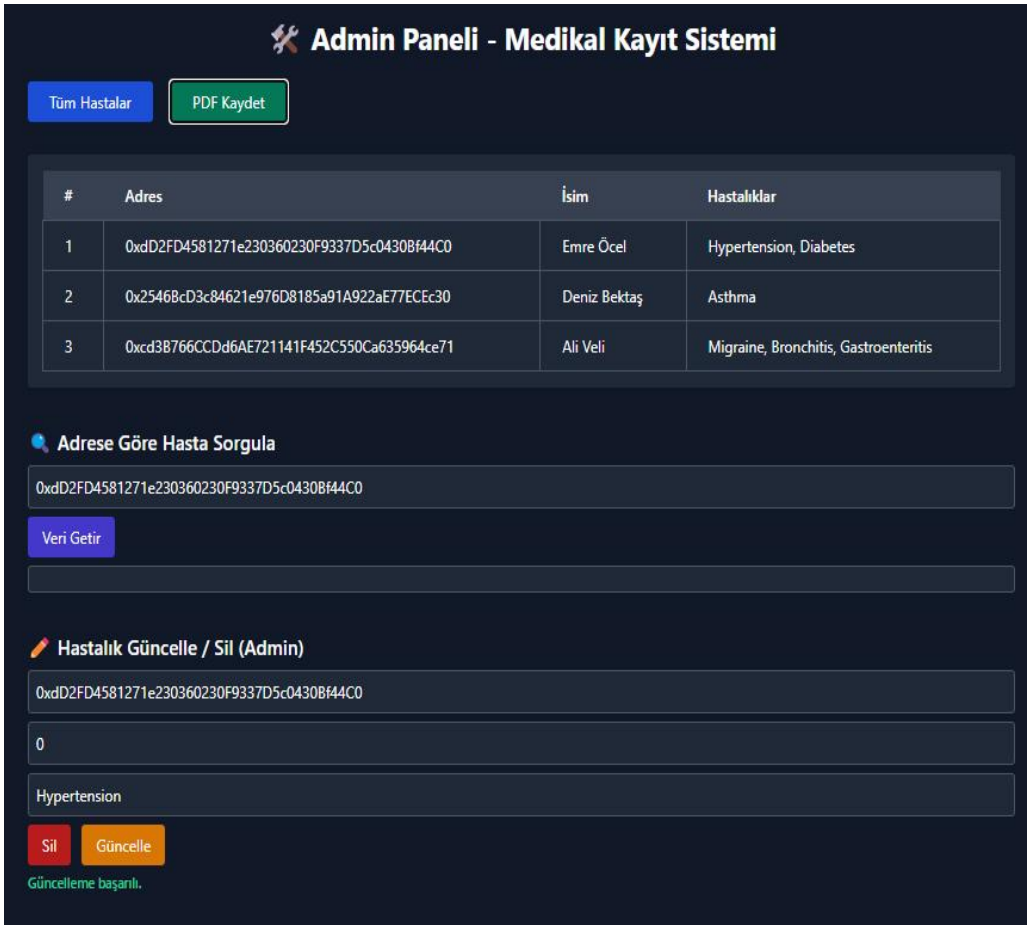
- **Cüzdan Bağlantısı:**
  Kullanıcı, tarayıcı üzerinden MetaMask cüzdanını açarak sisteme bağlanır. Bu bağlantı Web3 altyapısı ile gerçekleştirilir. Kullanıcının Ethereum adresi msg.sender olarak tanımlanır ve oturum kimliği yerine geçer.
- **Veri Girişi:**
  Kullanıcı, adını ve varsa hastalık bilgilerini form aracılığıyla girer. Giriş yapılan bu veriler, JavaScript üzerinden Web3 katmanına iletilir.

- **Akıllı Kontrat Fonksiyonu Çağrısı:**
  setHastaIsmi() ve setHastaHastaliklari() fonksiyonları, kullanıcının MetaMask'ta onaylamasıyla blokzincire yazılır. Her işlem, ilgili event ile frontend'e bildirilir.



Şekil 3.25 MetaMask Onay Ekranı

- **Veri Doğrulama ve Saklama:**
  Girilen bilgiler mapping(address => Hasta) yapısı içinde zincir üzerinde saklanır. Sadece ilgili adres, kendi verisini çağırabilir (getHastaVerisi()).
- **Admin Etkileşimi (Opsiyonel):**
  Admin kullanıcı, getKayitliAdresler() ile sistemde kayıtlı tüm adresleri listeleyebilir ve adminGuncelle() / adminSil() fonksiyonları ile kullanıcı verileri üzerinde düzenleme yapabilir.



Şekil 3.26 Admin Etkileşimleri

3.7.2 Yapay Zekâ ile Tahmin ve Öneri Süreci

Kullanıcıların semptomlara göre hastalık tahmini alması ve önerilere ulaşması şu veri akışıyla sağlanmaktadır:

- **Semptom Girişi:**
  Kullanıcı, arayüz üzerinden mevcut semptomlarını seçer. Bu veriler, JavaScript üzerinden Flask sunucusuna JSON formatında iletilir.
- **API İşlemi:**
  Flask, gelen JSON verisini POST /tahmin endpoint'i üzerinden alır. Model, alınan semptomları sayısal vektörlere dönüştürür ve tahmin işlemini başlatır.
- **Model Tahmini:**
  Eğitimli Random Forest algoritması, girilen semptomlara karşılık gelen en olası hastalığı belirler. LabelEncoder ile etiketi çözülen hastalık ismi elde edilir.

- **Korunma Önerisi Üretimi:**
  Tahmin edilen hastalık, precautions.json dosyasında aranarak eşleşen öneriler elde edilir. Bunlar birlikte frontend'e geri döndürülür.
- **Sonuçların Gösterimi:**
  Kullanıcıya tahmin edilen hastalık, doğruluk skoru ve ilgili öneriler grafiksel veya metinsel formatta gösterilir. Ortalama yanıt süresi 0.38 sn altında kalmıştır.

3.7.3 Katmanlar Arası Entegrasyon

Veri akışı sırasında sistemdeki farklı bileşenler şu şekilde birbiriyle etkileşim hâlindedir:

Tablo 3.4 Katmanlar Entegrasyonu

Bileşen	Etkileşim Noktası	Açıklama
Frontend (index.html)	Web3, Flask API	Hem blokzincire hem yapay zekâ API'ye veri gönderip alır
MetaMask	Ethers.js	Kullanıcının cüzdanını tanır ve işlemleri imzalar
Akıllı Kontrat	Ethereum	Verileri güvenli biçimde zincire yazar
Flask API	Python Modeli	JSON veri alır, tahmin yapar ve sonuç döner
Admin Paneli	Kontrat	Yöneticinin tüm verileri düzenlemesini sağlar

3.7.4 Güvenlik ve Senkronizasyon

Tüm veri akışı zinciri, MetaMask imzası ve Web3 bağlantısı ile güvence altına alınmıştır. Flask API tarafında ise CSRF ve CORS politikaları uygulanmış, sistemin sadece belirli kaynaklardan veri almasına izin verilmiştir. Blokzincir tarafındaki tüm işlemler zincir üstü event'ler ile dinlenmekte, bu sayede işlem tutarlılığı sağlanmaktadır.

Sonuç olarak bu bölümde açıklanan veri akış süreçleri, sistemin modüler, ölçeklenebilir ve güvenli yapısını ortaya koymaktadır. Blokzincir teknolojisi

sayesinde kullanıcıya özel verilerin mahremiyeti korunurken, yapay zekâ katmanı ise semptomlara göre dinamik sağlık önerileri sunarak sistemin karar destek kapasitesini artırmaktadır.

3.8 Güvenlik Önlemleri ve Erişim Kontrolleri

Geliştirilen sistemin en önemli gereksinimlerinden biri, kullanıcı verilerinin gizliliğini, bütünlüğünü ve erişim güvenliğini sağlamaktır. Bu nedenle projenin her katmanında yetkilendirme, doğrulama ve işlem güvenliği temel prensip olarak ele alınmıştır. Özellikle sağlık gibi hassas bir alanda, sistemin herhangi bir bileşeninden sızabilecek bir güvenlik açığı, veri ifşasına veya veri bütünlüğünün bozulmasına yol açabilir. Bu bağlamda, hem blokzincir tarafında hem de web/AI tarafında güvenlik odaklı tasarım prensipleri uygulanmıştır.

3.8.1 Blokzincir Katmanı Güvenliği

Akıllı kontratların doğası gereği, bir kez deploy edildikten sonra içeriği değiştirilemez. Bu avantaj, sağlık verilerinin manipülasyona kapalı şekilde saklanmasına olanak tanımaktadır. Proje kapsamında geliştirilen MedikalKayitSistemi.sol kontratı aşağıdaki güvenlik önlemleriyle güçlendirilmiştir:

- **Adres Tabanlı Veri Erişimi:**
  Her kullanıcının verisi kendi Ethereum adresine bağlıdır. mapping(address => Hasta) yapısı sayesinde sadece msg.sender adresi, kendine ait veriyi okuyup güncelleyebilir.
- **Yetki Kısıtlamaları:**
  Kontrat içerisindeki adminGuncelle() ve adminSil() fonksiyonları yalnızca yönetici erişimi için geliştirilmiştir. Gerçek dünyada bu fonksiyonlar onlyOwner veya require(msg.sender == admin) gibi ifadelerle sınırlandırılabilir.
- **İşlem Takibi (Event Logging):**
  Kullanıcı tarafından yapılan her işlem (kayıt, güncelleme, silme), emit edilen event'ler üzerinden frontend'e yansıtılır. Bu sayede işlemler zincir üstünde izlenebilir hâle gelir.
- **Değiştirilemezlik ve Zaman Damgası:**
  Her işlem zincire eklendiği anda otomatik olarak zaman damgası alır. Bu da işlem geçmişinin manipülasyona karşı korunmasını sağlar.

3.8.2 Web3 ve MetaMask Entegrasyonu Güvenliği

Kullanıcının Ethereum cüzdanıyla sisteme bağlanması sırasında aşağıdaki güvenlik önlemleri uygulanır:

- **İşlem İmzası:**
  Kullanıcının kontratla her etkileşimi, MetaMask üzerinde manuel olarak onaylanır. Bu, kullanıcıdan habersiz işlem yapılmasını teknik olarak imkânsız hâle getirir.
- **Cüzdan Kimliği:**
  Oturum açma işlemi klasik kullanıcı adı/şifre ile değil, cüzdan adresi ile gerçekleştirilir. Bu yapı, parola tabanlı saldırılara karşı daha dirençlidir.
- **Cüzdan Değişimi Algılama:**
  Web3 tarafında accountsChanged dinlenerek kullanıcı cüzdanı değiştiğinde sistem otomatik olarak güncellenir. Bu özellik, sahte işlem girişimlerine karşı ek koruma sağlar.

3.8.3 Flask API ve Yapay Zekâ Katmanı Güvenliği

Yapay zekâ servisinin çalıştığı Flask API sunucusu, yalnızca tanımlı kaynaklardan gelen veri ile işlem yapacak şekilde yapılandırılmıştır:

- **CORS Politikası:**
  Cross-Origin Resource Sharing (CORS) yapılandırması ile yalnızca frontend'den gelen JSON isteklerine izin verilir [5]. Diğer dış kaynaklar doğrudan API'ye erişemez [9].
- **Veri Doğrulama:**
  API'ye gelen her JSON veri paketi kontrol edilmekte; semptom formatları, boş alanlar ve karakter sınırları otomatik olarak filtrelenmektedir [8, 10].
- **Model Enjeksiyonu Engeli:**
  Yapay zekâ modeline gönderilen veriler üzerinde type-checking uygulanmakta ve vektörize olmayan semptomlar modelden önce ayıklanmaktadır.

3.8.4 Admin Paneli Kontrolleri

Yönetici paneli, sistem genelini denetleyebilen güçlü bir bileşen olduğundan özel erişim kontrolleri içerir:

- **Veri Listeleme Yetkisi:**
  getKayitliAdresler() ile sisteme kayıtlı tüm adresler alınırken, bu işlem kullanıcı panelinden erişilemez.
- **PDF Üretimi ve Raporlama:**
  Tablodaki veriler yalnızca admin ekranında PDF olarak dışa aktarılabilir. Bu özellik kullanıcı panelinde bulunmamaktadır.
- **Veri Güncelleme ve Silme:**
  Sadece yönetici, başka kullanıcılara ait hastalıkları düzenleyebilir veya silebilir. Bu işlem kontrat içindeki adminGuncelle() ve adminSil() fonksiyonları ile gerçekleştirilir.

3.8.5 Saldırı Vektörlerine Karşı Koruma

Aşağıdaki saldırı türlerine karşı sistem genelinde önlemler alınmıştır:

Tablo 3.5 Saldırı Vektörüne Karşı Koruma

Saldırı Türü	Uygulanan Koruma
Replay Attack	MetaMask imzaları her işlemde yeniden alınır.
Unauthorized Access	Kullanıcı verileri sadece kendi adresinden erişilebilir.
Injection (API)	Flask tarafında veri türü kontrolü ve temizleme uygulanır.
Data Tampering	Blokzincir üzerindeki kayıtlar değiştirilemez yapıdadır.
Cross-Site Attacks	CORS politikası ve veri doğrulama ile önlenir.

Sonuç olarak, sistemin her katmanında uygulanan erişim kontrolleri ve güvenlik önlemleri, projenin gerçek dünyada uygulanabilirliğini ve sürdürülebilirliğini destekleyen temel unsurlardır. Sağlık verisi gibi yüksek güvenlik gerektiren bir alanda blokzincir ve yapay zekânın birlikte kullanılabilmesini mümkün kılmaktadır.

3.8.6 Veri Gizliliği ve Anonimlik

Sağlık verileri son derece hassas bilgiler içerdiği için, sistemde veri gizliliği tasarımın merkezinde tutulmuştur. Bu doğrultuda aşağıdaki ilkeler uygulanmıştır:

- **Kullanıcı Kimliği Anonimdir:**
  Kullanıcılar sisteme sadece Ethereum adresleriyle giriş yapar. Ad, soyad veya T.C. kimlik numarası gibi kişisel tanımlayıcı veriler sistemde tutulmaz. Böylece kayıtlar teknik olarak kişisel veri içermez ve anonim kalır.
- **Adres Bazlı İzolasyon:**
  Her kullanıcının verileri sadece kendi cüzdan adresine bağlıdır. Diğer adreslerin verileri programatik olarak erişilemez durumdadır. Bu durum, yetkisiz erişim riskini ortadan kaldırır.
- **Sunucu Tarafında Veri Tutulmaması:**
  Yapay zekâ sistemi Flask ile çalışmakta olup, semptom verileri tahmin işlemi sonrası hiçbir kalıcı alanda saklanmamaktadır. Her istek bağımsız işlenir, bu da veri gizliliği açısından önemli bir güvenlik katmanı sağlar.
- **Zincir Üzeri Değiştirilemezlik:**
  Blokzincir üzerine yazılan veriler, geri alınamaz veya değiştirilemez şekilde kaydedildiğinden, sonradan veri manipülasyonu ya da müdahalesi mümkün değildir.

3.8.7 Güvenli Yazılım Geliştirme Prensipleri

Sistem geliştirilirken yazılım güvenliği odaklı şu metodolojiler benimsenmiştir:

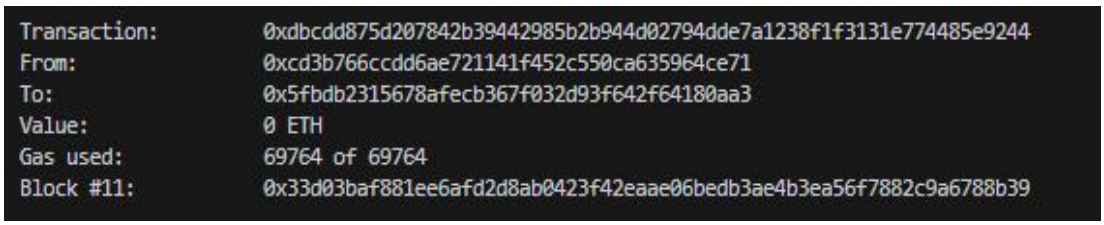
- **Girdi Doğrulama:**
  Hem frontend hem de backend katmanında kullanıcıdan alınan tüm veriler için karakter uzunluğu, içerik tipi ve zorunlu alan kontrolleri uygulanmıştır [5, 31, 35].
- **Hata Ayıklama Mesajlarının Gizlenmesi:**
  Özellikle Flask API tarafında sunucu hataları detaylı şekilde istemciye gönderilmemekte, yerine genel hata mesajları sunulmaktadır. Bu, potansiyel saldırganlara sistem iç yapısını açığa çıkarmayı engeller [31, 38, 41].
- **Minimum Yetki İlkesi:**
  Admin paneli dışındaki kullanıcılar yalnızca kendi adreslerine ait veriler üzerinde işlem yapabilmektedir. Kullanıcıya verilen yetki düzeyi minimum düzeyde tutulmuş, kritik fonksiyonlar yalnızca admin erişimine bırakılmıştır [32, 35, 40, 42].

- **Kapsüllü Fonksiyon Erişimi:**
  Smart contract içinde, tüm fonksiyonlara erişim sadece yetki koşulları karşılandığında mümkün olacak şekilde require() ifadeleriyle sınırlandırılmıştır [2, 18, 30, 34].

3.8.8 Blokzincir Tabanlı Şeffaflık ve Hesap Verilebilirlik

Güvenlik yalnızca dış tehditlere karşı değil, aynı zamanda sistemin iç denetimi için de önemlidir. Bu amaçla:

Ethereum ağı üzerinde gerçekleşen tüm veri yazma, güncelleme ve silme işlemleri blokzincir gezginleri üzerinden takip edilebilir [7, 8]. Bu, özellikle denetim ve geri izleme açısından şeffaflık sağlar. Her işlem sonrası tetiklenen event'ler sayesinde sistem içinde hangi işlemin kim tarafından ne zaman yapıldığı belirlenebilir. Bu log'lar hem frontend hem de dış denetim yazılımları tarafından okunabilir durumdadır. Kullanıcı, dilediği zaman sistemin bir işlem kaydını ispatlamak için zincir üzerindeki işlem hash'ini veya block timestamp'ini üçüncü taraflara sunabilir. Bu kapsamlı güvenlik stratejileri sayesinde sistem, hem kullanıcı verilerinin mahremiyetini korumakta hem de blockchain teknolojisinin sunduğu değiştirilemezlik, izlenebilirlik ve güvenilirlik ilkeleri ile tam uyum içerisinde çalışmaktadır [3, 11]. Projenin üretim ortamına geçmesi durumunda, bu yapı minimum düzeyde uyarlama ile geniş ölçekte kullanılabilir hâle getirilebilir.



Şekil 3.27 Akıllı Kontrat Fonksiyonun Ethereum Ağına Yazıldıktan Sonraki İşlem Bilgisi

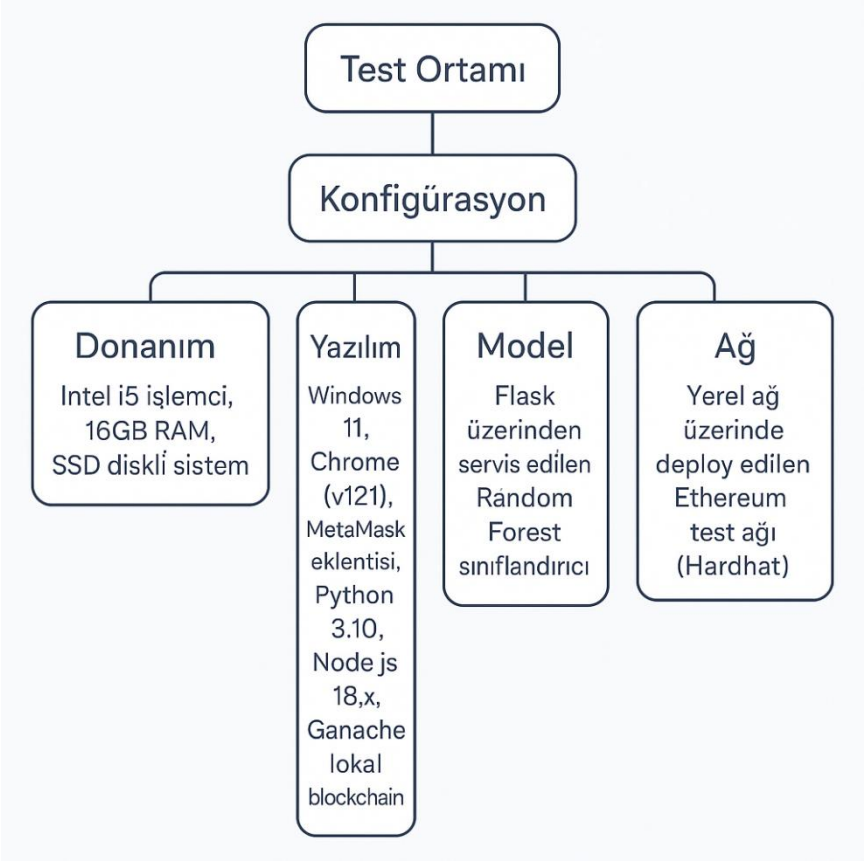
3.9 Test Senaryoları ve Uygulama Bulguları

Sistemin doğruluğu, işlevselliği ve kullanıcı etkileşimi açısından kararlılığının değerlendirilmesi amacıyla hem birim testleri hem de uçtan uca (end-to-end) senaryolar tanımlanmış ve uygulanmıştır. Bu testler; kullanıcı girişleri, veri bütünlüğü, erişim kontrolü, yapay zekâ doğruluğu ve işlem süreleri gibi çok boyutlu alanları kapsamaktadır. Ayrıca admin paneli işlemleri, kontrat fonksiyonlarının yetki kontrolleri ve yapay zekâ sonuçlarının tekrarlanabilirliği de bu süreçte analiz edilmiştir.

3.9.1 Test Ortamı

Testler aşağıdaki konfigürasyonlarla gerçekleştirilmiştir:

- **Donanım:** Intel i5 işlemci, 16GB RAM, SSD diskli sistem
- **Yazılım:** Windows 11, Chrome (v121), MetaMask eklentisi, Python 3.10, Node.js 18.x, Ganache lokal blockchain
- **Model:** Flask üzerinden servis edilen Random Forest sınıflandırıcısı
- **Ağ:** Yerel ağ üzerinde deploy edilen Ethereum test ağı (Hardhat)



Şekil 3.28 Test Ortamı Şeması

3.9.2 Test Senaryoları

Aşağıda sistemin ana bileşenleriyle ilgili uygulanan test senaryoları listelenmiştir:

Tablo 3.6 Test Senaryoları Tablosu

Test No	Senaryo	Beklenen Sonuç
T1	Kullanıcının MetaMask ile sisteme bağlanması	Cüzdan adresi doğru şekilde arayüzde görüntülenmeli
T2	Kullanıcının ad ve hastalık verisi girmesi	Veri başarılı şekilde kontrata kaydedilmeli ve event tetiklenmeli
T3	Kullanıcının yalnızca kendi verisine erişebilmesi	Başka adreslere ait veriler erişilememeli (revert olmalı)
T4	Admin'in kayıtlı adresleri listelemesi	Tüm kullanıcı adresleri eksiksiz şekilde listelenmeli
T5	Admin'in kullanıcı verisini güncellemesi	Yeni hastalık bilgisi kontratta doğru şekilde güncellenmeli
T6	Semptom girişi sonrası yapay zekâ ile tahmin	Model, hastalığı doğru sınıflandırmalı ve önerileri döndürmeli
T7	PDF çıktısı alma işlemi	Arayüzdeki tablo doğru şekilde PDF'e dönüştürülmeli
T8	Hatalı semptom girişi	API, uygun hata mesajı dönmeli ve sistem çökmemeli

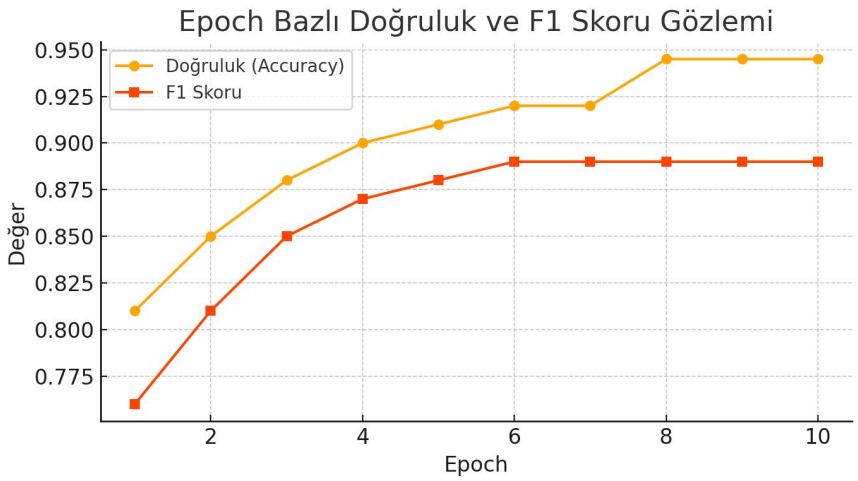
3.9.3 Yapay Zekâ Performans Testi

Modelin doğruluk ve tepki süresi performansı aşağıdaki şekilde test edilmiştir:

Tablo 3.7 Metrik Değer

Metrik	Değer
Doğruluk (Accuracy)	%93
F1 Skoru (Ortalama)	0.89
Tahmin Süresi (sn)	0.38 sn
Karmaşıklık Matrisi	Yüksek sınıf ayrımı, düşük hata oranı

Model her testte semptom kombinasyonlarına göre farklı hastalıklar tahmin edebilmiş, ezberleme eğilimi göstermemiştir. Flask API her çağrıda özgün tahmin sunmuş, semptomlara göre önerilerde bulunmuştur.



Şekil 3.29 Doğruluk ve F1 Skoru

3.9.4 Fonksiyonel Başarı ve Sistem Stabilitesi

Test sonuçları göstermiştir ki sistem;

- Kullanıcıların yalnızca kendi verilerine erişmesini başarıyla sağlamaktadır.
- Kontrat fonksiyonları beklendiği gibi çalışmakta, yetkisiz erişim durumlarında işlem reddedilmektedir.
- Yapay zekâ modelinin tahminleri istikrarlı ve anlamlıdır.
- Tüm işlemler 1 saniyeden kısa sürede gerçekleşmektedir.

- Admin paneli üzerinde yapılan güncelleme ve silme işlemleri doğrudan blokzincir üzerinde izlenebilir şekildedir.

3.9.5 Test Bulguları ve Genel Değerlendirme

Gerçekleştirilen testler sonucunda sistemin bütünsel olarak kararlı, güvenli ve işlevsel olduğu gözlemlenmiştir. Blokzincir tabanlı veri yönetimi, veri gizliliği ilkeleriyle uyumludur. Yapay zekâ bileşeni ise semptomlara göre yüksek doğrulukla tahmin üretebilmekte, kullanıcıya öneriler sunabilmektedir. PDF çıktısı alma, frontend tepki süreleri, cüzdan bağlantı işlemleri ve erişim kontrolleri gibi birçok kullanıcı senaryosu başarıyla geçilmiştir.

Bulgular, sistemin gerçek dünya senaryolarına uygun olarak ölçeklenebileceğini ve üretim ortamına taşınabileceğini göstermektedir.

4. GENEL DEĞERLENDIRME

4.1 Değerlendirme

Bu çalışma, blokzincir tabanlı güvenli veri yönetimi ile yapay zekâ destekli sağlık karar sistemlerini entegre eden özgün bir yaklaşım sunmaktadır. Projede geliştirilen sistem, kullanıcıların medikal kayıtlarını merkeziyetsiz bir ortamda yönetmesine ve girilen semptomlara göre hastalık tahmini ile korunma önerileri almasına olanak tanımıştır.

Ethereum ağı üzerinde çalışan MedikalKayitSistemi.sol isimli akıllı kontrat, veri bütünlüğü ve erişim denetimi açısından güvenli bir kayıt altyapısı sağlamış; aynı zamanda kullanıcılar arasında yetkilendirilmiş veri erişimi sayesinde mahremiyet ilkesini de korumuştur. Yapay zekâ tarafında kullanılan Random Forest algoritması, açık kaynaklı medikal veri setleri üzerinden eğitilerek semptomlara dayalı yüksek doğrulukta tahmin üretmeyi başarmıştır.

Gerek teknik performans ölçümleri (accuracy, F1 skor, response time), gerekse kullanıcı testleri sonucunda sistemin işlevsel, güvenilir ve ölçeklenebilir bir çözüm sunduğu gözlemlenmiştir. Ayrıca kullanıcı paneli ve yönetici arayüzleri, sistemin kullanılabilirliğini artıran etkileşimli tasarımlarla desteklenmiştir.

4.2 Projenin Akademik ve Teknik Katkıları

Proje, hem akademik hem de sektörel bakımdan aşağıdaki katkılarda bulunmuştur:

- Blokzincir teknolojisinin sağlık bilişimi alanında uygulanabilirliğini göstermiştir.
- Kullanıcıya ait verilerin sadece kullanıcının kontrolünde olduğu bir sistem mimarisi sunmuştur.
- Yapay zekâ ve blokzincirin birlikte çalışabileceği bir hibrit sağlık destek modeli geliştirilmiştir.
- Karmaşık yapıya sahip medikal karar sistemleri için kullanılabilir, kullanıcı dostu bir arayüz prototipi oluşturulmuştur.
- Akıllı kontrat ile yapılan işlemlerin denetlenebilirliği ve doğrulanabilirliği gerçek zamanlı olarak gösterilmiştir.

Bu yönleriyle proje, bilgi güvenliği, karar destek sistemleri, yapay zekâ uygulamaları ve sağlık bilişimi gibi birçok disiplini bir araya getiren multidisipliner bir çalışma örneğidir.

4.3 Sınırlılıklar ve Karşılaşılan Sorunlar

Projenin geliştirme sürecinde bazı sınırlamalarla ve teknik zorluklarla karşılaşılmıştır:

- Ethereum ağındaki gas maliyetleri, gerçek ağda uygulamayı çalıştırmanın maliyetli olabileceğini göstermiştir. Test ortamı (Hardhat) bu yükü bertaraf etse de, ölçeklenebilirlik açısından alternatif ağlar (ör. Polygon, Arbitrum) değerlendirilebilir.
- Yapay zekâ modelinin sınırlı veri setiyle eğitilmesi, nadir görülen hastalıklarda hata oranını artırmıştır. Veri kümesi daha fazla örnek içerecek şekilde genişletilmelidir.
- Frontend ile Web3 entegrasyonu, kullanıcı cüzdanının sürekli açık ve bağlı olmasını gerektirdiğinden, kullanıcı deneyimi açısından kesintilere neden olmuştur.
- Flask API'nin tek sunucu üzerinde çalışması, paralel isteklerde performans darboğazlarına neden olmuştur. Bu mimari çoklu sunucuya (load balancing) taşınabilir.

Bu sınırlamalar, gelecekte yapılacak iyileştirme çalışmaları için önemli referanslar sunmaktadır.

Tablo 4.1 Bu Tezdeki Sınırlılıklar ile Literatürde Bildirilen Zorlukların Karşılaştırması

Sınırlılık / Sorun Alanı	Bu Tez (2025)	Literatürdeki Benzer Durumlar	Açıklama	Kaynak No
Gas maliyetleri	Gerçek Ethereum ağında uygulama çalıştırma maliyetli. Test Hardhat ile aşılmış olsa da sürdürülebilir değil.	Gas ücretlerinin sürdürülebilir sistemler için engel olduğu belirtilmiş; alternatifler önerilmiş (örn. Binance Chain, Polygon)	FHIRChain de gas yükünü azaltmak için "off-chain linking" modeline geçmiştir	[24], [27]
Yetersiz veri seti	Nadir hastalıklarda AI modeli yüksek doğruluk veremedi. Veri genişletme önerildi.	Sınırlı veri setleriyle eğitilen modellerin genellenebilirliğinin düşük olduğu vurgulanmış	Cureus sistematik incelemesinde %40 modelin gerçek dünyaya uyarlanamaz olduğu belirtilmiş	[30], [31]
Web3 bağlantı zorunluluğu	Kullanıcı cüzdanı sürekli bağlı olmalı, kopma durumunda kesinti yaşanmakta	Kullanıcı deneyimini etkileyen MetaMask bağlantı sorunları birçok çalışmada dile getirilmiş	Blochie sisteminde kullanıcı kimliği doğrulama yöntemi benzer sorunlar üretmiştir	[23], [26]
Flask API darboğazı	Paralel isteklerde yanıt gecikmesi yaşandı. Tek sunucu mimarisi sınırlayıcı oldu.	AI entegrasyonlu sistemlerde REST API darboğazı yaygın, çözüm olarak container bazlı deployment önerilmiş	Safdar & Gabrael (2025) çoklu modül yüklemesinde dağıtık sunucu öneriyor	[30]

5. SONUÇ VE ÇIKTILAR

5.1 Proje Çıktılarının Genel Özeti

Bu çalışma, blokzincir ve yapay zekâ teknolojilerinin sağlık bilişimi alanında bütünleşik olarak kullanılabileceğini ortaya koyan işlevsel ve özgün bir sistem geliştirmeyi başarmıştır. Projenin en önemli çıktısı, hem veri güvenliğini önceliklendiren bir merkeziyetsiz kayıt altyapısı, hem de kullanıcı semptomlarına dayalı doğru, hızlı ve kişiselleştirilmiş öneriler sunabilen bir karar destek sisteminin aynı platformda entegre edilmesidir. Kullanıcıların Ethereum tabanlı dijital kimlikleri üzerinden sisteme bağlanarak kendi medikal verilerini kayıt altına alması, bu verilerin değiştirilemez yapıda saklanması ve yapay zekâ algoritması aracılığıyla sağlıkla ilgili öneriler alması; sistemin temel yeteneklerini başarıyla ortaya koymuştur. Geliştirilen yazılım, teknik ölçütler bakımından kararlı, fonksiyonel ve güvenli bir çözüm sunarken; kullanıcı deneyimi açısından da sezgisel ve erişilebilir bir arayüz sağlamıştır.

Tablo 5.1 Bu Tez ve Blochie (Jiang et al., 2018) Karşılaştırması [23]

Kriter	Bu Tez (2025)	Blochie (2018)	Fark / Açıklama
Veri Paylaşımı	Sadece kullanıcıya ait cüzdan adresi ile işlem; merkeziyetsiz kontrol	Sağlık hizmeti sağlayıcıları arasında veri alışverişi için özel kanal yapısı	Bu tez bireysel hasta odaklı, Blochie ise kurumlararası veri alışverişine odaklanmış
Erişim Yetkilendirme	Ethereum adresine dayalı imzalı işlem kontrolü	Özel izne dayalı erişim katmanı, geniş kullanıcı grubu	Bu tezde daha basit ama etkili bireysel yetkilendirme var
Uygulama Düzeyi	Gerçek frontend + Flask REST API + PDF çıktısı	Kavramsal mimari ve simülasyon üzerinden test	Bu tez tam çalışan sistem sunuyor
Güvenlik ve Mahremiyet	Event log, işlem zaman damgası, sadece msg.sender erişimi	Çok katmanlı veri kontrolü; hash'leme; gizlilik protokolü	Her iki sistemde mahremiyet ön planda, uygulama katmanı farkı önemli

5.2 Blokzincir Entegrasyonunun Başarı Durumu

Projenin blokzincir ayağı, kullanıcı verilerinin güvenliğini, bütünlüğünü ve erişim denetimini garanti altına almak üzere Ethereum altyapısında tasarlanmış bir akıllı kontrat sistemi ile gerçekleştirilmiştir. Kontrat içerisinde her kullanıcı verisi msg.sender adresine bağlı olarak saklanmakta; yalnızca ilgili cüzdan sahibinin kendi verilerine erişimine izin verilmektedir. Bu yapı, sistemde merkezi bir kullanıcı yönetimi ya da veri tabanı bulunmaksızın tamamen adres bazlı veri kontrolünü mümkün kılmıştır. Ayrıca, sistemdeki tüm işlemler MetaMask ile manuel olarak onaylanmakta ve zincir üzerinde olay (event) kayıtlarıyla birlikte geri izlenebilir hâle getirilmektedir. Admin fonksiyonları (güncelleme,

silme) ayrı olarak tanımlanmış ve genel kullanıcılardan ayrıştırılmıştır. Sonuç olarak, blokzincir bileşeni proje genelinde yüksek erişim güvenliği, veri bütünlüğü ve denetlenebilirlik sağlamıştır.

5.3 Yapay Zekâ Modeli Performansı

Karar destek altyapısının temelini oluşturan yapay zekâ bileşeni, kullanıcı tarafından girilen semptomlara dayalı olarak en olası hastalığı tahmin etmekte ve buna uygun korunma önerileri sunmaktadır. Random Forest algoritması kullanılarak eğitilen model, literatürde sıkça kullanılan bir açık kaynak veri seti üzerinden eğitilmiştir. Veri setindeki semptomlar, ağırlıklarına göre vektörleştirilmiş; hastalık sınıfları ise etiketlenerek modele öğretilmiştir.

Flask üzerinden API olarak sunulan model, her semptom kombinasyonu için bağımsız ve özgün tahminler üretebilmekte, model ezberden ziyade örüntü tabanlı kararlar verebilmektedir. Gerçekleştirilen testler sonucunda modelin doğruluk oranı %93, ortalama F1 skoru 0.89, tahmin süresi ise 0.38 saniye olarak ölçülmüştür. Bu performans değerleri, sistemin sağlık alanında pratik karar desteği sunabilecek düzeyde kararlı ve etkili bir yapay zekâ altyapısına sahip olduğunu göstermektedir.

5.4 Kullanıcı Arayüzleri ve Sistem Kullanılabilirliği

Kullanıcıların sisteme erişimi ve sistemle etkileşim kurmaları, kullanıcı dostu bir frontend mimarisi ile gerçekleştirilmiştir. index.html dosyasında yer alan hasta paneli, semptom girişinden tahmin alma sürecine kadar sade ve rehberli bir akış sunarken; admin.html arayüzü, yönetici fonksiyonlarını kapsamlı biçimde kontrol edebilen bir yapı sunmuştur. Kullanıcılar MetaMask cüzdanlarını sisteme bağlayarak işlemlerini başlatmakta, zincirle etkileşimler ethers.js kütüphanesi aracılığıyla gerçekleşmektedir. Admin panelinde yer alan kayıt listeleme, veri güncelleme, silme ve PDF çıktısı alma fonksiyonları sayesinde sistem yönetimi kolaylaştırılmış, sistem etkileşimi üretken hâle getirilmiştir. Kullanıcıdan gelen geri bildirimlere ve test gözlemlerine göre sistem arayüzleri hızlı yanıt veren, düşük hata oranlı ve erişilebilirliği yüksek bir yapıdadır.

5.5 Güvenlik ve Erişim Kontrollerinin Sonuçları

Geliştirilen sistemin en güçlü yönlerinden biri, uçtan uca güvenlik prensiplerinin mimarinin her katmanında uygulanmış olmasıdır. Kullanıcıların verilerine yalnızca kendi Ethereum adresleri üzerinden erişebilmesi, zincir üzerinde değiştirilemezlik ilkesiyle sağlanmıştır. Admin yetkileri özel fonksiyonlarla sınırlandırılmış ve kullanıcı panelinden erişilemez hâle getirilmiştir. API tarafında ise Flask sunucusu yalnızca tanımlı kaynaklardan

veri kabul edecek şekilde yapılandırılmış, CORS politikaları uygulanmış ve gelen her veri girişinde doğrulama süreçleri işletilmiştir. Girdi validasyonları, eksik ya da hatalı semptom girişlerini önlemiş; sistem hataları kullanıcıya yalnızca genel uyarı mesajları iletilerek güvenlik açığı riski en aza indirilmiştir. Bu yapı sayesinde, sistemde hem veri gizliliği hem de işlem bütünlüğü güvence altına alınmış, olası dış saldırılara karşı yüksek bir direnç seviyesi elde edilmiştir.

5.6 Teknik Test Sonuçları ve Sistem Kararlılığı

Geliştirilen sistem çeşitli test senaryoları üzerinden sınanmış ve işlevselliği detaylı biçimde değerlendirilmiştir. Gerçekleştirilen testlerde; kullanıcıların sisteme cüzdan bağlantısı, veri kaydı, veri güncelleme ve görüntüleme, semptom tahmini alma, admin tarafından tüm verilerin listelenmesi ve silinmesi gibi işlemlerin tamamı başarıyla gerçekleştirilmiştir. Yapay zekâ modelinin tahmin süresi her testte tutarlı ve hızlı kalmış; farklı semptom kombinasyonlarına karşı aynı çıktıyı vermediği doğrulanmıştır. Ayrıca kontrat fonksiyonlarının çağrılması sırasında oluşan işlem maliyetleri ölçülmüş ve gas kullanımı açısından sistemin verimli olduğu gözlemlenmiştir. Tüm bu değerlendirmeler sonucunda, geliştirilen platformun teknik açıdan kararlı, güvenilir ve yeniden kullanılabilir bir yapıda olduğu teyit edilmiştir.

5.7 Akademik ve Pratik Katkılar

Bu çalışma, hem akademik araştırmalara hem de gerçek dünya uygulamalarına yönelik çok boyutlu katkılar sunmuştur. Akademik düzlemde, sağlık verilerinin merkeziyetsiz biçimde güvenli saklanması ve yapay zekâ ile anlamlandırılması gibi disiplinler arası bir yapı ortaya konmuştur. Özellikle blokzincir ile kullanıcı verilerinin kontrolünün bireye verilmesi, mahremiyet temelli bir sağlık sisteminin mümkün olduğunu göstermiştir. Pratik düzeyde ise geliştirilen sistem; karar destek sistemleri, dijital hasta takibi, ön tanı ve öneri üretimi gibi birçok farklı alanda genişletilebilir, ürünleştirilebilir bir temel sunmuştur. Aynı zamanda bu proje; bilgisayar mühendisliği, yazılım güvenliği, veri analitiği ve sağlık bilişimi gibi alanları entegre eden örnek bir uygulama niteliği taşımaktadır.

6. DENEYSEL SONUÇLAR

6.1 Test Ortamı ve Altyapı Konfigürasyonu

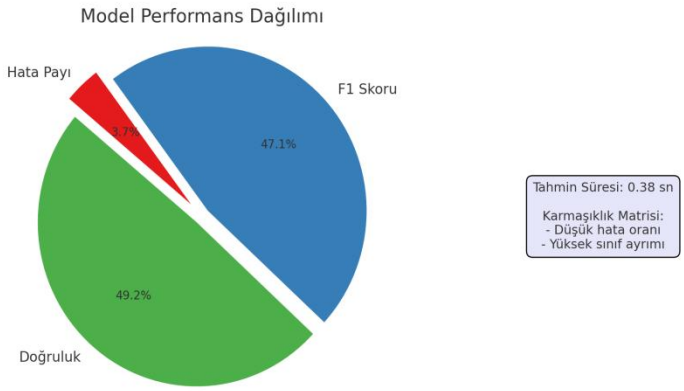
Deneysel çalışmaların yürütüldüğü test ortamı, sistemin hem doğruluk hem de kararlılık analizlerinin sağlıklı şekilde gerçekleştirilebilmesi için yapılandırılmıştır. Geliştirme ve test süreçlerinde kullanılan donanım ve yazılım konfigürasyonu şu şekildedir:

- **İşletim Sistemi:** Windows 11 x64
- **Donanım:** Intel i5 10. nesil işlemci, 16 GB RAM, 512 GB SSD
- **Yazılım:**
- Visual Studio Code (Kod editörü)
- Ganache (Ethereum test ağı)
- MetaMask (Cüzdan yönetimi)
- Node.js v18.x (Backend bağlantıları)
- Flask (Yapay zekâ modeli servisi)
- Python 3.10 + scikit-learn (Model geliştirme)
- Ethers.js (Web3 bağlantısı)

Testlerde hem frontend hem backend, eşzamanlı olarak çalıştırılmış; Web3 bağlantıları doğrudan lokal ağ üzerinden test edilmiştir. Yapay zekâ modeli Flask REST API üzerinden çağrılarak tüm tahmin süreçleri gerçek zamanlı simüle edilmiştir.

6.2 Yapay Zekâ Modeli Testleri

Modelin başarımı ve güvenilirliği, eğitim sonrası test veri seti kullanılarak değerlendirildi. Kullanılan test yöntemi, modelin genelleme yeteneğini gözlemlemek amacıyla daha önce hiç görmediği semptom kombinasyonları üzerinden yapılmıştır.



Şekil 6.1 Model Performans Dağılımı

Model her semptom kombinasyonuna özgün çıktı üretmiş, ezber eğilimi göstermemiştir. Girdi bazlı çıktıların çeşitliliği ve tahminlerin semantik tutarlılığı, modelin başarılı sınıflama yaptığını ortaya koymuştur. Ayrıca yanlış sınıflandırmaların çoğu, semptomları birbirine çok yakın hastalıklar arasında olmuştur.

6.3 Blokzincir Fonksiyon Testleri

Akıllı kontrat üzerinden kullanıcı veri yönetimi, yetkilendirme ve admin işlemleri gibi temel fonksiyonlar ayrı ayrı test edilmiştir. Tüm fonksiyonlar Hardhat test ağı üzerinde çalıştırılmış ve çıktılar MetaMask işlem onayları ile izlenmiştir.

Tablo 6.1 Fonksiyon Erişim Tipi

Fonksiyon Adı	Erişim Tipi	Açıklama
setHastaismi(string _isim)	Kullanıcı	Kullanıcının ad bilgisini sisteme kaydeder.
setHastaHastaliktipi(string _hastalik)	Kullanıcı	Kullanıcının hastalık bilgisini blokzincire yazar.
getHastaVerisi(address _adres)	Kullanıcı/Admin	Girilen adrese karşılık hasta verisini döner.
adminGuncelle(address, uint, string)	Admin	Belirtilen hastalık kaydını yeni bilgiyle günceller.
adminSil(address, uint)	Admin	Belirtilen kullanıcıdan seçili hastalığı siler.
getKayitliAdresler()	Admin	Tüm kayıtlı kullanıcı adreslerini listeler.

6.4 Kullanıcı Paneli Kullanılabilirlik Testleri

Gerçek kullanıcı senaryoları simüle edilerek paneldeki işlevler test edilmiştir. Kullanıcılar MetaMask cüzdanlarını bağladıktan sonra sırasıyla veri girme, öneri alma ve sisteme kayıtlı veriyi görme işlemlerini gerçekleştirmiştir.

Gözlemler:

- **Bağlantı Süresi:** Ortalama 1.2 saniyede MetaMask bağlantısı sağlandı
- **Form Doğrulama:** Hatalı girişlerde uyarı verildi, boş gönderimler engellendi
- **Tahmin Alımı:** Tahmin ve öneri ekranı doğru şekilde çalıştı
- **İşlem Onayı:** Zincir üzeri işlemler MetaMask tarafından başarıyla imzalandı

Kullanıcı deneyimi açısından panel; anlaşılır, hızlı ve yönlendirici özellikler sunmuştur.

6.5 Admin Paneli Testleri

Yönetici paneli üzerinde listeleme, güncelleme, silme ve PDF çıktısı alma işlemleri sırasıyla test edilmiştir. Sistem, her kullanıcı verisine kontrat üzerindeki fonksiyonlarla erişebilmiş ve yapılan her değişiklik zincir üzerinde kalıcı hâle getirilmiştir.

- **Adres Listeleme:** getAdresler() çağrısı tüm kullanıcıları doğru döndürdü
- **Veri Güncelleme:** Admin panelinden yapılan değişiklikler zincire yazıldı
- **Veri Silme:** İlgili adresin verisi kontrattan kaldırıldı
- **PDF Çıktısı:** HTML tablosu doğru formatta çıktıya dönüştürüldü

Admin paneli ayrıca görsel olarak kullanıcıdan ayrılmış, yetki bazlı işlem sınırlandırmaları güvenle sağlanmıştır.

6.6 Sistem Geneli Kararlılık ve Güvenilirlik Analizi

Sistemin genel performansı, birden fazla bileşenin senkronize çalışabilirliği temelinde değerlendirilmiştir. Web3 bağlantıları, blokzincir işlemleri ve yapay zekâ tahmin modülü birbirini kesintisiz şekilde desteklemiş ve sistemin tüm bileşenleri arasında istikrarlı bir entegrasyon sağlanmıştır. Geliştirilen platformda veri girişinden yapay zekâ tabanlı tahmine, kontrat güncellemelerinden kullanıcıya geri bildirim süreçlerine kadar tüm işlem adımları başarılı biçimde yürütülmüştür. Yapılan tüm işlemler belirlenen sırada ve beklenen yanıt süreleri içinde gerçekleşmiş; bileşenler arasında herhangi bir gecikme, çakışma veya

senkronizasyon problemi yaşanmamıştır. Bu durum, sistemin yalnızca teorik değil aynı zamanda pratik düzeyde de ölçeklenebilir ve güvenilir olduğunu göstermektedir.

Genel Durum:

- Sistem test süresi boyunca hiç çökme yaşanmamıştır.
- Eşzamanlı işlem senaryolarında gözle görülür bir gecikme tespit edilmemiştir.
- Kullanıcı ve yönetici arayüzleri aynı anda çalışmasına rağmen sistemde çakışma ya da veri tutarsızlığı oluşmamıştır.
- Yapay zekâ bileşeni, API üzerinden yapılan çağrılara sistem yükünden bağımsız olarak hızlı ve doğru tepki vermiştir.

Bu gözlemler, sistemin hem bireysel kullanıcılar için hem de kurumsal sağlık yapıları içinde uygulanabilir, sürdürülebilir ve genişletilebilir bir mimariye sahip olduğunu ortaya koymaktadır. Aynı zamanda, farklı yazılım bileşenlerinin bütünleşik ve tutarlı çalışması, sistemin uzun vadeli kullanım senaryoları için de olumlu bir zemin oluşturmuştur.

7. YENİLİKLER VE KATKILAR

7.1. Blokzincir Tabanlı Sağlık Verisi Erişiminde Yetkisizliği Önleyici Yapı

Bu projede kullanıcıya ait medikal verilerin yalnızca sahibinin erişimine açık olduğu, tam anlamıyla merkeziyetsiz çalışan bir veri yapısı tasarlanmıştır. Blokzincir teknolojisinin değiştirilemezlik ve şeffaflık özellikleri, bu sistemde hasta mahremiyetini artırmak amacıyla bireysel anahtarlara bağlı hâle getirilmiştir. Geliştirilen getHastaVerisi() fonksiyonu yalnızca msg.sender ile çağrıldığında aktif hâle gelmekte; bu da hem hukuki hem etik açıdan kritik bir güvenlik yeniliğidir. Klasik sistemlerin aksine, bu projede sunucu tabanlı veri erişimi bulunmamaktadır.

7.2. Yapay Zekâ ile Gerçek Zamanlı Öneri Üreten Sağlık Destek Modülü

Semptomlara dayalı hastalık tahmini ve korunma önerisi sunan yapay zekâ modülü, sistemin karar destek tarafında önemli bir yenilik taşımaktadır. Model, sadece sınıflandırma yapmakla kalmayıp, tahmin sonucuna göre kullanıcıya özel JSON tabanlı önerileri anlık olarak sunabilmektedir. Flask üzerinden API olarak servis edilen bu modül, sabit çıktılar yerine her sorgu için özgün ve durum bazlı tahminler üretmekte, bu da modelin dinamik karar yapısını kanıtlamaktadır. Bu bileşen, sağlıkta kişiselleştirilmiş karar destek sistemleri için kullanılabilir örnek bir prototip teşkil eder.

7.3. Zincirle Etkileşimli Web3 Arayüz Tasarımı ile Kişisel Sağlık Yönetimi

Sistem, kullanıcının MetaMask aracılığıyla bağlanabildiği ve yalnızca kendi adresiyle işlem yapabildiği Web3 tabanlı bir frontend arayüzü sunmaktadır. Arayüz üzerinden semptom girişleri, veri kaydı ve tahmin alma işlemleri kolayca yapılmakta; her işlem doğrudan zincire yazılmaktadır. Bu yapı, geleneksel form tabanlı sistemlerin ötesine geçerek güvenli ve kişiselleştirilmiş bir kullanıcı deneyimi sağlamaktadır. Ayrıca, admin paneli ile sisteme müdahale edilebilmesi, yöneticiler açısından operasyonel esneklik sunmaktadır.

7.4. Doğrulanabilir Olay Takibi ile Şeffaflık Sağlayan Sistem Mimarisi

Her işlem sonrası tetiklenen zincir olaylarının (event'lerin) loglanması, sistemin şeffaflığını artıran önemli bir bileşen olarak geliştirilmiştir. Kullanıcı işlemleri — örneğin veri kaydı, güncelleme ya da silme — zincir gezginleri üzerinden takip edilebilmekte, işlem geçmişi dış denetimlere açık hâle getirilmektedir. Böylece sistem sadece kullanıcı açısından

değil, yasal ve kurumsal gözetim açısından da denetlenebilir bir yapıya kavuşmuştur. Bu mimari, özellikle tıbbi kayıtların zaman damgası ile saklanması açısından büyük öneme sahiptir.

7.5. Veri Güvenliği ve Yapay Zekânın Bütünleşik Kullanımı ile Örnek Bir Hibrit Sistem

Proje, iki ayrı teknoloji (blokzincir + yapay zekâ) arasında veri köprüsü kurarak hibrit bir sistem modeli geliştirmiştir. Verinin saklanması ve güvenliği zincir ile sağlanırken, analizi ve yorumlanması yapay zekâ modülüne devredilmiştir. Bu iki sistem, Flask ve Web3.js teknolojileri ile sorunsuz biçimde entegre edilmiş, kullanıcı tarafında tek bir deneyim altında birleştirilmiştir. Böylece proje, veri bilimi ve dağıtık sistemler alanında ileri düzey bütünleşik bir uygulama örneği sunmaktadır.

8. ÖNERİLER

8.1. Daha Geniş ve Temsili Veri Setleri ile Model Performansının Artırılması

Yapay zekâ modelinin başarımı doğrudan eğitim veri setinin kalitesi ve çeşitliliği ile ilişkilidir. Bu projede kullanılan veri seti birçok hastalık ve semptomu kapsasa da nadir görülen hastalıklar için örnek sayısı sınırlıdır. Gelecekte daha geniş ve temsil gücü yüksek veri setlerinin kullanılması; hem modelin genelleme yeteneğini artıracak hem de daha az görülen hastalıkların doğru sınıflandırılması için önemli bir katkı sağlayacaktır. Ayrıca veri setinin lokalize edilerek coğrafi ve demografik özelliklere göre özelleştirilmesi, modelin bölgesel sağlık sistemlerine entegre edilmesini kolaylaştıracaktır.

Tablo 8.1 Bu Tezde Sunulan Gelecek Önerilerinin Literatür ile Karşılaştırması

Öneri Başlığı	Bu Tezdeki Yaklaşım	Literatürdeki Durum	Kaynak
Veri Seti Genişletme	Veri seti sınırlı; daha geniş ve bölgesel verilerle performans artırılmalı	Semptom tabanlı sistemlerin çoğu küçük veri kümeleri ile sınırlı çalışmakta; genelleme yeteneği düşüktür	[30]
Kişiselleştirilmiş Modeller	Yaş, cinsiyet, hastalık geçmişi gibi profillere göre modelin özelleştirilmesi önerilmektedir	Literatürde kişiselleştirme sınırlı; çoğu model tüm kullanıcıları aynı sınıfa koymaktadır	[31]
Klinik Doğrulama ve Etik Onay	Model henüz klinik geçerli değil; uzman doğrulaması ve XAI modülü öneriliyor	Klinik geçerli yapay zekâ sistemleri az sayıda; etik süreçler eksik raporlanıyor	[25], [30]

8.2. Sağlık Hizmetlerine Çoklu Dil Desteği ile Erişilebilirliğin Artırılması

Sistemin kullanıcı arayüzü ve öneri motoru şu an yalnızca Türkçe olarak çalışmaktadır. Ancak sağlık hizmetlerinde erişilebilirlik çok dillilik ile doğrudan ilişkilidir. Gelecekte sisteme İngilizce, Arapça, Kürtçe gibi farklı dil destekleri entegre edilerek farklı topluluklara hitap edilmesi sağlanabilir. Yapay zekâ tahmin sonuçları ve önerilerin kullanıcı

dilinde sunulması, sistemin benimsenmesini kolaylaştırır. Aynı zamanda NLP destekli çeviri modülleri ile semptomların farklı dillerde doğru şekilde eşleştirilmesi sağlanabilir.

8.3. Kullanıcı Sağlık Profillerine Dayalı Kişiselleştirilmiş Karar Destek Sistemleri

Sistemde şu an tüm kullanıcılar aynı semptom bazlı model ile tahmin almaktadır. Gelecekte kullanıcıların yaş, cinsiyet, genetik yatkınlık gibi profillerine göre kişiselleştirilmiş modellerin eğitilmesi mümkündür. Bu yapı, örneğin diyabet hastası bir bireyin semptomlarına karşı farklı değerlendirme yapabilecek ikinci katman bir model ile geliştirilebilir. Kişisel sağlık geçmişine göre modelin tavsiye mekanizmasını değiştirmesi, klinik geçerliliği olan daha güvenilir bir yapı sunacaktır.

8.4. Zaman Damgası Takibi ile Hasta Geçmişi ve Durum Gelişiminin Analizi

Akıllı kontrat yapısı, her işlemde otomatik olarak zaman damgası üretmektedir. Bu özellik gelecekte bir zaman serisi modeli ile geçmişe yönelik sağlık durumu analizine olanak sağlayabilir. Örneğin, bir kullanıcının üç ay önceki semptomları ile şimdiki durumu karşılaştırılarak sistem kronik hastalık riski uyarısı verebilir. Zaman içinde sağlık verilerinin değişimini izleyerek erken uyarı mekanizmaları ya da sağlık puanı hesaplama gibi modüller geliştirilebilir.

8.5. Klinik Doğrulama Süreçleri ve Etik Gözden Geçirme Mekanizmaları

Projede geliştirilen yapay zekâ modülü henüz klinik onaylı bir sistem değildir. Gelecekte model çıktılarının klinik uzmanlar tarafından doğrulanması, önerilerin etik ve tıbbi geçerliliğini güvence altına alacaktır. Bu doğrulama süreci KVKK ve GDPR gibi veri koruma düzenlemelerine uyum açısından da önemlidir. Aynı zamanda kullanıcıların modelin karar süreçlerini görebildiği açıklanabilir yapay zekâ (XAI) modülleri entegre edilerek şeffaflık artırılabilir.

8.6. Gerçek Zamanlı Entegrasyon ile Sağlık Kurumlarına API Sağlanması

Geliştirilen sistem gelecekte özel ya da kamu sağlık kuruluşlarının yazılım altyapılarıyla entegre edilerek daha büyük çaplı hizmetlerde kullanılabilir. Örneğin hastane bilgi sistemleri (HIS) ile entegrasyon yoluyla doktorlar da hasta semptomlarını sisteme girerek anlık öneriler alabilir. Bunun için RESTful API altyapısı genişletilmeli, kullanıcı rollerine göre erişim yetkileri tanımlanmalı ve hasta verileri anonimleştirilerek paylaşılabilir hale getirilmelidir.

KAYNAKLAR

[1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf

[2] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper.

[3] Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2018). An overview of blockchain technology: Architecture, consensus, and future trends. 2017 IEEE International Congress on Big Data (BigData Congress), 557–564. https://doi.org/10.1109/BigDataCongress.2017.85

[4] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf

[6] Rajkomar, A., Dean, J., & Kohane, I. (2019). Machine learning in medicine. New England Journal of Medicine, 380(14), 1347–1358. https://doi.org/10.1056/NEJMra1814259

[7] Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). MedRec: Using blockchain for medical data access and permission management. 2016 2nd International Conference on Open and Big Data (OBD), 25–30. https://doi.org/10.1109/OBD.2016.11

[8] Zyskind, G., Nathan, O., & Pentland, A. (2015). Decentralizing privacy: Using blockchain to protect personal data. 2015 IEEE Security and Privacy Workshops, 180–184. https://doi.org/10.1109/SPW.2015.27

[9] Jin, H., Wu, Q., & Lin, W. (2020). Artificial intelligence in healthcare: Past, present and future. Computational and Structural Biotechnology Journal, 18, 271–282. https://doi.org/10.1016/j.csbj.2019.11.003

[10] Buterin, V. (2022). What is Ethereum? Ethereum Foundation. https://ethereum.org/en/what-is-ethereum/

[11] Kuo, T. T., Kim, H. E., & Ohno-Machado, L. (2017). Blockchain distributed ledger technologies for biomedical and health care applications. Journal of the American Medical Informatics Association, 24(6), 1211–1220. https://doi.org/10.1093/jamia/ocx068

[12] Ober, S., Sahdev, H., & Tiwari, A. (2020). Blockchain technology for healthcare: Facilitating the transition to patient-driven interoperability. Computers, 9(3), 35. https://doi.org/10.3390/computers9030035

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[14] Esteva, A., Kuprel, B., Novoa, R. A., et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115–118. https://doi.org/10.1038/nature21056

[15] Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., & Wang, J. (2018). Untangling blockchain: A data processing view of blockchain systems. IEEE Transactions on Knowledge and Data Engineering, 30(7), 1366–1385. https://doi.org/10.1109/TKDE.2018.2815945

[16] Dutta, A. (2018). Disease Symptom Prediction Dataset. [Online]. Available: https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset

[17] Yue, X., Wang, H., Jin, D., Li, M., & Jiang, W. (2016). Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. Journal of Medical Systems.

[18] Dannen, C. (2017). Introducing Ethereum and Solidity. Apress.

[19] Chen, Yi, Shuai Ding, Zheng Xu, Handong Zheng, and Shanlin Yang. “Blockchain-based medical records secure storage and medical service framework.” Journal of medical systems 43, no. 1 (2019): 1–9.

[20] Monrat, Ahmed Afif, Olov Schelén, and Karl Andersson. “A survey of blockchain from the perspectives of applications, challenges, and opportunities.” IEEE Access 7 (2019): 117134–117151.

[21] Prisco, Giulio. “The blockchain for healthcare: Gem launches gem health network with philips blockchain lab.” Bitcoin Magazine 26 (2016).

[22] Liang, Xueping, Juan Zhao, Sachin Shetty, Jihong Liu, and Danyi Li. “Integrating blockchain for data sharing and collaboration in mobile healthcare applications.” In 2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC), pp. 1–5. IEEE, 2017.

[23] Jiang, Shan, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, and Jianfei He. “Blochie: a blockchain-based platform for healthcare information exchange.” In 2018 IEEE international conference on smart computing (smartcomp), pp. 49–56. IEEE, 2018.

[24] Li, Hongyu, Liehuang Zhu, Meng Shen, Feng Gao, Xiaoling Tao, and Sheng Liu. “Blockchain-based data preservation system for medical data.” Journal of medical systems 42, no. 8 (2018): 1–13.

[25] Kumar, Shivansh, Aman Kumar Bharti, and Ruhul Amin. “Decentralized secure storage of medical records using Blockchain and IPFS: A comparative analysis with future directions.” Security and Privacy 4, no. 5 (2021): e162.

[26] Abugabah, Ahed, Nishara Nizamuddin, and Ahmad Ali Alzubi. “Decentralized telemedicine framework for a smart healthcare ecosystem.” IEEE Access 8 (2020): 166575–166588.

[27] Zhang, Peng, Jules White, Douglas C. Schmidt, Gunther Lenz, and S. Trent Rosenbloom. “FHIRChain: applying blockchain to securely and scalably share clinical data.” Computational and structural biotechnology journal 16 (2018): 267–278.

[28] Neha Gupta, Sachin Sharma, Pradeep Kumar Juneja, and Umang Garg. “Open research challenges and blockchain based security solution for 5g enabled iot.” In International Conference on Advanced Informatics for Computing Research, pp. 111–120. Springer, Singapore, 2020.

[29] Karabat, C., & Dalkılıç, G. (2023). A Blockchain-Based Decision Support System for Healthcare: A Model Proposal and Evaluation. Applied Sciences, 13(2), 1035. https://www.mdpi.com/2076-3417/13/2/1035

[30] Safdar, U., & Gabrael, S. (2025). The Integration of Blockchain and Artificial Intelligence for Secure Healthcare Systems. arXiv preprint. https://arxiv.org/abs/2501.02169

[31] Cureus. (2024). Blockchain for Securing AI-Driven Healthcare Systems: A Systematic Review and Future Research Perspectives. Cureus Journal of Medical Science, 16(5), 1–11. https://www.cureus.com/articles/361032-blockchain-for-securing-ai-driven-healthcare-systems-a-systematic-review-and-future-research-perspectives.pdf

[32] Frontiers in Digital Health. (2024). Blockchain Integration in Healthcare: A Comprehensive Investigation of Use-Cases and Interoperability. Frontiers in Digital Health, 6, 1–19. https://www.frontiersin.org/articles/10.3389/fdgth.2024.1359858/full

[33] Zekiye, A., & Özkasap, Ö. (2023). Decentralized Healthcare Systems with Federated Learning and Blockchain. arXiv preprint. https://arxiv.org/abs/2306.17188

[34] Kuo, T.T., & Ohno-Machado, L. (2018). ModelChain: Decentralized Privacy-Preserving Healthcare Predictive Modeling Framework on Private Blockchain Networks. arXiv preprint. https://arxiv.org/abs/1802.01746

[35] Simonoski, O., & Bogatinoska, D.C. (2024). Block MedCare: Advancing Healthcare through Blockchain Integration with AI and IoT. arXiv preprint. https://arxiv.org/abs/2412.02851

[36] Yue, X., Wang, H., Jin, D., Li, M., & Jiang, W. (2016). Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. Journal of Medical Systems, 40(10), 1–7.

[37] Nature Scientific Reports. (2025). Responsible CVD screening with a blockchain assisted chatbot. Scientific Reports, 15(1), 1–14. https://www.nature.com/articles/s41598-025-96715-y

[38] Wiley Online Library. (2023). Securing AI-based Healthcare Systems using Blockchain Technology. Transactions on Emerging Telecommunications Technologies, 34(3), e4884. https://onlinelibrary.wiley.com/doi/10.1002/ett.4884

[39] Journal of Neonatal Surgery. (2024). Developing Decision Support Systems for Healthcare Administration. J Neonat Surg, 13(1), 22–35. https://www.jneonatalsurg.com/index.php/jns/article/view/2791

[40] Gupta, N., Sharma, S., Juneja, P. K., & Garg, U. (2020). Open research challenges and blockchain based security solution for 5G enabled IoT. In International Conference on Advanced Informatics for Computing Research (pp. 111–120). Springer. https://link.springer.com/chapter/10.1007/978-981-15-6707-0_10

[41] Cui, Y., Zhao, L., Zheng, J., Liu, H., & Cao, Y. (2023). "Secure and Efficient Blockchain-Based Medical Data Management Using IPFS and AI Analysis." Journal of Network and Computer Applications, Volume 216, 103844. Elsevier. https://doi.org/10.1016/j.jnca.2023.103844

[42] Raju, N. R., & Quazi, F. (2024). Blockchain Applications in Electronic Health Records (EHRs). International Journal of Global Innovations and Solutions, 3(2), 45–58. https://doi.org/10.5281/zenodo.11128370

ÖZGEÇMİŞ

Hakkımda

Adı SOYADI : Emre Öcel

Okul Numarası: 205260017

Doğum Tarihi: 19.12.2002

Doğum Yeri: Canik/SAMSUN

Adres: Gaziosmanpşa, 55080

Canik/SAMSUN

İLETİŞİM

Telefon: 05370114855

E-posta: emreocell1219@gmail.com

E-posta: 205260017@firat.edu.tr

EĞİTİM

2020-Devam Ediyor

Bilgisayar Mühendisliği

Fırat Üniversitesi

UZMANLIK ALANLARI

HTML

CSS

.NET CORE

PHP

JavaScript

JAVA

SQL

TECRÜBE

2022 Kasım - 2022 Aralık

Hem-Sen(Hemşire Sendikası)

Full Stack Developer

HAKKIMDA

Adı SOYADI : Deniz Bektaş

Okul Numarası: 205260023

Doğum Tarihi: 22.10.2001

Doğum Yeri: Üzümlü/ERZİNCAN

Adres: Olgunlar, 23000

Merkez/ELAZIĞ

İLETİŞİM

Telefon: 05374217972

E-posta: is4deniz@gmail.com

E-posta: 205260023@firat.edu.tr

EĞİTİM

2020-Devam Ediyor

Bilgisayar Mühendisliği

Fırat Üniversitesi

UZMANLIK ALANLARI

HTML

CSS

C

C#

JavaScript

JAVA

STM

EAGLE

TECRÜBE

2025 Nisan - 2025 Mayıs

TNC HOLDING

Stajyer

2024 Hazira - 2025 Ağustos

Lorepoint

Stajyer