Benha University
Faculty of Computers & Artificial Intelligence

Electronic Medical Records System Using Blockchain Technology (MedRecChain)

A senior project submitted in partial fulfillment of the requirements for the degree of Bachelor of Computers and Artificial Intelligence.

Information Systems Department

Project Team

Ghada Abdul Wahab Awwad

Fatma Mohamed Sorour

Heba Walid Mohamed

Farouk Hamed Seliem

Ali Tharwat Mohamed

Aya Hamdy Abdalla

Fatma Samir El Sayed

Under Supervision of

Dr. Mona Mohamed Arafa

Eng. Esraa Hassan El Gamal

(June - 2023)

ACKNOWLEDGMENT

We would like to take this opportunity to express our deepest appreciation to the following individuals who have contributed directly or indirectly to the completion of this work.

First and foremost, we would like to thank our supervisors **Dr. Mona Arafa and Eng. Esraa El Gamal**, for their invaluable guidance, support, and encouragement throughout this project. Their expertise and feedback were crucial in shaping the direction of our research and improving the quality of our work.

We would also like to express our gratitude to **Prof. Tarek El Shishtawy and Dr. Fady Mohamed** who provided us with valuable insights and feedback on various aspects of this work. Their input and constructive criticism have helped us refine our ideas and improve the overall quality of our research.

We would also like to extend our appreciation to the faculty members of the Information System Department for their guidance and support throughout our studies.

Finally, we would like to express our heartfelt gratitude to our families for their unwavering support, encouragement, and understanding, without which this achievement would not have been possible.

Thank you all for your support and contributions

DECLARATION

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Artificial Intelligence in *Information System* is entirely our own work, that we have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

Signed: ______________________________

Date:

Abstract

MedRecChain is a decentralized solution designed to address the challenges faced by Electronic Medical Records (EMR) sharing systems. Leveraging the Ethereum network and IPFS technology, MedRecChain offers a secure and efficient platform for storing and exchanging medical records. Traditional EMR systems often encounter issues such as data breaches, lack of interoperability, and limited patient control over their own records. MedRecChain aims to overcome these challenges by employing blockchain technology, ensuring data immutability, transparency, and enhanced security. By utilizing the Ethereum network, MedRecChain establishes a decentralized infrastructure that allows healthcare providers, patients, and other authorized parties to access and share medical records in a seamless and trustless manner. The use of smart contracts further enhances the integrity and reliability of transactions, enabling automated and auditable interactions. In addition, MedRecChain incorporates IPFS (InterPlanetary File System) technology to efficiently store and distribute medical records. IPFS facilitates the decentralized storage and retrieval of data, eliminating reliance on a single central server and enabling greater scalability and resilience. The key features of MedRecChain include secure patient identification, granular access control, data privacy, and auditability. Patients have full control over their medical records, granting or revoking access to healthcare providers as needed. The system ensures data privacy through encryption techniques, while still allowing authorized parties to trace and audit record access and modifications. Through the implementation of MedRecChain, the healthcare industry can benefit from a decentralized and secure solution for EMR sharing. It has the potential to streamline record management, enhance patient privacy and control, and foster interoperability among different healthcare entities.

Table of Contents

Table of Figures

Chapter 1: Introduction

1.1 Background

In recent years, the healthcare industry has undergone a significant digital transformation with the adoption of Electronic Medical Records (EMRs). EMRs have revolutionized the way medical information is stored, accessed, and shared, offering numerous advantages over traditional paper-based records. The transition from paper records to electronic systems has improved data accuracy, reduced errors, enhanced patient safety, and streamlined healthcare workflows. EMRs allow healthcare providers to access patient information more efficiently, make informed decisions, and provide timely and coordinated care. Additionally, electronic records facilitate data analysis, research, and population health management, leading to improved healthcare outcomes.However, despite the numerous benefits, traditional centralized EMR systems have inherent limitations that hinder their full potential. These systems typically rely on a single central authority, such as a hospital or healthcare organization, to store and manage patient data. While this approach offers convenience in terms of data consolidation and management, it introduces several challenges and risks. One significant concern is the security and privacy of patient information. Centralized EMR systems have become attractive targets for hackers and malicious actors seeking to exploit vulnerabilities and gain unauthorized access to sensitive medical records. Data breaches can result in significant harm to patients, including identity theft, insurance fraud, or the exposure of highly personal health information. Moreover, the lack of interoperability among different EMR systems remains a major hurdle in the healthcare industry. Healthcare providers often use various EMR platforms that operate in silos, making it difficult to share patient data seamlessly. This fragmentation hampers effective care coordination, slows down the exchange of medical information, and can lead to fragmented or incomplete medical records, potentially compromising patient safety and continuity of care. Furthermore, the dependence on a single authority for data management creates a centralized point of failure. System outages, hardware failures, or other disruptions can lead to the unavailability of patient records, impacting healthcare delivery and patient care. Additionally, the reliance on a central authority puts healthcare providers and patients at the mercy of that entity's policies, practices, and potential limitations in terms of data access and sharing.To address these challenges and limitations, there is a growing need for decentralized solutions for EMR sharing systems. Decentralization aims to distribute the storage, control, and access of EMRs among multiple entities, eliminating the reliance on a single authority. By leveraging distributed ledger technology, such as blockchain, and decentralized storage solutions like IPFS (Interplanetary File System), it becomes possible to create secure, transparent, and interoperable EMR sharing ecosystems.

1.2 Problem Statement

The traditional centralized Electronic Medical Records (EMR) systems have several inherent limitations and challenges that hinder the efficient and secure sharing of patient information. These challenges necessitate the development of a decentralized solution, which forms the core problem that the MedRecChain project aims to address.

1- Security and Privacy Risks

One of the primary concerns with centralized EMR systems is the vulnerability to security breaches and unauthorized access. Healthcare data is highly valuable and attractive to hackers, making centralized systems a prime target for data breaches. A single breach can lead to the exposure of sensitive patient information, resulting in identity theft, insurance fraud, and potential harm to individuals. The centralization of data storage also increases the risk of insider threats and unauthorized access by individuals within the authority managing the EMR system.

2- Lack of Interoperability

Centralized EMR systems often operate in silos, making it challenging to share medical records seamlessly across different healthcare providers and organizations. Lack of interoperability hampers care coordination and continuity, as healthcare professionals struggle to access complete patient information from various sources. This fragmentation of data can lead to redundant tests, delays in treatment, and compromised patient safety.

3- Single Point of Failure

Relying on a single authority for data storage and management introduces the risk of a single point of failure. System outages, hardware failures, or other disruptions can render the entire EMR system inaccessible, potentially impacting patient care and impeding healthcare provider operations. In such instances, the unavailability of patient records can cause delays, compromise diagnosis and treatment, and hinder overall healthcare delivery.

4- Limited Control and Data Access

Centralized EMR systems give the managing authority significant control over patient data, including who can access and share it. This authority may impose restrictions and limitations on data access, potentially hindering collaboration and hindering patient care. Healthcare providers and patients often face challenges in obtaining their own medical records or sharing them with other authorized entities due to the centralized control and limitations imposed by the managing authority.

5- Lack of Transparency and Trust

In centralized EMR systems, there is often limited transparency regarding how data is stored, accessed, and shared. Patients and healthcare providers may lack visibility into the processes and mechanisms employed by the managing authority, raising concerns about data integrity, tampering, and accountability. This lack of transparency undermines trust in the system and poses challenges to establishing a robust and reliable EMR ecosystem.

6- Fragmented Patient Experience

The limitations of centralized EMR systems result in a fragmented patient experience, with medical records stored across multiple disconnected systems. Patients often need to provide their medical history repeatedly, leading to inefficiencies and potential gaps in information. This fragmented experience can impede personalized care, hinder informed decision-making, and compromise patient satisfaction and engagement.

1.3 Objectives

The MedRecChain project has several objectives that guide its development as a decentralized solution for Electronic Medical Records (EMR) sharing systems. These objectives encompass the core aspects that MedRecChain aims to achieve to address the limitations of centralized EMR systems and enhance the efficiency, security, and interoperability of EMR sharing.

1. Developing a Secure and Private Platform

One of the primary objectives of MedRecChain is to create a secure and private platform for sharing EMRs among healthcare providers, patients, and authorized entities. By leveraging blockchain technology, MedRecChain ensures the immutability and integrity of EMR data. The use of cryptographic techniques and access controls helps to protect patient privacy, preventing unauthorized access and ensuring that sensitive medical information remains confidential.

2. Enhancing Interoperability and Data Accessibility

   MedRecChain aims to improve interoperability between different EMR systems by establishing a standardized framework for data exchange. The project endeavors to overcome the limitations of siloed EMR systems and facilitate seamless information sharing between healthcare providers and organizations. By enabling efficient and standardized data interoperability, MedRecChain aims to enhance care coordination, reduce redundancies, and provide a comprehensive view of a patient's medical history.

3. Leveraging Blockchain Technology for Data Integrity

   Blockchain technology is a key component of MedRecChain, and the project aims to leverage its inherent characteristics to ensure data integrity and reliability. MedRecChain utilizes the Ethereum network to create a distributed and decentralized ledger for recording and storing EMR transactions. This distributed ledger offers immutability and transparency, reducing the risk of data tampering and providing a reliable audit trail for all EMR activities.

4. Establishing a Decentralized Network

   The central objective of MedRecChain is to establish a decentralized network for EMR sharing systems. By eliminating the reliance on a single authority or central server, MedRecChain aims to create a distributed network where EMRs are stored across multiple nodes, enhancing system resilience, and reducing the risk of a single point of failure. The decentralized architecture allows for increased data availability, fault tolerance, and scalability, ensuring continuous access to EMRs even in the event of localized disruptions.

5. Ensuring User-Friendly Experience

   MedRecChain acknowledges the importance of user experience for both healthcare providers and patients. The project aims to design and develop a user-friendly interface that simplifies the process of accessing and sharing EMRs. Intuitive interfaces and streamlined workflows will facilitate efficient navigation and ease of use, ensuring that healthcare providers can access the necessary patient information seamlessly and patients can have better control over their own medical records.

6. Contributing to Industry Standards and Best Practices

   MedRecChain aims to contribute to the establishment of industry standards and best practices for decentralized EMR sharing systems. By conducting rigorous research and development, the project aims to provide insights and guidelines that can be adopted by the broader healthcare community. Through knowledge dissemination and collaboration, MedRecChain seeks to facilitate the adoption of decentralized EMR systems and drive positive change in the healthcare industry.

   By achieving these objectives, MedRecChain endeavors to revolutionize EMR sharing systems by providing a decentralized, secure, and interoperable platform. The project's focus on privacy, data integrity, user experience, and industry standards aims to address the challenges of centralized EMR systems and pave the way for a more efficient and patient-centric approach to EMR sharing.

1.4 Scope

After talking about these problems, we will try to solve them through a web application to serve patients and doctors.

- First: we will solve the main challenges existing in the current way hospitals manage their data and the lack of sharing with other institutes as they can easily find all the information from different institutes organized in one place.
- Second: patients can have all their medical information as it is difficult for them to keep up with their medical history, either be it complicated or long.
- Third: specialist's diagnosis will be more accurate as it will depend on the patient's condition in similar situations that he will get it from the patient's medical history.

The project As contributes to save people's lives through their knowledge of their health and knowledge of their diseases that can be detected early from their medical history such as hepatitis B, C, cancer, diabetes, and herpes virus which are the most diseases that can be predictable from the patient's medical history like it's updatable test blood and his past diseases or from their family medical history as they can share their information with other patients like their family. Each person will have a medical file containing its own medical history.

Provides medical history will help doctors to diagnose the disease as by knowing the patients' old diseases will increase the percentage of accurate diagnose in a lot of diseases as we mentioned before and save doctors and patient's time.

Our project will provide a single format for all patient's medical record so it will be more readable and easier to record it from different institutes.

1.5 Stakeholders

1. Healthcare Providers

   Healthcare providers, including doctors, nurses, specialists, and other medical professionals, are crucial stakeholders in the MedRecChain project. They directly interact with Electronic Medical Records (EMRs) and require a secure, efficient, and user-friendly system for accessing and managing patient data. Their input and involvement are critical to ensuring that MedRecChain meets their specific needs, enhances their workflows, and improves the overall quality of patient care.

2. Patients

   Patients are key stakeholders in the MedRecChain project as they are the subjects of the medical records being managed and shared. They have a vested interest in the privacy, security, and accessibility of their EMRs. Patients expect their sensitive health information to be safeguarded and shared only with authorized healthcare providers. Involving patients in the design and development of MedRecChain ensures that their rights and preferences are respected, promoting trust and patient-centric care.

3. Healthcare Organizations and Institutions

   Healthcare organizations and institutions, such as hospitals, clinics, and medical centers, are important stakeholders in the successful implementation of MedRecChain. These organizations manage vast amounts of patient data and require a reliable and interoperable system for EMR sharing. Collaboration with healthcare organizations is crucial to align MedRecChain with their existing infrastructure, workflows, and regulatory requirements.

   Their involvement ensures that the system integrates seamlessly into healthcare environments and promotes interoperability across different organizations.

Chapter 2: Background and Related work

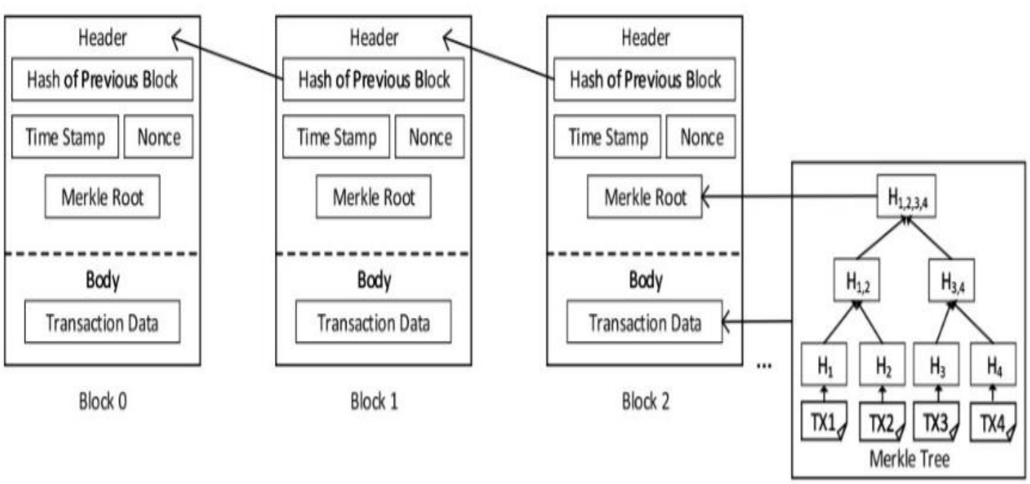
2.1 Blockchain

2.1.1 What is Blockchain?

Blockchain is a decentralized, trustworthy distributed ledger on a peer-to-peer network that consists of a list of chronologically ordered blocks that cannot be altered retroactively, without the alteration of all subsequent blocks. Every block consists of a hash of a previous block and hence forming a chain. The first block in the blockchain is the genesis block and the block before a given block is called its parent block.

A block contains a header and body. The block header consists of:

- Version: Specifies the block validation rules.
- Previous block hash: This is to ensure that the previous block cannot be changed, without changing the current block header.
- Timestamp: The current block creation time.
- Merkle root hash: The Merkle root is obtained from the hash values of all the transactions present in the block. This is to ensure that the transactions cannot be modified without changing the header.
- Nonce: A nonce is a 4-byte unique number that is used only once in communication.



2.1.2 History of Blockchain

1. **1991 - Blockchain Invention**

   Scott Stornetta and Stuart Haber invented Blockchain. They created a cryptographically protected chain of blocks which would prevent anyone from tampering with document timestamps.

2. **2008-2013 - Bitcoin Emergence**

   Blockchain technology gained popularity due to Bitcoin in 2008, as it is the first application in the blockchain. Bitcoin was conceptualized by Satoshi Nakamoto. In 2009, Nakamoto published a white paper on Bitcoin. First Bitcoin was purchased for 10,000BTC in 2010. Bitcoin crosses $1 billion in 2013.

3. **2013-2015 - Ethereum Development**

   Ethereum was conceptualized by Vitalik Buterin. Ethereum has additional features like smart contracts compared to Bitcoin. The development of Ethereum has proven to be an important moment in the history of Blockchain. Ethereum is used for cryptocurrencies and in many other decentralized applications. because of smart contracts.

4. **2015 - Hyperledger**

   Hyperledger was developed by Linux Foundation in 2015. It allows development of open source blockchain. The different Hyperledger frameworks are Hyperledger Fabric, Hyperledger Iroha, Hyperledger Sawtooth, and Hyper-ledger Burrow.

5. **2015 - Future**

   Many cryptocurrencies and applications have been developed after the emergence of Bitcoin and Ethereum. Nowadays, Blockchain technology is used by various companies and organizations.

2.1.3 Type of Blockchain

Blockchain is divided into four types, public, private, consortium and hybrid blockchains based on permission and assess.

1. **Public Blockchain**

It is a permissionless distributed technology. As the name suggests, anyone with the internet is allowed to access the public blockchain and participate in the transactions. Each peer in this network has their own copy of the ledger. Consensus mechanisms like Proof of Work (PoW), Proof of Stake (PoS), etc., are required to reach consensus while adding new blocks and during verification of transactions. Public blockchains are fully decentralized because all the peers have equal authority, which in terms makes it secure as no one can have full control over the blockchain network. In turn, ensuring data security helps in keeping the ledger immutable. Some examples are Bitcoin, Litecoin, and Ethereum.

2. **Private Blockchain**

It is also known as permissioned blockchain. There are limitations on who can be part of the network and who can contribute to the transactions. A private blockchain is used by an organization or company for its internal usage. They are centralized i.e.; one organization has authority in the network. It provides transparency and security to the participants. For example, Hyperledger Fabric.

3. **Consortium Blockchain**

In this blockchain, more than one organization can manage the network, hence it is not fully centralized. It is used by organizations that need both public and private blockchain functionality. For example, Quorum and Hyper ledger Fabric.

4. **Hybrid Blockchain**

It is a mix of the private and public. In the blockchain, the peers determine who has access to which data. Some processes are kept private, while others are made public. On an open ledger, businesses can protect background transactions with business partners while still providing the product details to customers. For example, Dragon chain.

2.1.4 Consensus mechanisms

1. Proof of Work

Bitcoin uses (PoW) as the consensus mechanism. Miners “mine” a block to connect to the blockchain by solving cryptographic puzzles. This method requires a significant amount of energy and computation. The puzzles have been created in such a way that they are challenging and demanding on the system. When a miner completes the puzzle, they send their block for verification to the network. The method of determining whether a block belongs in the chain or not is extremely easy. A 51% attack is a possible attack in the blockchain, in which a miner or a group of miners with more than 51% of the computing power will prevent new blocks from being produced and establish false transaction records that benefit the attackers.

2. Proof of Stake

It is a more energy-efficient version of Pow. Nodes with the most stakes (for example, currency) are thought to be less likely to strike the network. The disadvantage here is the monopoly when it comes to both technological and economic aspects of the scheme, the main stakeholder has full influence and authority.

2.1.5 Main Features of blockchain

Blockchain has a lot of features, and the following is the most basic features of why we are using blockchain not regular database.

- Immutability.

There are some exciting blockchain features but among them “Cannot be Corrupted” is undoubtedly one of the key features of blockchain technology.
It means something that can’t be changed or altered. This is one of the blockchain features that help to ensure that the technology will remain as it is – a permanent, unalterable network.

- Decentralized Technology

The network is decentralized meaning it doesn't have any governing authority or a single person looking after the framework. Rather a group of nodes maintains the network making it decentralized.

- Security

Using encryption ensures another layer of security for the system, every information on the blockchain is hashed cryptographically. In simple terms, the information on the network hides the true nature of the data. Any input data gets through a mathematical algorithm that produces a different kind of value. And each user will have a private key to access the data but will have a public key to make transactions.

- Irreversible

Hashing is quite complex, and it's impossible to alter or reverse it. No one can take a public key and come up with the private key. Also, a single change in the input could lead to a completely different ID, so small changes aren't a luxury in the system. Accessing and hacking millions of computers is next to impossible and costly.

- Distributed Ledgers

It is like a spreadsheet that contains every node in a network and records every purchase made by that node. The digital signature prevents tampering with the information in the digital ledger and assures its security. The most remarkable aspect of this ledger is while the data can be viewed by anybody, it cannot be changed.

2.1.6 Ethereum vs Hyperledger Fabric

Hyperledger and Ethereum are the most popular blockchain platforms and networks, respectively. Both are free and open source. They have also aided in the development of a huge number of blockchain applications.

Table 0-12.1.6 Ethereum vs Hyperledger Fabric

Feature	Ethereum	Hyperledger
Confidentiality	Public blockchain	Private blockchain
Purpose	Client-side B2C applications	Enterprise-level B2B applications
Participation	Anyone	Only authorized individuals can participate
Governance	Ethereum Developers	Linux Foundation
Consensus Mechanism	Proof-of-stake	RBFT

We use Ethereum because: -

1. Public or B2C applications: Anyone can connect to the network and set up a node (new patient). Every such node has a complete copy of the blockchain.
2. Roles and tasks of nodes are Identical, Differentiated, and straightforward.
3. And easy to learn it

2.1.7 Smart Contract

A smart contract is a method of digitally formalizing and securing network interactions. The fundamental idea behind smart contracts is to enable the embedding of various contractual provisions, collateral, bonding, and property rights in computer software or hardware, reducing the likelihood of an intentional breach of the contracts. A smart contract is described in this context as an application that runs on the blockchain network and is executed by all network members. Smart contracts are computer programs that manage blockchain transactions and specify the terms of mutually agreed-upon contracts.

Many blockchain-based projects, including the Ethereum platform and Hyperledger, have recently added smart contracts. They enable trustworthy agreements and transactions to be made between different, anonymous entities without the necessity for a centralized authority or an external enforcement mechanism. The Ethereum platform enables the creation of smart contracts that meet the needs of the intended system. When smart contracts are used in EARs systems, they enable the construction of scalable and dynamic conditions, terms, and norms for the safe exchange and sharing of academic records.

2.1.8 IPFS Technology

It is a distributed file storage system that enables computers across the world to store and serve information as part of a massive peer-to-peer network. in a peer-to-peer (P2P) network, each computer acts as both a server and a client supplying and receiving files with bandwidth and processing distributed among all members of the network.

Every file added to IPFS is given a unique address derived from a hash of the file's content. This address is called a Content Identifier (CID) and it combines the hash of the file and a unique identifier for the hash algorithm used into a single string.

2.2 Medical Record

2.2.1 What is an electronic health record?

It is digital version of a patient's paper chart that make information available instantly and securely to authorized user. it contains the medical, treatment histories of patients, diagnoses, medications, treatment plans, immunization dates, allergies, radiology images, and laboratory and test results. so, it is an amalgam of all the data acquired and created during a patient's course through the health care system.

Electronic health record (EHR) is a repository of electronically maintained information about an individual's lifetime health status and health care. Also, it can integrate multimedia information such as radiology images and echocardiographic video loops that were never part of the traditional medical record

2.2.2 Use Blockchain in Electronic health record.

Blockchain has the potential to change the way electronic health records are kept and exchanged for patients. It has the potential to create a more secure, transparent, and traceable foundation for health information transmission. The technology offers the ability to integrate numerous data management systems that are now functioning in silos, resulting in a connected and interoperable electronic health record system.

Healthcare data can be stored immutably in a decentralized manner instead of storing in one database. As a result, there would be no single point of entry for a hacker to retrieve the data. That’s how blockchain can ensure better data security for health information.

Also, patients have access to their data. Once the data has been saved on the blockchain and allocated to the public key, the patient can restrict access to the data to just the authorities that need it.

2.2.3 Existing Electronic Medical Record Systems

In this section, we will provide additional information about the existing electronic medical record (EMR) systems, including MEDREC, Medical Chain, MedBloc, and Medshare, in order to enhance the understanding of their respective functionalities and contributions.

1. MEDREC:

MEDREC is a decentralized system proposed in 2016 that utilizes blockchain technology to handle health records. It played a significant role in facilitating the emergence of data economies by providing researchers with access to vast amounts of data. Patients and service providers have the freedom to choose whether to publish their data, enabling greater control over data sharing.

2. MedicalChain

MedicalChain is an EMR system that harnesses blockchain technology to develop a user-focused electronic health record. It ensures the existence of a single true version of the user's data, reducing data inconsistencies and promoting data accuracy. Additionally, MedicalChain serves as a platform for the creation of applications that enhance the user experience. Users can leverage their medical information to power a variety of applications and services .

3. MedBloc

MedBloc is a blockchain-based system that enables healthcare organizations and patients to securely access and share health records. It utilizes encryption and smart contract-based access control mechanisms to safeguard medical data. One of the key features of MedBloc is that it allows patients to take ownership of their medical records by granting or denying others access. The immutability of blockchain ensures that unauthorized modifications to the data are not possible. The transparency of the blockchain enables data auditability and provenance, allowing patients to track who accessed their medical records and how their data were used .

4. Medshare:

Medshare is a system designed to facilitate the sharing of health data and provide control over cloud-based warehouses between hospitals. It aims to address the challenges of interoperability and data exchange in the healthcare industry. While further details are not available, it can be inferred that Medshare enables healthcare institutions to securely share patient data, ensuring data availability and accessibility.

These existing EMR systems have made valuable contributions to the field by leveraging blockchain technology to enhance data security, privacy, and interoperability. However, each system has its own unique features and focus areas. The MedRecChain project aims to build upon these existing solutions by introducing its own advancements, such as utilizing the Ethereum network and IPFS technology, emphasizing patient control and ownership, and implementing enhanced security and privacy measures. By doing so, MedRecChain seeks to provide a comprehensive and innovative solution for decentralized EMR sharing systems.

2.2.4 Comparison Between The Relevant Work And Our Project

System	Technology Used	Focus/Goal	Patient Control	Data Privacy and Security
MEDREC	Blockchain (Ethereum-based)	Enable secure and decentralized EMR	Limited patient control	Robust data encryption and access control
MedicalChain	Blockchain (Ethereum-based)	User-focused EMR with single source	Patient control and consent	Encryption and secure access to data
MedBloc	Blockchain (Not specified)	Secure health data sharing	Patient ownership and control	Encryption and smart contract-based access control
Medshare	Cloud-based	Health data sharing between hospitals	Limited patient control	Cloud-based security measures
MedRecChain	Ethereum network, IPFS	Decentralized EMR sharing	Patient control and ownership	Enhanced security and privacy measures

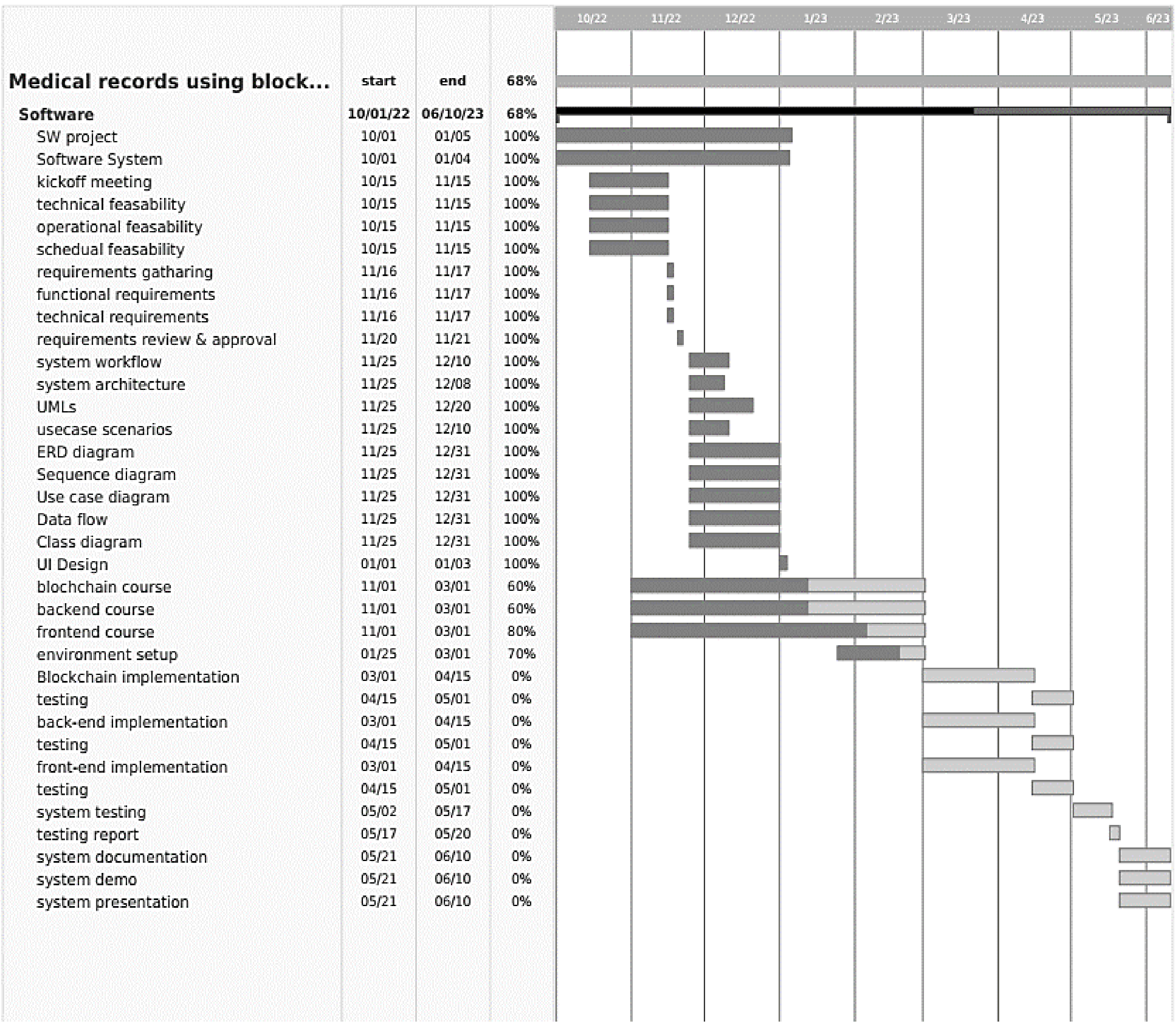
Chapter 3: Systems Analysis and Design

3.1 User Requirements

1. The user can use the app with him at any place and any time so our app should be available at any home and any time.
2. The user easy to see his medical history.
3. The system secures users' medical history and profiles to avoid data breaches.

a. Gantt Chart

This chart describes the workflow of building the system during the whole period. The workflows are moderated through agile methodology.



b. System Requirements

1. Browser: The application works on any browser ex." Chrome, Microsoft edge, Internet explorer"
2. Internet connection: This app uses a cloud server to store data, so you need to be connected to the internet to fetch and view it

c. Develop Requirements and Tools

We have used a combination of technology and tools that help build a decentralized medical record system:

1. **Front-End**:
   - HTML, CSS, JavaScript
   - React.js
   - Bootstrap
2. **Back-End:**
   - Web3.js (library that allows to interact with a Ethereum node, smart contract, and your app)
   - Solidity (Programming language for implementing smart contracts on blockchain network)
3. **Blockchain:**
   - Ethereum network
   - Truffle (Farmwork for Dapp that provide EVM (Ethereum Virtual Machine) so we can deploy and test)
   - Ganache (It is local Ethereum blockchain. It provided us with number of accounts allowed us to test and deploy the application and dividing them among application different end users.)
   - MetaMask (Browser extension that is gateway to blockchain apps by make account to manage Ethereum smart contract)
   - IPFS (It is a protocol and peer-to-peer network for data storage and sharing in a distributed file system.)
4. **Workspace**
   - Visual Studio Code & GitHub

d. Risk Management

a) Schedule Risk

	Risk	Probability rate	Impact rate
Schedule Risk	Wrong time estimate	Low likely	High
	Resources like team and their skills are not tracked properly	Mid likely	High
	Failure to identify time required to develop function	Mid likely	Mid
	Unexpected project growth	Most likely	High

Can be avoided by:

- ✓ Following up the Gantt chart regularly and reducing the critical paths in the project network.
- ✓ Look after the major tasks that the other tasks depend on.
- ✓ Scheduling the risky tasks first.

b) Operational Risk

	Risk	Probability rate	Impact rate
Operational Risk	No Resource planning	Low likely	mid
	No communication in the team	low likely	High
	No distribution for responsibility	Mid likely	High
	Not enough training	Mid likely	High

Can be avoided by:

- ✓ Monitoring and evaluations at regular intervals.
- ✓ Cutting the project into very small tasks.
- ✓ Full training before proceeding with the work.

c) Technical Risk

	Risk	**Probability rate**	**Impact rate**
Technical Risk	Changing requirements	Most likely	High
	Database is not efficient	low likely	Mid
	Application failure	Mid likely	High

Can be avoided by:

- ✓ Understanding the platform\tools used.
- ✓ Not rushing the analysis phase.
- ✓ Specifying the desired outcome.
- ✓ Detailed test cases and regulations.

3.2 Functional Requirements

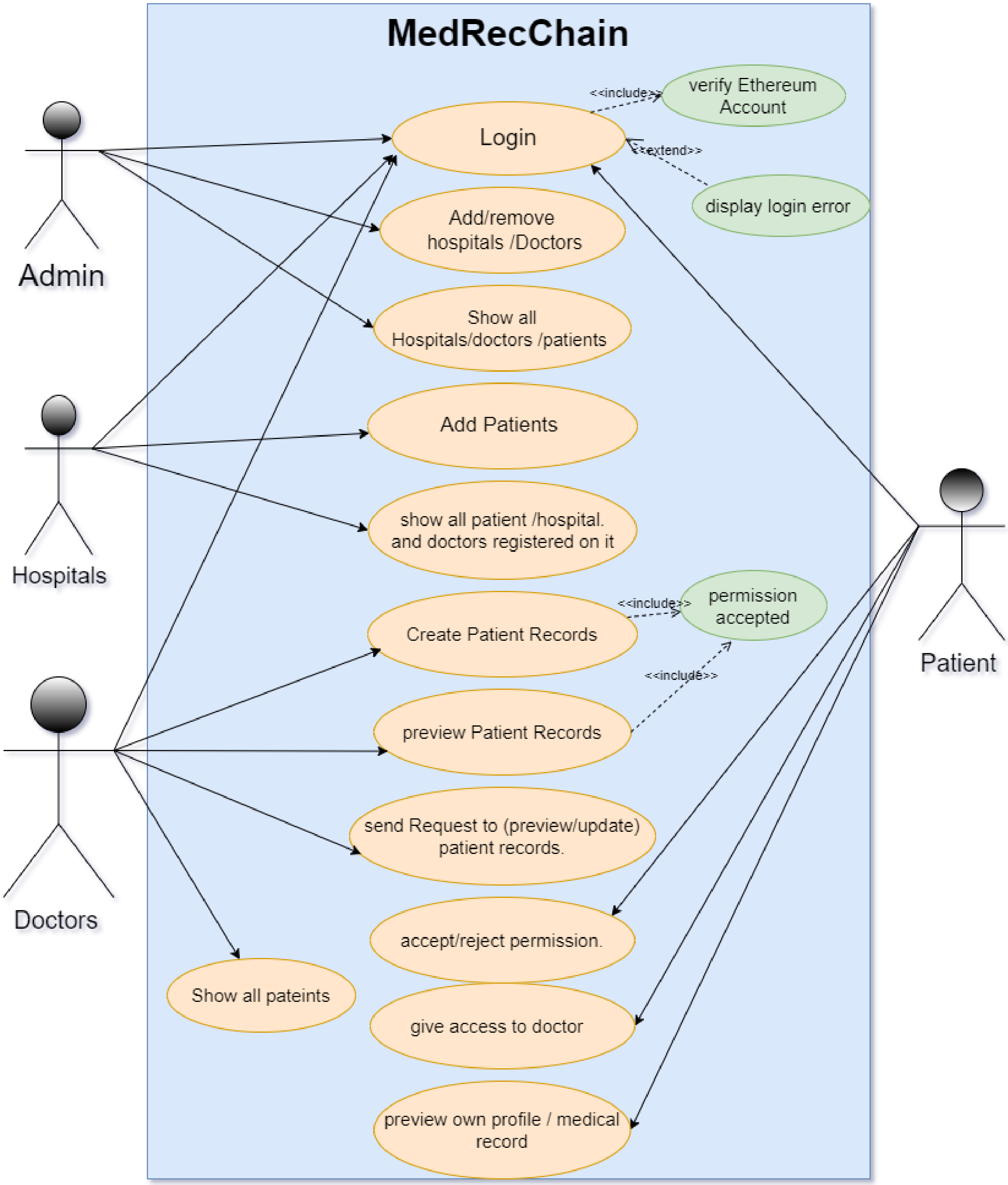
1. **User Registration:** Users should be able to create accounts and register on the MedRecChain platform.
2. **EMR Uploading**: Users should be able to upload their electronic medical records to the MedRecChain platform.
3. **EMR Sharing**: Users should be able to share their medical records with other authorized users on the platform.
4. **Access Control**: The system should enforce access control mechanisms to ensure that only authorized users can view and modify medical records.
5. **Data Integrity:** The platform should maintain the integrity of medical records to prevent unauthorized modifications.
6. **Search and Retrieval**: Users should be able to search for specific medical records based on various criteria and retrieve them efficiently.
7. **Audit Trail**: The system should keep a log of all activities related to medical records, including uploads, modifications, and access, for auditing purposes.

.

e. Non-Functional Requirements

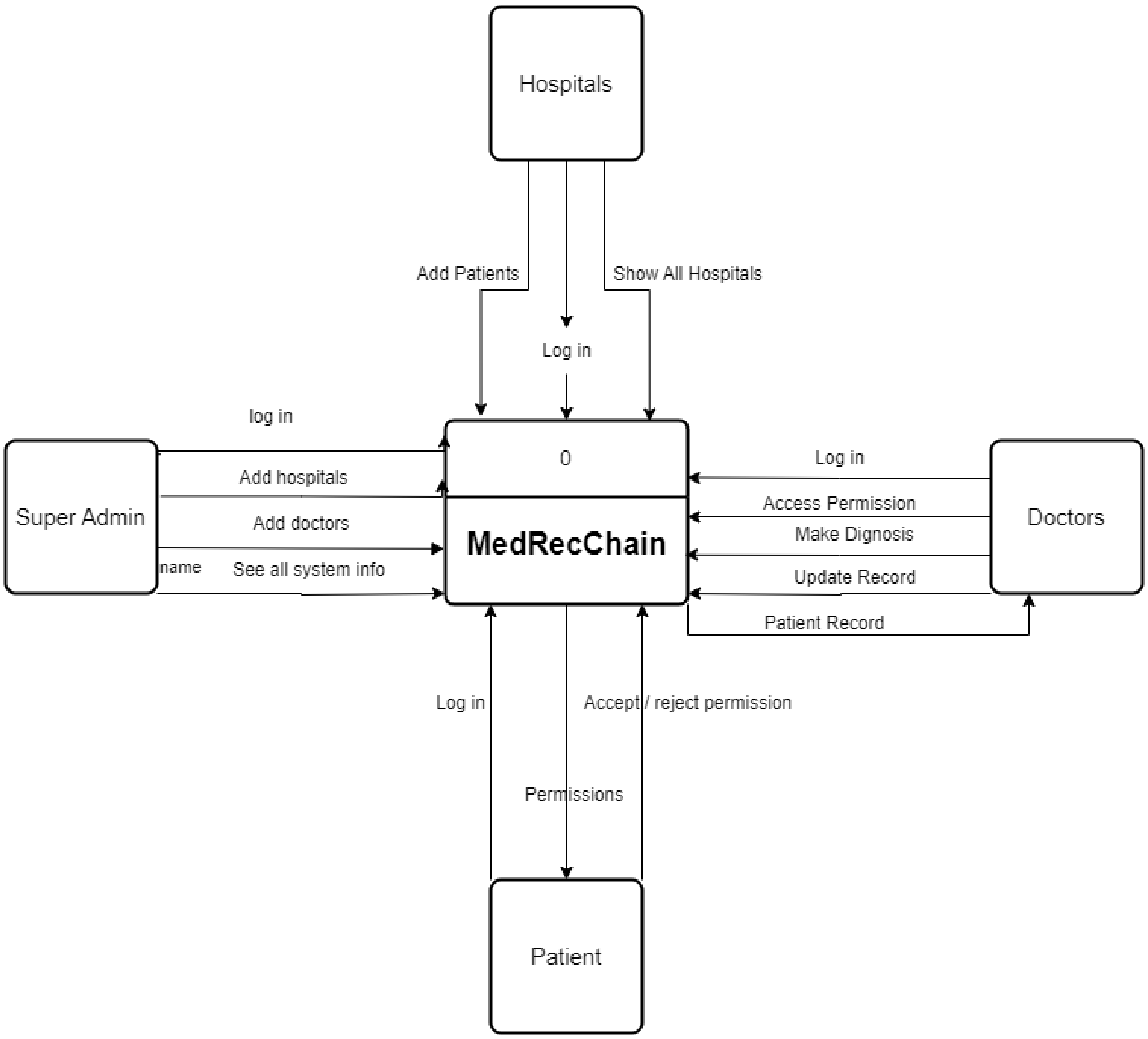
1. **Security**: The platform should employ robust security measures to protect the confidentiality and privacy of medical records.
2. **Scalability**: The system should be able to handle a large number of users and medical records without compromising performance.
3. **Reliability**: The platform should be reliable and available for users to access their medical records at any time.
4. **Interoperability**: The system should be compatible with existing EMR systems to facilitate seamless data exchange.
5. **Usability**: The platform should have a user-friendly interface and intuitive navigation to ensure ease of use for both technical and non-technical users.
6. **Performance**: The system should have fast response times for uploading, sharing, and retrieving medical records to provide a smooth user experience.
7. **Compliance**: The platform should adhere to relevant legal and regulatory requirements, such as data protection laws and healthcare industry standards.

3.8 Use case Diagram.

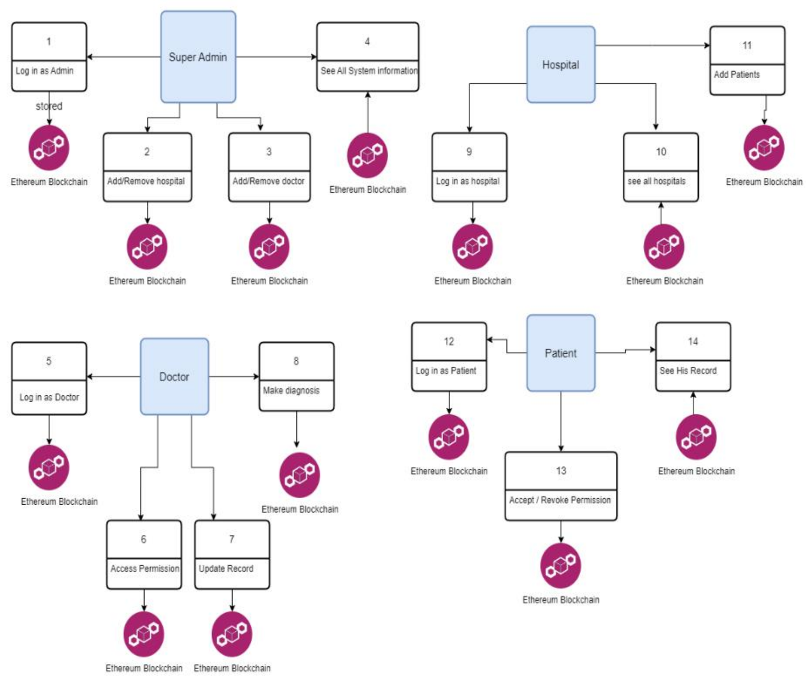


3.9Data flow Diagram.

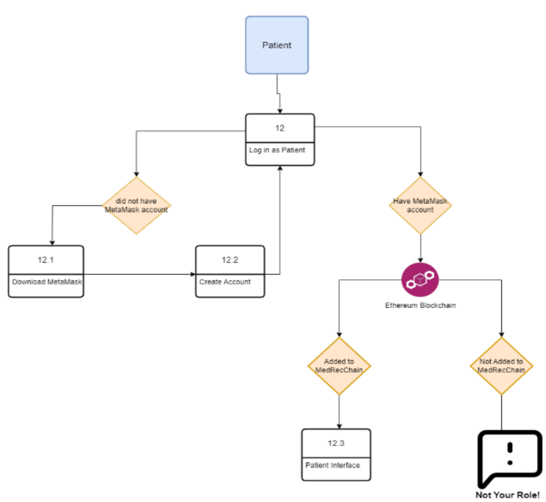
1- Context Diagram



2- Level -1

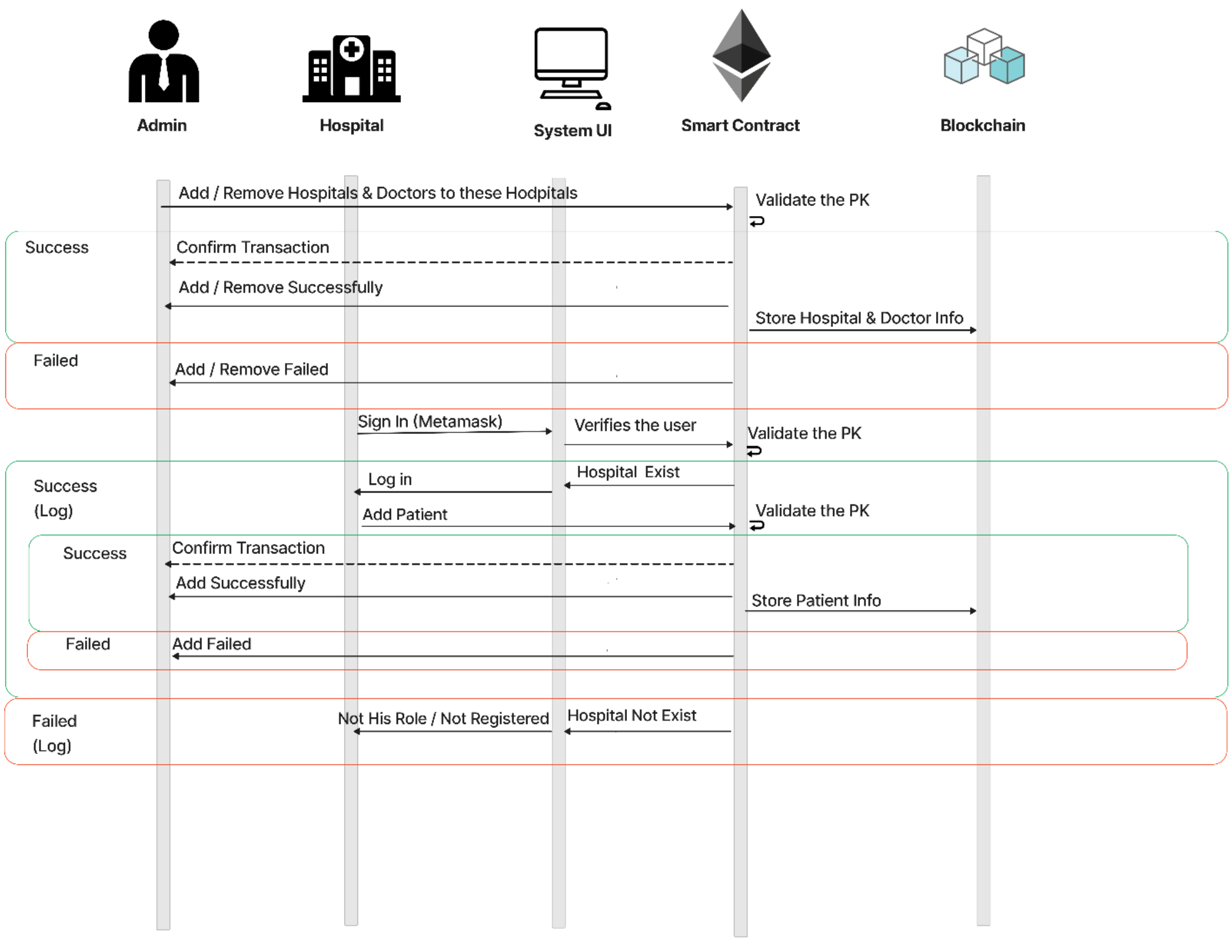


3- Level -2

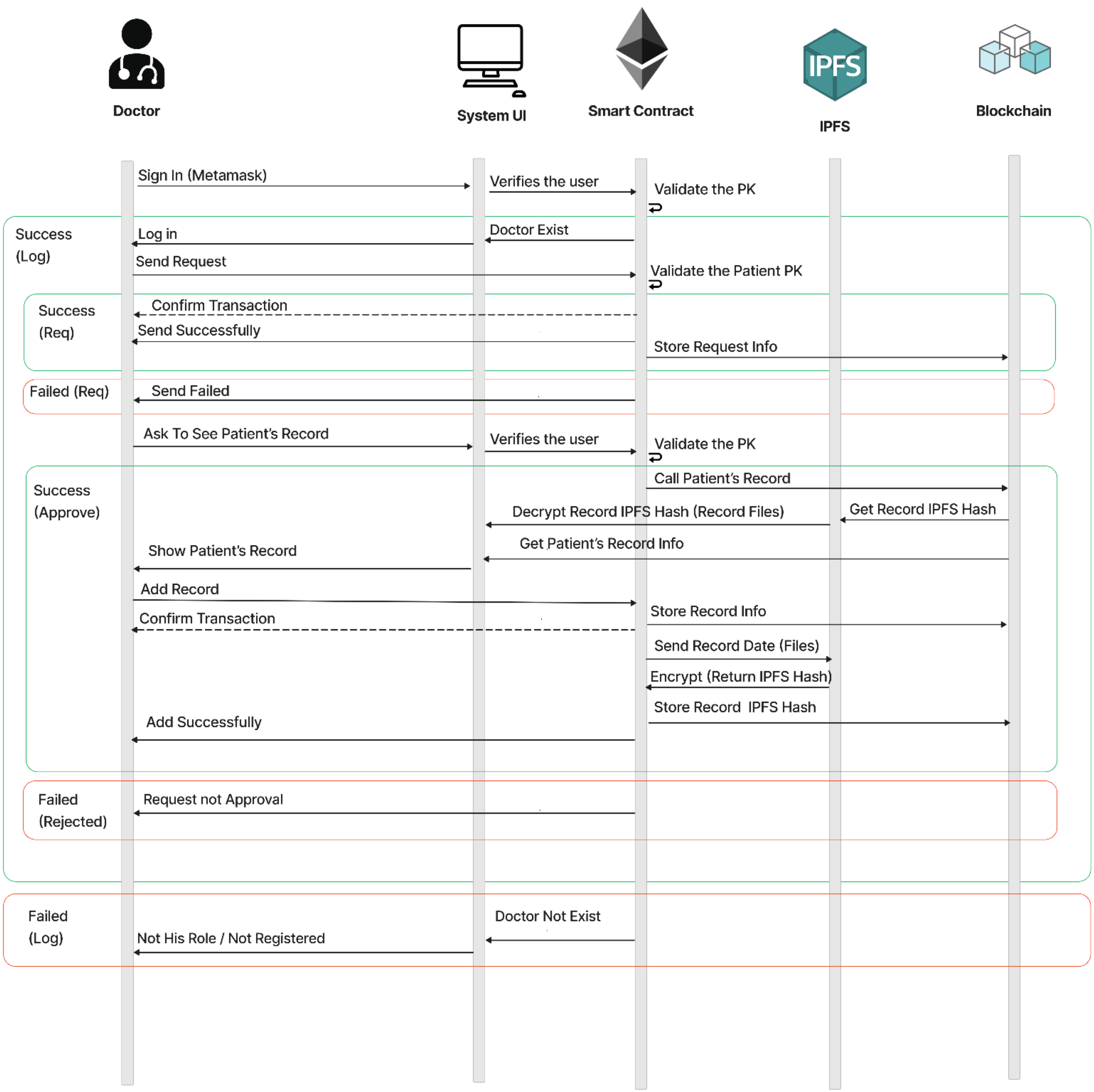


3.10 Sequence Diagram

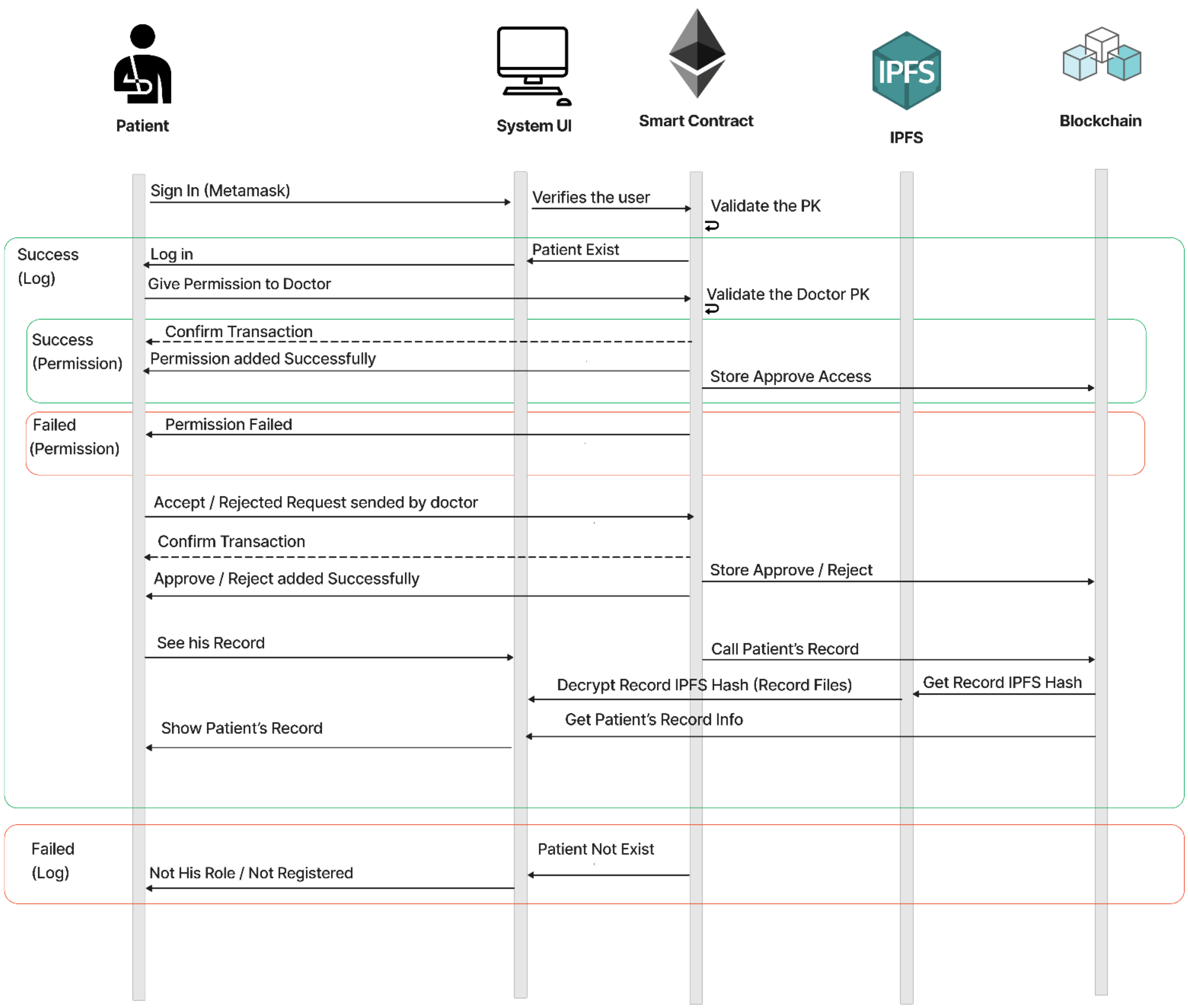
1- (Admin -Hospital)



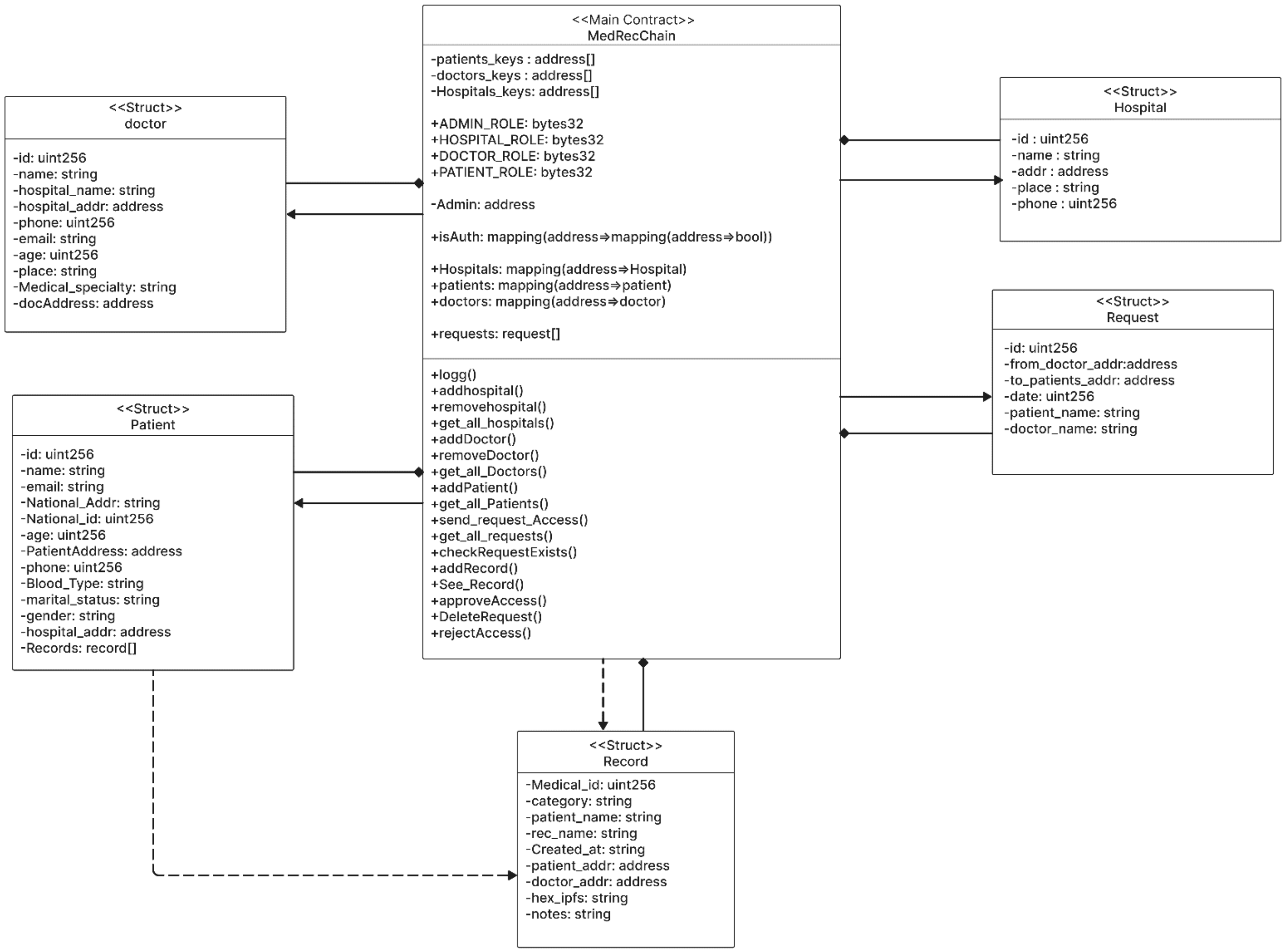
2- Doctor


Doctor
System UI
Smart Contract
IPFS
IPFS
Blockchain
Sign In (Metamask)
Verifies the user
Validate the PK
Success
(Log)
Log in
Doctor Exist
Send Request
Validate the Patient PK
Success
(Req)
Confirm Transaction
Send Successfully
Store Request Info
Failed (Req)
Send Failed
Ask To See Patient's Record
Verifies the user
Validate the PK
Success
(Approve)
Call Patient's Record
Decrypt Record IPFS Hash (Record Files)
Get Record IPFS Hash
Show Patient's Record
Get Patient's Record Info
Add Record
Store Record Info
Confirm Transaction
Send Record Date (Files)
Encrypt (Return IPFS Hash)
Store Record IPFS Hash
Add Successfully
Failed
(Rejected)
Request not Approval
Failed
(Log)
Doctor Not Exist
Not His Role / Not Registered

3- Patient


Patient
System UI
Smart Contract
IPFS
Blockchain
Sign In (Metamask)
Verifies the user
Validate the PK
Success
(Log)
Log in
Patient Exist
Give Permission to Doctor
Validate the Doctor PK
Success
(Permission)
Confirm Transaction
Permission added Successfully
Store Approve Access
Failed
(Permission)
Permission Failed
Accept / Rejected Request sended by doctor
Confirm Transaction
Approve / Reject added Successfully
Store Approve / Reject
See his Record
Call Patient's Record
Decrypt Record IPFS Hash (Record Files)
Get Record IPFS Hash
Show Patient's Record
Get Patient's Record Info
Failed
(Log)
Patient Not Exist
Not His Role / Not Registered

3.11 Class Diagram



3.12 System Architecture

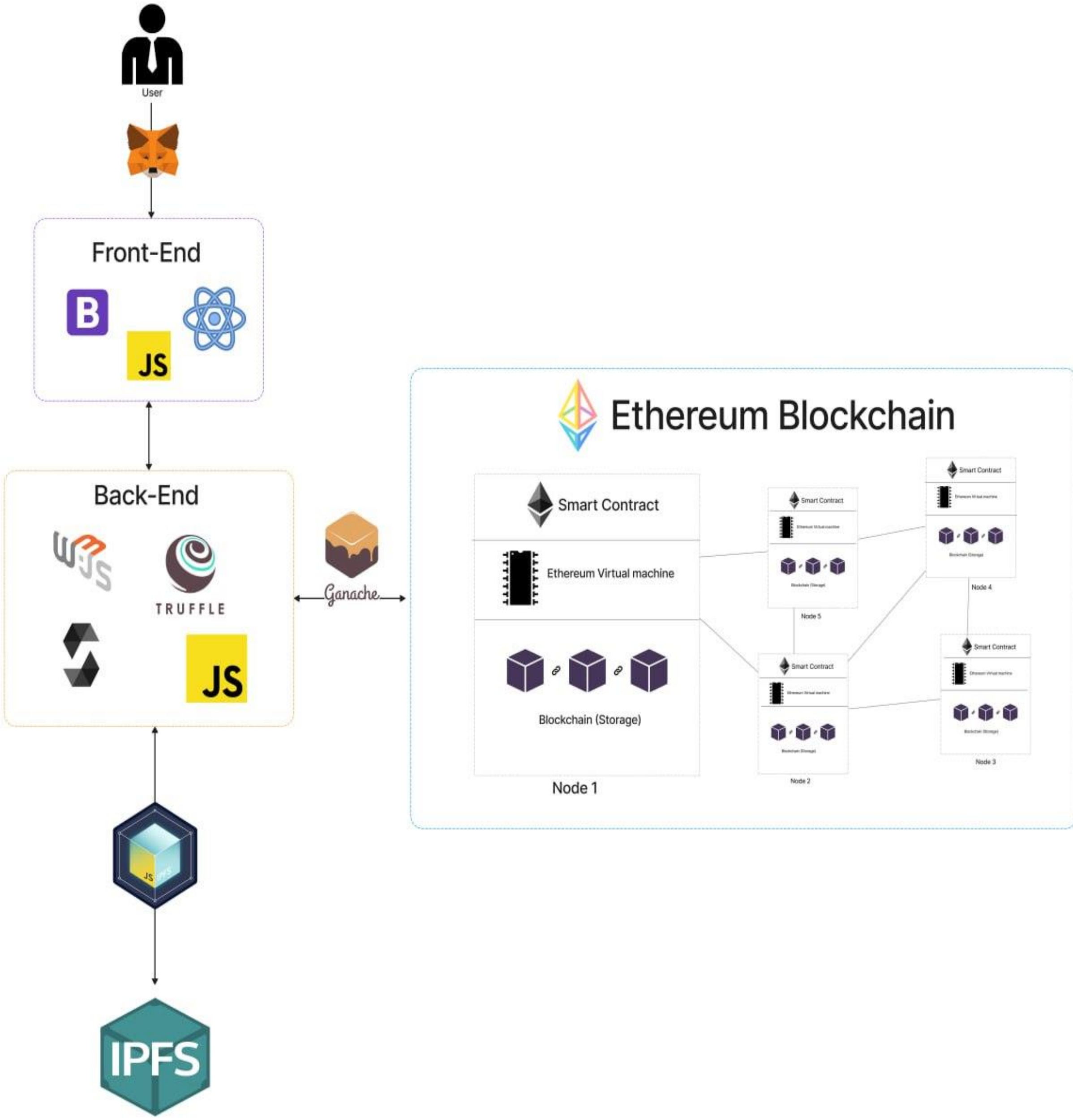


Figure 0-1system architecture

Chapter 4: Implementation

4.1Smart Contract

The contract defines various roles (admin, hospital, doctor, patient) using the AccessControl contract from the OpenZeppelin library.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/access/AccessControl.sol";

contract MedRecChain is AccessControl {
    //ROLES
    bytes32 public constant ADMIN_ROLE = keccak256("ADMIN_ROLE");
    bytes32 public constant HOSPITAL_ROLE = keccak256("HOSPITAL_ROLE");
    bytes32 public constant DOCTOR_ROLE = keccak256("DOCTOR_ROLE");
    bytes32 public constant PATIENT_ROLE = keccak256("PATIENT_ROLE");

```

This a struct of Hospital.

```

struct Hospital {
    uint256 id;
    string name;
    address addr;
    string place;
    uint256 phone;
}

mapping(address => Hospital) Hospitals;
address[] public Hospitals_keys;
```

Struct of Patient

```
struct patient {
    uint256 id;
    string name;
    string email;
    string National_Addr;
    uint256 National_id;
    uint256 age;
    address PatientAddress;
    uint256 phone;
    string Blood_Type;
    string marital_status;
    string gender;
    address hospital_addr;
    record[] Records;
}

mapping(address => patient) patients;
address[] private patients_keys;
```

Doctor

```
struct doctor {
    uint256 id;
    string name;
    string hospital_name;
    address hospital_addr;
    uint256 phone;
    string email;
    uint256 age;
    string place;
    string Medical_specialty;
    address docAddress;
}
```

Record

```
struct record {
    uint Medical_id;
    string category;
    string patient_name;
    string rec_name;
    string Created_at;
    address patient_addr;
    address doctor_addr;
    string hex_ipfs; // file, image date
    string notes;
}
```

This modifier to check who deploy or act this function.

```
//To check the role of who deploye the contract.
modifier onlyAdmin() {
    require(
        hasRole(ADMIN_ROLE, msg.sender),
        "Only superAdmin has permission to do that action"
    );
    _;
}

modifier onlyHospital() {
    require(
        hasRole(HOSPITAL_ROLE, msg.sender),
        "This account is not Hospital"
    );
    _;
}
modifier onlydoctors() {
    require(hasRole(DOCTOR_ROLE, msg.sender), "This account is not Doctor");
    _;
}
modifier onlypatient() {
    require(
        hasRole(PATIENT_ROLE, msg.sender),
        "This account is not patient"
    );
    _;
}
```

Initializes the contract and sets the ADMIN_ROLE for the contract deployer.

```
constructor() {
        _setupRole(ADMIN_ROLE, Admin);
  }
```

```
// function to know who logg in
  function logg() public view returns (uint role) {
      if (hasRole(ADMIN_ROLE, msg.sender)) {
          return 1;
      } else if (hasRole(HOSPITAL_ROLE, msg.sender)) {
          return 2;
      } else if (hasRole(DOCTOR_ROLE, msg.sender)) {
          return 3;
      } else if (hasRole(PATIENT_ROLE, msg.sender)) {
          return 4;
      } else {
          return 0;
      }
  }
```

Add Hospital function that deployed only by admin.

```
function addhospital(
      string memory _name,
      address _address,
      string memory _place,
      uint256 _phone
  ) public onlyAdmin returns (bool success) {
      require(
          !hasRole(HOSPITAL_ROLE, _address),
          "This hospital is really exisit!! "
      );
      require(!hasRole(ADMIN_ROLE, _address), "This Account is Admin!! ");
      require(!hasRole(DOCTOR_ROLE, _address), "This Account is Doctor!! ");
      require(!hasRole(PATIENT_ROLE, _address), "This Account is Patient!! ");

      _setupRole(HOSPITAL_ROLE, _address);
      Hospital_index = Hospital_index + 1;
      Hospitals[_address] = Hospital(
          Hospital_index,
          _name,
          _address,
          _place,
          _phone
      );
      Hospitals_keys.push(_address);
      return true;
  }
```

Delete Hospital function

```
function removehospital(
    address _address
) public onlyAdmin returns (bool success) {
    require(
        hasRole(HOSPITAL_ROLE, _address),
        "This hospital is not exisit"
    );
    _revokeRole(HOSPITAL_ROLE, _address);
    delete Hospitals[_address];
    removeitem_hos(_address);
    return true;
}
```

Add Doctor function that deployed only by Hospital

```
function addDoctor(
        string memory _name,
        string memory _hospital_name,
        address _hospital_addr,
        uint256 _phone,
        string memory _email,
        uint256 _age,
        string memory _place,
        string memory _Medical_specialty,
        address _docAddress
    ) public onlyAdmin returns (bool success) {
        require(
            !hasRole(DOCTOR_ROLE, _docAddress),
            "This account already a doctor"
        );
        require(!hasRole(ADMIN_ROLE, _docAddress), "This Account is Admin!! ");
        require(
            !hasRole(HOSPITAL_ROLE, _docAddress),
            "This Account is Hospital!! "
        );
        require(
            !hasRole(PATIENT_ROLE, _docAddress),
            "This Account is Patient!! "
        );
        require(
            hasRole(HOSPITAL_ROLE, _hospital_addr),
            "This hospital is really exisit!! "
        );
        _setupRole(DOCTOR_ROLE, _docAddress);
        Doctor_index = Doctor_index + 1;
        doctors[_docAddress] = doctor(
            Doctor_index,
            _name,
            _hospital_name,
            _hospital_addr,
            _phone,
            _email,
            _age,
            _place,
            _Medical_specialty,
            _docAddress
        );
        doctors_keys.push(_docAddress);
        return true;
    }

```

Delete doctor function.

```
function removeDoctor(
    address _docAddress
) public onlyAdmin returns (bool success) {
    require(hasRole(DOCTOR_ROLE, _docAddress), "This account not exisit");
    delete doctors[_docAddress];
    _revokeRole(DOCTOR_ROLE, _docAddress);
    removeitem_doc(_docAddress);
    return true;
}
```

Add Patient function

```
function addPatient(
        string memory _name,
        string memory _email,
        string memory _National_Addr,
        uint256 _National_id,
        uint256 _age,
        address _PatientAddress,
        uint256 _phone,
        string memory _Blood_Type,
        string memory _marital_status,
        string memory _gender
    ) public onlyHospital returns (bool success) {
        require(
            !hasRole(PATIENT_ROLE, _PatientAddress),
            "This account already a patient"
        );
        require(
            !hasRole(ADMIN_ROLE, _PatientAddress),
            "This Account is Admin!! "
        );
        require(
            !hasRole(DOCTOR_ROLE, _PatientAddress),
            "This Account is Doctor!! "
        );
        require(
            !hasRole(HOSPITAL_ROLE, _PatientAddress),
            "This Account is Hospital!! "
        );
        _setupRole(PATIENT_ROLE, _PatientAddress);
        Patient_index = Patient_index + 1;
        patients[_PatientAddress].id = Patient_index;
        patients[_PatientAddress].PatientAddress = _PatientAddress;
        patients[_PatientAddress].name = _name;
        patients[_PatientAddress].email = _email;
        patients[_PatientAddress].National_Addr = _National_Addr;
        patients[_PatientAddress].National_id = _National_id;
        patients[_PatientAddress].age = _age;
        patients[_PatientAddress].phone = _phone;
        patients[_PatientAddress].Blood_Type = _Blood_Type;
        patients[_PatientAddress].marital_status = _marital_status;
        patients[_PatientAddress].gender = _gender;
        patients[_PatientAddress].hospital_addr = msg.sender;

        patients_keys.push(_PatientAddress);
        return true;
    }
```

Add Record by doctor

```
function addRecord(
        string memory _category,
        string memory _patient_name,
        string memory _rec_name,
        string memory _Created_at,
        address _patientAddr,
        address _doctor_addr,
        string memory _hex_ipfs,
        string memory _notes
    ) public onlydoctors returns (bool) {
        require(
            hasRole(PATIENT_ROLE, _patientAddr),
            "This patients is not exisit"
        );
        require(
            isAuth[_patientAddr][msg.sender],
            "No permission to add Records"
        );
        require(_doctor_addr == msg.sender, "No permission to this Doctor");
        Record_index = Record_index + 1;
        patients[_patientAddr].Records.push(
            record(
                Record_index,
                _category,
                _patient_name,
                _rec_name,
                _Created_at,
                _patientAddr,
                _doctor_addr,
                _hex_ipfs,
                _notes
            )
        );
        return true;
    }
```

Send request function that used by authorized doctor in system

```
function send_request_Access(
    address _patient,
    address _doctor
) public returns (bool success) {
    require(
        hasRole(PATIENT_ROLE, _patient),
        "The person you are requesting access to is not a Patient"
    );
    require(
        !checkRequestExists(_doctor, _patient),
        "already send a request"
    );
    Request_index = Request_index + 1;
    patient memory pati = get_patient_by_address(_patient);
    doctor memory doc = get_doctor_by_address(_doctor);
    requests.push(
        request(
            Request_index,
            _doctor,
            _patient,
            block.timestamp,
            pati.name,
            doc.name
        )
    );
    emit RequestAccess(msg.sender, _patient);

    return true;
}
```

Accept request function by patient

```

function approveAccess(address _doctor) public onlypatient {
    require(hasRole(DOCTOR_ROLE, _doctor), "this acount is not a doctor ");
    require(
        (isAuth[msg.sender][_doctor]) == false,
        "This Request alraedy been Approved!! "
    );
    isAuth[msg.sender][_doctor] = true;
    emit AccessGranted(_doctor, msg.sender);
}
```

reject request function.

```
function rejectAccess(address _doctor) public onlypatient {
    require(
        hasRole(DOCTOR_ROLE, _doctor),
        "Doctor does not have the doctor role"
    );
    //   require((isAuth[msg.sender][_doctor]) == true, "This Request alraedy been Reject!! ");
    isAuth[msg.sender][_doctor] = false;
    DeleteRequest(_doctor, msg.sender);
    emit AccessRejected(_doctor, msg.sender);
}
```

Functions to get all data about (Patients , doctors ,hospital , record)

```

  function get_all_Patients() public view returns (patient[] memory) {
      patient[] memory pat = new patient[](patients_keys.length);
      for (uint256 i = 0; i < patients_keys.length; i++) {
          pat[i] = patients[patients_keys[i]];
      }
      return pat;
  }

function get_all_Doctors() public view returns (doctor[] memory) {
      doctor[] memory doc = new doctor[](doctors_keys.length);
      for (uint256 i = 0; i < doctors_keys.length; i++) {
          doc[i] = doctors[doctors_keys[i]];
      }
      return doc;
  }

      function get_all_hospitals() public view returns (Hospital[] memory) {
      Hospital[] memory hos = new Hospital[](Hospitals_keys.length);
      for (uint256 i = 0; i < Hospitals_keys.length; i++) {
          hos[i] = Hospitals[Hospitals_keys[i]];
      }
      return hos;
  }


  function See_Record() public view returns (record[] memory) {
      require(
          patients[msg.sender].Records.length > 0,
          "patient record doesn't exist"
      );
      return (patients[msg.sender].Records);
  }

```

4.2 Interface

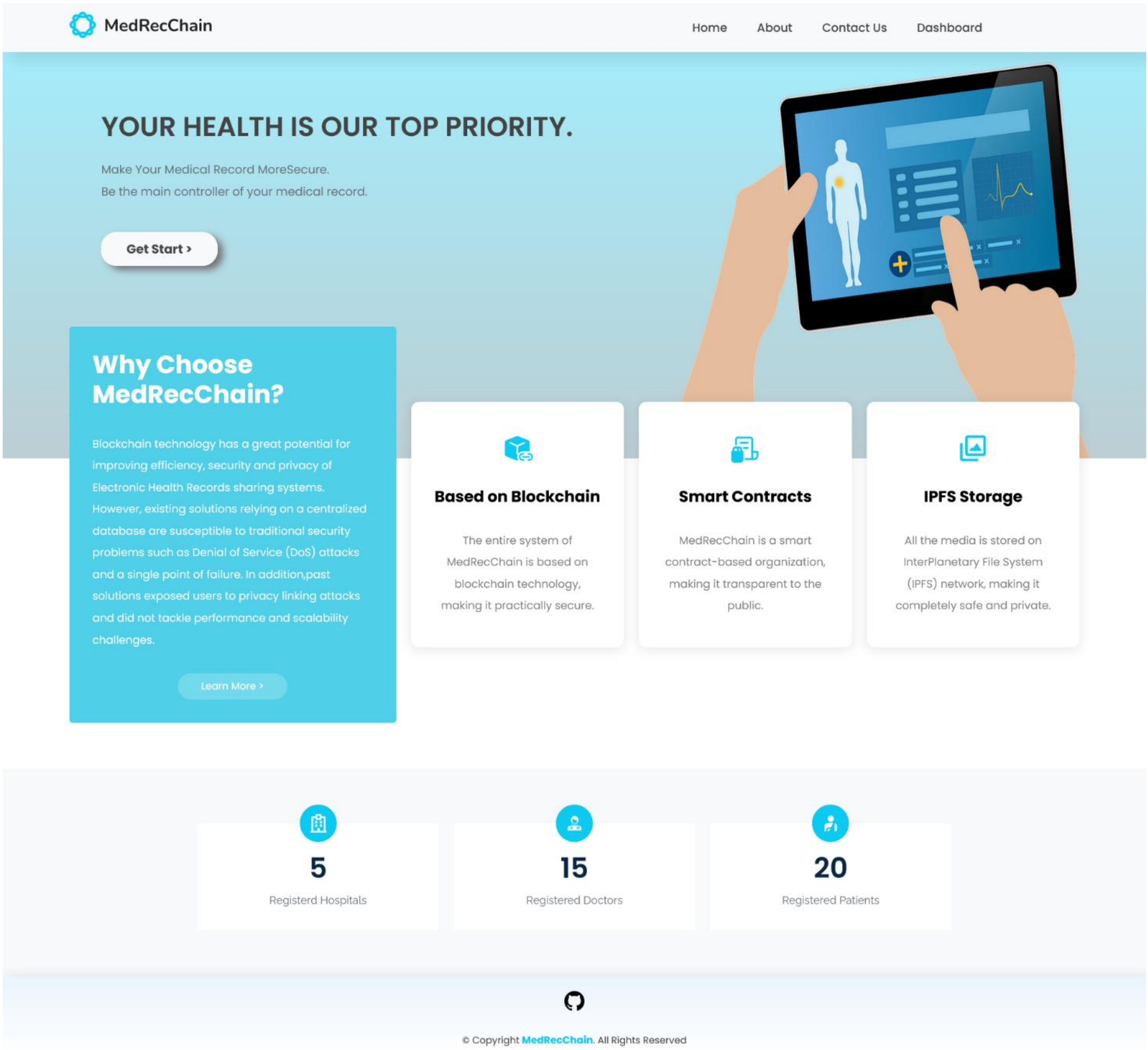


Figure 0-2Home page

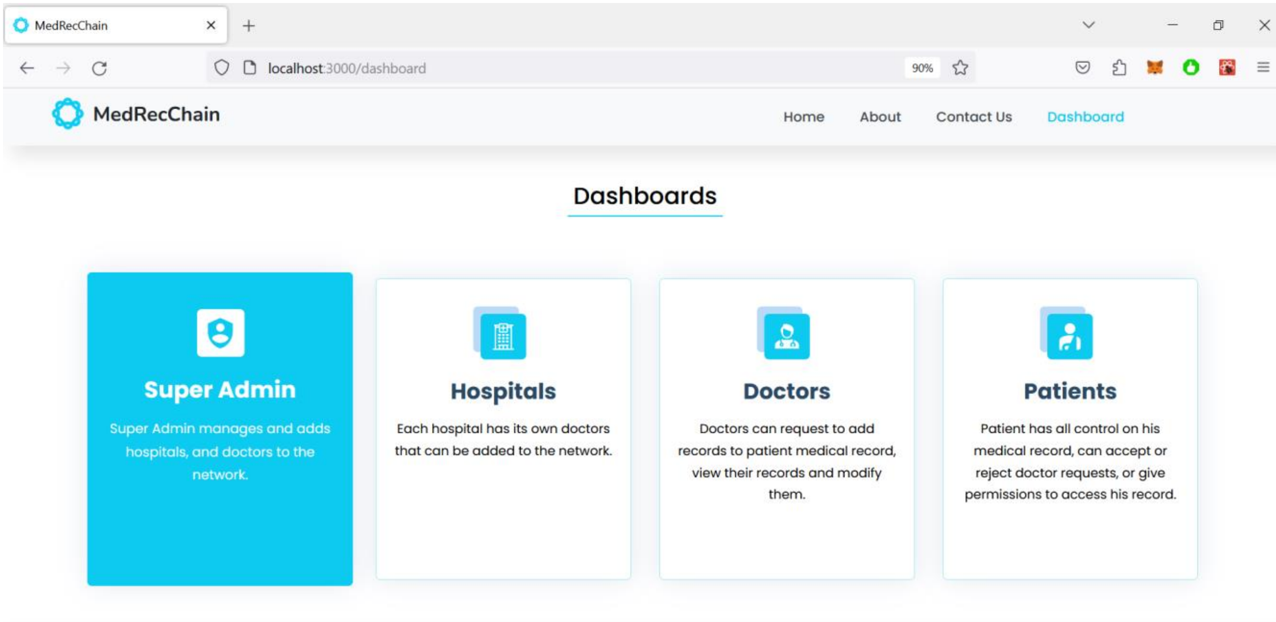


Figure 0-3 main Dashboard

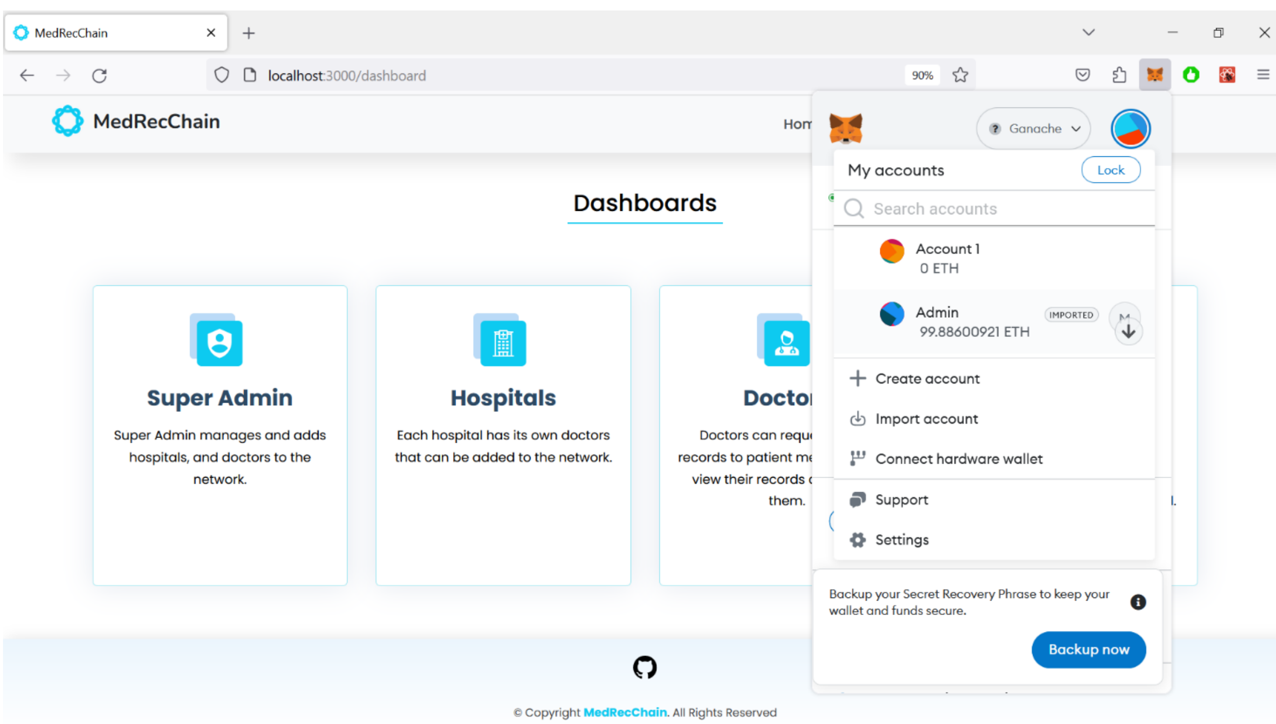


Figure 0-4connect to Meta mask

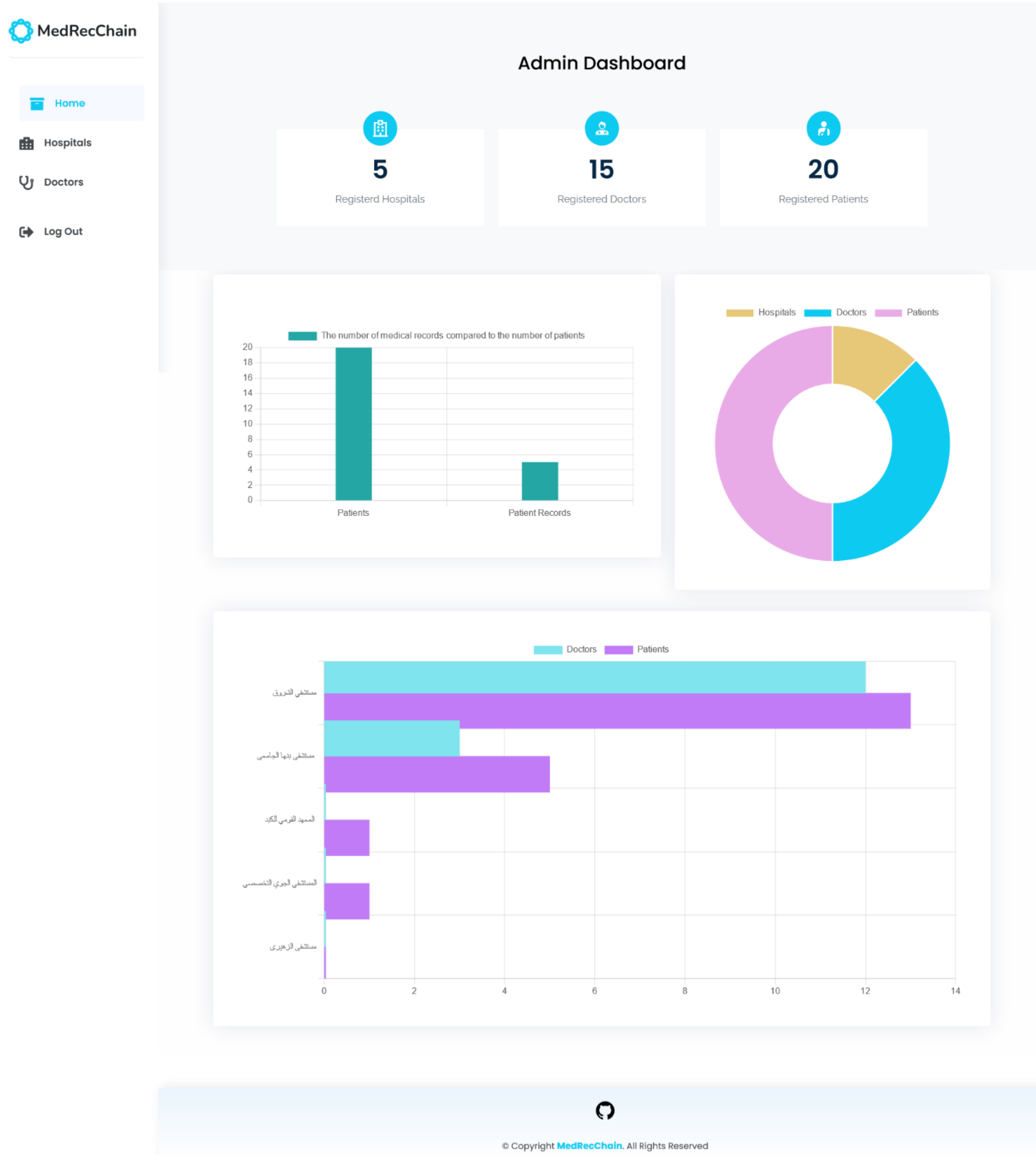


Figure 0-5admin Dashboard

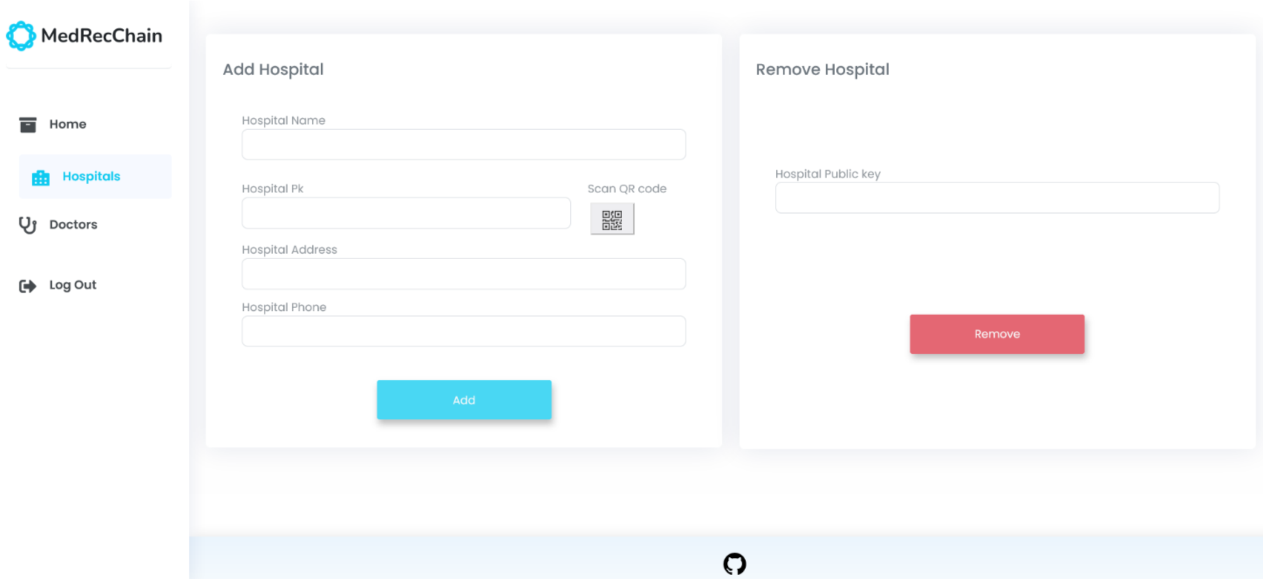


Figure 0-6 Add hospital by admin

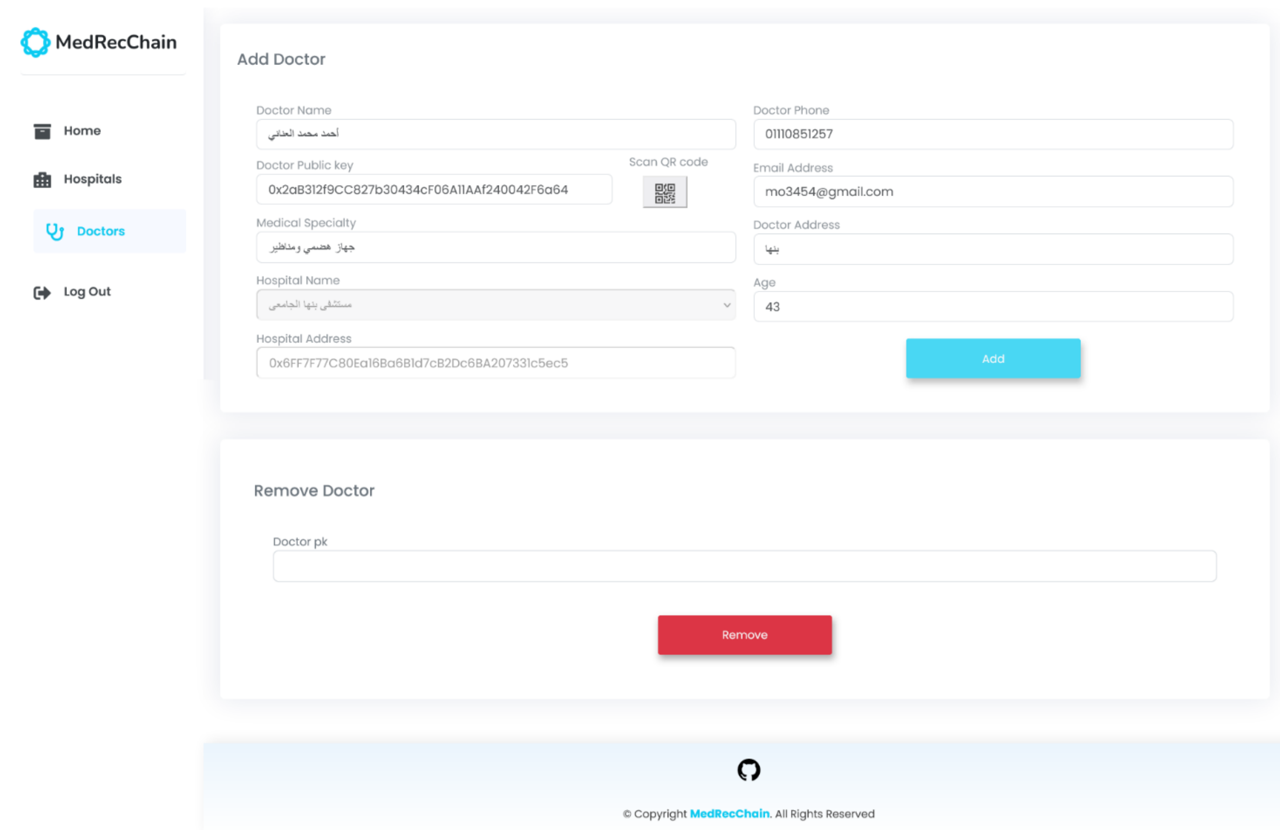
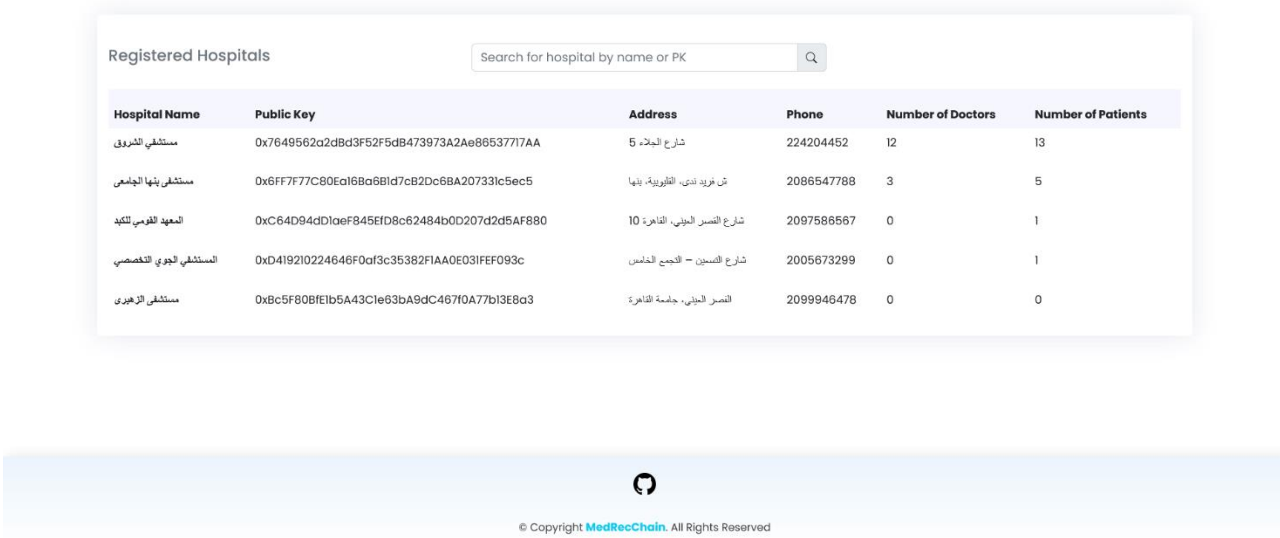


Figure 0-7 Add Doctors by admin



Registered Hospitals

Hospital Name	Public Key	Address	Phone	Number of Doctors	Number of Patients
مستشفي الشروق	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA	5 شارع الجلاء	224204452	12	13
مستشفى بنها الجامعى	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5	ش فريد ندى، القليوبية، بنها	2086547788	3	5
المعهد القومي للكبد	0xC64D94dD1aeF845EfD8c62484b0D207d2d5AF880	10 شارع القصر العيني، القاهرة	2097586567	0	1
المستشفي الجوي التخصصي	0xD419210224646F0af3c35382F1AA0E031FEF093c	شارع التسعين – التجمع الخامس	2005673299	0	1
مستشفى الزهيرى	0xBc5F80BfE1b5A43C1e63bA9dC467f0A77b13E8a3	القصر العيني، جامعة القاهرة	2099946478	0	0

Figure 0-8 show All hospital in system

Registered Doctors

Search for doctor

Doctor Name	Doctor PK	Medical Specialty	Hospital PK
أحمد محمد العناني	0xA1Caa1D43EAb9aA8C8cdFB94486f8b4c76ec99bb	أنف وأذن وحنجرة	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
محمد عادل	0xC36339c6B2eB1755569Ae3C4ac3a76d0b638b7DC	جراحة مخ واعصاب	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
احمد القدوسي	0x9840206CB8bc9a34961CDb05aDDEdB9Bd2215429	أسنان	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
سعيد الأمام	0xdF4fFB18705B85E87e55d12160a071C233C41314	جراحة مخ واعصاب	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
أحمد صلاح	0x71D21FcE596EEF0c161930e322a3063aC7beC754	أشعة وتحاليل	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
ابراهيم عبد العظيم	0x6D4271331d37A30fb781531485253f768ead656A	اطفال وحديثي ولادة	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
عبد الله سعيد	0x6f45597635746bEa411e5F958f29f520Cf73983B	باطنة	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
عبد الرحمن اسامة	0x9f30e3D82E7E61B11Ea179175639Af80c40C34ac	قلب واوعية دموية	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
محمود عبد المنعم	0xc18Bb8a381Ad560685A214D4a002c49F2E025210	صدر وجهاز تنفسي	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
محمد الحسيني	0x2E22E04d83fF5A290d669dB1dD007b1A1fe8A620	جهاز هضمي ومناظير	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
اسراء	0x8562B409fAD3502116C4F08D26963f451C20Ea44	جلدية	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
حازم عمران	0xe51BAE06244B74A660154Af40074EB4599DAAC5C	أسنان	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
علي اسماعيل	0xF8A274c9BCa06156Fe82923a7Bba9E01479cE2Cf	جهاز هضمي ومناظير	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
حسن ابو العنين	0x70cdc7773cA202183DaBB7CfbA19c9C00A239505	اطفال وحديثي ولادة	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
عبد الرحمن عصام	0x2aB312f9CC827b30434cF06A11AAf240042F6a64	أسنان	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA

Figure 0-9 show all Doctors

Registered Patients

Search for patient by name or PK

Patient Name	Public Key	Hospital Address
محمود عبدالله ابراهيم	0xB24a7a4C6C38e436Ea9A140d35A331c9946e8f20	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
احمد حسام ماهر	0xA9E3a023720051eb801668a7583D493168e035Bd	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
سارة السيد هاني	0x46C4D7996ddc70ba83E2Cc5B8bfE20D896A88Ae5	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
وفاء جمال منصور	0x74311AAc8Fe5e0251c3f5a9e02FC6D018a66b679	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
محمد عماد السيد	0xe8cF332CBF902F7dd9C3d89034538886C7a58fbD	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
حبيبة مصطفي احمد	0xeeB43A3E8f24E3E3399100cF1Cb997C8cdC94265	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
محمود طلعت احمد	0x048c66eD1835FcEb63F0330B729d963b2404797F	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
عمر مجدي عبد الحكيم	0xCf9115Dd5c3b74a07a7519Aff605C2F79F846458	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
شروق مؤمن اسماعيل	0xC57C26B44D07E1064182B5ED16d04A5742e737E1	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
سليمان محمد الامام	0x7E146c54d28658A970f09c7D6ec39bb7c1C86832	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
محمود السيد مصطفي	0xB68f9826A24A06Dd4b319461Cf6F71fDAC2174D5	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
شيماء ابراهيم سعيد	0xa72393c77a452b21C12F12f28070A95218Ec63FB	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
سهيلة عبد الله مرعي	0x4e3222D8dd7e2Dc2eBA11EF787337e37F453CDF8	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
محمد فؤاد صبري	0x2f28C5256856F183BcBbC02DcC2B0e8E6eDA7D62	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
مروان عبد العزيزسالم	0x1A1e43d9A27a51870c656ba91E0140C1A4563e59	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
ياسمين محمد عبد العليم	0xCa53B655505A78c70cC636FE0b46187B8A0A32bb	0x6FF7F77C80Ea16Ba6B1d7cB2Dc6BA207331c5ec5
اسماء عبد الله مسعود	0x2A1697494FdAD3521c4548D01130EC25C7dAF54a	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
مصطفي علي احمد	0x22a6315De362cdCc5A95c742Ef7e3A5598CDDaD4	0x7649562a2dBd3F52F5dB473973A2Ae86537717AA
محمد علي ابراهيم	0x0474FA4B0772223CDa09Fd874eb2F65090628fD6	0xD419210224646F0af3c35382F1AA0E031FEF093c
صبري محمد سعد	0x47d21d615899A2a4809DcC76CB7F8af295a0445D	0xC64D94dD1aeF845EfD8c62484b0D207d2d5AF880

Figure 0-10 show   all Patients

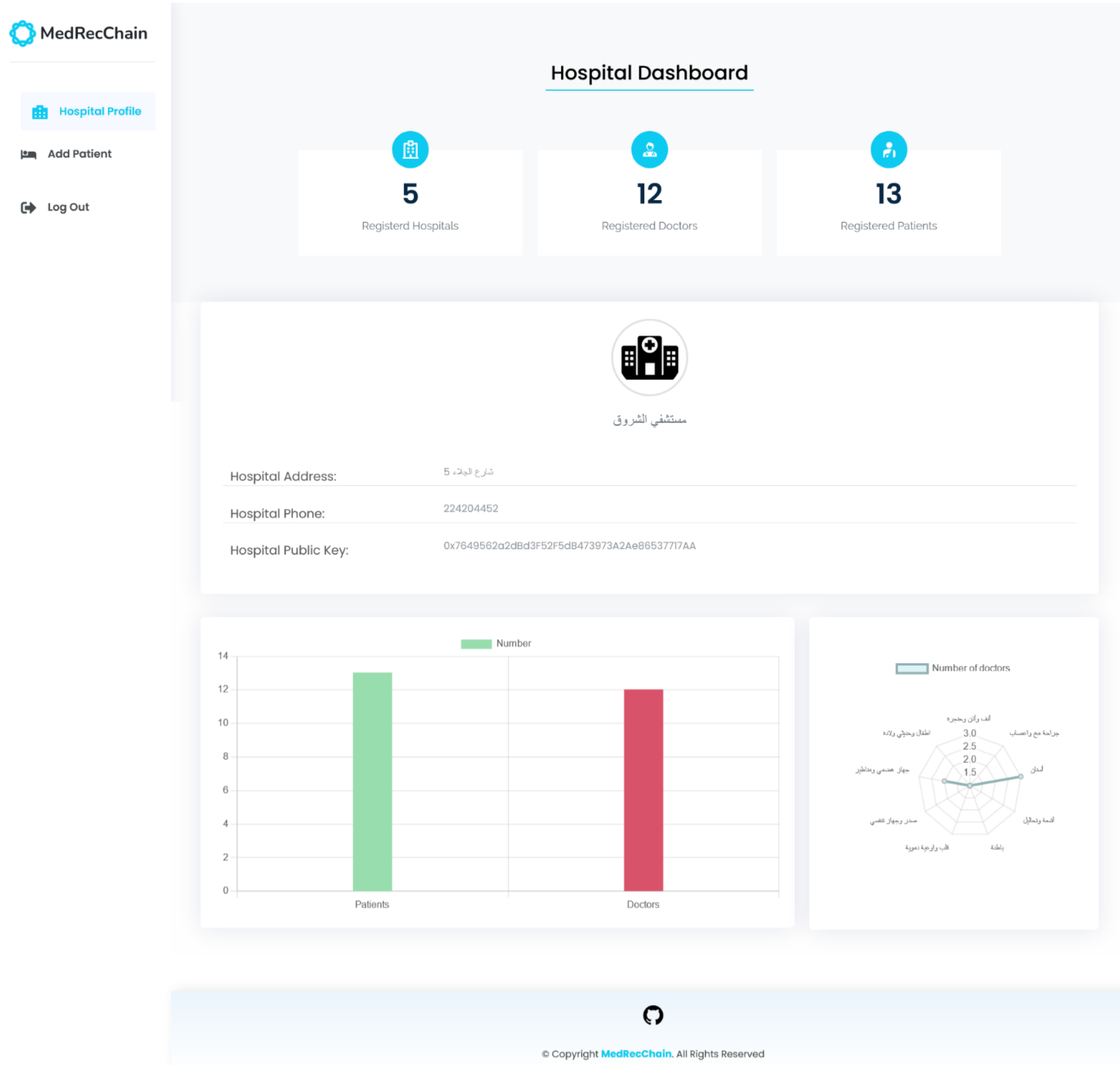


Figure 0-11 Hospital Dashboard

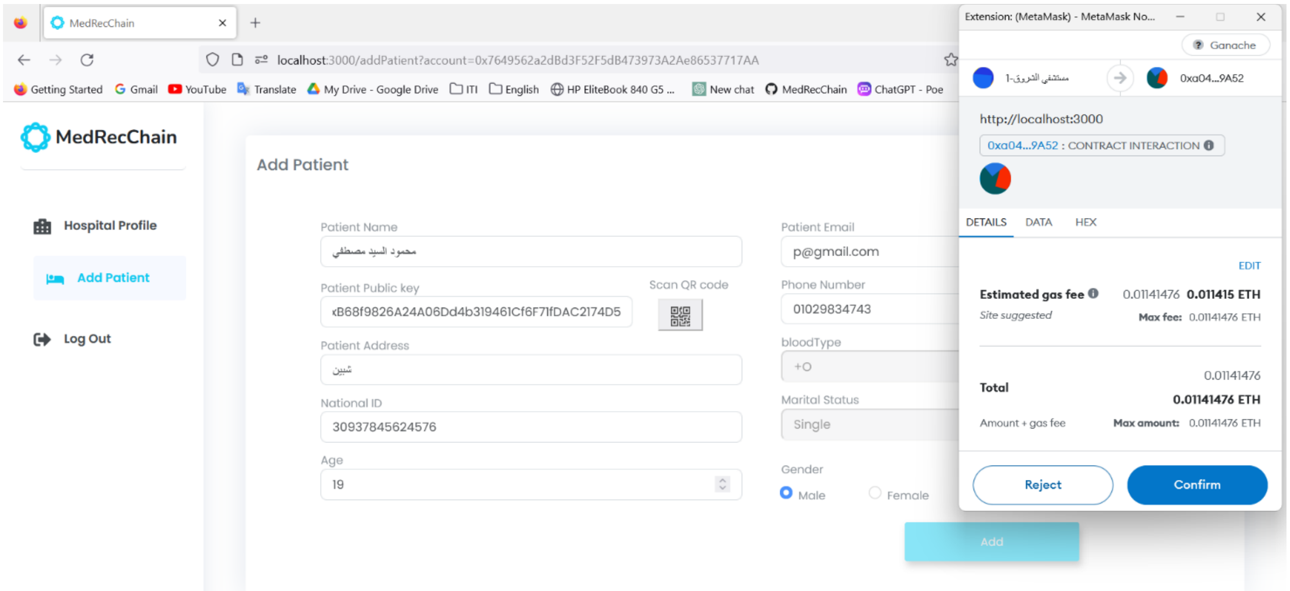


Figure 0-12Add patient.

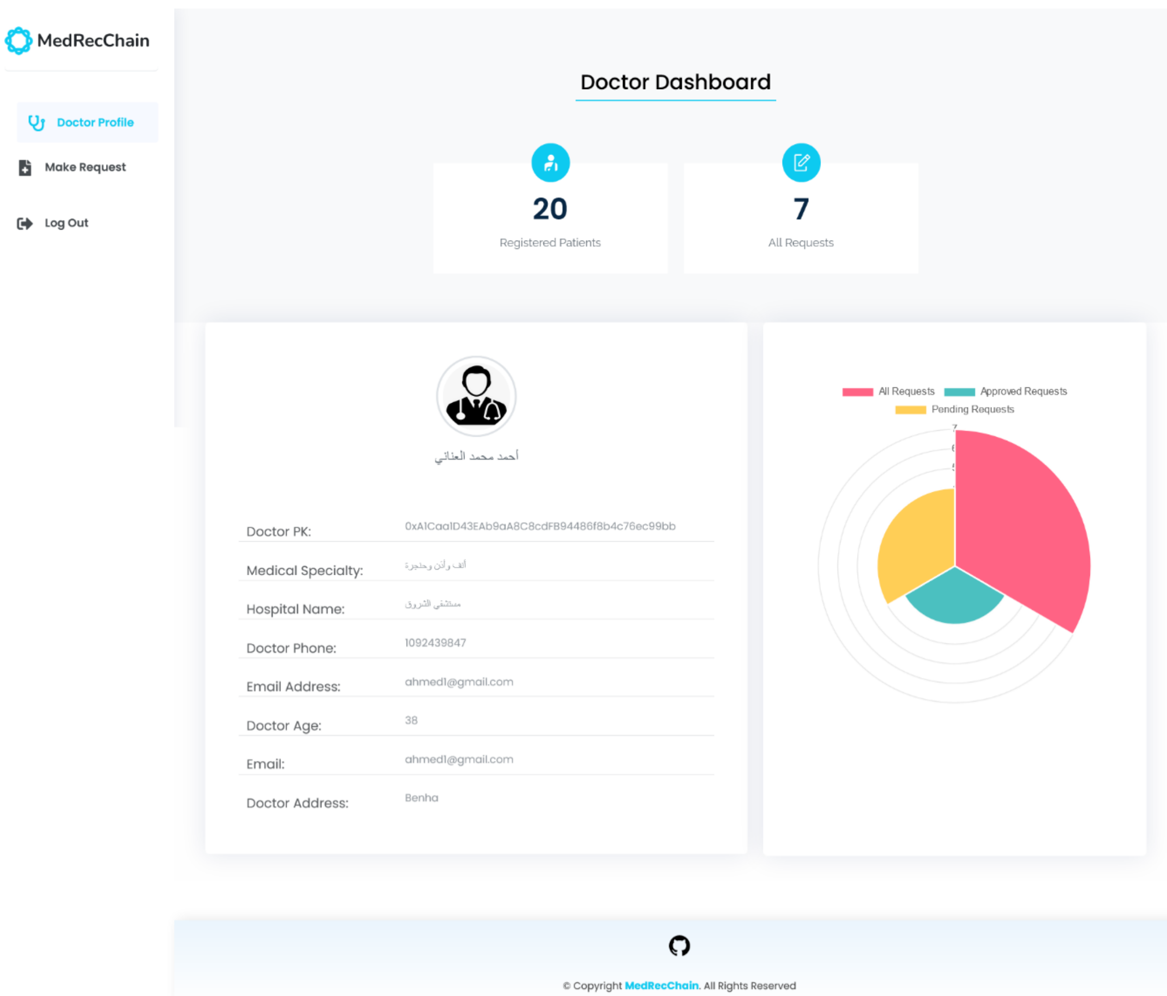


Figure 0-13 doctor Profile

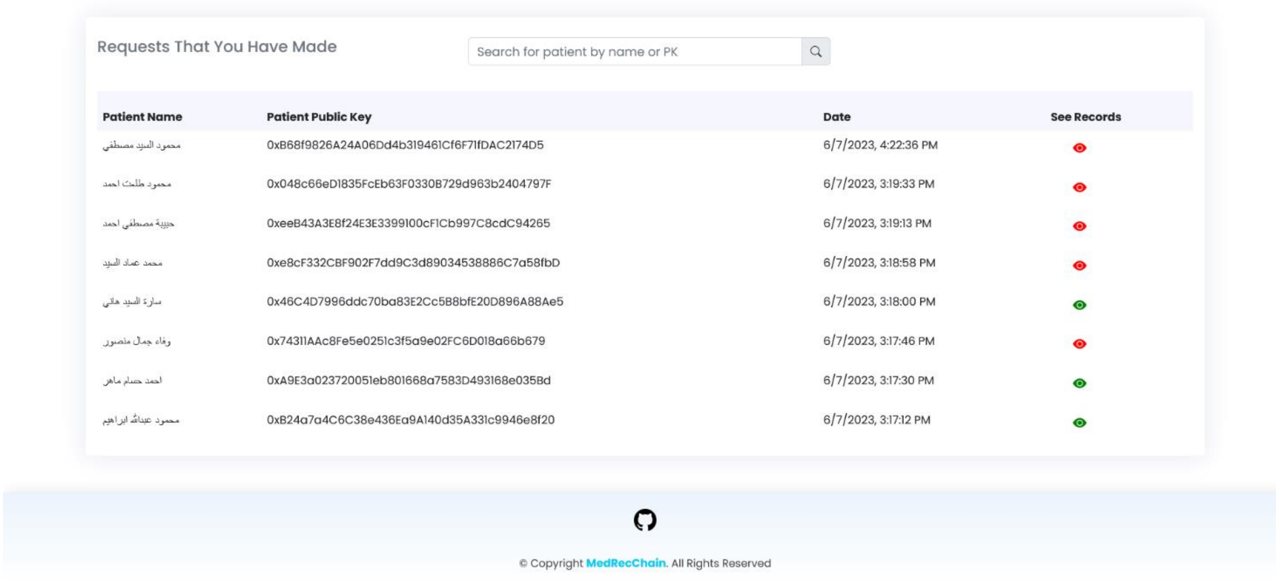


Figure 0-14 doctor All Requests

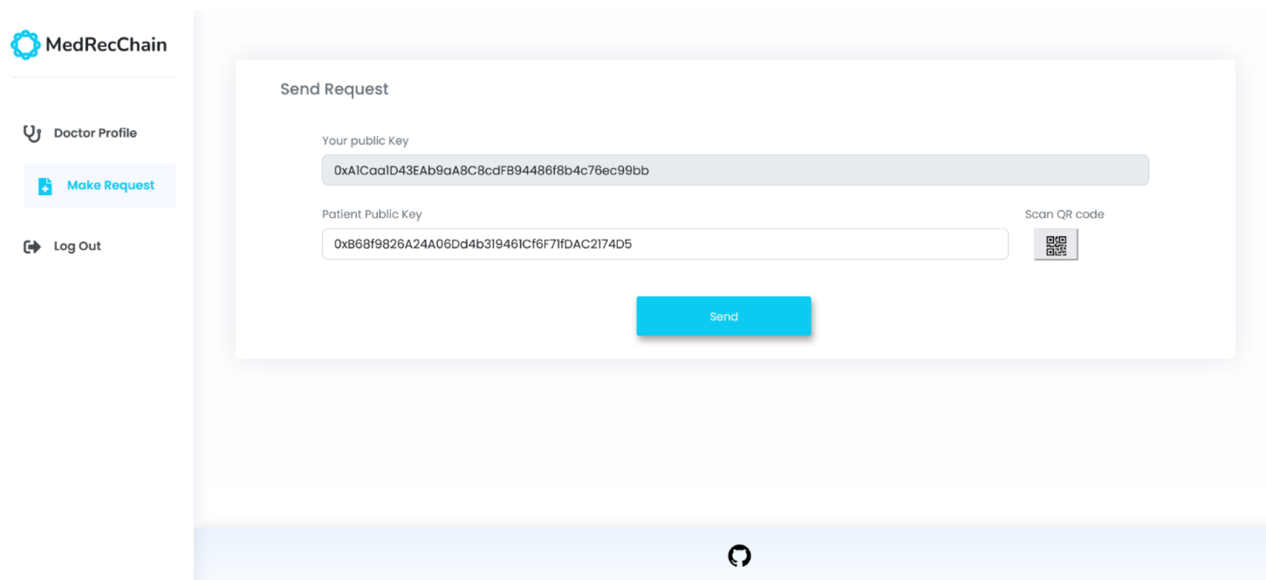


Figure 0-15 doctor send Requests

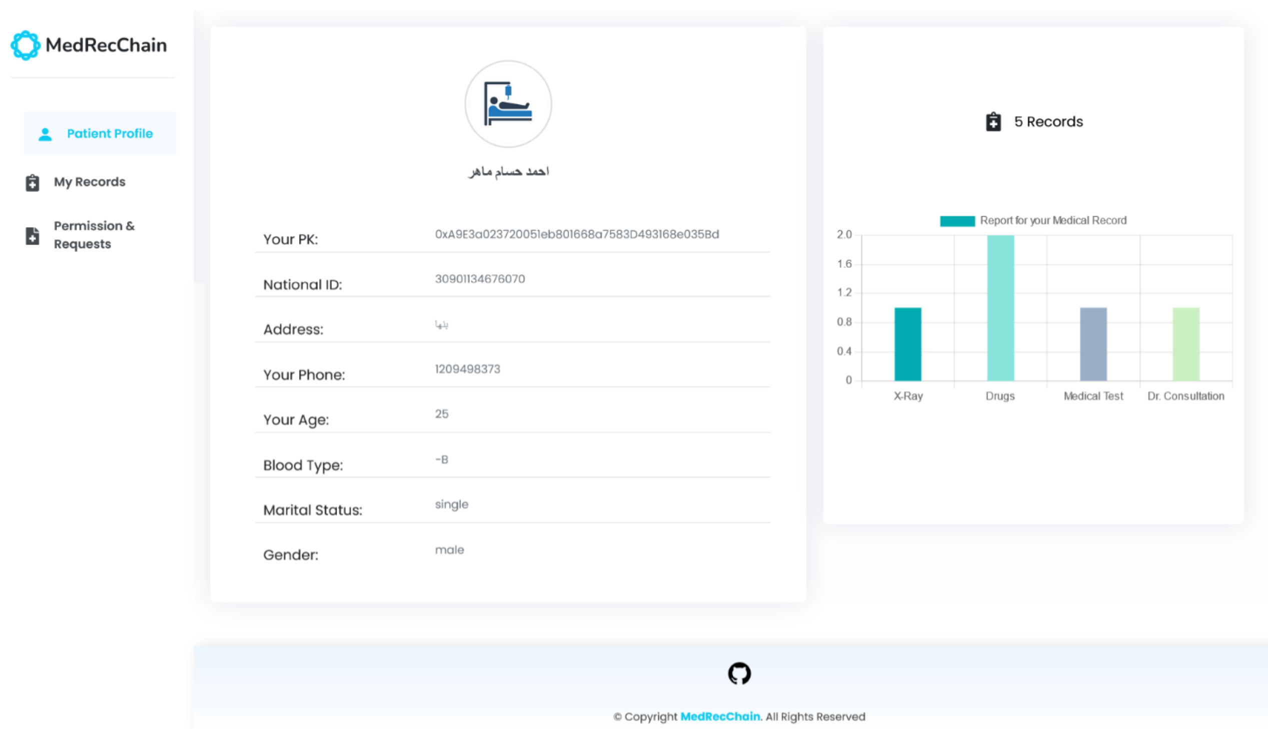


Figure 0-16 Patient profile

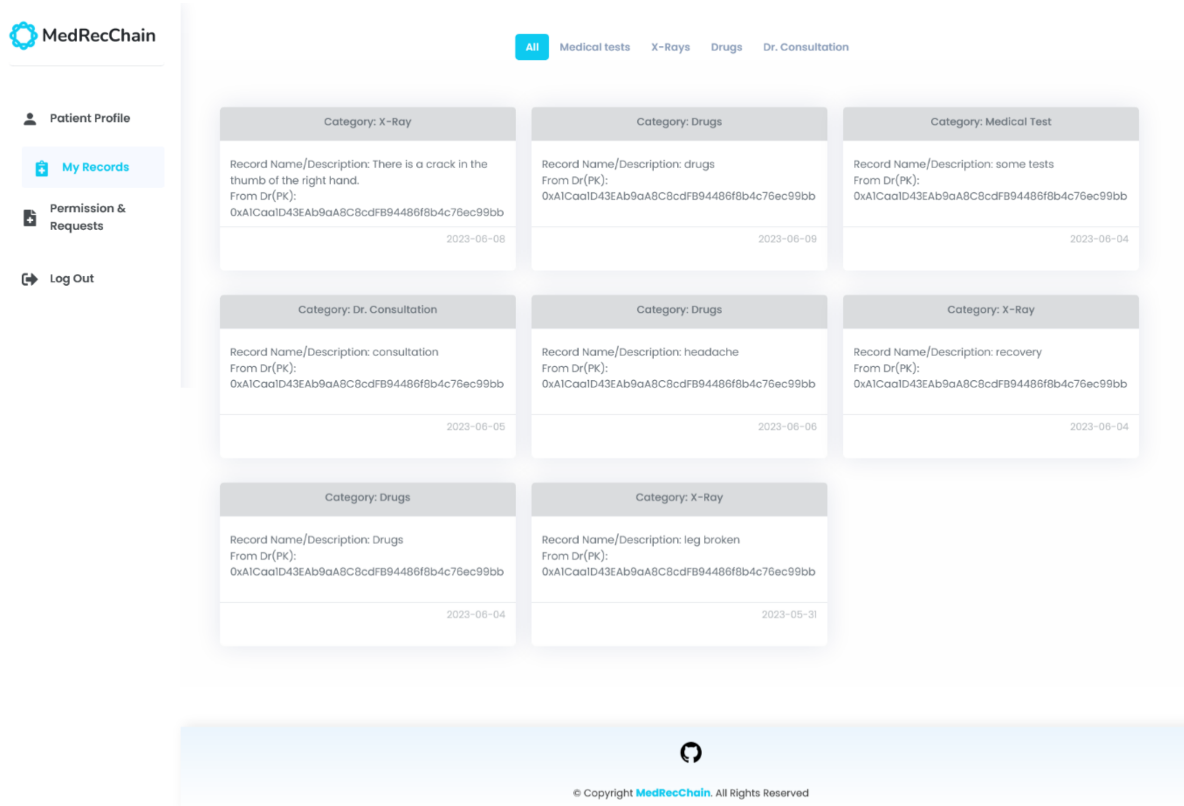


Figure 0-17 patient see his records

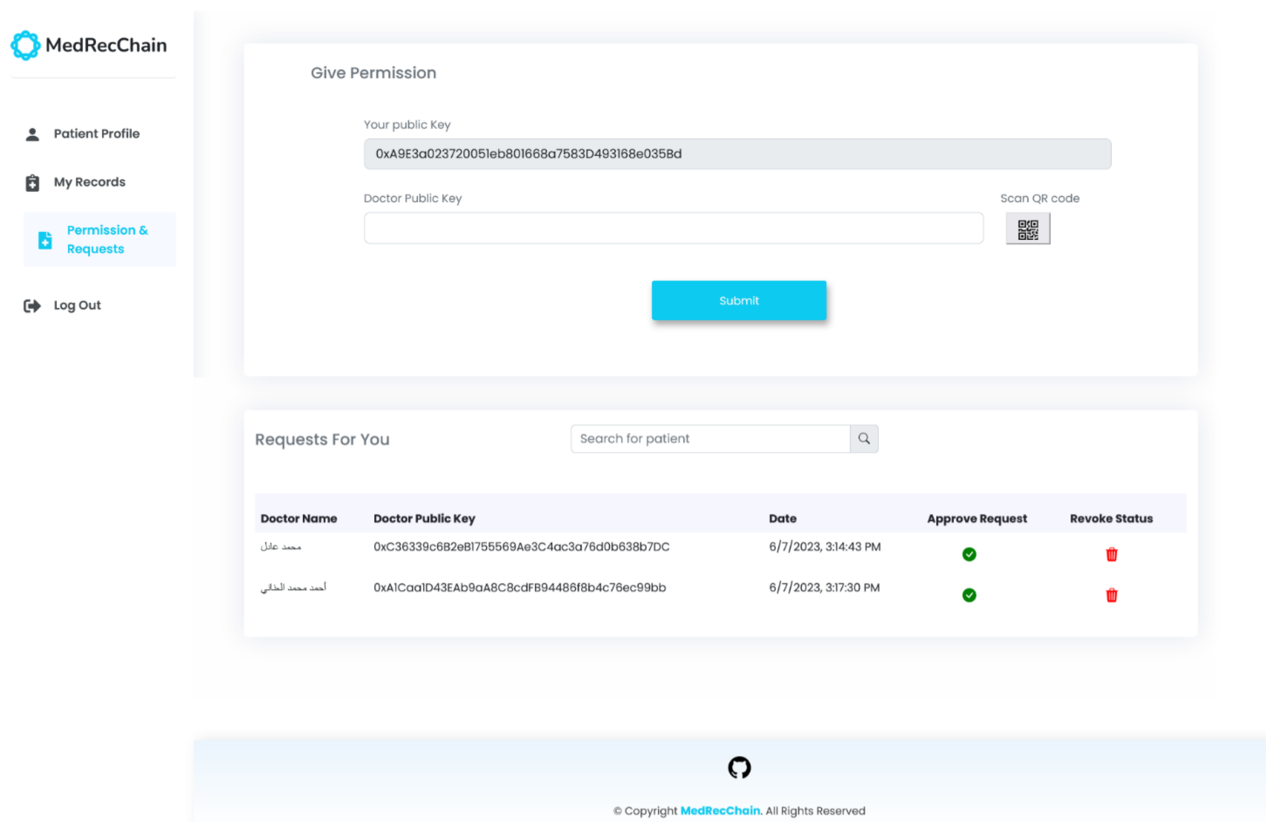


Figure 0-18 patient send /accept/reject permission

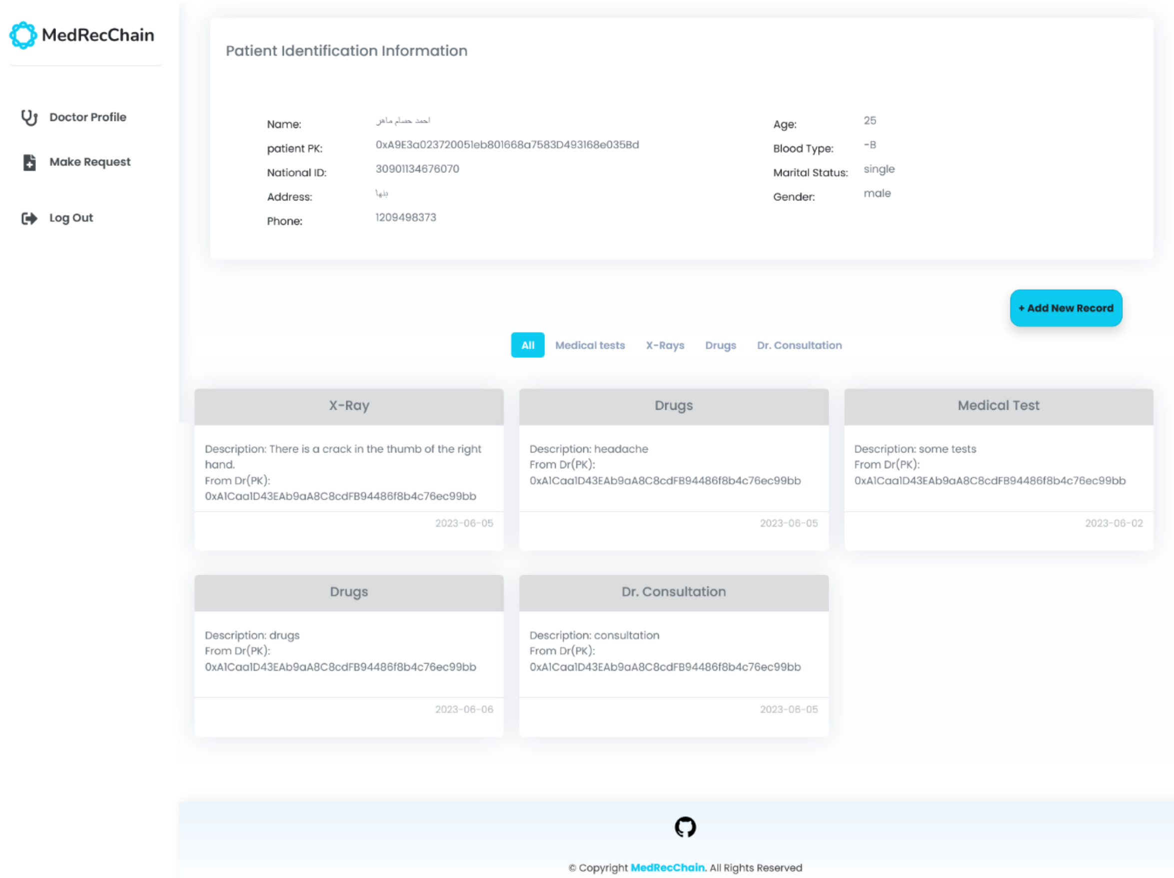


Figure 4.2-17 doctor preview record

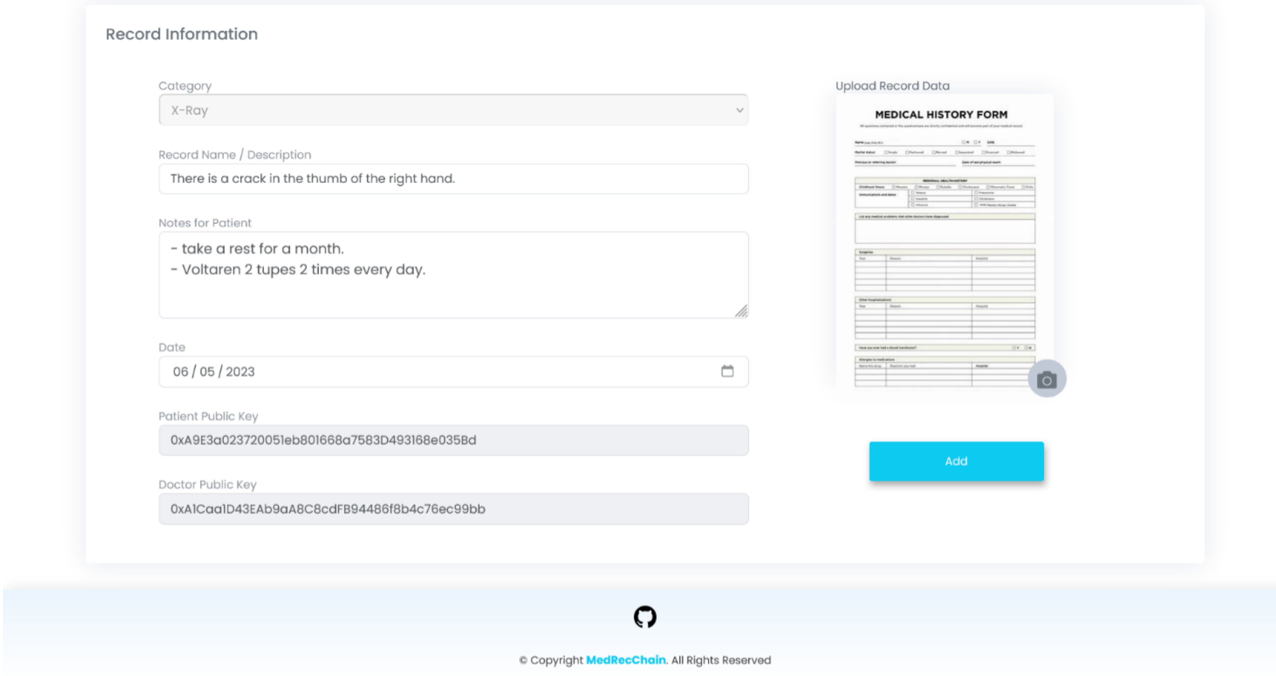


Figure 4.2-18 doctor adds record.

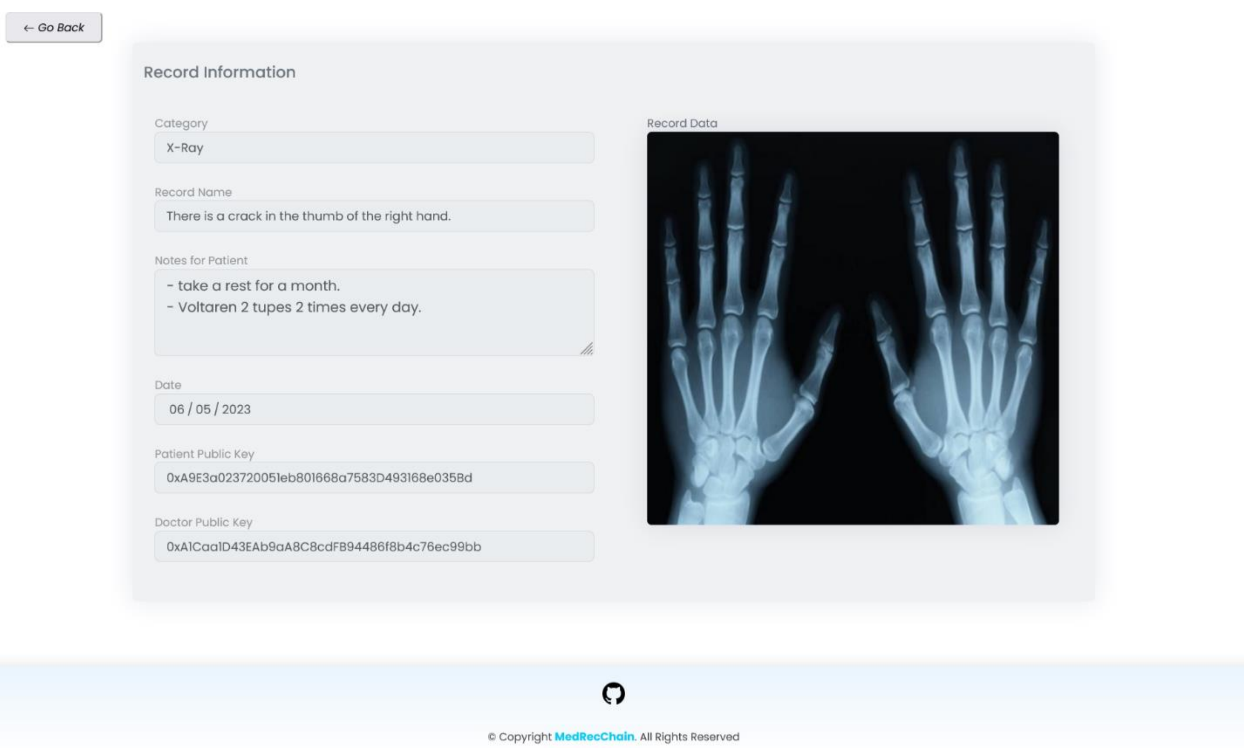


Figure 4.2-19 doctor preview record

Chapter 5: Future work and Conclusion

Future Work

There are several areas of future work that can further enhance MedRecChain and its decentralized solution for Electronic Medical Records (EMR) sharing systems. Here are some potential avenues for future development:

1. Interoperability with other blockchain platforms: While MedRecChain currently utilizes the Ethereum network, exploring interoperability with other blockchain platforms could enhance its reach and connectivity with different healthcare systems. Integration with other blockchain networks, such as Hyperledger or Corda, could enable seamless data exchange and collaboration across diverse platforms.

2. Integration with emerging technologies: As technology continues to evolve, incorporating emerging technologies like artificial intelligence (AI) and machine learning (ML) into MedRecChain can unlock new possibilities. AI and ML algorithms can be employed to analyze medical records, identify patterns, and provide valuable insights for healthcare providers, ultimately improving patient care and treatment outcomes.

3. Enhancing privacy and data protection: While blockchain technology inherently provides security and immutability, ongoing efforts to enhance privacy and data protection are crucial. Exploring privacy-preserving techniques like zero-knowledge proofs or differential privacy can further safeguard sensitive medical information while maintaining the benefits of a decentralized system.

4. Scalability and performance optimization: As the usage of MedRecChain expands, addressing scalability challenges will be important. Exploring solutions like sharding, off-chain storage, or layer-2 scaling techniques can enhance system performance and accommodate a larger volume of medical records and users.

5. User experience and interface design: Continued focus on user experience and interface design will ensure that MedRecChain remains accessible and intuitive for both technical and non-technical users. Incorporating user feedback, conducting usability studies, and employing modern design principles can further enhance the platform's usability and adoption.

Conclusion:

In conclusion, MedRecChain represents a decentralized solution for Electronic Medical Records (EMR) sharing systems, utilizing the Ethereum network and IPFS technology. By leveraging the transparency, security, and immutability of blockchain and the distributed storage capabilities of IPFS, MedRecChain offers a promising approach to address the challenges associated with centralized EMR systems.

The combination of the Ethereum network and IPFS technology empowers patients, healthcare providers, and relevant parties to securely store, access, and share electronic medical records. It promotes data integrity, privacy, and control, while facilitating efficient interoperability and collaboration among healthcare systems.

Looking ahead, future work can focus on areas such as interoperability with other blockchain platforms, integration with emerging technologies, enhancing privacy and data protection, addressing scalability and performance challenges, and refining the user experience and interface design.

Overall, Mordechai's decentralized solution has the potential to revolutionize the management and sharing of electronic medical records, improving efficiency, security, and accessibility in the healthcare industry. As advancements are made and new opportunities arise, MedRecChain can continue to evolve and contribute to the advancement of healthcare systems worldwide.

Reference

[1]- Jill C. Muhrer, MSN, FNP-C The importance of the history and physical in diagnosis | CE Article | NursingCenter, 2014

[2]- Vanessa McMains, Study Suggests Medical Errors Now Third Leading Cause of Death in the U.S., May 3, 2016

[3]- Summerton N. The medical history as diagnostic technology. Br J Gen Pract. 2008 Apr;58(549):273-6. doi: 10.3399/bjgp08X279779. PMID: 18387230; PMCID: PMC2277113

[4]-https://www.hipaajournal.com/june-2022-healthcare-data-breach report/#:~:text=Over%20the%20past%2012%20months,of%2057.67%20breaches%20a%20month.

[5]- Haber, S., Stornetta, W.S. How to time-stamp a digital document. J. Cryptology 3, 99–111 (1991). https://doi.org/10.1007/BF00196791

[6]- Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008

[7]- N. Nchinda, A. Cameron, K. Retzepi and A. Lippman, "MedRec: A Network for Personal Information Distribution," 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 2019, pp. 637-641, doi: 10.1109/ICCNC.2019.8685631.

[8]- Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du and M. Guizani, "MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain," in IEEE Access, vol. 5, pp. 14757-14767, 2017, doi: 10.1109/ACCESS.2017.2730843.

[9]- https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf

[10]- J. Huang, Y. W. Qi, M. R. Asghar, A. Meads, and Y. Tu, “MedBloc: A Blockchain-Based Secure EHR System for Sharing and Accessing Medical Data,” in 2019 18th IEEE International

[11]- Edward H. Shortliffe, James J. Cimino, Biomedical Informatics (Computer Applications in Health Care and Biomedicine), 2006.

[12]- Martin Valenta, Philipp Sandner, Comparison of Ethereum, Hyperledger Fabric and Corda, FSBC Working Paper, 2017

[13]- https://docs.ipfs.tech/concepts/what-is-ipfs/#decentralization

[15]- Fries, Martin; P. Paal, Boris (2019). Smart Contracts (in German). Mohr Siebeck.

[16]- https://101blockchains.com/introduction-to-blockchain-features/

[17]- https://perfectial.com/blog/leveraging-private-blockchains/

[18]- A. Shahnaz, U. Qamar and A. Khalid, "Using Blockchain for Electronic Health Records," in IEEE Access, vol. 7, pp. 147782-147795, 2019, doi: 10.1109/ACCESS.2019.2946373