**MEDICHAIN – A USER CONTROLLED EHEALTH SHARING SYSTEM**

**PROJECT DESIGN REPORT**

Submitted by

**ABEL T VARGHESE**

**PTA19CS001**

**AJU SEBASTIAN**

**PTA19CS004**

**ALAN FRANCIS JOSE**

**PTA19CS006**

**SONA CHACKO**

**PTA19CS048**

**TINSU MARIYAM THOMAS**

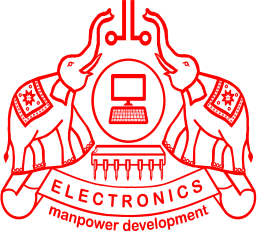
**PTA19CS052**

To

*The APJ Abdul Kalam Technological University*

*in partial fulfilment of the requirements for the award of the*

*Degree of Bachelor of Technology in Computer Science and Engineering*



**Department of Computer Science & Engineering**

College of Engineering Kallooppara

Pathanamthitta

December 2022

**DECLARATION**

We hereby declare that the project report “**MEDICHAIN**”, submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Mrs. Anitha Jose . This submission represents our ideas in our own words and where ideas or words of others have been included; we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

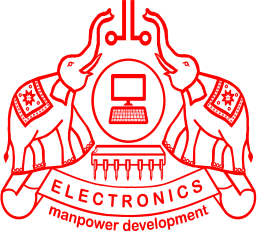
Place : Kallooppara

Date :27 December,2022

**DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING**

**COLLEGE OF ENGINEERING KALLOOPPARA**

**PATHANAMTHITTA- 689603**



**CERTIFICATE**

This is to certify that the project design report entitled title submitted by Abel T Varghese(PTA19CS001), Aju Sebastian(PTA19CS004), Alan Francis Jose(PTA19CS006), Sona Chacko(PTA19CS048) and Tinsu Mariyam Thomas(PTA19CS052) in partial fulfillment with the requirements of the award of the Degree of Bachelor of Technology in Computer Science and Engineering of APJ Abdul Kalam Technological University is a bonafide work carried out by him/her under our supervision.

**Coordinator**  **Guide Head of Department**

**Ms. Leeba Merin Sam Mrs. Anitha Jose** **Dr. Renu George**

Assistant Professor, Assistant Professor, Assistant Professor,

Department of Department of Department of

Computer Science & Computer Science & Computer Science &

Engineering Engineering Engineering

**ABSTRACT**

This project tackle the problem of sharing eHealth data across different jurisdictions. As a general rule, and due to the sensitive nature of the information, different national regulations impose severe limits on what can be exchanged, even in case of emergencies. Furthermore, different systems in different jurisdictions do not communicate. We propose **Medichain** as a scheme that allows eHealth data to be securely exchanged, with the data subject always in the position of mediation. We combine several technologies namely Blockchain, OAuth and User-Managed Access, and concept Receipts, to achieve our aim.

**ACKNOWLEDGEMENT**

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to Dr. Nisha Kuruvilla ,The Principal, Dr. Renu George ,The Head of the Department, our esteemed guides Mrs. Anitha Jose and Ms. Leeba Merin Sam of the Department of Computer Science and Engineering, College of Engineering, Kallooppara, for providing us with the right guidance and advice at the crucial junctures and for showing us the right way. We would like to thank the other faculty members also, at this occasion. Last but not leaving the one, we would like to thank our parents for their motivation and our friends who gave us with valuable contributions regarding our topic and the encouragement throughout the course of work.

Abel T Varghese

Aju Sebastian

Alan Francis Jose

Sona Chacko

Tinsu Mariyam Thomas

**CONTENTS**

ACKNOWLEDGEMENT………………………………………………………………………...4

ABSTRACT……………………………………………………………………………………….5

LIST OF FIGURES………………………………………………………………………………..7

ABBREVATIONS………………………………………………………………………………...8

Chapter 1. INTRODUCTION……………………………………………………………………10

1.1 General Background………………………………………………………………10

1.2 Objective………………………………………………………………………….10

1.3 Purpose and Scope………………………………………………………………...11

Chapter 2. LITERATURE SURVEY…………………………………………………………….12

2.1 Introduction……………………………………………………………………….12

2.2 EHealth Data Exchange…………………………………………………………...12

2.2.1 Blockchain in Data Exchange………………………………………………14

2.2.2 OAuth ……………………………………………….……………………..15

2.2.3 User Managed Access…..………………………………………………….16

2.2.4 Concept Receipts…………………………………………………………...18

Chapter 3. PROBLEM STATEMENT…………………………………………………………...19

Chapter 4. PROPOSED SYSTEM……………………………………………………………….21

4.1 Architecture……………………………………………………………………….21

4.2 System Requirement……………………………………………………………...23

4.2.1 Software Requirements…………………………………………………….23

4.2.1.1 Node JS…………………………………………………………..23

4.2.1.2 Web3.js…………………………………………………………..24

4.2.1.3 Ethereum PoS…………………………………………………….25

4.2.1.4 Infura……………………………………………………………..25

4.2.1.5 JavaScript………………………………………………………...26

4.2.1.6 HTML……………………………………………………………27

4.2.1.7 CSS………………………….……………………………………27

4.2.1.8 MetaMask………………………………………………………...28

Chapter 5. DESIGN DIAGRAM………………………………………………………………...29

5.1 Data Flow Diagram………………………………………………………………..29

5.1.1 Level 0……………………………………………………………………...29

5.1.2 Level 1……………………………………………………………………...30

5.2 Use Case Diagram…………………………………………………………………31

Chapter 6. CONCLUSION……………………………………………………………………….34

6.1 Conclusion and Future Scope……………………………………………………...34

REFERENCES………………………………………………………………………………….. 35

**LIST OF FIGURES**

4.1 Architecture…………………………………………………………………………………. 21

5.1 DFD Level 0………………………………………………………………………………… 30

5.2 DFD Level 1………………………………………………………………………………… 31

5.3 Use Case Diagram……………………………………………………………………………31

**ABBREVIATIONS**

DS Data Subject

RO Resource owner

RqP Requesting party that wants to access the health records.

AS Authorization server

LAS Local Authorization Server

RAS Remote Authorization Server

RS Resource Servers

LRS Local Resource Server

RRS Remote Resource Server

PT Permission Token

VIT Verified Identity Token

RPT Requesting Party

kN,M Shared secret key of entity N and M

RN,M Service request sent from entity N to M.

R||R2 Concatenation of request R1 and R2.

IN Identity of entity N.

signN (RN,M) Request RN,M signed by entity N.

enckN,M (RN,M) Request RN,M encrypted with key kN,M.

REN,M Receipt of transaction between entity N and M.

hash(R) Digest of request

**CHAPTER-1**

**INTRODUCTION**

* 1. **GENERAL BACKGROUND**

People like to travel and may visit different doctors across different jurisdictions (such as countries). eHealth data of individuals is managed by the diverse health service providers and stored in different locations. Although there are many agreements between different jurisdictions (such as countries), in general they do not allow eHealth data to be shared externally. Often, sharing is not even allowed within the same country between different healthcare providers. There are several reasons but, as a general theme, it is due to a lack of trust or data disclosure considerations stemming from compliance and regulations. There is a consensus that healthcare data is sensitive personal information that must be well protected.

Here, we tackle the problem of sharing eHealth data across different jurisdictions. The overall challenge stems from three fundamental problems. First, there needs to be full accountability (e.g., non-repudiation) when sharing data; accountability also refers to the possibility of someone sharing healthcare data without the authorization of the patient. Second, since there is no global infrastructure to discover the locations of eHealth data, it must be a truly decentralized scheme. Third, before sharing eHealth data, explicit consent must be obtained from the data subject. The eHealth data custodian needs to be able to demonstrate that appropriate measures had been taken to address these problems if there is an audit or breach in the future.

* 1. **OBJECTIVE**

The problem of sharing eHealth data across different jurisdictions stems from these three fundamental problems.

* There needs to be full accountability (e.g., non-repudiation) when sharing data.
* There is no global infrastructure to discover the locations of eHealth data.
* Before sharing eHealth data, explicit consent must be obtained from the data subject.
  1. **PURPOSE AND SCOPE**

Our project focus on auditability, confidentiality, and distributedness in cross-jurisdiction eHealth data exchange. Firstly, minimize the information exchanged in the sharing network. It is designed to verify identity information in the local jurisdiction and it is not necessary to check and share identity information cross-jurisdiction. Secondly, they propose a scheme where lightweight and short-lived authorization tokens are created and shared between entities to access eHealth data cross region. For the identity information, a token without identity information is produced by authorization party to represent the verified identity. All entities share and verify these tokens with each other to create an individual identity while not disclosing information. Tokens are shared through blockchain with smart-contracts, which are system-agnostic. Our project tackle the problem of sharing eHealth data across different jurisdictions.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 RELATED PAPER**

**2.1.1 “User-Controlled, Auditable, Cross-Jurisdiction Sharing of Healthcare Data Mediated by a Public Blockchain”- Xiaohu Zhou, Vitor Jesus, Yonghao Wang and Mark Josephs-2020**

This paper tackle the problem of eHealth data is managed by diverse health service providers with an agreement between different jurisdictions not to share the data externally due to lack of trust or data disclosure considerations by introducing a decentralized scheme known as BRUE.

The above paper propose a BRUE scheme:-

* Centering all information exchange and control on data subject,
* Combine a set of technologies and standards (Blockchain, Receipts, UMA for eHealth data sharing).

**2.1.2 “Blockchain technology: Application in health care”- S. Angraal, M. H. Krumholz, and L. W. Schulz-2017**

Assess several healthcare applications based on blockchain. Point out that key limitations of blockchain technology expanded to large-scale production deployment in future research are system scalability, security and cost-effectiveness

**2.1.3 “Metrics for assessing blockchain-based healthcare decentralized apps,”-P. Zhang, M. A. Walker, J. White, D. C. Schmidt, and G. Lenz-2017**

Define set of evaluation metrics for blockchain based healthcare decentralized applications to guild development of blockchain applications in health domain. It Includes cost effectiveness, patient-centered care model, system scalability, inter operability, user identification and so on. Concern only HIPAA requirements.(Health care regulation in US).

**2.1.4 “Blockchain in healthcare applications: Research challenges and opportunities”-T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He-2019**

It compares nine types of existing blockchain based applications in healthcare and give tips on how blockchain technology meet which requirements of the healthcare industry. They also present limitations and technical issues of blockchain technology, such as mining incentives and standardization, although its applications have potential benefits for the healthcare industry.

**2.1.5 “Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems”- R. Guo, H. Shi, Q. Zhao, and D. Zheng-2018**

MA-ABS scheme is using blockchain to exchange encapsulated electronic medical record with an attribute-based signature scheme authorized by multiple authorities .In this approach, although the exchanged message is endorsed by participants without any information disclosure, it has a scalability problem due to a significant volume of storage of EMRs data in the blockchain

**2.1.6 “Blockchain: A panacea for healthcare cloud-based data security and privacy?”- C. Esposito, A. De Santis, G. Tortora, H. Chang, and K. R. Choo-2018**

Suggest an approach that uses conventional or distributed database to store medical data and an online chain to record hash values of those data.It uses a private blockchain to store eHealth data and a consortium blockchain to record the secure indexes of eHealth data.These approaches have benefits on access control and confidentiality because of storing all data in the blockchain, they are not compliant with regulations and specifications.

**2.2 TECHNOLOGY**

**2.2.1 EHEALTH DATA EXCHANGE**

**2.2.1.1 Blockchain in Healthcare Data Exchange**

Healthcare is a sensitive domain that poses and processes a large amount of personal medical data daily. Regarding the requirement of eHealth data exchange, the application of blockchain (Distribute ledger) technology in eHealth data exchange is a continuous hot topic. Prior literature about eHealth data exchange with distributed ledger commonly review the current research situation and evaluate the existing distributed application. This raises a problem of performance feasibility as there is no any implementation work prove in practice. Angraal et al. assess several healthcare applications based on blockchain and point out that key limitations of blockchain technology expanded to large-scale production deployment in future research are system scalability, security and cost-effectiveness. Zhang et al define a set of evaluation metrics for blockchain-based healthcare decentralized applications to guild the development of blockchain applications in the healthcare domain, which include cost-effectiveness, patient-centered care model, system scalability, interoperability, user identification, and Turing complete operations. However, those metrics concern only the requirements of HIPAA1, the key Healthcare regulation in the United States. McGhin et al compare nine types of existing blockchain-based applications in healthcare and give tips on how blockchain technology meet which requirements of the healthcare industry. They also present limitations and technical issues of blockchain technology, such as mining incentives and standardization, although its applications have potential benefits for the healthcare industry. Mackey et. recommend a ‘fit-for-purpose’ framework as a guide of health blockchain application design.

Some approaches attempt to store all data with a public, or private, or consortium blockchain instead of traditional data storage; other approaches use a mixed approach in data storage. MA-ABS scheme is a typical approach for medical data exchange with blockchain, which is using blockchain to exchange encapsulated electronic medical record (EMR) with an attribute-based signature scheme authorized by multiple authorities. In this approach, although the exchanged message is endorsed by participants without any information disclosure, it has a scalability problem due to a significant volume of storage of EMRs data in the blockchain. Another example is a blockchain-based secure and privacy-preserving personal health information sharing (BSPP) scheme for diagnosis improvement. It uses a private blockchain to store eHealth data and a consortium blockchain to record the secure indexes of eHealth data. These approaches have benefits on access control and confidentiality because of storing all data in the blockchain, they are not compliant with regulations and specifications, such

as EU/GDPR2, which requires data subjects to have the right to erase personal information. However, depending on the type of blockchain, it may not be possible to modify or delete after-the-fact.

MedBlock is a blockchain-based information management system, which attempts to provide large scale data retrieval and share without extra costs and network congestion. While it does not consider participants’ incentive in the system, MeDShare is a blockchain application to improve security and data authentication in medical data sharing. MedRec in turn, built upon existing databases that support open standards of healthcare exchange to facilitate data share and authentication. The key focus is on designing mining incentives and is not focused on the security issue of the existing database.

The design focus on auditability, confidentiality, and distributedness in cross-jurisdiction eHealth data exchange. Firstly, minimize the information exchanged in the sharing network. It is designed to verify identity information in the local jurisdiction and it is not necessary to check and share identity information cross-jurisdiction. Secondly, propose a scheme where lightweight and short-lived authorization tokens are created and shared between entities to access eHealth data cross region. For the identity information, a token without identity information is produced by authorization party to represent the verified identity. All entities share and verify these tokens with each other to create an individual identity while not disclosing information. Tokens are shared through blockchain with smart-contracts, which are system-agnostic. for the existing infrastructure. It improves system scalability. Furthermore, since tokens can be revoked at any time with immediate effect, it promotes compliance with virtually all regulations. Thirdly, we reuse the notion of Personal Data Receipt from the Data Protection communities . **MEDICHAIN** provides acknowledgement Receipts for all operations of the involved entities. Receipts not only meet accountability requirements (for data controllers) but also provide the data subject with means to trace accesses to the past. Finally, cryptographic access is required for any exchange between two entities. For this matter, we adapt the well-known Diffie Hellman secret sharing scheme to be used in a blockchain providing the useful result of proving, beyond any doubt, that the two parties were engaged. It further provides forward secrecy and confidentiality.

**2.2.1.2 OAuth**

OAuth (Open Authorization) is an open standard authorization framework for token-based authorization on the internet. OAuth, which is pronounced "oh-auth," enables an end user's account information to be used by third-party services, such as Facebook and Google, without exposing the user's account credentials to the third party. It acts as an intermediary on behalf of the end user, providing the third-party service with an access token that authorizes specific account information to be shared. The process for obtaining the token is called an authorization flow.

OAuth 1.0 was first released in 2007 as an authorization method for the Twitter application program interface (API). In 2010, the IETF OAuth Working Group published the first draft of the OAuth 2.0 protocol. Like the original OAuth, OAuth 2.0 provides users with the ability to grant third-party application access to web resources without sharing a password. However, it is a completely new protocol, and is not backward compatible with OAuth 1.0. Updated features include a new authorization code flow to accommodate mobile applications, simplified signatures and short-lived tokens with long-lived authorizations.

The authorization flow in a typical OAuth 2.0 implementation is a six-step process. In the example below, an online calendar creation application needs to be able to access a user's photos stored on their Google Drive:

1. The calendar creation application (the client) requests authorization to access protected resources, in this case image files, owned by the user (resource owner) by directing the user to the authorize endpoint.

2. The resource owner authenticates and authorizes the resource access request from the application, and the authorize endpoint returns an authorization grant to the client. The OAuth 2.0 protocol defines four types of grants: Authorization Code, Client Credentials, Device Code and Refresh Token.

3. The client then requests an access token from the authorization server by presenting the authorization grant returned from the authorize endpoint along with authentication of its own identity to the token endpoint. A token endpoint is a URL such as https://your\_domain/oauth2/token.

4. If the client identity is authenticated and the authorization grant is valid, the authorization server or authentication provider -- Google's Authorization Server in this instance -- will issue an access token to the client.

5. The client can now request the protected resources from the resource server -- Google Drive in this example -- by presenting the access token for authentication.

6. If the access token is valid, the resource server returns the requested resources to the calendar creation application (client).

Now the calendar creation application can access and import the user's photos to create a calendar. Depending on the grant type issued in step two, the authorization flow may differ slightly. However, it still largely follows these core steps.

Examples of OAuth

OAuth is often used to consolidate user credentials and streamline the login process for users, so that when they access an online service, they don't have to reenter information that many of their other online accounts already possess.

OAuth is the underlying technology used for website authentication by sites that let users register or login using their account with another website such as Facebook, Twitter, LinkedIn, Google, GitHub or Bitbucket. For example, a user clicks on the Facebook login option when logging into another website, Facebook authenticates them, and the original website logs them in using permission obtained from Facebook.

**2.2.1.3 User Managed Access**

User-Managed Access (UMA) 2.0 is a federated authorization standard protocol built on top of Open Authentication (OAuth) 2.0 which enables party-to-party sharing. The award winning protocol was introduced by the Kantara Initiative.

UMA gives a resource owner the ability to control access to resources (digital assets) from a centralized authorization server by creating authorization policies regardless of where the resources reside. For example, a patient sharing health-related data such as medical prescriptions and blood reports with doctors, the insurance company, and family members.

Sharing resources can be done selectively which allows individuals to gain control of their resource sharing reliably and securely. The specialty of this protocol is that resource owners need not be online during access as cross-party sharing is driven by predefined policies.

Using UMA 2.0 standards enables individuals achieve the following:

· Context: The right moment to make the decision to share.

· Control: The power to share only what the resource owner desires.

· Choice: The ability to make an independent decision.

· Respect: Regard for one’s preferences.

UMA Versions

The first version, UMA 1.0, was released in 2015. The Kantara Initiative officially announced the approval and publication of UMA Version 2.0’s technical specifications in February 2018. UMA 2.0 was designed to be closely associated with the well-known OAuth protocol, making it easier to implement while improving its security.

UMA 2.0 Specification

The protocol consists of two specifications:

1. UMA 2.0 Grant for OAuth 2.0 Authorization: Specifies how a client should use a permission ticket to request an OAuth 2.0 access token to gain access to a protected resource asynchronously from the time a resource owner authorizes access.

2. Federated Authorization for UMA 2.0: Defines a means for an UMA-enabled authorization server and resource server to be loosely coupled, or federated, in a secure and authorized resource owner context. This specification is optional to be used with an UMA grant.

Anatomy of UMA 2.0

The UMA protocol consists of five roles:

1. Resource Owner: An entity which grants access to a protected resource.

2. Requesting Party: A person that seeks access to a protected resource by the support of a client.

3. Client: An application that makes requests for protected resources on behalf of the requesting party with the resource owner’s authorization.

4. Resource Server: A server that hosts protected resources and has the capability of reacting to requests for protected resources.

5. Authorization Server: A server that protects resources hosted by a resource server on behalf of the resource owner.

The manner in which the resource owner manages resources at the resource server and how policies are defined at the authorization server are out of scope for the UMA specification.

**2.2.1.4 Concept Receipts**

This approach use receipts for transaction records to assure auditability. When a new information is shared between entities, a receipt is generated. For the **MEDICHAIN**, there are two kinds of receipts:

• Receipt for services request. If RqP starts a service request to access eHealth data for DS, a receipt is generated and a hash digest of the request is included.

• Receipt for new token generated and shared. When entity N generates a token and then pushes it into the blockchain, a receipt is returned to N which has the hash value of this transaction.

**CHAPTER 3**

**PROBLEM STATEMENT**

Suppose a person ‘A’ with a health issue born in France registered a GP(FGP),after 10 years, moved to UK then to Canada when felt sick and visited CGP. Returned to UK, visited BGP. Here there is a need of sharing her previous medical records from Canada and France.

● Four entities :- Data Subject(A), Requesting Party(BGP),Data controllers-FGP and CGP-Independent parties in different jurisdictions, no global system to communicate.

● ‘A’ needs to intermediate request and grant access, in turn needs to authenticate against FGP and CGP.

The key requirements of a distributed cross-jurisdiction eHealth data exchange architecture needs to target auditability, non-repudiation, confidentiality, and compatibility.

There is a consensus that healthcare data is sensitive personal information that must be well protected. Therefore different countries have imposed severe national regulations on sharing of healthcare data cross jurisdiction .In case of emergencies, it is very difficult. To illustrate the problem, we informally analyze a simple working scenario of international eHealth data exchange. Alice (A) has had heart trouble since she was born in France. She has registered a GP (FGP) in her original city. Then she moved to the UK after 10 years. When she was 20 years old and travelled to Canada, she fell sick and visited a GP (CGP) to get e temporary treatment in Canada. After that trip, she returned to the UK and visited her British doctor (BGP, where GP stands for ” General Practitioner”). She would like to share the British GP with her previous medical data which is stored respectively in Canada and France. However, she doesn’t want to simply give British GP her personal credential but would rather authorize British GP which could gain access using GP’s credential. In the scenario, there are four entities: the data subject A, a requesting party BGP, and data controllers which are healthcare service providers FGP and CGP. To note that A, BGP, CGP, and FGP, are independent parties located in different jurisdictions. Some of them are not known to the others; their only point of contact is their relationship with A, the patient and data subject. There is no global system in place that allows all parties to directly communicate, find each other, or self-certify. In other words, BGP needs to access A’s medical data from FGP and CGP but FGP and CGP do not recognize BGP so A needs to intermediate the request and grant access. A, in turn, needs to authenticate against FGP and CGP. Healthcare is a sensitive domain that poses and processes a large amount of personal medical data daily. Regarding the requirement of eHealth data exchange, the application of blockchain (Distribute ledger) technology in eHealth data exchange is a continuous hot topic. The above problem is trying to be solved by combining such different technologies.

**CHAPTER 4**

**PROPOSED SYSTEM**

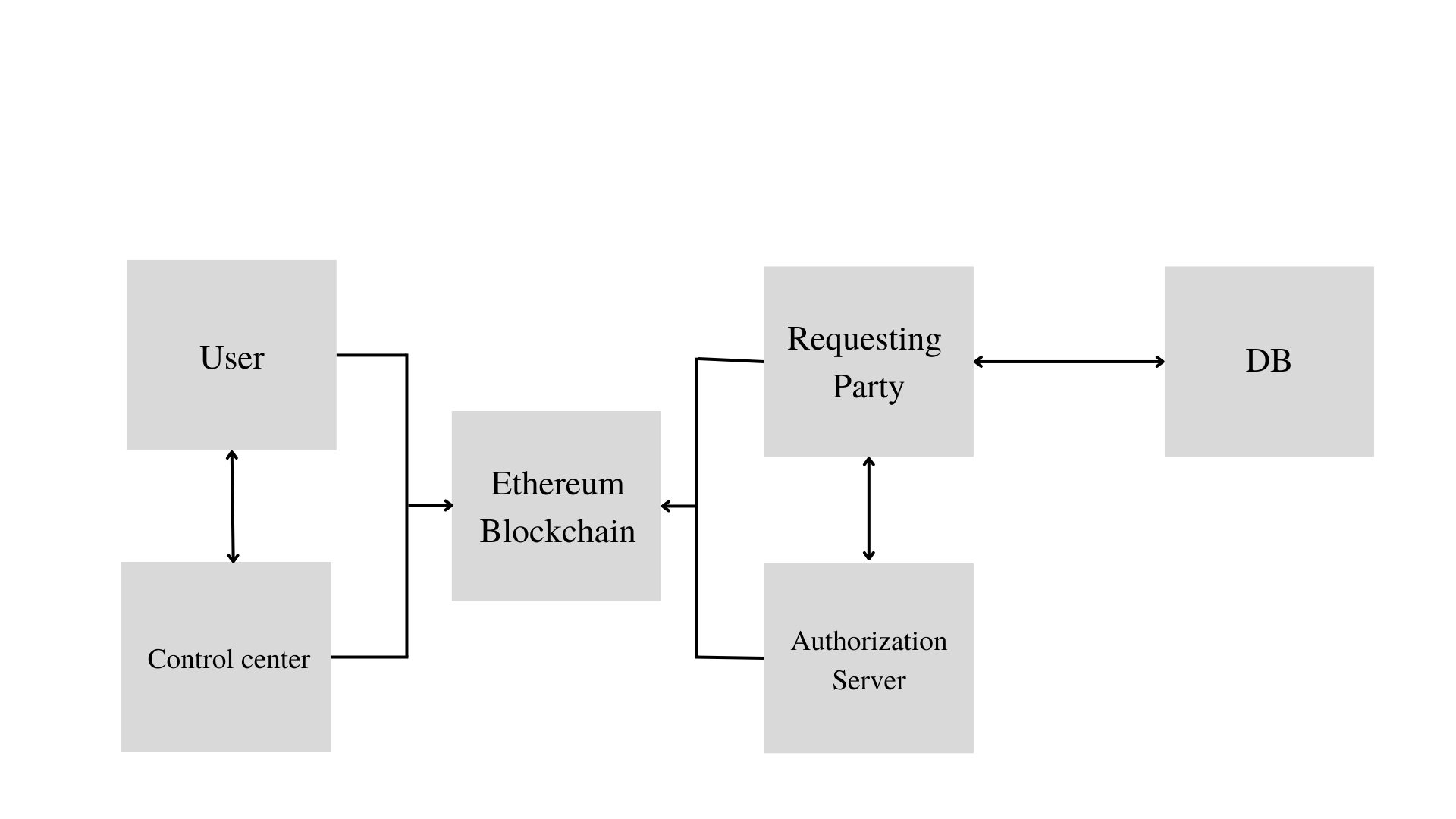
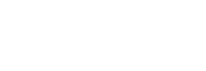
**4.1 ARCHITECTURE**

Fig 4.1 Architecture



The proposed system follows an approach which provides the feasibility of cross-jurisdiction eHealth data exchange. It produces receipt record as participants’ consent integrated with UMA standards to target auditability and compatibility; in addition to, runs on a public blockchain to achieve non-repudiation and confidentiality with blockchain’s features.

At first, Requesting Party (RqP) requests service from the data subject. Resource Owner (RO) processes this request on behalf of the data subject. RO checks service request and then returns a PT. Then, RqP pulls Permission Token (PT) and then sends to Local Authorization Server (LAS) with identity information for identity verification. LAS returns a Verified Identity Token(VIT) to RqP after verification. RqP uses VIT to Remote Authorization Server (RAS) for identity and authorization check in the cross-jurisdiction. RAS then gives a Requesting Party (RPT) to RqP for final progress of eHealth data access. RqP shares RPT with Remote Resource Server (RRS) to get eHealth data. After permission check, RRS returns the required eHealth data to RqP. The whole data flow terminates at this point. During transactions progresses, all authorized tokens are pushed to the blockchain firstly and then pulled by entities always mediated by the blockchain. In the blockchain, smart-contracts are triggered by transactions to process tokens sharing. A receipt is always generated following transaction of new information, such as a new token generation and push. All data flow of transactions from entities go through the blockchain network for provenance and accountability. A receipt of each transaction is also produced following the data flow. In the sketch, N and M start by preparing R-wallet to receive receipts, N then requests services from M by invoking smart-contracts running on the blockchain. M processes work pulled from the blockchain (via notifications) to complete the request from N. When M returns the expected outcome and pushes the result to the blockchain, N then obtains the outcome from the blockchain itself. The fact that the communication channel is the blockchain itself guarantees traceability. The receipt as proof of an intermediate transaction is generated and follows the data flow in both directions. The peer-to-peer data exchange between N and M terminates at this point. The receipt generated while the data flow progresses is composed of the individual audit records. Should a concern or dispute arise in the future, this receipt holds all the evidence needed to keep all participants accountable. Consider a general scenario of a cross-jurisdiction eHealth data exchange flows with multiple parties: data subject, a requesting party, the data controller, and a verifier/authorizer. Each entity contributes to the overall data flow work done. The topology of participants is established before it starts because of services request and data flow direction. It is also static for its duration. Assume that, without loss of work generalization and rejection, that the graph is acyclic when each entity is a requester or responder. In other words, data flows the graph with a path by order such that no entity is a requester or a responder twice. If a particular data exchange workflow uses the same participant twice at different times, the model has a different node in the graph. The system is agnostic in terms of the actual eHealth data format.

**4.2 Stakeholders**

Patients: Patients are a major stakeholder this project that involves the sharing of healthcare data, as they have a direct interest in the accuracy and security of the data that is being shared.

Healthcare organizations: Healthcare organizations, including hospitals, clinics, and other healthcare facilities, are stakeholders in the project, as they will be responsible for managing and sharing healthcare data with patients and other providers and some may be the requesting parties themselves.

**4.3 ENVIRONMENT**

**4.3.1 Web3 Application**

Web3 enhances the internet as we know it today with a few other added characteristics. web3 is:

* Verifiable
* Trustless
* Self-governing
* Permissionless
* Distributed and robust
* Stateful
* Native built-in payments

In web3, developers don't usually build and deploy applications that run on a single server or that store their data in a single database usually hosted on and managed by a single cloud provider.

Instead, web3 applications either run on blockchains, decentralized networks of many peer to peer nodes (servers), or a combination of the two that forms a [crypto-economic protocol](https://thegraph.com/blog/modeling-cryptoeconomic-protocols-as-complex-systems-part-1).

To achieve a stable and secure decentralized network, network participants (developers) are incentivized and compete to provide the highest quality services to anyone using the service.

These protocols may often offer a variety of different services like compute, storage, bandwidth, identity, hosting, and other web services commonly provided by cloud providers in the past.

People can make a living by participating in the protocol in various ways, in both technical and non-technical levels.

**4.3.2 MySQL**

MySQL is one of the most recognizable technologies in the modern big data ecosystem. Often called the most popular database and currently enjoying widespread, effective use regardless of industry, it’s clear that anyone involved with enterprise data or general IT should at least aim for a basic familiarity of MySQL.

With MySQL, even those new to relational systems can immediately build fast, powerful, and secure data storage systems. MySQL’s programmatic syntax and interfaces are also perfect gateways into the wide world of other popular query languages and structured data stores.

MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL).

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or a place to hold the vast amounts of information in a corporate network. In particular, a relational database is a digital store collecting data and organizing it according to the relational model. In this model, tables consist of rows and columns, and relationships between data elements all follow a strict logical structure. An RDBMS is simply the set of software tools used to actually implement, manage, and query such a database.

MySQL is integral to many of the most popular software stacks for building and maintaining everything from customer-facing web applications to powerful, [data-driven B2B services](https://www.talend.com/resources/business-intelligence-data-analytics/).

**4.4 SYSTEM REQUIREMENTS**

**4.4.1 Node JS**

Node.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) server environment. Node.js is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) and runs on Windows, Linux, Unix, and macOS. Node.js is a [back-end](https://en.wikipedia.org/wiki/Front_end_and_back_end) [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [runtime environment](https://en.wikipedia.org/wiki/Runtime_system). Node.js runs on the [V8](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) [JavaScript Engine](https://en.wikipedia.org/wiki/JavaScript_Engine) and executes JavaScript code outside a [web browser](https://en.wikipedia.org/wiki/Web_browser).

Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting). The functionality of running scripts server-side produces [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying [web-application](https://en.wikipedia.org/wiki/Web_application) development around a single programming language, rather than different languages for server-side and client-side scripts.

Node.js has an [event-driven architecture](https://en.wikipedia.org/wiki/Event-driven_architecture) capable of [asynchronous I/O](https://en.wikipedia.org/wiki/Asynchronous_I/O). These design choices aim to optimize [throughput](https://en.wikipedia.org/wiki/Throughput) and [scalability](https://en.wikipedia.org/wiki/Scalability) in web applications with many input/output operations, as well as for [real-time Web](https://en.wikipedia.org/wiki/Real-time_Web) .

The Node.js [distributed development](https://en.wikipedia.org/wiki/Distributed_development) project was previously governed by the Node.js Foundation, and has now merged with the [JS Foundation](https://en.wikipedia.org/wiki/JS_Foundation) to form the [OpenJS Foundation](https://en.wikipedia.org/wiki/OpenJS_Foundation" \o "OpenJS Foundation). OpenJS Foundation is facilitated by the [Linux Foundation](https://en.wikipedia.org/wiki/Linux_Foundation)'s Collaborative Projects program.

Node.js allows the creation of [Web servers](https://en.wikipedia.org/wiki/Web_server) and networking tools using [JavaScript](https://en.wikipedia.org/wiki/JavaScript) and a collection of "modules" that handle various core functionalities. Modules are provided for [file system](https://en.wikipedia.org/wiki/File_system) I/O, networking , [binary](https://en.wikipedia.org/wiki/Binary_file) data (buffers), [cryptography](https://en.wikipedia.org/wiki/Cryptography) functions, [data streams](https://en.wikipedia.org/wiki/Stream_(computing)), and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.

JavaScript is the only language that Node.js supports natively, but many [compile-to-JS](https://en.wikipedia.org/wiki/Source-to-source_compiler) languages are available. As a result, Node.js applications can be written in [CoffeeScript](https://en.wikipedia.org/wiki/CoffeeScript" \o "CoffeeScript), [Dart](https://en.wikipedia.org/wiki/Dart_(programming_language)), [TypeScript](https://en.wikipedia.org/wiki/TypeScript), [ClojureScript](https://en.wikipedia.org/wiki/ClojureScript" \o "ClojureScript) and others.

Node.js is primarily used to build network programs such as Web servers. The most significant difference between Node.js and [PHP](https://en.wikipedia.org/wiki/PHP) is that most functions in PHP [block](https://en.wikipedia.org/wiki/Asynchronous_I/O) until completion (commands execute only after previous commands finish), while Node.js functions are [non-blocking](https://en.wikipedia.org/wiki/Asynchronous_I/O) (commands execute [concurrently](https://en.wikipedia.org/wiki/Concurrent_computing) or even in [parallel](https://en.wikipedia.org/wiki/Parallel_computing), and use [callbacks](https://en.wikipedia.org/wiki/Callback_(computer_programming)" \o "Callback (computer programming)) to signal completion or failure).

**4.4.2 Web3.js**

Web3.js is a collection of libraries that allow developers to interact with a remote or local Ethereum node using HTTP, IPC, or WebSocket. Using this library, websites or clients that can interact with the blockchain are developed. This can be actions like sending Ether from one user to another, checking data from smart contracts, creating smart contracts, among other things. Ethereum nodes provide interfaces to users in order to complete transactions: of which, nodes receive this information through a JSON RPC interface. This is an encoding format that allows running processes to receive new and verify existing data. Web3.js helps to make the process of running and selecting nodes participating in the Ethereum network simpler and easier to grasp.

Web3.js helps developers and users alike to interact with the Ethereum blockchain network. Web3.js itself represents a JavaScript language binding for the aforementioned JSON RPC interface. This allows for the library to be inherently usable and flexible (since Javascript is natively supported in most popular web browsers), but also allows use on the server side in Node.js applications or Electron-based ones. The most common use is the one frequently stated in this piece, however: as accessing the Ethereum blockchain seems to be the chief reason that someone seeks out Web3.js. Users utilize Ethereum nodes via HTTP, which can either be locally hosted, by a Dapp (decentralized application) provider, or through public gateways like Moralis or Infura.

Generally, those creating Dapps or integrated web browser applications into the Ethereum blockchain utilize the Metamask browser extension in conjunction with Web3.js. Metamask is an Ethereum wallet that is hosted in-browser and natively puts a Web3 provider object into said browser: which, in brief, a Web3 provider object is a data-structure that provides links to publicly accessible Ethereum nodes. Using Web3.js and Metamask, users and developers are able to manage private keys and verify transactions directly from their preferred browser.

**4.4.3 Ethereum PoS**

Ethereum is a blockchain with a computer embedded in it. It is the foundation for building apps and organizations in a decentralized, permissionless, censorship-resistant way.

In the Ethereum universe, there is a single, canonical computer (called the Ethereum Virtual Machine, or EVM) whose state everyone on the Ethereum network agrees on. Everyone who participates in the Ethereum network (every Ethereum node) keeps a copy of the state of this computer. Additionally, any participant can broadcast a request for this computer to perform arbitrary computation. Whenever such a request is broadcast, other participants on the network verify, validate, and carry out ("execute") the computation. This execution causes a state change in the EVM, which is committed and propagated throughout the entire network.

Requests for computation are called transaction requests; the record of all transactions and the EVM's present state gets stored on the blockchain, which in turn is stored and agreed upon by all nodes.

Cryptographic mechanisms ensure that once transactions are verified as valid and added to the blockchain, they can't be tampered with later. The same mechanisms also ensure that all transactions are signed and executed with appropriate "permissions"

Ethereum uses a [proof-of-stake-based consensus mechanism](https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/). Anyone who wants to add new blocks to the chain must stake at least 32 ETH into the deposit contract and run validator software. They then can be randomly selected to propose blocks that other validators check and add to the blockchain. In this model, there is usually only one chain, but network latency and dishonest behavior can cause multiple blocks to exist at the same position near the head of the chain. To resolve this, a fork-choice algorithm selects one canonical set of blocks. The blocks selected are the ones that form the heaviest possible chain, where 'heavy' refers to the number of validators that have endorsed the blocks (weighted by the ETH they have staked). There is a system of rewards and penalties that strongly incentivize participants to be honest and online as much as possible.

**4.4.4 Infura**

**Infura is a Web3 backend and Infrastructure-as-a-Service (IaaS) provider that offers a range of services and tools for blockchain developers. This includes the Infura API (Application Programming Interface) suite. The flagship Infura Ethereum API is at the heart of the Infura Web3 service. However, connectivity with both the Inter Planetary File System (IPFS) and Filecoin are in the pipeline. Some Infura alternatives currently offer broader cross-chain connectivity. Despite Ethereum being the number-one programmable blockchain for launching decentralized applications (dApps), many blockchain developers are now looking elsewhere for Infura alternatives. This coincides with the rise in popularity of Binance Smart Chain (BSC) and Polygon Network .**

Infura offers top-of-the-range documentation and resources to help developers build decentralized applications (dApps) quickly. This is achieved by reducing the time spent building infrastructure from scratch. Infura offers enterprise-ready infrastructure using a distributed cloud-hosted network of nodes. This removes much of the friction associated with the development and ownership of proprietary computing and storage facilities.

Infura aims to create an ideal environment for developers and enterprises looking to create exciting blockchain-based products and Web3 applications with a low barrier to entry. In turn, this is enabling developers to build innovative products and services that harness the power of the Ethereum blockchain to be used sector-wide.

**4.4.5 JavaScript**

JavaScript is a scripting or programming language that allows to implement complex features on web pages every time a web page does more than just sit there and display static information for to look at displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes. It is the third layer of the layer cake of standard web technologies.

**4.4.6 HTML**

The Hyper Text Markup Language or HTML is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for documents designed to be displayed in a [web browser](https://en.wikipedia.org/wiki/Web_browser). It can be assisted by technologies such as [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [scripting languages](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript).

[Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [web server](https://en.wikipedia.org/wiki/Web_server) or from local storage and [render](https://en.wikipedia.org/wiki/Browser_engine) the documents into multimedia web pages. HTML describes the structure of a [web page](https://en.wikipedia.org/wiki/Web_page) [semantically](https://en.wikipedia.org/wiki/Semantic_Web) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/HTML_element#Images_and_objects) and other objects such as [interactive forms](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. HTML provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes, and other items. HTML elements are delineated by tags, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript), which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as [HTML5](https://en.wikipedia.org/wiki/HTML5), is used to display video and audio, primarily using the <canvas> element, in collaboration with javascript.

**4.4.7 CSS**

[CSS](https://developer.mozilla.org/en-US/docs/Glossary/CSS) (Cascading Style Sheets) allows to create great-looking web pages.CSS can be used for very basic document text styling — for example, for changing the [color](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value) and [size](https://developer.mozilla.org/en-US/docs/Web/CSS/font-size) of headings and links. It can be used to create a layout — for example, [turning a single column of text into a layout](https://developer.mozilla.org/en-US/docs/Web/CSS/Layout_cookbook/Column_layouts) with a main content area and a sidebar for related information. It can even be used for effects such as [animatio](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations)n.

CSS is no different — it is developed by a group within the W3C called the [CSS Working Group](https://www.w3.org/Style/CSS/). This group is made of representatives of browser vendors and other companies who have an interest in CSS. There are also other people, known as invited experts, who act as independent voices; they are not linked to a member organization.

New CSS features are developed or specified by the CSS Working Group — sometimes because a particular browser is interested in having some capability, other times because web designers and developers are asking for a feature, and sometimes because the Working Group itself has identified a requirement. CSS is constantly developing, with new features becoming available. However, a key thing about CSS is that everyone works very hard to never change things in a way that would break old websites.

**4.4.8 MetaMask**

MetaMask is a [software](https://en.wikipedia.org/wiki/Software) [cryptocurrency wallet](https://en.wikipedia.org/wiki/Cryptocurrency_wallet) used to interact with the [Ethereum](https://en.wikipedia.org/wiki/Ethereum) [blockchain](https://en.wikipedia.org/wiki/Blockchain). It allows users to access their Ethereum wallet through a [browser extension](https://en.wikipedia.org/wiki/Browser_extension) or [mobile app](https://en.wikipedia.org/wiki/Mobile_app), which can then be used to interact with [decentralized applications](https://en.wikipedia.org/wiki/Decentralized_application). MetaMask is developed by [ConsenSys Software Inc.](https://en.wikipedia.org/wiki/ConsenSys" \o "ConsenSys), a blockchain [software company](https://en.wikipedia.org/wiki/Software_company) focusing on [Ethereum](https://en.wikipedia.org/wiki/Ethereum)-based tools and infrastructure. MetaMask allows users to store and manage account [keys](https://en.wikipedia.org/wiki/Key_(cryptography)), broadcast transactions, send and receive [Ethereum](https://en.wikipedia.org/wiki/Ethereum)-based [cryptocurrencies](https://en.wikipedia.org/wiki/Cryptocurrency) and tokens, and securely connect to [decentralized applications](https://en.wikipedia.org/wiki/Decentralized_application) through a compatible [web browser](https://en.wikipedia.org/wiki/Web_browser) or the [mobile app](https://en.wikipedia.org/wiki/Mobile_app)'s built-in browser.

Websites or other decentralized applications are able to connect, [authenticate](https://en.wikipedia.org/wiki/Authentication), and/or integrate other [smart contract](https://en.wikipedia.org/wiki/Smart_contract) functionality with a user's MetaMask wallet and any other similar blockchain wallet browser extensions via [JavaScript](https://en.wikipedia.org/wiki/JavaScript) code that allows the website to send action prompts, signature requests, or transaction requests to the user through MetaMask as an intermediary.

The application includes an integrated service for exchanging [Ethereum](https://en.wikipedia.org/wiki/ERC-20) tokens by aggregating several [decentralized exchanges](https://en.wikipedia.org/wiki/Decentralized_exchange) (DEXs) to find the best [exchange rate](https://en.wikipedia.org/wiki/Exchange_rate).

**CHAPTER 5**

**DESIGN DIAGRAM**

**5.1 DATA FLOW DIAGRAM**

A data-flow diagram is a way of representing a flow of data through a [process](https://en.wikipedia.org/wiki/Process) or a system (usually an [information system](https://en.wikipedia.org/wiki/Information_system)). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a [flowchart](https://en.wikipedia.org/wiki/Flowchart).

**5.1.1 LEVEL 0**

Tokens are designed to grant access; in the process, a record is generated in the form of a receipt that a party keeps in its possession if later is audited. Entities share tokens with each other only via a public blockchain running smart-contracts. Two kinds of tokens are defined. One is generated between RO or AS with RqP. When RO or AS generates a new signed token, they send it to the blockchain encrypted by key kRO, RqP or kAS, RqP . Then, RqP gets it from the blockchain and decrypts it with a shared key kRO, RqP or kAS, RqP . Another token is used between RqP and AS or RS. RqP shares an authorization token PT with LAS, or shares VIT with RAS, or shares RP T with RS through blockchain. Before the token exchange, RqP needs to sign and encrypt the token. For example, a shared token PT between RqP with LAS is delivered as enckLAS, RqP (signRqP (PT)). After the token is pushed, AS or RS pulls the protected tokens from the blockchain and decrypts with key kAS,RqP or kRS,RqP .

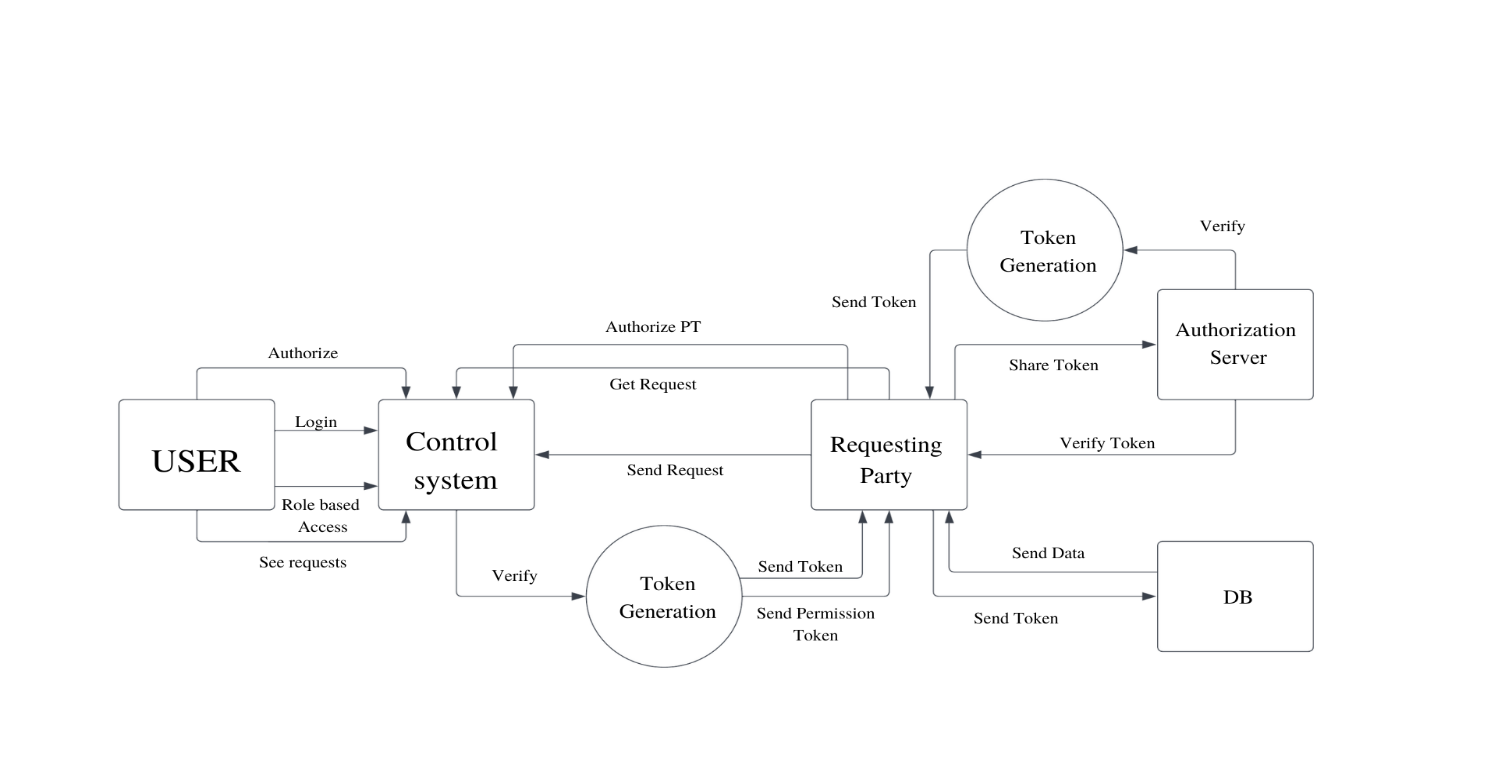
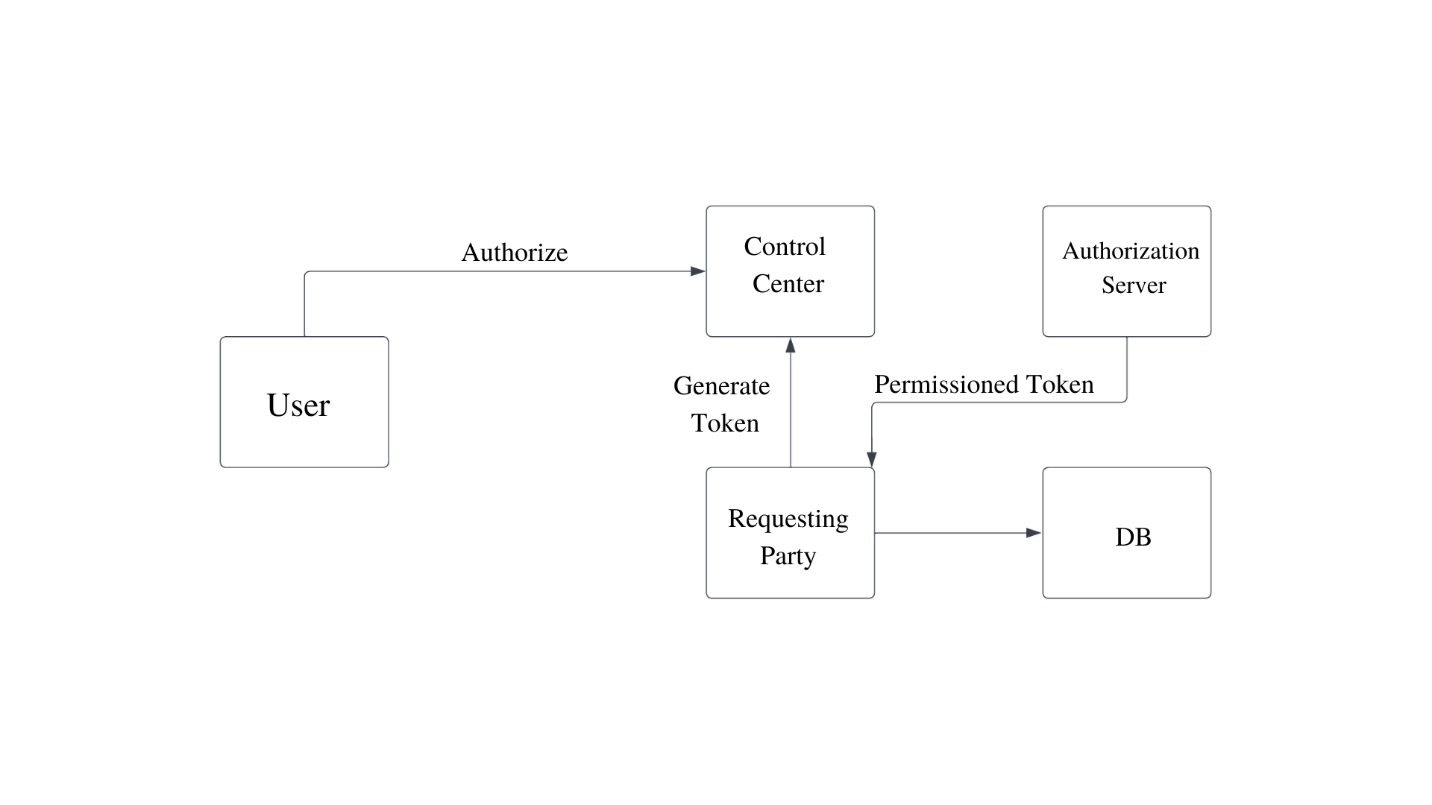


Fig 5.1 DFD Level 0

**5.1.2 LEVEL 1**

In this system, the requesting party sends a request for data to the control system, the user can log in and view the requests and gives permission token to the requesting party. The permission token is send over to the authorization server along with identification details and after the identity has been verified, another token is generated and send back to requesting party after successful verification. This token can be used to access the data from the server where the data is stored. During transactions progresses, all authorized tokens are pushed to the blockchain firstly and then pulled by entities always mediated by the blockchain. In the blockchain, smart-contracts are triggered by transactions to process tokens sharing. A receipt is always generated following transaction of new information, such as a new token generation and push.

Fig 5.2 DFD Level 1

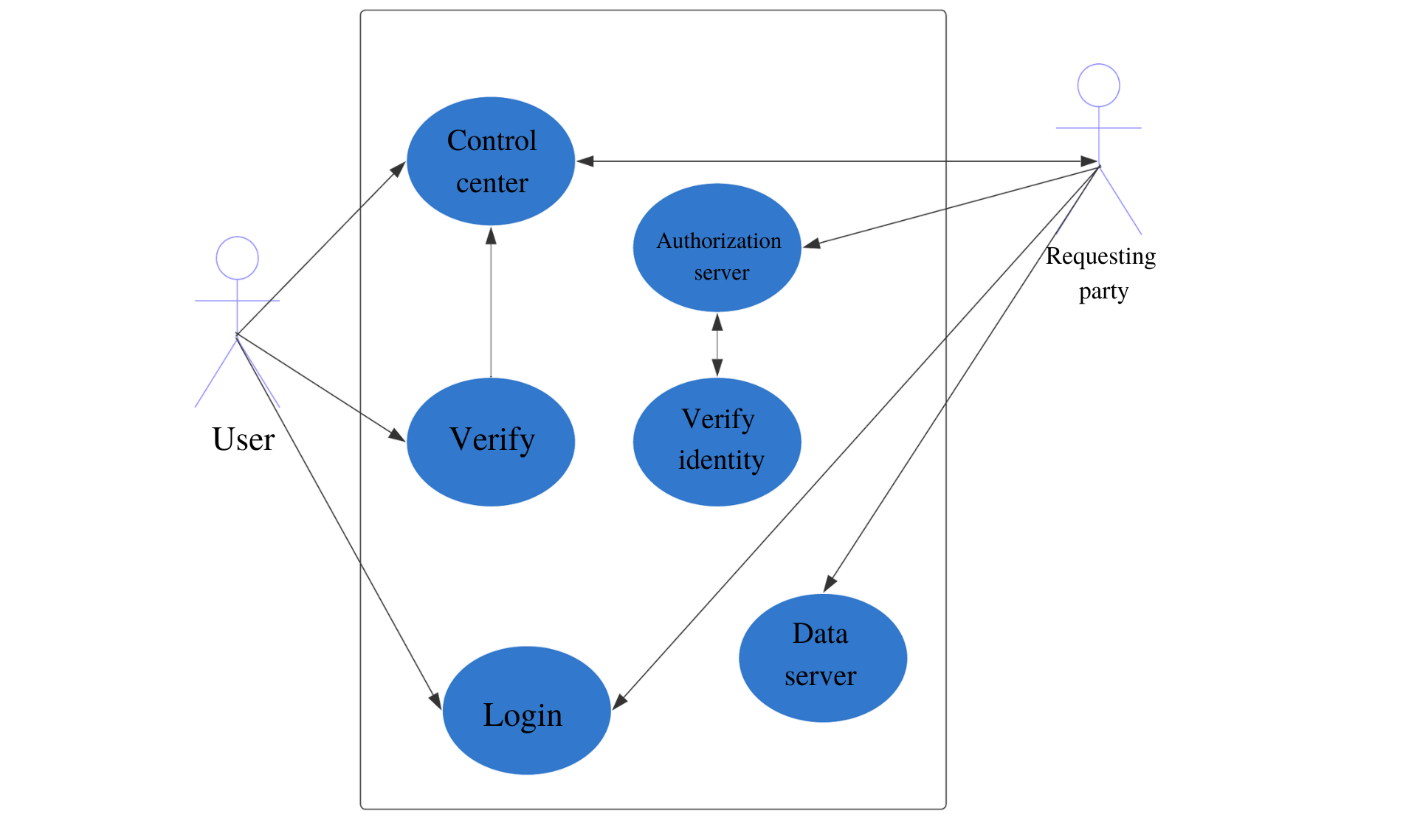
**5.2 USE CASE DIAGRAM**

Fig 5.3 Use Case Diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system. The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

**Actors:**

User: The patient who can review requests and generate permission tokens

Requesting Party: represents the individuals or organizations that will use the system to request and access healthcare data

Control System: represents the system that receives and processes requests for healthcare data from users

Authorization Server: represents the server that verifies the identity of requesting parties and issues tokens for accessing healthcare data

**Use Cases:**

Request Healthcare Data: represents the ability of users to request healthcare data from the control system

Verify Identity: represents the ability of the authorization server to verify the identity of requesting parties

Issue Token: represents the ability of users to issue permission tokens for accessing healthcare data

Generate Token: represents the ability of the authorization server to generate tokens for accessing healthcare data after verifying the identity of requesting parties

Access Healthcare Data: represents the ability of users to access healthcare data using the tokens issued by the authorization server

Share Data: represents the ability of the system to share healthcare data with other users who have permission to access it.

**CHAPTER 6**

**CONCLUSION**

**6.1 CONCLUSION AND FUTURE SCOPE**

Designed the proposed system for eHealth data exchange. It relies on open protocols so that it is easier to adopt in existing systems. The system seems able to satisfy key requirements such as auditability, confidentiality, non-repudiation, and compatibility when supporting cross jurisdiction eHealth data exchange. Preliminary results of our prototype show it is not complex to implement and the user interface can be simple for non-technical users.

This project also opens new research directions. On one hand, there essential features that we should be able to accommodate. permissions. Furthermore, the identity of participants is deliberately overlooked.

**GANTT CHART**

**REFERENCES**

[1] E. Maler (ed.), “User-managed access (uma) 2.0 grant for oauth 2.0 authorization,” Kantara Initiative Recommendation, 2018.

[2] M. Lizar and D. Turner (eds.), “Consent receipt specification,” Kantara Initiative Recommendation, 2017.

[3] S. Angraal, M. H. Krumholz, and L. W. Schulz, “Blockchain technology: application in health care,” Circulation: Cardiovascular Quality and Outcomes, vol. 10, no. 9, Sep 2017.

[4] P. Zhang, M. A. Walker, J. White, D. C. Schmidt, and G. Lenz, “Metrics for assessing blockchain-based healthcare decentralized apps,” in 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), 2017, pp. 1–4.

[5] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, “Blockchain in healthcare applications: Research challenges and opportunities,” Journal of Network and Computer Applications, vol. 135, pp. 62–75, Jun 2019.

[6] T. K. Mackey, T. Kuo, B. Gummadi, A. K. Clauson, G. Church, D. Grishin, K. Obbad, R. Barkovich, and M. Palombini, “’fit-for purpose?’ - challenges and opportunities for applications of blockchain technology in the future of healthcare,” BMC Medicine, vol. 17, no. 1, p. 68, 2019.

[7] R. Guo, H. Shi, Q. Zhao, and D. Zheng, “Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems,” IEEE Access, vol. 6, pp. 11 676–11 686, 2018.

[8] A. Zhang and X. Lin, “Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain,” Journal of Medical Systems, vol. 42, no. 140, June 2018.

[9] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, “Medblock: Efficient and secure medical data sharing via blockchain,” Journal of Medical Systems, vol. 42, June 2018.

[10] 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,” IEEE Access, vol. 5, pp. 14 757–14 767, 2017.

[11] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, “Medrec: Using blockchain for medical data access and permission management,” in 2016 2nd International Conference on Open and Big Data (OBD), 2016, pp. 25–30.

[12] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K. R. Choo, “Blockchain: A panacea for healthcare cloud-based data security and privacy?” IEEE Cloud Computing, vol. 5, no. 1, pp. 31–37, Jan 2018.

[13] X. Zhou, A. Nehme, V. Jesus, Y. Wang, M. Josephs, and K. Mahbub, “Towards blockchain-based auditing of data exchanges,” in Smart Blockchain. Cham: Springer, 2019, pp. 43–52.

[14] P. Zhang, J. White, D. C. Schmidt, and G. Lenz, “Chapter one - blockchain technology use cases in healthcare,” in Blockchain Technology: Platforms, Tools and Use Cases, ser. Advances in Computers, P. Raj and G. C. Deka, Eds. Elsevier, 2018, vol. 111, pp. 1 – 41.

[15] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, “Fhirchain: Applying blockchain to securely and saleably share clinical data,” Computational and Structural Biotechnology Journal, vol. 16, pp. 267 – 278, 2018.

[16] HL7, “Fast healthcare interoperability resources (FHIR),” online: https://www.hl7.org/fhir/overview.html, 2011.

[17] V. Jesus, “Towards an accountable web of personal information: the web-of-receipts,” IEEE Access,, vol. 8, 2020.

[18] W. Diffie and M. Hellman, “New directions in cryptography,” IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644–654, 1976.