A

Major Project Report

on

Transplantation and Organ Donation using Blockchain

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

By

Anjana Reddy
(20EG105504)

Nithin Reddy
(20EG105553)

Puli Rajasri
(20EG105554)

Under the Guidance
Mrs. T Veda Reddy
Assistant Professor
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANURAG UNIVERSITY VENKATAPUR-500088
TELANGANA
(2023-2024)

DECLARATION

I declare that the Report entitled “**TRANSPLANTATION AND ORGAN DONATION USING BLOCKCHAIN**” submitted for the award of Bachelor of technology Degree is my own work and the Report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma

Place: Anurag University, Hyderabad

Anjana
(20EG105504)

Nithin
(20EG105553)

Rajasri
(20EG105554)

CERTIFICATE

This is to certify that the Report entitled **Transplantation And Organ Donation Using Blockchain** that is being submitted by **Ms. Anjana Reddy** bearing Hall Ticket number **(20EG105504), Mr. Nithin Reddy** bearing Hall Ticket number **(20EG105553),** and **Ms. Puli Rajasri** bearing Hall Ticket number **(20EG105554)** in partial fulfilment for the award of **Bachelor of Technology** in **Computer Science and Engineering** to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this report have not been submitted to any other university or Institute for the award of any degree or diploma

Signature of Supervisor	Dean CSE
T. Veda Reddy	Dr. G. Vishnu Murthy
Assistant Professor	
Department of CSE	

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **T. Veda Reddy** for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. Her patience, guidance and encouragement made this project possible.

We would like to express my special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B.Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy** , Dean, Dept. of CSE ,Anurag University. We also express my deep sense of gratitude to **Dr.V.V.S.S.S.Balaram**, Academic co-ordinator and **Dr. Pallam Ravi** Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated me during the crucial stage of our project work.

ABSTRACT

Cloud Health Record (CHR) has recorded the process of occurrence, development, and treatment of diseases, therefore it has great medical value. Cloud Health Records are personal and vital records for every patient, owing to the privacy and sensitive nature of medical data for patients, the data sharing, security and privacy preservation are critical issues in CHR. Technology advancement have created so much benefits which includes improve security, user experiences that are key factors to the success of Cloud Health Records in recent years. However, there are still enormous issues being faced regarding the security of medical records, users absolute ownership of data with integrity. The appropriate solution to the identified issues could be the novel technology, i.e., Blockchain. The technology is meant to provide well architected secure, tampered proof platform that allows medical records to be stored with other healthcare related information. Cryptographic techniques are employed to preserve privacy. Features of the model include fine-grained and flexible access control, revocability of consent, auditability, and tamper resistance. A detailed security analysis would show that the model is provably secure for privacy and tamper resistance. The performance analysis would show that the model achieves a better overall performance compared with the existing approach in the literature review.

Keywords: Cloud Health Record, Blockchain technology

LIST OF FIGURES

CONTENTS

INTRODUCTION

1.1 Background Statement

Before the advent of modern technology,healthcare sector used paper based system to store the medical records, i.e., using handwritten mechanism. This paper-based medical record system was inefficient, insecure, unorganized and was not temper-proof. It also faced the issue of data- duplication and redundancy as all the institutions that patient visited had various copies of patient’s medical records

1.2 Problem Statement

Below are some of the reasons that would led to the development of this system;

Information Asymmetry*:*There is inconsistency in information exchange between health providers and patients which consist of medical errors, customer frustration, over- and under- treatment mostly in the western world and predominantly in the developing world. Equalizing the information exchange will ensures there is an engagement with patients, improvement with outcomes and reduction in unnecessary healthcare expenditure

Organ donation is currently not widespread in India, and even among the educated sections of society, there is a lack of knowledge and positive attitudes toward organ transplantation. This has resulted in organ scarcity in the country. Some of the main concerns contributing to this scarcity include a lack of awareness and consciousness among the population, as well as the presence of mythological beliefs and misconceptions surrounding organ donation, often influenced by religious and cultural barriers. This paper proposes the use of a private Ethereum blockchain system to facilitate organ donation and transplantation in a decentralized, secure, trackable, auditable, private, and trustworthy manner. Organ donation has significantly impacted the medical field, and many individuals express a desire to donate their organs, whether they are alive, deceased, or brain dead. One of the main challenges in organ donation is the delay in organ supply, leading to the unfortunate loss of many lives in need. To address this issue, the authors suggest utilizing blockchain technology, which is a distributed database capable of handling dynamic datasets.

Data Breaches:

According to US Department of Health and Human Services Office for Civil Rights, Breaches in Healthcare, have been an issue across the healthcare sector for ages. Between 2009 and 2019 there have been over 3,054 healthcare data breaches involving more than 500 records. Those breaches have resulted in the loss, theft, exposure, or impermissible disclosure of more than 230,954,151 healthcare records. That equates to more than 69.78% of the population of the United States.

Given the rapid expansion in Cloud Health Record deployment since 2012, as well as the expected increase in cloud-based services provided by vendors supporting predictive analytics, personal health records, health-related sensors, and gene sequencing technology, the frequency and scope of electronic health care data breaches are likely to increase

1.3 Project Motivation

Cloud Health Record (CHR) is increasingly being implemented in many developing countries. It is the need of the hour because it improves the quality of healthcare. Recent news of security breaches has put a question mark on this system. Despite its increased usefulness, and increasing enthusiasm in its adoption, not much attention is being paid to the ethical issues that might arise. The frequency of data breaches in healthcare prompted this project.

The motivation of this project is to improve and extend on existing work by implementing a blockchain based Cloud Health Record/medical record for security, integrity, privacy, information asymmetry and data interoperability.

1.4 Project Objectives

The objectives of this project is to develop a blockchain-based Cloud Health Record system that;

- Secures the integrity, confidentiality and privacy of Cloud Health Record.
- Solves the issue of information asymmetry which is allowing patients have better access to their health records
- Implementation of ‘a’, ‘b’ and ‘c’ above.

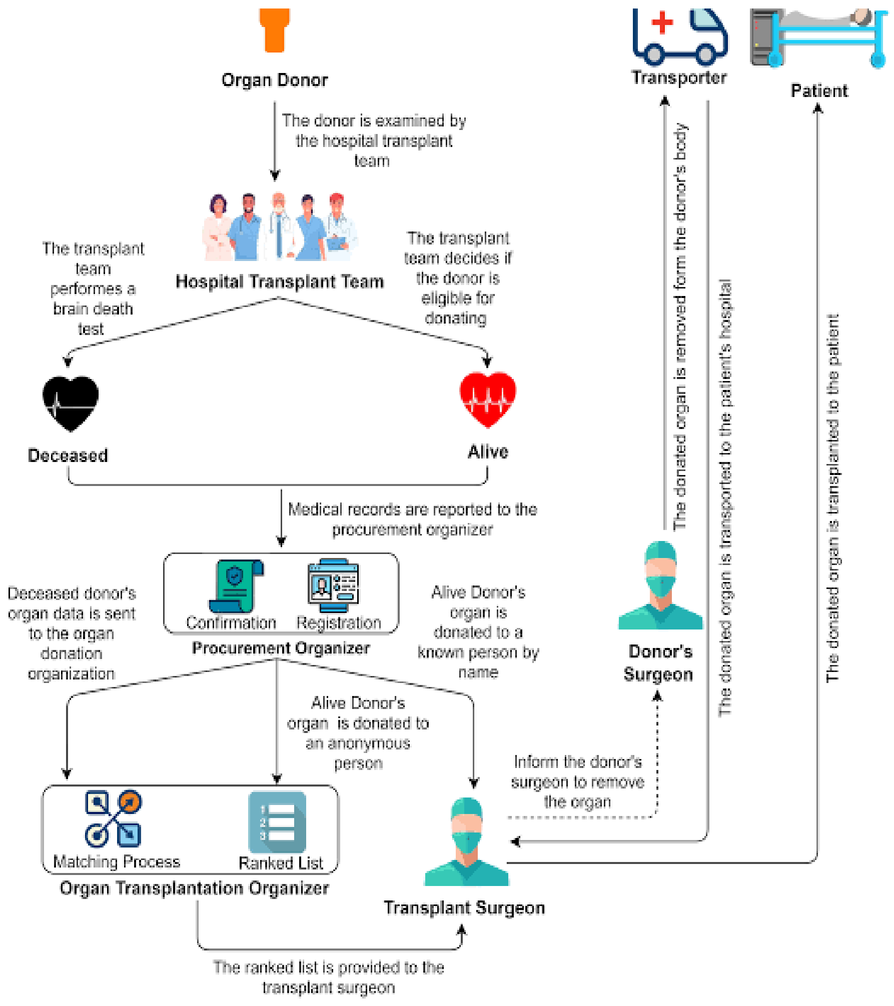
1.5 **Contribution to Knowledge**

- Secure the Cloud Health Records/ medical records of patients efficiently providing integrity
  privacy and confidentiality by storing the health records in a blockchain
- Resolve the issues of information asymmetry and data interoperability between health providers.

1.6 **Possible Challenges**

- Learning a whole new concept of blockchain technology and the implementation

- The duration of time for the completion of the project



Organ Donor
The donor is examined by the hospital transplant team
Hospital Transplant Team
The transplant team performes a brain death test
The transplant team decides if the donor is eligible for donating
Deceased
Alive
Medical records are reported to the procurement organizer
Confirmation
Registration
Procurement Organizer
Deceased donor's organ data is sent to the organ donation organization
Alive Donor's organ is donated to a known person by name
Alive Donor's organ is donated to an anonymous person
Matching Process
Ranked List
Organ Transplantation Organizer
Transplant Surgeon
The ranked list is provided to the transplant surgeon
Inform the donor's surgeon to remove the organ
Donor's Surgeon
The donated organ is removed form the donor's body
The donated organ is transported to the patient's hospital
The donated organ is transplanted to the patient
Transporter
Patient

Fig 1.6: Process of organ donation

LITERATURE SURVEY

[1]Kulshrestha, A., Mitra, A., & Amisha. (2020). Securing Organ Donation using Blockchain. International Journal of Scientific & Engineering Research, 11(6), June 2020.

In this paper, we are proposing a secure method of organ donation over a decentralized platform. This system will be implemented via a web portal that connects organ donors with organ receivers and administered by hospitals. We are trying to completely avoid third-party interference and protecting the integrity of the patient data and identification of the donated organs. This will be attained with the help of smart contracts. Smart contracts will contain the protocols that will govern our organ transaction process and facilitate smooth transactions without intermediaries.

[2]Douville, F., Godin, G., & Vézina-Im, L. A systematic review of interventions targeting health professionals and their impact on organ and tissue donation in clinical settings. This study is available under PMID: 24628967, PMCID: PMC4003858, and DOI: 10.1186/2047- 1440-3-8.

The number of patients awaiting organ or tissue transplantation continues to grow throughout the world [1-4]. The shortage of organ and tissue donors is widely studied and several factors explaining why individuals accept or refuse to consent to organ and tissue donation are reported in the literature [5]. Simpkin *et al*. [6] conducted a review of modifiable factors that influence relatives' decisions to allow organ donation. This review suggests that the skills of individuals making the request to donate may have a significant impact on consent rates. Based on this information, evaluating the efficacy of interventions among HPs to increase donation seems relevant.

[3]Abbasi, M., Kiani, M., Ahmadi, M., & Salehi, B. Perspectives on Knowledge and Ethical Issues in Organ Transplantation and Organ Donation from Iranian Health

Personnel. The authors of this study are Mahmou Abbasi, Mehrzad Kiani, Mehdi Ahmadi, and Bahare Salehi.

The transplantation of human organs in the case of the irreversible failure of an organ has been raised for a long time in scientific and social committees, and this topic has been addressed from scientific, moral, religious, political, and legal perspectives [1–4]. Organ donation options, such as the heart, lungs, kidneys, the liver, and the eyes, from a patient with brain death before cardiac arrest, are particularly important because these organs can save another patient's life. Organ transplantation is one of the most critical topics explored in medical ethics, which is presently commonplace in many countries. Owing to the prevalence of organ transplants in the world, various issues need careful attention from researchers. Organ transplantation within the study of medical ethics contains hundreds of topics.

[4]Jones, C. P., Papadopoulos, C., Randhawa, G., & Asghar, Z. A research protocol titled "General Practice Organ Donation Intervention - A Feasibility Study" (GPOD) by Catrin Pedder Jones, Chris Papadopoulos, Gurch Randhawa, and Zeeshan Asghar.

The primary intervention element, prompted choice, requires general practice to ask patients in consultations if they wish to join the NHS ODR. Two additional intervention techniques will be used to support prompted choice: staff training and leaflets and posters. The intervention will run for 3 months (April–July 2018) followed by a period of data collection. The following methods will be used to assess feasibility, acceptability and fidelity: registration data, a training evaluation survey, focus groups with staff and online surveys for staff and patients.

Introduction to Blockchain

Blockchain technology has rapidly emerged as one of the most transformative innovations of the digital age, revolutionizing the way data is stored, shared, and secured across various sectors. At its core, blockchain represents a decentralized, distributed ledger system that enables peer-topeer transactions without the need for intermediaries. This introductory section provides a foundational understanding of blockchain, exploring its conceptual framework, historical evolution, and core principles.

3.1 Definition and Conceptual Framework

Blockchain can be defined as a decentralized digital ledger that records transactions across a network of computers in a secure and transparent manner. Unlike traditional centralized databases, which rely on a single authority to validate and authenticate transactions, blockchain operates on a distributed network of nodes, where each participant maintains a copy of the entire ledger. This decentralized architecture ensures consensus among network participants and eliminates the risk of a single point of failure or manipulation.

At its most basic level, a blockchain consists of a series of blocks, each containing a bundle of transactions, linked together in a chronological order to form a continuous chain. Every block contains a cryptographic hash of the previous block, creating an immutable record of transaction history. This cryptographic linkage ensures the integrity and tamper-proof nature of the blockchain, as any alteration to a single block would necessitate the modification of all subsequent blocks, making it computationally infeasible to tamper with the data.

While the concept of blockchain gained widespread recognition with the advent of Bitcoin in 2008, its roots can be traced back to earlier cryptographic and distributed systems. The seminal whitepaper titled "Bitcoin: A Peer-to-Peer Electronic Cash System," authored by the pseudonymous Satoshi Nakamoto, introduced the foundational principles of blockchain technology as the underlying framework for the Bitcoin cryptocurrency. Since then, blockchain has evolved beyond its cryptocurrency origins to encompass a diverse range of applications across industries, including finance, supply chain, healthcare, and more.

3.2 Core Principles: Decentralization, Transparency, Immutability

Three core principles underpin the functionality and value proposition of blockchain technology:

Decentralization: Blockchain operates on a peer-to-peer network of nodes, where transactions are validated and recorded collectively by network participants, rather than relying on a central authority. This decentralized architecture enhances security, resilience, and censorship resistance, as there is no single point of control vulnerable to manipulation or attack.

Transparency: The transparent nature of blockchain allows all participants to view and verify transactional data in real-time. Each transaction is recorded on the blockchain in a transparent and immutable manner, providing a comprehensive audit trail and fostering trust among network participants.

Immutability: Once recorded on the blockchain, data becomes immutable and tamperproof, thanks to cryptographic hashing and consensus mechanisms. This immutable record ensures the integrity and reliability of the transaction history, mitigating the risk of fraud, corruption, or data manipulation.

In summary, blockchain technology represents a paradigm shift in how data is managed and transactions are conducted, offering unparalleled levels of security, transparency, and efficiency. As we delve deeper into the intricacies of blockchain technology in subsequent sections, we will explore its technical foundations, diverse applications, potential challenges, and future prospects, illuminating the transformative potential of this groundbreaking innovation.

Use Cases of Blockchain Technology

Blockchain technology has transcended its origins as the underlying framework for cryptocurrencies like Bitcoin and has found applications across a multitude of industries. Its decentralized, secure, and transparent nature makes it suitable for a wide range of use cases. Below are some prominent examples of how blockchain is being utilized:

Financial Services and Cryptocurrencies: Perhaps the most well-known application of blockchain is in the realm of digital currencies. Cryptocurrencies such as Bitcoin, Ethereum, and others leverage blockchain technology to facilitate peerto-peer transactions without the need for intermediaries like banks. Blockchain ensures the security and immutability of transactions, making it an ideal solution for digital payments, remittances, and cross-border transfers.

Supply Chain Management: Blockchain is revolutionizing supply chain management by enhancing transparency, traceability, and efficiency throughout the entire supply chain process. Companies can use blockchain to track the movement of goods from the point of origin to the end consumer, ensuring authenticity, preventing counterfeit products, and optimizing logistics. For example, food traceability solutions like IBM Food Trust enable retailers and consumers to trace the journey of food products from farm to table, improving food safety and quality.

Healthcare: Blockchain has the potential to transform the healthcare industry by securely managing and sharing electronic health records (EHRs) and other sensitive medical data. Patients can have greater control over their health information, while healthcare providers can access accurate and up-to-date patient records securely.

Blockchain-based healthcare platforms also facilitate interoperability among disparate healthcare systems, enabling seamless data exchange and improving patient care outcomes.

Identity Management: Blockchain offers a decentralized and secure solution for managing digital identities, reducing the risk of identity theft and fraud. By leveraging blockchain technology, individuals can have self-sovereign control over their digital identities, allowing them to selectively share personal information with trusted parties. Governments and enterprises can use blockchain-based identity management systems for secure authentication, KYC (Know Your Customer) processes, and identity verification.

Smart Contracts and Decentralized Applications (DApps): Ethereum, a blockchain platform with smart contract functionality, has opened up a myriad of possibilities for decentralized applications (DApps) across various industries. Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They enable automated and trustless transactions, eliminating the need for intermediaries and reducing costs. DApps built on Ethereum and other blockchain platforms span domains such as decentralized finance (DeFi), gaming, decentralized social networks, and more.

Real Estate: Blockchain technology is being used to streamline and digitize real estate transactions, including property sales, leasing agreements, and title deeds. By recording property ownership and transfer of assets on a blockchain, real estate transactions can be executed more efficiently, securely, and transparently. Blockchain-based land registry systems ensure the integrity of land records, reduce fraud, and minimize disputes over property ownership.

Voting Systems: Blockchain offers a solution to enhance the integrity and transparency of electoral processes by creating tamper-proof and auditable voting systems. Blockchain-based voting platforms enable secure and verifiable electronic voting, allowing voters to cast their ballots remotely while ensuring the confidentiality and integrity of their votes. These systems eliminate the risk of tampering, manipulation, or voter fraud, thereby enhancing trust in democratic processes.

Intellectual Property Management: Blockchain technology can be utilized to protect and manage intellectual property rights, including patents, trademarks, and copyrights. By timestamping and recording intellectual property transactions on a blockchain, creators and inventors can establish proof of ownership and enforce their rights more effectively. Blockchain-based IP management platforms also facilitate transparent and fair royalty distribution mechanisms, ensuring that creators are fairly compensated for their work.

These are just a few examples of the diverse range of use cases for blockchain technology. From finance to healthcare, supply chain management to identity verification, blockchain is disrupting traditional industries and unlocking new opportunities for innovation and efficiency.

Fig 3.2: Blockchain

3.3 Layers of Blockchain

Blockchain technology consists of several layers, each serving a specific purpose in the operation and functionality of the blockchain network. Below are the typical layers of a blockchain system:

Networking Layer:

- The networking layer establishes the communication infrastructure of the blockchain network. It enables nodes (computers or devices participating in the network) to connect and exchange data with each other.
- Peer-to-peer (P2P) networking protocols are commonly used to facilitate communication among nodes, allowing them to broadcast transactions, share blocks, and synchronize the blockchain ledger.

Consensus Layer:

- The consensus layer is responsible for achieving agreement among network participants on the validity of transactions and the state of the blockchain ledger.
- Various consensus mechanisms, such as Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), and Practical Byzantine Fault Tolerance (PBFT), are used to ensure consensus and maintain the integrity of the blockchain.
- Consensus algorithms determine how new blocks are created and added to the blockchain, as well as how conflicts or discrepancies in the network are resolved.

Transaction Layer:

o The transaction layer handles the creation, validation, and propagation oftransactions within theblockchain network.

- Transactions typically include information about the sender, receiver, amount transferred, and any additional data or metadata associated with the transaction.
- Smart contracts, which are self-executing contracts with predefined conditions written in code, are executed at this layer, enabling automated and trustless transactions.

Smart Contract Layer:

- Smart contracts, also known as the application layer or contract layer, facilitate the execution of self-executing contracts on the blockchain.

- Smart contracts are programmable scripts that automatically execute predefined actions when certain conditions are met. They enable decentralized applications (DApps) to perform functions without the need for intermediaries.
- Smart contracts are typically written in high-level programming languages such as Solidity (used in Ethereum) and deployed on blockchain platforms that support smart contract functionality.

Data Layer:

- The data layer encompasses the storage and management of data within the blockchain network.
- Blockchain technology relies on distributed ledger technology (DLT) to maintain a secure, tamper-resistant record of transactions and other relevant information.
- Blocks in the blockchain contain a collection of transactions, and each block is linked to the previous block through cryptographic hashes, forming a continuous chain of blocks.

User Interface Layer:

- The user interface (UI) layer provides interfaces and tools for users to interact with the blockchain network.
- This layer includes wallets, blockchain explorers, decentralized applications (DApps), and other user-facing applications that enable users to view transaction history, send and receive digital assets, interact with smart contracts, and access blockchain-based services.

These layers collectively form the foundational infrastructure of a blockchain system, enabling secure, decentralized, and transparent transaction processing and data management. Each layer plays a crucial role in ensuring the integrity, efficiency, and functionality of the blockchain network.

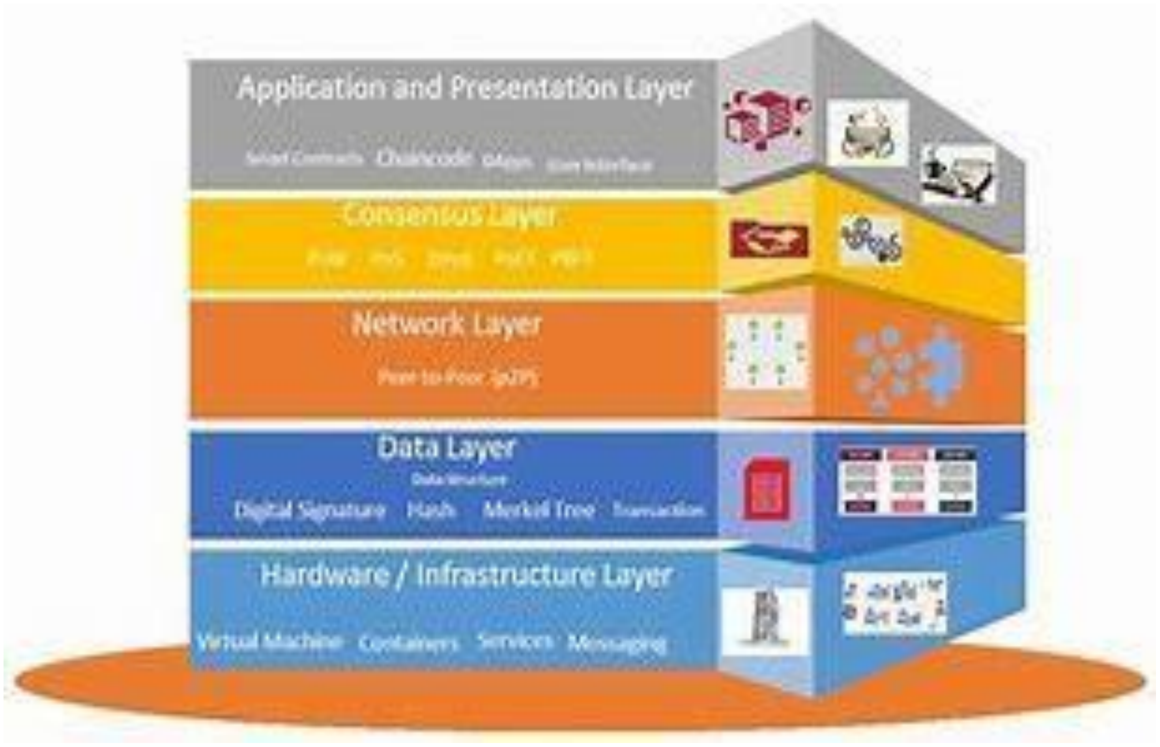


Fig 3.3: Layers in Blockchain

3.4 Types of Blockchain

Blockchain technology has evolved since its inception, leading to the development of different types of blockchains tailored to specific use cases, governance models, and levels of decentralization. Here are the main types of blockchains:

Public Blockchains:

- Public blockchains are open networks where anyone can participate, transact, and validate transactions without permission. Examples include Bitcoin and Ethereum.
- Participants in public blockchains retain full control over their data and assets, and transactions are transparent and immutable.
- Public blockchains rely on decentralized consensus mechanisms, such as Proof of Work (PoW) or Proof of Stake (PoS), to validate and confirm transactions.

Private Blockchains:

- Private blockchains are permissioned networks where access to participate, transact, and validate transactions is restricted to authorized entities.
- In private blockchains, participants are known, and identities are verified, allowing for greater privacy, confidentiality, and control over data.
- Private blockchains are often used by enterprises, governments, and consortia to build internal systems for specific use cases, such as supply chain management or financial transactions.

Consortium Blockchains:

- Consortium blockchains are a hybrid model that combines elements of both public and private blockchains.
- In consortium blockchains, a predefined group of participants collaborates to operate and maintain the network, sharing control over governance and decision-making.
- Consortium blockchains offer increased scalability, privacy, and efficiency compared to public blockchains, while still retaining some degree of decentralization and trustlessness.

Permissioned Blockchains:

- Permissioned blockchains require participants to obtain explicit permission or authorization to join and interact with the network.
- Permissioned blockchains are often used in enterprise settings where strict access control, regulatory compliance, and privacy requirements are paramount.
- Participants in permissioned blockchains are typically known entities, such as businesses, institutions, or government agencies, who operate nodes and validate transactions.

Hybrid Blockchains:

- Hybrid blockchains combine elements of both public and private blockchains, allowing for flexibility and customization to meet specific use case requirements.
- In hybrid blockchains, certain aspects of the network may be public and open to anyone, while other aspects are private and restricted to authorized participants.

- Hybrid blockchains enable organizations to leverage the benefits of both public and private blockchains, such as transparency, security, scalability, and privacy, depending on the context of the application.

Permissionless Blockchains:

- Permissionless blockchains, also known as open or public blockchains, allow anyone to participate in the network without requiring permission.
- Participants in permissionless blockchains can join, transact, and validate transactions anonymously, without the need for identity verification or authorization.
- Permissionless blockchains prioritize decentralization, censorship resistance, and inclusivity, enabling anyone to contribute to the network's security and consensus process.

These types of blockchains cater to different use cases, governance models, and levels of decentralization, offering flexibility and versatility to developers, enterprises, and organizations seeking to leverage blockchain technology for various applications. The choice of blockchain type depends on factors such as security requirements, data privacy, scalability, regulatory compliance, and the specific needs of the intended use case.

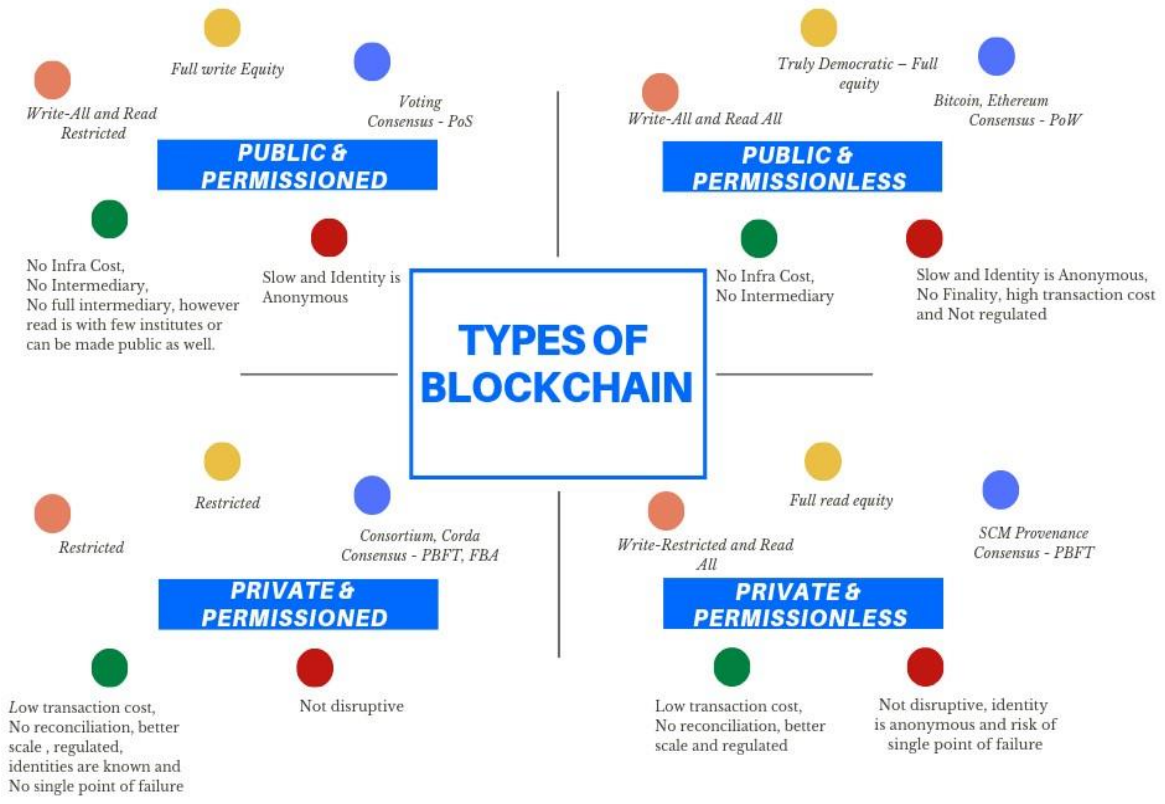

TYPES OF BLOCKCHAIN
PUBLIC & PERMISSIONED
Write-All and Read Restricted
Full write Equity
Voting Consensus - PoS
No Infra Cost, No Intermediary, No full intermediary, however read is with few institutes or can be made public as well.
Slow and Identity is Anonymous
PUBLIC & PERMISSIONLESS
Write-All and Read All
Truly Democratic – Full equity
Bitcoin, Ethereum Consensus - PoW
No Infra Cost, No Intermediary
Slow and Identity is Anonymous, No Finality, high transaction cost and Not regulated
PRIVATE & PERMISSIONED
Restricted
Restricted
Consortium, Corda Consensus - PBFT, FBA
Low transaction cost, No reconciliation, better scale , regulated, identities are known and No single point of failure
Not disruptive
PRIVATE & PERMISSIONLESS
Write-Restricted and Read All
Full read equity
SCM Provenance Consensus - PBFT
Low transaction cost, No reconciliation, better scale and regulated
Not disruptive, identity is anonymous and risk of single point of failure

Fig 3.4: Types of Blockchain

PROPOSED METHOD

4.1 EXISTING SYSTEM

Cloud computing has emerged as a new enterprise IT architecture. However, privacy concern has remained a primary barrier pre-venting the adoption of cloud computing by a broader range of users/applications. When sensitive data are outsourced to the cloud, data owners naturally become concerned with the privacy of their data in the cloud and beyond. However, how encrypted data can be effectively utilized becomes another new challenge.

Symmetric cryptography-based schemes are clearly not suitable for this setting due to the high complexity of secret key management. Although authorized keyword search can be realized in single-owner setting by explicitly defining a server-enforced user list that takes the responsibility to control legitimate users' search capabilities, i.e. search can only be carried out by the server with the assistance of legitimate users' complementary keys on the user list, these schemes did not realize fine-grained owner-enforced search authorization and thus are unable to provide differentiated access privileges for different users within a dataset. Asymmetric cryptography is better suited to this dynamic setting by encrypting individual contributions with different public keys.

4.2 SHORTCOMINGS OF THE CURRENT SYSTEM:

- Symmetric cryptography-based schemes are clearly not suitable for this setting due to the high complexity of secret key management.
- These schemes did not realize fine-grained owner-enforced search authorization and thus are unable to provide differentiated access privileges for different users within a dataset.
- Privacy risk during file sharing.

4.3 PROPOSED SYSTEM

Blockchain technology is a game-changer with the potential to impact not one or two industries, but the complete landscape of how business is done. When 200 healthcare executives were surveyed, 16 percent expected to have a commercial blockchain solution at scale sometime this year. The key players for blockchain adoption will be regulators, industry groups and market makers. Managing and securing data within healthcare and supply chain management are two great examples of principal concepts influencing and being impacted by possible blockchain adoption. Let's take a brief look at each one:

- **Healthcare:** Better data sharing between healthcare providers means a higher probability of accurate diagnoses, more effective treatments, and the overall increased abilityof healthcare organizations to deliver cost-effective care. Blockchain technology can allow various stakeholders in the healthcare value-chain to share access to their networks without compromising data security and integrity, by allowing them to track data provenance as well as any changes made.
- **Health Chain Management:** One of the most Merkle Hash Tree applicable aspects of blockchain technology is that it enables more secure and transparent monitoring of transactions. With blockchain, the transactions can be documented in a permanent decentralized record reducing time delays, added costs and human errors.

4.4 HEALTHCARE INDUSTRY ON SECURITY

In the current system, security and trust are the most common concerns shared by businesses regarding the information shared between different entities. Information can be entered anywhere along the line of communication, and this leads to trust issues, especially in the healthcare industry. There are also concerns where multiple vendors hold different versions of the same patient record that are not validated, resulting in various errors, inconsistency and incompleteness. Add to that reports of security breaches, tampering of personal data and the ever-present hacking threat, it's not surprising healthcare officials are concerned.

Since blockchain are cryptographically secure and the data present there can be authenticated using digital signatures that are unique to each person, this technology could be the answer to most of these concerns.

IMPLEMENTATION

We will use Ethereum blockchain to save student data/certificate. For that we need write Smart Contract that is an interface to connect on blockchain.

5.1 SMART CONTRACTS

Solidity is a language used for smart contracts on the Ethereum blockchain and it is a set of code and data that have permanent address on the Ethereum blockchain. In Object Oriented Programming language, it is similar to class where it includes state variables & functions. Smart Contracts and blockchain are the basis of all Decentralized Applications. Contracts and blockchain have immutable and distributed feature as common feature. If they are on blockchain then it will be painful to upgrades contracts.

State Variables-variables that hold values that are permanently stored on the Blockchain. We will use state variables to hold Student name, Course detail, Certificate number and validity date.

Functions-Functions are the executables of smart contracts. They are what we will call interacting with the Blockchain, and they have different levels of visibility, internally and externally. Keep in mind that whenever you want to change the value/state of a variable, a transaction must occur-costing Ether.

Events-Whenever an event is called, the value passed into the event will be logged in the transaction's log. This allows JavaScript callback functions or resolved promises to view the certain value you wanted to pass back after a transaction. This is because every time you make a transaction, a transaction log will be returned. We will use an event to log the ID of the newly created Candidate, which we‟ll display.
Smart contracts are self-executing contracts with predefined conditions written in code. They are designed to automatically enforce and execute the terms of an agreement when

certain conditions are met, without the need for intermediaries or manual intervention. Smart contracts run on blockchain platforms that support smart contract functionality, such as Ethereum, Binance Smart Chain, and others.

Here's a closer look at smart contracts and how they work:

Code-Based Contracts: Smart contracts are written in programming languages specifically designed for blockchain platforms, such as Solidity for Ethereum. These contracts encode the rules and conditions of an agreement into executable code.

Decentralized Execution: Smart contracts operate in a decentralized manner, running on every node of the blockchain network. Once deployed to the blockchain, smart contracts are immutable and tamper-proof, ensuring that the contract's logic cannot be altered or manipulated.

Automated Execution: Smart contracts automatically execute predefined actions when specific conditions are met. For example, in a simple escrow smart contract, funds are released to the seller when the buyer confirms receipt of goods, eliminating the need for a trusted intermediary to facilitate the transaction.

Trustless Transactions: Smart contracts enable trustless transactions, as the execution of the contract's terms is enforced by the underlying blockchain protocol. Participants can interact with smart contracts directly, knowing that the outcome will be executed as programmed without the need to trust a central authority.

Transparency and Immutability: Smart contracts leverage the transparency and immutability of blockchain technology. The code and execution of smart contracts are visible to all participants on the blockchain, providing transparency and auditability of contract execution. Once deployed, smart contracts cannot be altered or tampered with, ensuring the integrity of the contract's execution.

Diverse Use Cases: Smart contracts have a wide range of applications across various industries and use cases, including decentralized finance (DeFi), supply chain management, voting systems, digital identity, insurance, and more. They enable the automation of complex processes and the creation of decentralized applications (DApps) that operate without centralized control.

Gas Fees: In blockchain platforms like Ethereum, smart contract execution requires payment of transaction fees, known as gas fees. Gas fees compensate network validators (miners or stakers) for the computational resources required to execute smart contracts. The cost of executing smart contracts depends on factors such as the complexity of the contract and network congestion.

Overall, smart contracts are a powerful tool that enables the automation, transparency, and trustless execution of agreements and transactions on blockchain networks. They represent a fundamental building block of decentralized applications and are driving innovation across industries by streamlining processesand eliminating intermediaries.

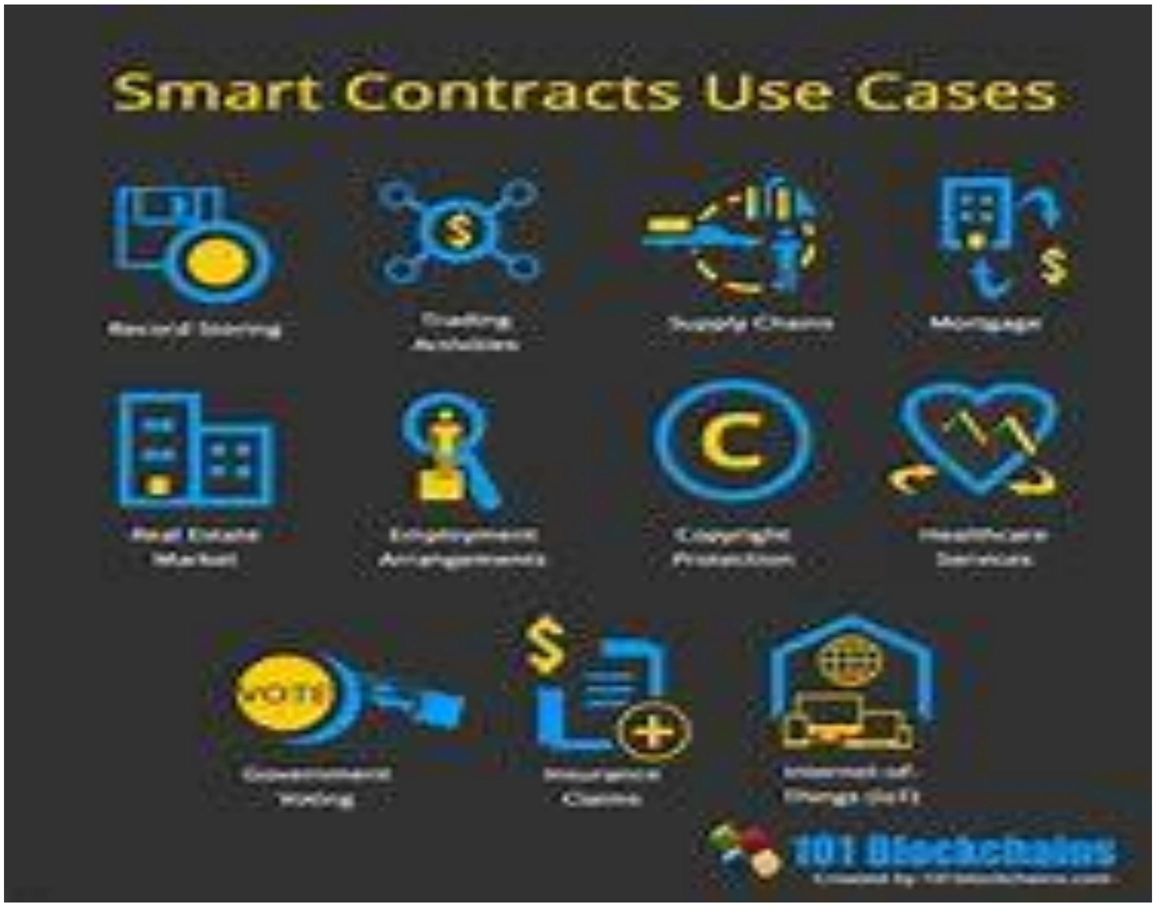


Fig 5.1: Smart Contracts

5.2 THE ETHEREUM VIRTUAL MACHINE(EVM) :

Ethereum Virtual Machines is implemented in C++, Go, Haskell, Java, JavaScript, and Python. It is the runtime environment for smart contracts in Ethereum. It handles the internal state and computation of the entire Ethereum Network

The Ethereum Virtual Machine (EVM) is a crucial component of the Ethereum blockchain platform, responsible for executing smart contracts and running decentralized applications (DApps) on the network. Here's a detailed overview of the Ethereum Virtual Machine:

- **Execution Environment**: The Ethereum Virtual Machine provides a runtime environment for executing smart contracts written in high-level programming languages, primarily Solidity. It is a sandboxed environment, meaning that code execution is isolated from the underlying blockchain and other smart contracts.
- **Turing-Complete**: The EVM is Turing-complete, meaning it can execute any algorithm or computation that a traditional computer can perform. This property enables developers to build complex decentralized applications and smart contracts with arbitrary logic and functionality.
- **Deterministic Execution**: The EVM ensures deterministic execution of smart contracts, meaning that given the same input and state, the outcome of a contract's execution will always be the same. This deterministic behavior is critical for achieving consensus among all nodes in the Ethereum network.
- **Gas Mechanism**: Smart contract execution on the EVM requires payment of gas, which is a measure of computational resources consumed during execution. Gas is used to compensate network validators (miners or stakers) for processing transactions and executing smart contracts. Gas limits prevent infinite loops and denial-of-service attacks by constraining the computational resources that a contract can consume.
- **Opcode Instructions**: The EVM operates using a set of low-level instructions called opcodes. These opcodes define the operations that the EVM can perform, such as arithmetic operations, memory manipulation, storage access, and control flow instructions. Smart contracts are compiled into bytecode, a series of EVM bytecode instructions that the EVM can execute.
- **State Transition Function**: The EVM implements a state transition function that updates the state of the Ethereum blockchain based on the execution of smart contracts. This function defines how transactions modify the account balances, contract storage, and other state variables on the blockchain.
- **Decentralized Applications (DApps)**: The EVM enables the development and execution of decentralized applications (DApps) on the Ethereum blockchain.

DApps leverage smart contracts to implement business logic and interact with users, enabling a wide range of use cases, including decentralized finance (DeFi), decentralized exchanges (DEXs), gaming, governance, and more.

- **Cross-Chain Compatibility**: While the EVM is primarily associated with the Ethereum blockchain, its design and architecture have inspired the development of other blockchain platforms and virtual machines, such as Binance Smart Chain (BSC), which is EVM-compatible. This compatibility allows developers to port existing Ethereum smart contracts and DApps to other EVM-compatible chains with minimal modifications.

Overall, the Ethereum Virtual Machine plays a central role in the Ethereum ecosystem, enabling the execution of smart contracts and powering the decentralized applications that drive innovation and value creation on the blockchain.

The Ethereum Virtual Machine (EVM) serves as the backbone of the Ethereum blockchain platform, enabling a wide range of use cases and applications. Here are some of the primary uses of the EVM:

Smart Contract Execution: The EVM is primarily used for executing smart contracts, which are self-executing contracts with predefined conditions written in code. Smart contracts enable the automation of agreements and transactions on the Ethereum blockchain without the need for intermediaries. They are used in various industries and use cases, including decentralized finance (DeFi), supply chain management, digital identity, decentralized exchanges (DEXs), gaming, and more.

Decentralized Applications (DApps): DApps are applications built on blockchain platforms, such as Ethereum, that leverage smart contracts to implement business logic and interact with users. The EVM provides the runtime environment for executing these DApps, enabling developers to build decentralized, censorship-resistant applications with transparent and auditable code execution.

Token Standards: The EVM has played a significant role in the proliferation of tokenization on the Ethereum blockchain. Standards such as ERC-20 (fungible tokens) and ERC-721 (non-fungible tokens or NFTs) define common interfaces and conventions for the creation and management of tokens on Ethereum. These token standards have fueled the growth of token economies, enabling crowdfunding, tokenized assets, digital collectibles, and more.

Decentralized Finance (DeFi): DeFi refers to a broad category of financial services and applications built on blockchain platforms, primarily Ethereum. The EVM powers a wide range of DeFi protocols and applications, including decentralized exchanges (DEXs), lending and borrowing platforms, yield farming, liquidity pools, automated market makers (AMMs), derivatives, stablecoins, and asset management protocols.

Governance and DAOs: The EVM facilitates the creation and operation of decentralized autonomous organizations (DAOs), which are self-governing entities governed by smart contracts and token holders. DAOs use smart contracts to implement governance mechanisms, voting systems, and decision-making processes, enabling decentralized governance and community-driven initiatives.

Blockchain Oracles: Oracles are services that provide external data to smart contracts, enabling them to interact with off-chain resources, such as web APIs, IoT devices, and external data feeds. The EVM is used to execute oracle contracts that fetch and verify external data, enabling smart contracts to make informed decisions based on real-world events and data sources.

Layer 2 Scaling Solutions: Layer 2 scaling solutions, such as state channels, sidechains, and rollups, aim to improve the scalability and throughput of the Ethereum network by processing transactions off-chain or aggregating them before submitting to the main Ethereum chain. These solutions leverage the EVM to execute smart contracts and settle transactions securely and efficiently while reducing congestion and gas fees on the main chain.

Interoperability and Cross-Chain Bridges: The EVM serves as a common interoperability standard for blockchain platforms and virtual machines. Bridges and interoperability protocols allow assets and data to move seamlessly between different blockchain networks that support the EVM, enabling cross-chain communication and interoperability of decentralized applications.

Overall, the Ethereum Virtual Machine (EVM) is a versatile and powerful tool that enables the execution of smart contracts, the development of decentralized applications (DApps), and the creation of innovative blockchain solutions across a wide range of industries and use cases.

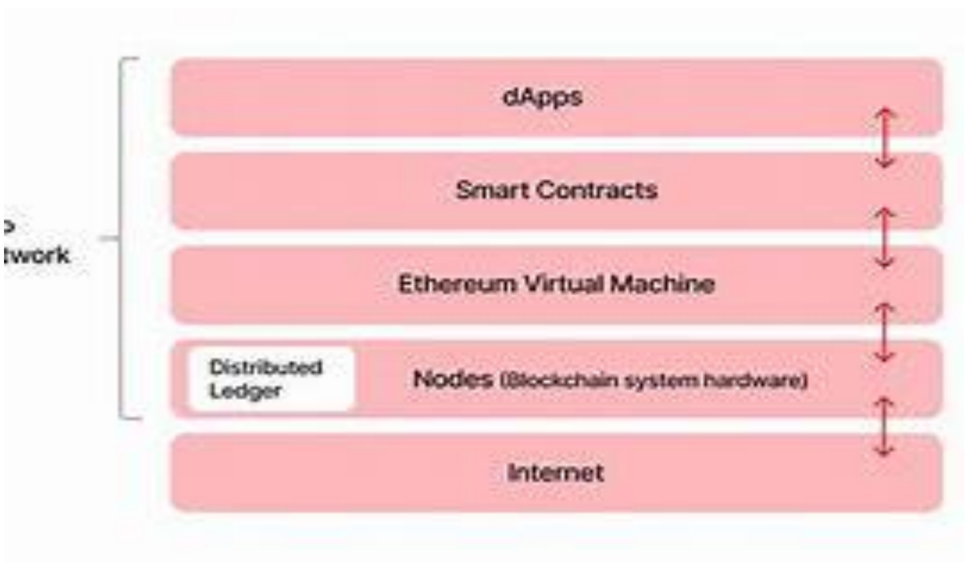


Fig 5.2: The Ethereum Virtual Machine

.5.3 GAS:

Gas is the internal pricing that we have to pay for running a transaction or contract in Ethereum blockchain**.** A certain number of gas occurred whenever there is an operation performed by transaction or contract on the Ethereum platform.

Any computer code (complex or short) can be run inside EVM, A short code can result in more computation work as compare to complex code. It means that short code ode not guarantee less computation work. Gas depends upon the calculation done inside the EVM; our focus should be on less computation work that will result in less amount of Gas. Its payment is charged as a certain number of ether. The transaction fee is Transaction fee is combination of total gas used multiplied by gas price.

We will also use below tools:

Web3.js is a JavaScript API and with the help of this API We can interact with the Blockchain - making transactions and calls to smart contracts. Developer can focus on the content of their application as this API abstracts the communication with Ethereum Clients.

Truffle is a testing development framework for Ethereum. It includes a development blockchain, compilation and migration scripts to deploy your contract to the Blockchain, contract testing, and so on. It makes development easier.

Gas in blockchain refers to a unit of measurement representing the computational effort required to execute operations or transactions on a blockchain network, particularly in systems like Ethereum. It serves as a measure of computational work and resource consumption, including CPU time, memory usage, and storage access. Gas is an essential concept in blockchain systems, especially those that

Supportsmart contracts, as it helpsensure the efficient and secure operation of the network. Here's a deeper look into gas in blockchain:

- **Transaction Fees**: In blockchain networks like Ethereum, every operation or transaction requires a certain amount of gas to be executed. Gas acts as a mechanism for fee estimation and payment, ensuring that participants compensate network validators (miners or stakers) for the computational resources consumed during transaction processing and smart contract execution.

Gas Limit and Gas Price: When sending a transaction or invoking a smart contracton Ethereum, users specify two parameters: gas limit and gas price. The gas limitdefines maximum amount of gas that can be consumed by the transaction orensure the efficientand secure operation of the network. Here's a deeper look intogas in blockchain:

Transaction Fees: In blockchain networks like Ethereum, every operation or transaction requires a certain amount of gas to be executed. Gas acts as a mechanism for fee estimation and payment, ensuring that participants compensate network validators (miners or stakers) for the computational resources consumed during transaction processing and smart contract execution.

Gas Limit and Gas Price: When sending a transaction or invoking a smart contracton Ethereum, users specify two parameters: gas limit and gas price. The gas limit defines the maximum amount of gas that can be consumed by the transaction.

Dynamic Gas Pricing: Gas prices are determined by supply and demand dynamics in the Ethereum network. Users can adjust gas prices based on network congestion and transaction priority, with higher gas prices incentivizing miners to prioritize their transactions. Gas prices can fluctuate based on network activity, congestion, and market conditions.

In summary, gas in blockchain serves as a crucial mechanism for fee estimation, resource management, and preventing network abuse. It ensures the efficient and secure operation of blockchain networks by aligning incentives, optimizing resource allocation, and enabling fair compensation for computational work performed by network validators.

Gas in blockchain networks, particularly in systems like Ethereum, serves several important purposes, primarily related to transaction execution and smart contract operations. Here are the key uses of gas in blockchain:

Transaction Execution Fees: Gas is used to determine the fee required for executing transactions on the blockchain. Every operation within a transaction, such as sending tokens, interacting with smart contracts, or updating data on the blockchain, consumes gas. The sender of the transaction specifies the gas limit (the maximum amount of gas that can be consumed) and the gas price (the amount of cryptocurrency paid per unit of gas). The total transaction fee is calculated as the product of gas consumed and the gas price. This fee is paid to miners or validators as compensation for including the transaction in a block and processing it on the network.

- **Smart Contract Execution**: Gas is essential for executing smart contracts on blockchain platforms. Smart contracts are self-executing contracts with predefined conditions written in code. Each operation or instruction within a smart contract consumes gas, including computations, storage operations, and external calls to other contracts. Gas ensures that the execution of smart contracts is bounded by computational resources and prevents infinite loops or excessive resource consumption. Smart contract developers need to optimize their code to minimize gas usage and ensure efficient execution.

- **Preventing Denial-of-Service (DoS) Attacks**: Gas helps prevent denial-of-service (DoS) attacks and spamming on the blockchain by imposing a cost for computational resources consumed during transaction processing and contract execution. The gas limit specifies the maximum amount of gas that can be used in a transaction or contract execution, preventing malicious actors from executing resource-intensive operations that could overload the network or disrupt its operation. Gas ensures that users pay for the resources they consume, discouraging abuse and incentivizing efficient use of the blockchain network.

- **Resource Management**: Gas acts as a measure of computational work and resource consumption on the blockchain. It helps manage the allocation of resources, such as CPU time, memory, and storage, by imposing limits on the amount of gas that can be consumed by transactions and smart contracts. Gas ensures that the blockchain network operates efficiently and fairly, providing a level playing field for all participants and preventing monopolization of network resources.

- **Dynamic Pricing Mechanism**: Gas prices are determined by supply and demand dynamics in the blockchain network. Users can adjust gas prices based on network congestion, transaction priority, and market conditions. Higher gas prices incentivize miners or validators to prioritize transactions with higher fees, leading to faster transaction confirmation and inclusion in blocks. Gas prices fluctuate based on network activity, congestion, and user preferences, providing a dynamic pricing mechanism that ensures efficient transaction processing and resource allocation.

Overall, gas plays a critical role in blockchain networks by facilitating transaction execution, smart contract operations, preventing network abuse, managing resources, and providing a dynamic pricing mechanism that balances supply and demand for computational resource .

5.4 MODULES USED

Data security Module

The Medical service provider must make sure that their data outsourced to the cloud is secure and the provider has to take security measures for securing the cloud data.

Privacy Data Module

The Online Medical service provider must make sure that all the critical data are Block chained and that only legal users have access to data in its entity. The digital identities and credentials must be secured as any data that the provider gathers the Patient activity in the cloud.

Data confidentiality Module

The cloud admin users ensure that their data contents are not made accessible or else disclosed to illegitimate doctor users. Only the Admin users can access the sensitive data even though the others should not access any data in the Medical cloud.

Fine-grained access control Module

Data owners Patients can restrict unauthorized users for accessing the data outsourced to the storage space of Blockchain Cloud. The data owner grants numerous access rights for the set of the user to access the data, while the others are not permitted to enable data access without permissions.

CODING ENVIORMENT

6.1 VS CODE

Visual Studio Code (VS Code) is a popular source-code editor developed by Microsoft. It's known for its versatility, extensibility, and robust feature set, making it a preferred choice among developers for various programming languages and technologies. Here's an overview of VS Code and its key features:

- **Cross-Platform**: VS Code is available for Windows, macOS, and Linux, making it accessible to developers using different operating systems. Its consistent user experience across platforms ensures a seamless workflow for developers working on multiple environments.

- **Intuitive User Interface**: VS Code features a clean and customizable user interface, with a minimalistic design that prioritizes productivity. It provides a variety of themes, color schemes, and layout options to suit individual preferences and workflows.

- **Code Editing**: VS Code offers powerful code editing capabilities, including syntax highlighting, IntelliSense (code completion), code navigation, and refactoring tools. It supports a wide range of programming languages out of the box and can be extended with additional language support through extensions.

- **Integrated Terminal**: VS Code includes an integrated terminal that allows developers to run command-line tools, execute scripts, and perform various tasks directly within the editor. The terminal supports multiple shells, such as PowerShell, Bash, and Command Prompt, and can be customized to meet specific requirements.

- **Extensions Marketplace**: One of the standout features of VS Code is its rich ecosystem of extensions. The Visual Studio Code Marketplace offers thousands of extensions developed by the community to enhance functionality, add new features, and integrate with third-party services. Extensions cover a wide range of categories, including language support, debugging, version control, productivity tools, and more

- **Version Control Integration**: VS Code provides built-in support for version control systems such as Git, enabling developers to manage code repositories, perform version control operations (e.g., commit, push, pull), and collaborate with team members directly within the editor. The Git integration includes features like visual diffing, branch management, and conflict resolution.

- **Debugging Tools**: VS Code offers comprehensive debugging tools for various programming languages and frameworks. It provides debuggers with breakpoints, watch variables, call stacks, and other debugging features, allowing developers to troubleshoot and diagnose issues in their code effectively.

- **Task Automation**: VS Code includes built-in task automation features that enable developers to define and execute tasks such as build scripts, test suites, and deployment workflows directly within the editor. Tasks can be configured using task runners like npm, Gulp, and Grunt, providing seamless integration with project workflows.

- **Integrations and Collaboration**: VS Code integrates seamlessly with other development tools and services, such as code repositories (GitHub, GitLab), issue trackers (Jira, Trello), and collaboration platforms (Slack, Microsoft Teams). It also supports Live Share, a real-time collaboration feature that allows multiple developers to work together on the same codebase simultaneously.

Overall, Visual Studio Code is a powerful and versatile code editor that caters to the needs of developers across different programming languages, platforms, and project types. Its rich feature set, extensibility, and vibrant community make it an essential tool for modern software development workflows.

Truffle is a popular development framework for Ethereum that simplifies the process of building, testing, and deploying smart contracts. Visual Studio Code (VS Code) provides excellent support for Truffle development through various extensions and integrations, enhancing the development experience for Ethereum developers. Here's how you can use Truffle in VS Code:

Truffle Suite Extension:

- The Truffle Suite extension for VS Code provides a suite of tools and features specifically designed for Truffle development. You can install the extension directly from the VS Code Marketplace.
- The Truffle Suite extension offers features such as project scaffolding, smart contract compilation, deployment management, and debugging support.
- It also includes integration with Ganache, a local Ethereum blockchain for testing and development, allowing you to easily launch and manage test networks from within VS Code.

Syntax Highlighting and IntelliSense:

- VS Code provides syntax highlighting and IntelliSense support for Solidity, the programming language used for writing smart contracts. This helps developers write Solidity code more efficiently by providing context-aware suggestions and auto-completion for keywords, variables, and functions.

- The Truffle Suite extension enhances Solidity support in VS Code, offering advanced syntax highlighting and IntelliSense features tailored specifically for Truffle projects.

Debugging with Truffle Debugger:

- VS Code integrates with the Truffle Debugger, allowing developers to debug smart contracts directly within the editor. You can set breakpoints, inspect variables, step through code execution, and diagnose issues in your smart contracts using the Truffle Debugger.
- The Truffle Suite extension provides seamless integration with the Truffle Debugger, enabling a streamlined debugging experience for Ethereum developers.

Task Automation and Scripting:

- VS Code's task automation features enable you to define and execute custom tasks and scripts for Truffle projects. You can configure tasks for common operations such as compiling smart contracts, running tests, and deploying contracts to Ethereum networks.
- The Truffle Suite extension enhances task automation in VS Code by providing predefined tasks for Truffle commands, making it easy to execute common development tasks without leaving the editor.

Extension Ecosystem:

- VS Code's extensive extension ecosystem includes a variety of extensions that complement Truffle development. You can find extensions for code linting, code formatting, Solidity code analysis, Ethereum network integration, and more.
- These extensions enhance the functionality of VS Code for Truffle development, providing additional features and tools to streamline the development workflow.

By leveraging the Truffle Suite extension and other VS Code features, Ethereum developers can benefit from a seamless and productive development experience for building decentralized applications (DApps) and smart contracts on the Ethereum blockchain.

Truffle is a comprehensive development framework for Ethereum that offers a suite of tools and utilities to streamline the process of building, testing, and deploying smart contracts and decentralized applications (DApps) on the Ethereum blockchain. Here are some key use cases and benefits of using Truffle in blockchain development:

Smart Contract Development:

- Truffle provides a structured environment for developing smart contracts using the Solidity programming language. Its project scaffolding feature allows developers to quickly set up new projects with predefined directory structures, configuration files, and sample contracts.
- Truffle's built-in compiler simplifies the process of compiling Solidity smart contracts into bytecode, which can then be deployed to the Ethereum blockchain.

Testing:

- Truffle includes a powerful testing framework that enables developers to write and execute automated tests for their smart contracts. Tests can verify the behavior and functionality of smart contracts under different conditions, ensuring reliability and robustness.
- Truffle's testing framework supports various testing methodologies, including unit tests, integration tests, and end-to-end tests. Developers can write tests using JavaScript or Solidity and run them using the Truffle CLI or integrated testing tools.

Deployment:

- Truffle streamlines the process of deploying smart contracts to Ethereum networks, whether it's a local development network or a public testnet/mainnet. Developers can define deployment scripts and configurations to deploy contracts with specific parameters and settings.
- Truffle's deployment process ensures that smart contracts are deployed securely and efficiently, with options for specifying gas limits, transaction parameters, and network configurations.

Network Management:

- Truffle simplifies network management by providing built-in support for connecting to different Ethereum networks, including local networks (e.g., Ganache), testnets (e.g., Ropsten, Rinkeby), and the Ethereum mainnet.
- Developers can configure and switch between different network environments seamlessly, allowing them to test and deploy smart contracts across multiple networks with ease.

Debugging:

- Truffle offers debugging tools that enable developers to diagnose and troubleshoot issues in their smart contracts. The Truffle Debugger provides features such as breakpoints, step-through execution, variable inspection, and stack tracing, making it easier to identify and resolve bugs in smart contract code.

Interoperability and Integration:

- Truffle integrates seamlessly with other development tools and frameworks commonly used in Ethereum development, such as web3.js, ethers.js, and Metamask. This interoperability allows developers to leverage existing tools and libraries to enhance their development workflow.

- Truffle also integrates with popular IDEs and editors like Visual Studio Code, providing additional features and extensions for Ethereum development.

Overall, Truffle serves as a valuable toolkit for Ethereum developers, offering a unified and efficient development environment for building decentralized applications and smart contracts on the Ethereum blockchain. Its comprehensive features, robust testing framework, and seamless integration with Ethereum networks make it an essential tool in the blockchain developer's toolkit.

6.2 GANACHE

Ganache is a popular and widely used personal blockchain for Ethereum development. It provides developers with a local Ethereum blockchain environment that can be used for testing, debugging, and deploying smart contracts without incurring the costs and complexities associated with deploying to the Ethereum mainnet or testnets. Here are some key features and use cases of Ganache:

- **Local Ethereum Blockchain**: Ganache creates a local, in-memory Ethereum blockchain that runs entirely on your local machine. This allows developers to interact with a blockchain environment that behaves similarly to the Ethereum mainnet or testnets but without the need for internet connectivity or external dependencies.

- **Quick Setup and Configuration**: Ganache offers a simple and user-friendly interface for setting up and configuring a local blockchain environment. Developers can customize various parameters such as the number of accounts, gas limits, block time, and network ID to simulate different network conditions and scenarios.

- **Accounts and Addresses**: Ganache generates a set of Ethereum accounts with associated private keys and addresses that developers can use for testing and development purposes. These accounts come preloaded with test Ether (fake ETH) for simulating transactions and interactions on the blockchain.

- **Transaction Simulation**: Ganache simulates Ethereum transactions, including contract deployments, token transfers, and contract interactions, allowing developers to test their smart contracts and DApps in a controlled environment. Developers can inspect transaction details, gas usage, and transaction receipts for debugging purposes.

- **Blockchain Explorer**: Ganache includes a built-in blockchain explorer that provides a visual interface for viewing blocks, transactions, and account balances on the local blockchain. Developers can use the blockchain explorer to monitor blockchain activity, track transactions, and inspect smart contract state changes.

- **Integration with Development Tools**: Ganache integrates seamlessly with popular Ethereum development tools and frameworks such as Truffle, Remix, and MetaMask. Developers can easily connect their development environment to Ganache and use it as the backend blockchain for compiling, deploying, and testing smart contracts and DApps.

- **Testing and Debugging**: Ganache is widely used for testing and debugging smart contracts and decentralized applications. Developers can write automated tests using frameworks like Truffle and run them against the Ganache blockchain to ensure the correctness and reliability of their code before deploying it to production networks.

- **Educational and Learning Purposes**: Ganache is often used for educational and learning purposes, allowing developers to experiment with Ethereum development concepts, practice writing smart contracts, and explore blockchain technology in a safe and controlled environment.

Overall, Ganache is a valuable tool for Ethereum developers, providing a lightweight, customizable, and feature-rich blockchain environment for testing, debugging, and deploying smart contracts and decentralized applications. Its ease of use, flexibility, and integration capabilities make it an essential component of the Ethereum development toolkit.

Ganache, as a personal blockchain for Ethereum development, offers various use cases and benefits for blockchain developers. Here are some of the key uses of Ganache in blockchain development:

- **Local Development Environment**: Ganache provides a local Ethereum blockchain environment that runs on your machine, allowing developers to build and test their smart contracts and decentralized applications (DApps) without needing to deploy to public testnets or the Ethereum mainnet. This speeds up the development cycle and reduces reliance on external networks, making it easier to iterate and experiment with blockchain applications.

- **Smart Contract Development**: Ganache is commonly used for smart contract development, allowing developers to write, compile, and deploy smart contracts to the local blockchain environment. This enables rapid prototyping and testing of smart contract logic, ensuring that contracts function as intended before deployment to production networks.

- **Testing and Debugging**: Ganache is a powerful tool for testing and debugging smart contracts and DApps. Developers can simulate various scenarios and edge cases, execute transactions, and inspect blockchain state changes in real-time using Ganache's built-in blockchain explorer. This facilitates thorough testing and debugging of smart contract code to identify and resolve issues early in the development process.

- **Integration with Development Tools**: Ganache seamlessly integrates with popular Ethereum development tools and frameworks such as Truffle, Remix, and MetaMask. Developers can easily connect their development environment to Ganache and use it as the backend blockchain for compiling, deploying, and testing smart contracts and DApps. This streamlines the development workflow and enhances productivity by leveraging familiar tools and workflows.

- **Contract Interactions and Transactions**: Ganache enables developers to interact with smart contracts deployed on the local blockchain environment, allowing them to send transactions, call contract functions, and inspect contract state changes. This facilitates rapid iteration and experimentation with contract functionality, empowering developers to build and refine blockchain applications efficiently.

- **Ethereum Network Simulation**: Ganache allows developers to simulate different Ethereum network conditions and configurations, including varying gas limits, block times, and network IDs. This provides developers with a flexible and customizable environment for testing and optimizing their smart contracts and DApps for different network scenarios.

- **Education and Learning**: Ganache is commonly used for educational purposes, providing students, researchers, and blockchain enthusiasts with a hands-on environment for learning about Ethereum development and blockchain technology. Its intuitive interface, documentation, and built-in features make it an accessible tool for exploring blockchain concepts, writing smart contracts, and understanding the Ethereum ecosystem.

Overall, Ganache serves as a versatile and indispensable tool for Ethereum developers, offering a lightweight, customizable, and feature-rich blockchain environment for building, testing, and deploying smart contracts and decentralized applications. Its ease of use, integration capabilities, and extensive features make it an essential component of the Ethereum development toolkit.

Visual Studio Code (VS Code) and Ganache are commonly used together in Ethereum development workflows to streamline the process of building, testing, and deploying smart contracts and decentralized applications (DApps). While they are separate tools, they can be seamlessly integrated to enhance the development experience. Here's how VS Code and Ganache are connected:

Development Environment Integration:

- VS Code serves as the primary integrated development environment (IDE) for writing and editing code, including smart contracts written in Solidity and client-side applications written in JavaScript, TypeScript, or other languages.
- Ganache provides a local Ethereum blockchain environment for testing and deploying smart contracts and DApps. It runs as a separate application on your machine and creates a local blockchain network that can be accessed from within VS Code.

Truffle Suite Integration:

- Truffle is a popular development framework for Ethereum that integrates seamlessly with both VS Code and Ganache. Truffle provides tools and utilities for managing Ethereum projects, compiling smart contracts, running tests, and deploying contracts to Ethereum networks.
- Developers can use Truffle's CLI commands to interact with Ganache from within VS Code. For example, they can compile and migrate smart contracts, run tests against the local blockchain, and deploy contracts to the Ganache network directly from the VS Code terminal or integrated terminal.

Visual Studio Code Extensions:

- VS Code offers a variety of extensions that enhance the Ethereum development experience and provide integration with Ganache. For example, the Truffle Suite extension for VS Code provides features such as project scaffolding, smart contract compilation, deployment management, and debugging support, all of which can be used in conjunction with Ganache.
- Other extensions for Ethereum development, such as Solidity syntax highlighting, code snippets, and IntelliSense, can also be used alongside Ganache to improve the coding experience in VS Code.

Debugging:

- VS Code's debugging capabilities can be used in conjunction with Ganache to debug smart contracts and client-side applications. Developers can set breakpoints, inspect variables, and step through code execution using VS Code's debugging tools while interacting with contracts deployed on the Ganache blockchain.

Workflow Integration

- The integration between VS Code and Ganache streamlines the Ethereum development workflow by providing a unified environment for writing code, testing contracts, and deploying applications. Developers can leverage the familiar interface and features of VS Code while utilizing the local blockchain environment provided by Ganache for testing and debugging.

Overall, the connection between Visual Studio Code and Ganache enhances the Ethereum development experience by providing a seamless and integrated environment for building, testing, and deploying blockchain applications. Developers can leverage the power of both tools to iterate quickly, debug effectively, and deploy confidently in their Ethereum projects.

EXPERIMENT

The proposed model works in the following way:

1. The donor or recipient visits a registered hospital and fills out a registration form for organ donation.
2. The hospital verifies the donor/recipient's identity and medical records.
3. Once the verification is complete, the hospital registers the donor/recipient on the Ethereum blockchain using a smart contract. The smart contract records the donor/recipient's information, medical records, and any other relevant details.
4. The donor/recipient is given a unique identifier, which can be used to track their information on the blockchain.
5. The donor/recipient can now access their information and medical records on the blockchain, securely and transparently.
6. When an organ becomes available, the hospital updates the smart contract to reflect the new status of the recipient.
7. The recipient can track the status of their organ transplant in real-time, securely and transparently.
8. Once the transplant is complete, the hospital updates the smart contract to reflect the new status of the donor and recipient.
9. The donor and recipient can access their updated information and medical records on the blockchain, securely and transparently.

Overall, the proposed model aims to provide a secure and transparent system for organ donation using blockchain technology. The use of smart contracts and off-chain decentralized databases ensures that the system is efficient ,cost-effective, and provides a high level of security and transparency.

7.1 EXPERIMENTAL SETUP

Architecture: The figure below displays the architecture of theDApp for organ donation management consists of threemain components: frontend, backend, and blockchain.

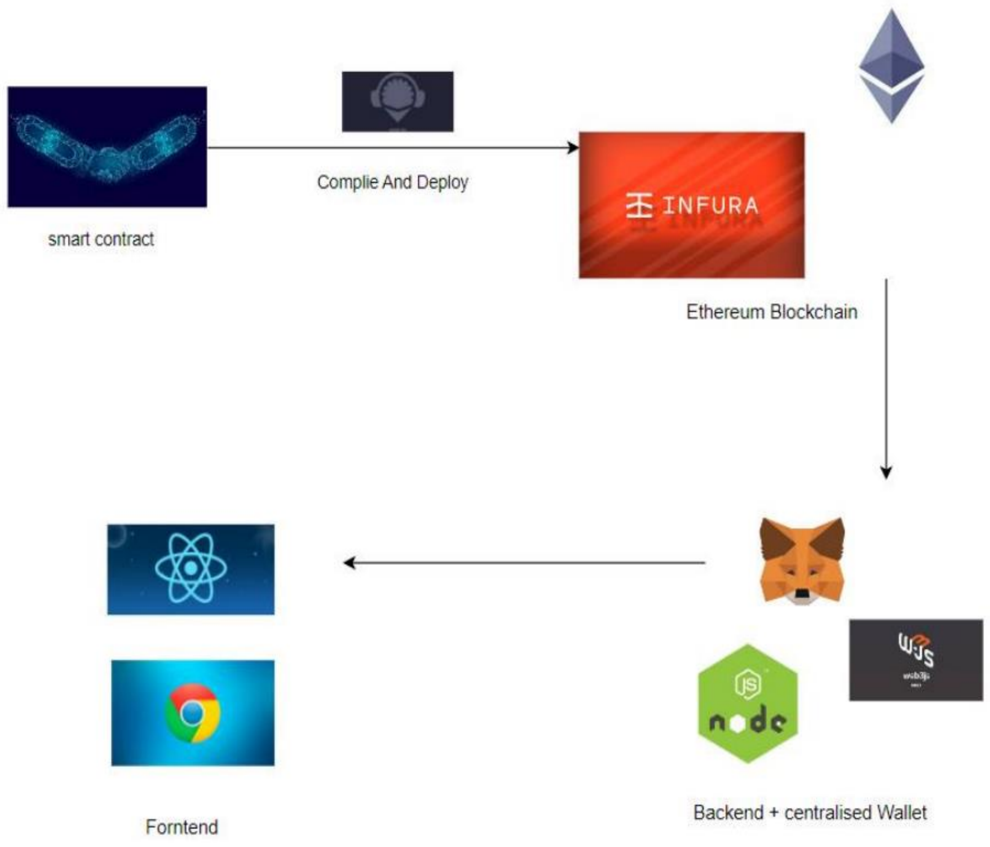


Fig 7.1: Experiment Setup

The implementation of the DApp for organ donation management involves several steps, including designing the user interface, developing the backend functionality, and integrating with the Ethereum blockchain. The following sections discuss the implementation details for each component of the architecture and the difficulties that may arise during the implementation process.

7.2 FRONTEND IMPLEMENTATION

The frontend of the DApp is built Utilizing ReactJS, as well-known JavaScript library designed for constructing user interfaces. React JS allows for the creation of reusable components, making it easier to develop complex user interfaces. The frontend of the DApp includes several pages, including a login page, a dashboard page, and a transaction history page. The login page is the first page that the user encounters when accessing the DApp. The login page provides the user with the option to access their account by entering their username and password, while also presenting a hyperlink for new users to register for a new account.. The dashboard page is the main page of the DApp, which provides users with an overview of their account information, including the balance of their funds, the status of their organ donations, and their transaction history. The dashboard page also allows users to initiate new organ donation transactions, view their pending transactions, and cancel transactions if necessary. The transaction history page provides users with a detailed view of their past transactions, including the amount of funds transferred, the status of the transaction, and the date and time of the transaction. This page is useful for users who want to review their past transactions and track their donation history.

7.3 BACKEND IMPLEMENTATION

The backend of the DApp is built using Node JS, which is a popular JavaScript runtime environment. Node JS allows for the development of server-side applications using JavaScript, making it easier to create a consistent development environment across the frontend and backend. The backend of the DApp is responsible for handling user requests, communicating with the centralized wallet, and interacting with the Ethereum smart contract. The centralized wallet is used to store and manage the funds used for organ donation. The Ethereum smart contract is responsible for executing the transactions on the blockchain and ensuring that the data is secure and tamper-proof. One of the main

difficulties that can arise during the backend implementation is handling security and data privacy. The backend must ensure that user data is stored securely and that transactions are executed correctly. This requires implementing robust security measures, such as encrypting user data, verifying user identities, and protecting against hacking attempts. Another challenge that can arise during the backend implementation is scaling the system to handle large numbers of users and transactions. The backend must be designed to handle high volumes of traffic and ensure that transactions are processed
quickly and efficiently.

7.4 BLOCKCHAIN INTEGRATION

The integration of the DApp with the Ethereum blockchain involves deploying the smart contract and interacting with the contract through the DApp frontend and backend. The smart contract is responsible for executing the transactions on the blockchain and ensuring that the data is secure and tamperproof. The smart contract contains the rules and logic that govern the organ donation process, including the amount of funds required to initiate a donation transaction, the conditions for canceling a transaction, and the process for verifying organ donor and recipient information. Another challenge that can arise during the blockchain integration is ensuring that the DApp is compatible with different Ethereum clients, such as Geth, Parity, and Infura.

7.5 WORKFLOW

The first step of the process is to register all hospitals that have been verified as eligible for organ donation.
This means that only hospitals that have been registered and authorized are allowed to perform organ transplantation.
1) Donor: The system is intended for individuals who are prepared to offer their organs for donation. Donors can register on the system by submitting their personal information

and specifying the organ they wish to donate. These particulars are then saved on both the blockchain network and electronic health record (EHR) files on IPFS.

To become an approved donor, the system performs verification checks. Once a donor is verified successfully, their status is changed to approved and their details are uploaded to the blockchain. The donor can then use their credentials to log into their account. When a suitable recipient is found, the donor can access information regarding the recipient and their hospital admission details is possible by retrieving the data from the blockchain. This system functions as a permissionless blockchain system, there is a risk of illegitimate use. Therefore, only verified donors are allowed to use the System.

2) Recipient: The system is specifically designed for individuals who are willing to donate their organs to others. Those in need of an organ can register on the system by providing personal information and specifying which organ they require. All of this information is securely stored on the blockchain network, along with electronic health records on IPFS. To ensure that only legitimate recipients are approved to use the system, verification checks are performed. Once approved, the recipient's status is changed and their details are uploaded to the blockchain. The system then uses a smart contract to automatically match recipients with suitable donors.

Recipients can access their account with their credentials and view the details of potential donors, which are pulled from the blockchain. Due to the open nature of the system, there is a risk of fraudulent activity. As a result, only verified recipients are allowed to use the platform.

3) Hospital: Once a hospital is registered by an authority, they can log into the system using their unique username and password. The profile of the hospital exhibits all pertinent information regarding the hospital. The hospital holds responsibility for various duties on the platform like :

Donor Approval: Any potential donors that visit the hospital must undergo a medical checkup. If they are verified, the hospital approves them on the system. The donor's details and electronic medical record (EMR) are retrieved from MongoDB, the details are removed and sent to IPFS to get their hashcode. This hashcode is then uploaded to the blockchain, along with the donor's EMR.

Recipient Registration: The hospital is responsible for registering patients in need of an organ transplant. The patient's details, EMR, and public key are uploaded to the blockchain in a similar fashion. The hospital sends these details to the IPFS server, gets the hashcodes, and uploads them to the blockchain network.

Transplantation: In the Transplant Match tab, the hospital can view a list of all the registered recipients in that hospital. If a hospital clicks the 'Match' button for any recipient, the match function in. Upon invoking the smart contract, the function retrieves the most suitable donor information from the blockchain. The function then provides the donor's public key, enabling the hospital to access the donor's information.

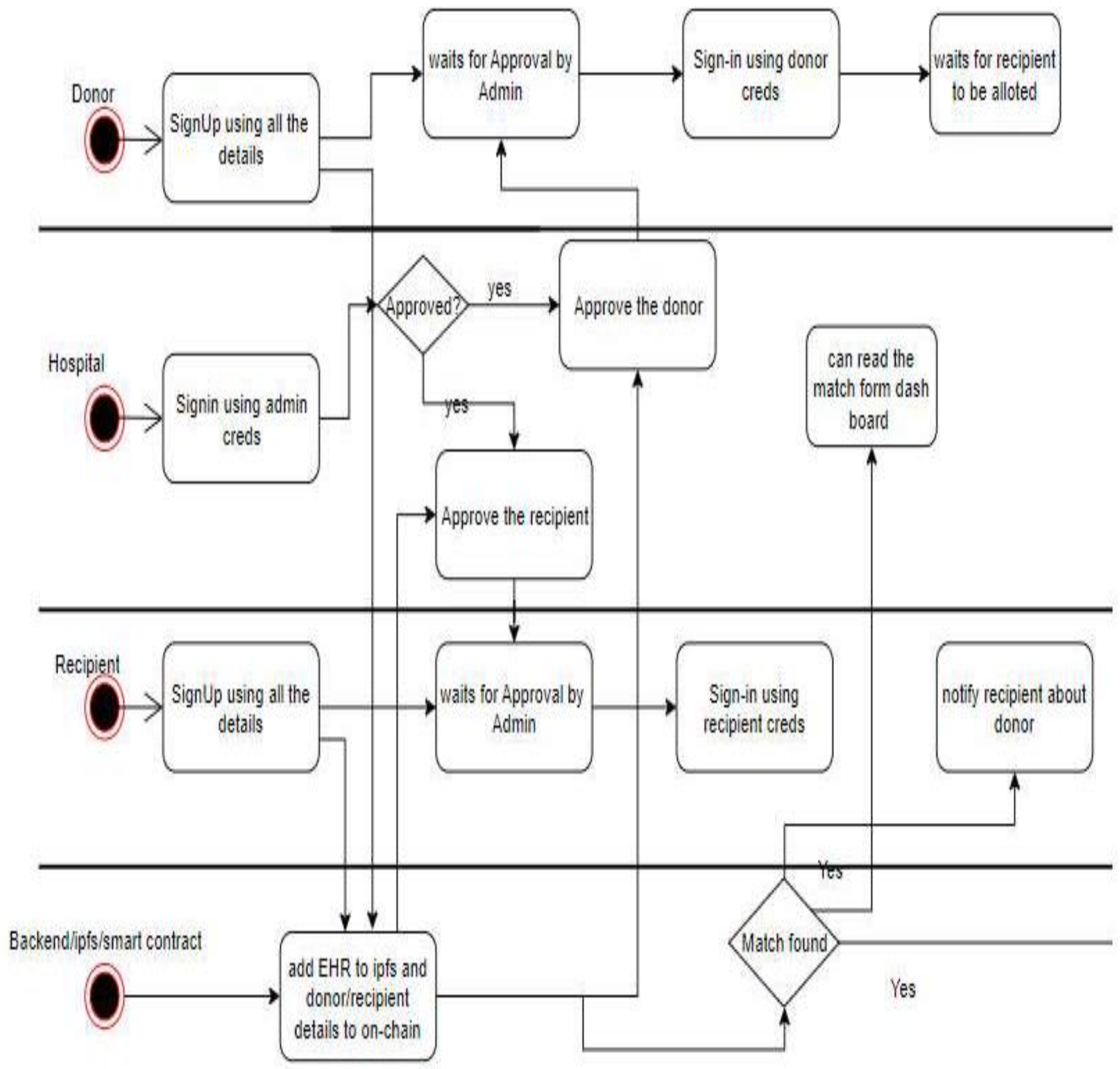

Donor
SignUp using all the details
waits for Approval by Admin
Sign-in using donor creds
waits for recipient to be alloted
Hospital
Signin using admin creds
Approved?
yes
Approve the donor
yes
Approve the recipient
can read the match form dash board
Recipient
SignUp using all the details
waits for Approval by Admin
Sign-in using recipient creds
notify recipient about donor
Backend/ipfs/smart contract
add EHR to ipfs and donor/recipient details to on-chain
Match found
Yes
Yes

Fig 7.5.1: Workflow

The proposed system works based on the existing system's one of the major problem. In this system, the organ donor's details are stored even if the organ donor dies. If any of the people dies, their tissues such as bone, skin ,heart valves, veins, tendons, ligaments and corneas can be donated within the first 24 hours of death. And even the organs that can be donated after death are heart, liver, kidneys, lungs, pancreas, and small intestines. These details can also be included into the database list. The block chain network is the

backbone of our proposed solution. It serves as the basis for recording the Transactions and events permanently to ensure accountability and data provenance. The developed smart contracts must be deployed on the block chain to ensure they are accessible at all times. However, it would not be ideal to deploy them on the main network during the testing phase. The proposed system is built on a private block chain, to which validation nodes and only authorized participants are added. The implementation of our proposed solution is mainly twofold: organ donation and organ transplantation.
In this project we will be using VS Code and Ganache.

There are mainly 3 instructions that we use to connect code, transfer data and execute in blockchain.

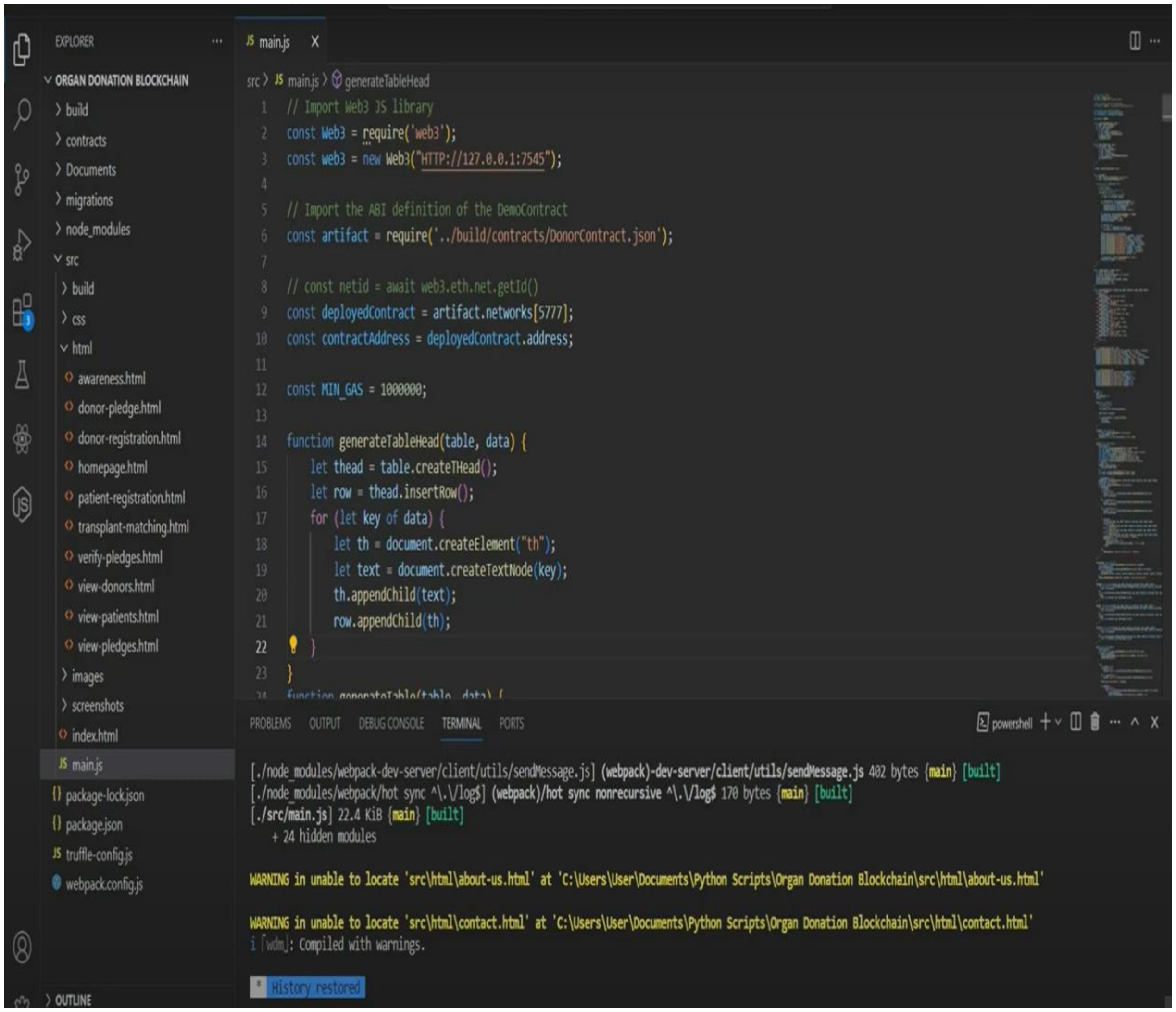
Truffle compile- to connect vscode's contracts into ganache

Truffle migrate- used to compile contracts and transfer data

Npm run dev: to run the code

After executing these 3 steps perfectly you will obtain a local host address where users can input information.

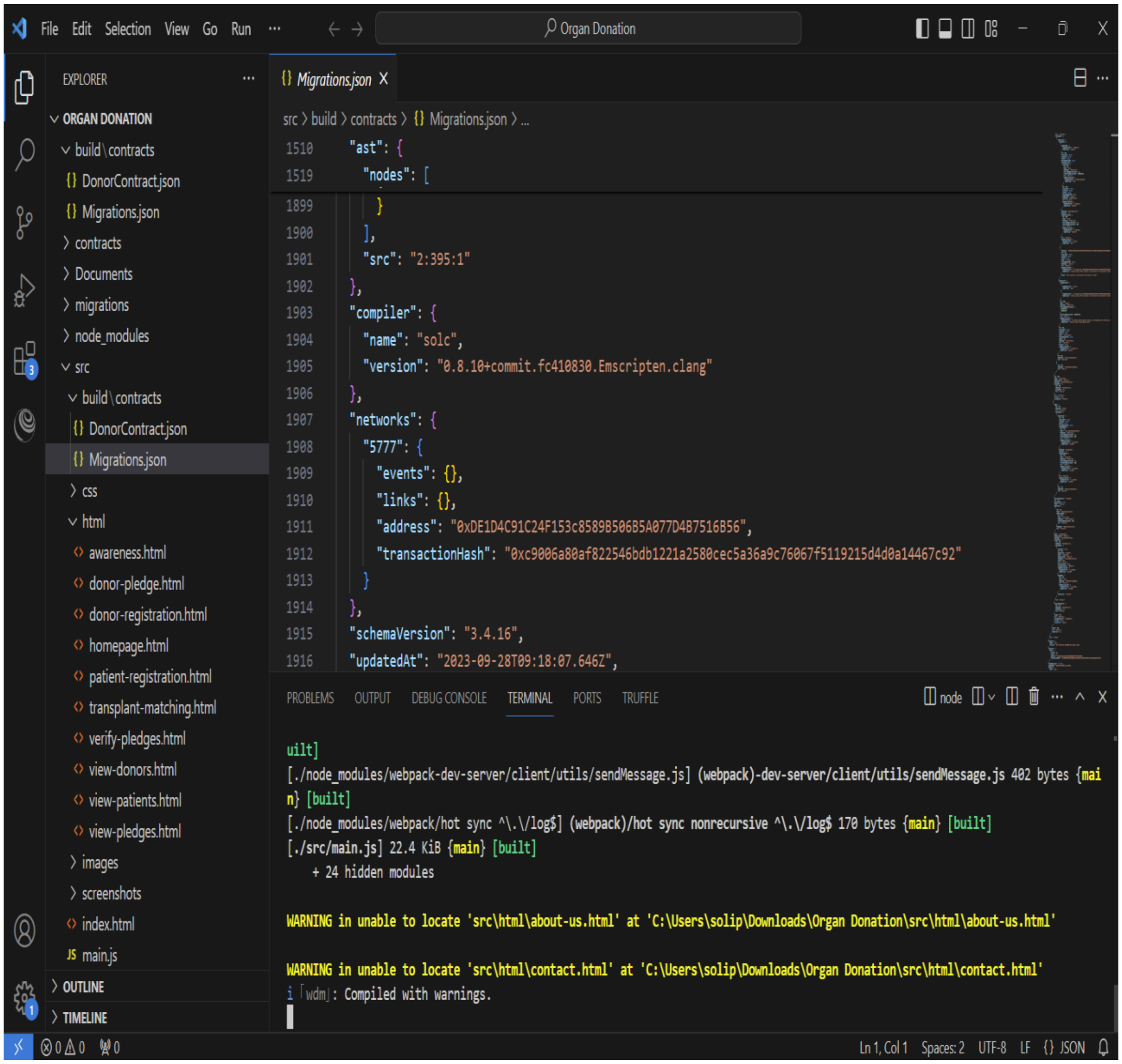
At the user interface they should be able to add patient information and find matches along with lists of all patients and donors



```javascript
// Import Web3 JS library
const Web3 = require('web3');
const web3 = new Web3("HTTP://127.0.0.1:7545");

// Import the ABI definition of the DemoContract
const artifact = require('../build/contracts/DonorContract.json');

// const netid = await web3.eth.net.getId()
const deployedContract = artifact.networks[5777];
const contractAddress = deployedContract.address;

const MIN_GAS = 1000000;

function generateTableHead(table, data) {
    let thead = table.createTHead();
    let row = thead.insertRow();
    for (let key of data) {
        let th = document.createElement("th");
        let text = document.createTextNode(key);
        th.appendChild(text);
        row.appendChild(th);
    }
}
function generateTable(table, data) {
```

```
[./node_modules/webpack-dev-server/client/utils/sendMessage.js] (webpack)-dev-server/client/utils/sendMessage.js 402 bytes {main} [built]
[./node_modules/webpack/hot sync ^\.\/log$] (webpack)/hot sync nonrecursive ^\.\/log$ 170 bytes {main} [built]
[./src/main.js] 22.4 KiB {main} [built]
    + 24 hidden modules

WARNING in unable to locate 'src\html\about-us.html' at 'C:\Users\User\Documents\Python Scripts\Organ Donation Blockchain\src\html\about-us.html'

WARNING in unable to locate 'src\html\contact.html' at 'C:\Users\User\Documents\Python Scripts\Organ Donation Blockchain\src\html\contact.html'
i ｢wdm｣: Compiled with warnings.

* History restored
```

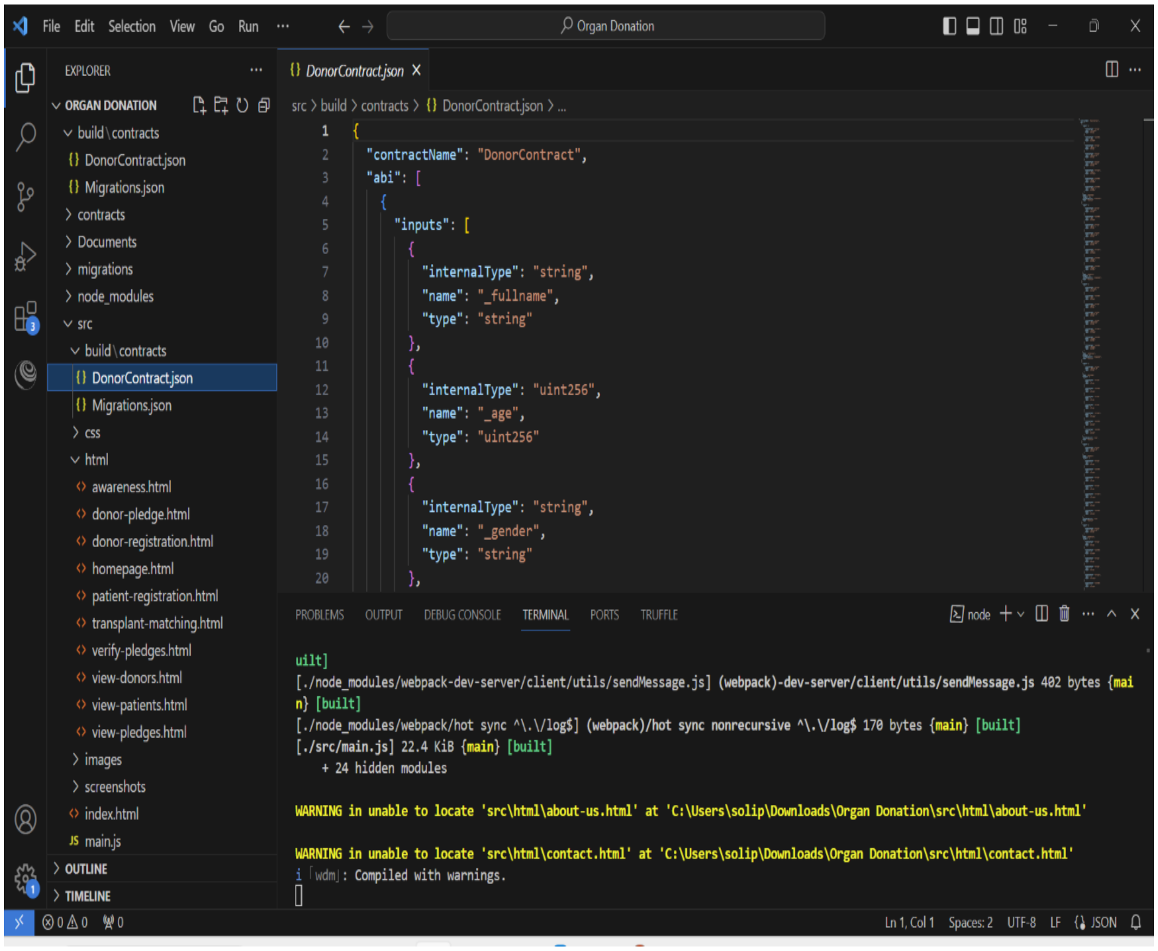
Fig 7.5.2: Main code



```
  "ast": {
    "nodes": [
      }
    ],
    "src": "2:395:1"
  },
  "compiler": {
    "name": "solc",
    "version": "0.8.10+commit.fc410830.Emscripten.clang"
  },
  "networks": {
    "5777": {
      "events": {},
      "links": {},
      "address": "0xDE1D4C91C24F153c8589B506B5A077D4B7516B56",
      "transactionHash": "0xc9006a80af822546bdb1221a2580cec5a36a9c76067f5119215d4d0a14467c92"
    }
  },
  "schemaVersion": "3.4.16",
  "updatedAt": "2023-09-28T09:18:07.646Z",
```

```
uilt]
[./node_modules/webpack-dev-server/client/utils/sendMessage.js] (webpack)-dev-server/client/utils/sendMessage.js 402 bytes {mai
n} [built]
[./node_modules/webpack/hot sync ^\.\/log$] (webpack)/hot sync nonrecursive ^\.\/log$ 170 bytes {main} [built]
[./src/main.js] 22.4 KiB {main} [built]
    + 24 hidden modules

WARNING in unable to locate 'src\html\about-us.html' at 'C:\Users\solip\Downloads\Organ Donation\src\html\about-us.html'

WARNING in unable to locate 'src\html\contact.html' at 'C:\Users\solip\Downloads\Organ Donation\src\html\contact.html'
i ｢wdm｣: Compiled with warnings.
```

Fig 7.5.3: Migration Contract with Address



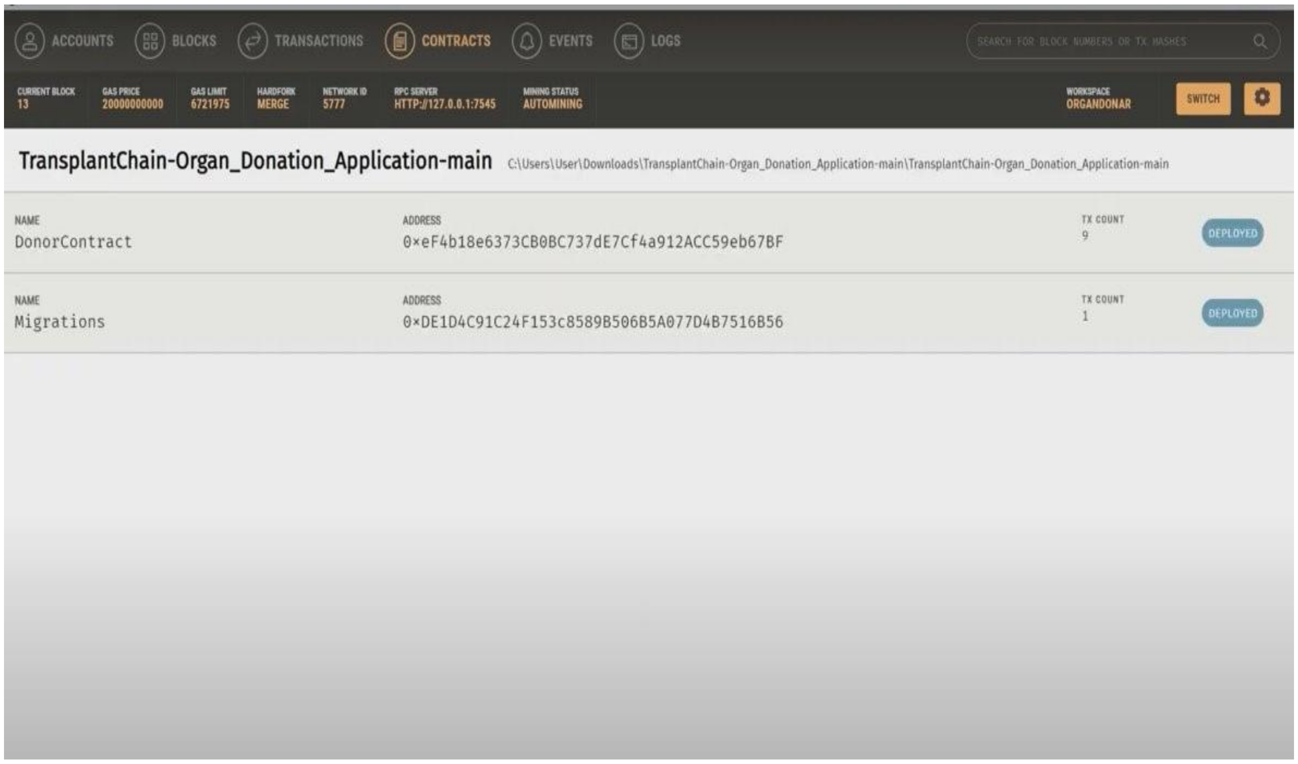
```
{
  "contractName": "DonorContract",
  "abi": [
    {
      "inputs": [
        {
          "internalType": "string",
          "name": "_fullname",
          "type": "string"
        },
        {
          "internalType": "uint256",
          "name": "_age",
          "type": "uint256"
        },
        {
          "internalType": "string",
          "name": "_gender",
          "type": "string"
        },
```

```
uilt]
[./node_modules/webpack-dev-server/client/utils/sendMessage.js] (webpack)-dev-server/client/utils/sendMessage.js 402 bytes {mai
n} [built]
[./node_modules/webpack/hot sync ^\.\/log$] (webpack)/hot sync nonrecursive ^\.\/log$ 170 bytes {main} [built]
[./src/main.js] 22.4 KiB {main} [built]
    + 24 hidden modules

WARNING in unable to locate 'src\html\about-us.html' at 'C:\Users\solip\Downloads\Organ Donation\src\html\about-us.html'

WARNING in unable to locate 'src\html\contact.html' at 'C:\Users\solip\Downloads\Organ Donation\src\html\contact.html'
i ｢wdm｣: Compiled with warnings.
```

Fig 7.5.4: Donor Contract

RESULTS

Here you can see two types of contracts. The first one is the donor contract where patients and donors information is taken. In the migration contract matches between donor and patient information is checked.



CURRENT BLOCK 13 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE ORGANDONAR

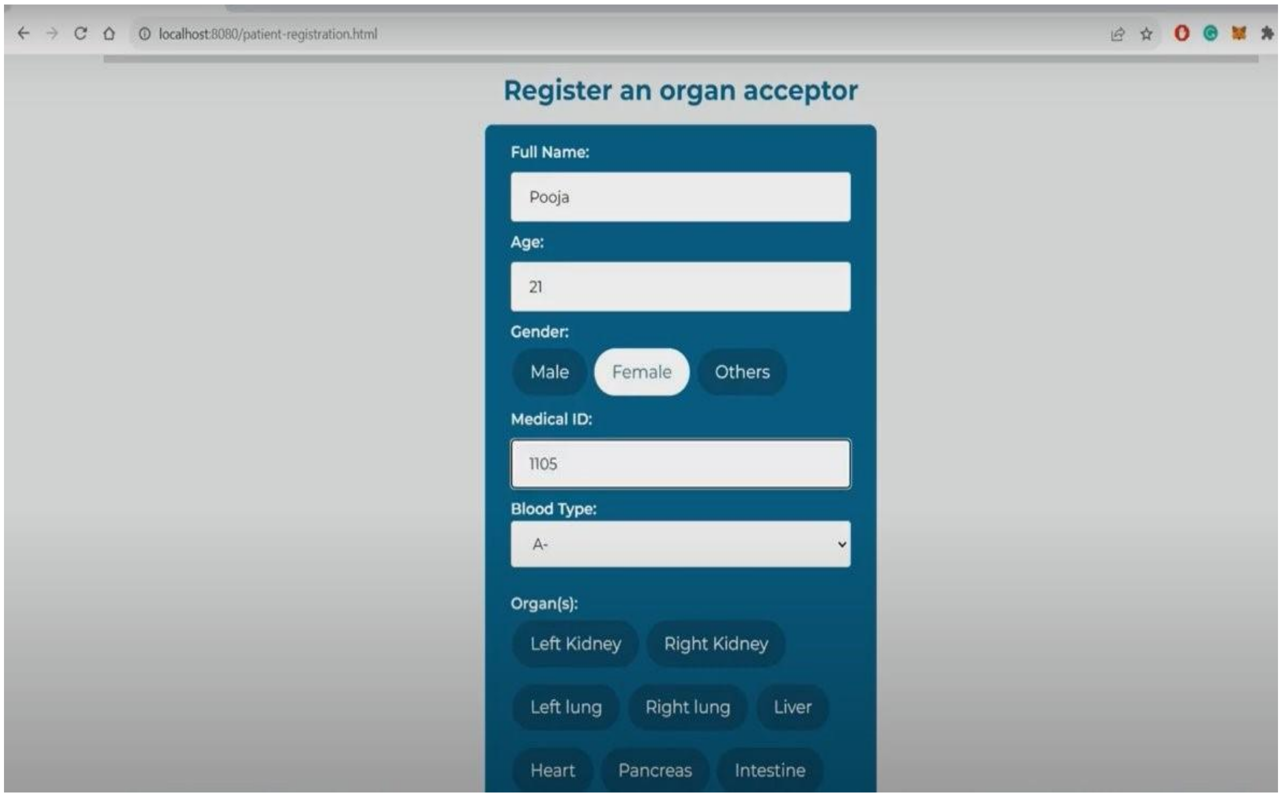
TransplantChain-Organ_Donation_Application-main C:\Users\User\Downloads\TransplantChain-Organ_Donation_Application-main\TransplantChain-Organ_Donation_Application-main

NAME	ADDRESS	TX COUNT	
DonorContract	0×eF4b18e6373CB0BC737dE7Cf4a912ACC59eb67BF	9	DEPLOYED
Migrations	0×DE1D4C91C24F153c8589B506B5A077D4B7516B56	1	DEPLOYED

Fig 8.1: Contracts in Ganache

After opening the local host address various options are given which include entering donor and patient information, gathering all donor information and patient information seperately and a place to find matches is also given.

When opening for inserting information this is what it will look like.

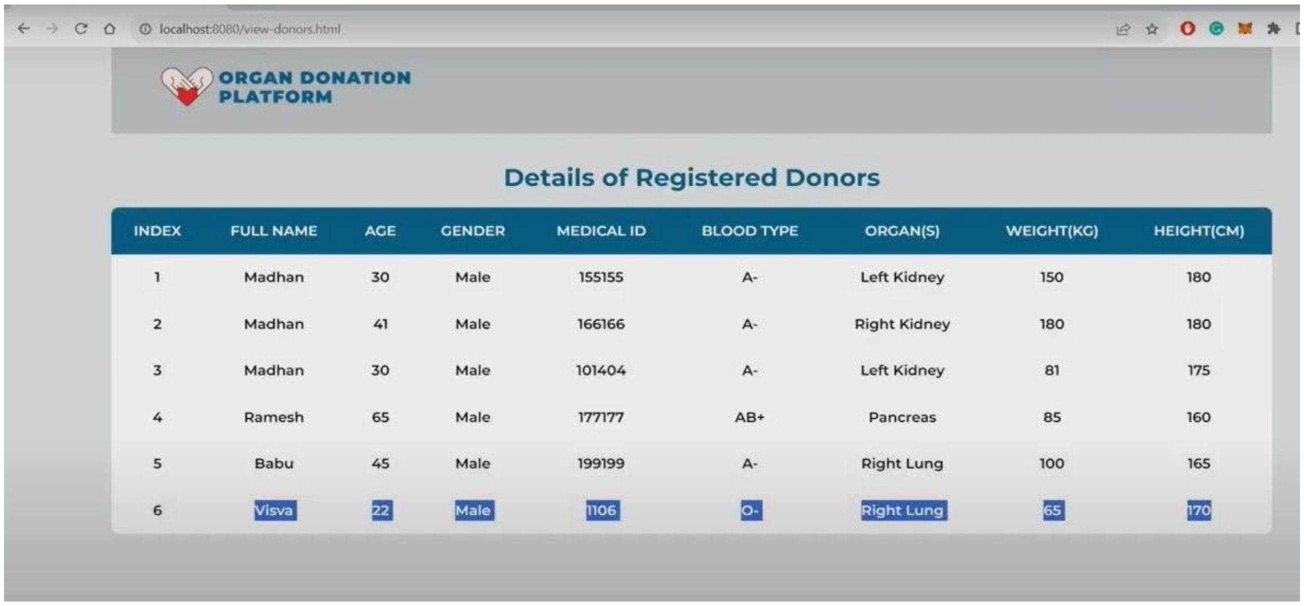

localhost:8080/patient-registration.html

Register an organ acceptor

Full Name:	Pooja
Age:	21
Gender:	Male [✓ Female] Others
Medical ID:	1105
Blood Type:	A-
Organ(s):	Left Kidney, Right Kidney, Left lung, Right lung, Liver, Heart, Pancreas, Intestine

Fig 8.2: Registration of Organ Donor and Acceptor

This is what the list of donors or patients would look like


localhost:8080/view-donors.html

ORGAN DONATION PLATFORM

Details of Registered Donors

INDEX	FULL NAME	AGE	GENDER	MEDICAL ID	BLOOD TYPE	ORGAN(S)	WEIGHT(KG)	HEIGHT(CM)
1	Madhan	30	Male	155155	A-	Left Kidney	150	180
2	Madhan	41	Male	166166	A-	Right Kidney	180	180
3	Madhan	30	Male	101404	A-	Left Kidney	81	175
4	Ramesh	65	Male	177177	AB+	Pancreas	85	160
5	Babu	45	Male	199199	A-	Right Lung	100	165
6	Visva	22	Male	1106	O-	Right Lung	65	170

Fig 8.3: Matches of Donors and Acceptors

CONCLUSION

In the EHR system the patient can access their report and can use the report for their lifetime with security. The private key is used for the patient which can be used for the further use of the reports. The one who don’t have the private key cannot involve in the process of retrieving data. Hence the Health Records of the patients are more secured with the Blockchain and can used with their own private key as well as they can make use of that for further reference. Currently, there is a huge obstacle for using Ethereum platform, which lays in difficulty of obtaining ETH units. The units can be obtained either by mining or purchased for fiat or cryptocurrency like Bitcoin. Once ETH units are available, the publishing and execution of smart contract is really smooth., we have developed a web application user interface for the health record system. For increased security, we can add account authentication features using some unique identity features.

In conclusion, the proposed Decentralized Application(DApp) using Ethereum Blockchain technology offers a solution to the problems currently faced by the organ transplantation system. The lack of communication between donors and recipients, the prevalence of illegal organ trade, and the high cost of transplantation due to monopolies and urgent needs of recipients are some of the issues that the proposed DApp addresses.

REFERENCES

[1] Kulshrestha, A., Mitra, A., & Amisha. (2020). Securing Organ Donation using Blockchain. International Journal of Scientific & Engineering Research, 11(6), June 2020.

[2] Douville, F., Godin, G., & Vézina-Im, L. A systematic review of interventions targeting health professionals and their impact on organ and tissue donation in clinical settings. This study is available under PMID: 24628967, PMCID: PMC4003858, and DOI: 10.1186/2047- 1440-3-8.

[3] Abbasi, M., Kiani, M., Ahmadi, M., & Salehi, B. Perspectives on Knowledge and Ethical Issues in Organ Transplantation and Organ Donation from Iranian Health Personnel. The authors of this study are Mahmou Abbasi, Mehrzad Kiani, Mehdi Ahmadi, and Bahare Salehi.

[4] Jones, C. P., Papadopoulos, C., Randhawa, G., & Asghar, Z. A research protocol titled "General Practice Organ Donation Intervention - A Feasibility Study" (GPOD) by Catrin Pedder Jones, Chris Papadopoulos, Gurch Randhawa, and Zeeshan Asghar.

[5] Kalpana, K., & Baby Shamili, M. Organ Donor Management System. This system was developed by K. Kalpana and M. Baby Shamili, who are students pursuing their Master of Computer Applications at Madanapalle Institute of Technology And Science in Madanapalle, India.

[6] Network policies regarding organ donation and transplantation were published on October 31, 2019, and can be accessed at the following website: https://optn.transplant.hrsa.gov/

[7] Statistics on organ donation can be found at https://www.organdonor.gov/statistics-stories/statistics.html. The statistics were published on October 31, 2019.

[8] The "Blood Gift" framework was developed by the Foundation of Data Innovation, Software Engineering, and Designing branch at Jahangirnagar College in Dhaka, Bangladesh. It is discussed in Volume 4 of their publication.

[9] The "UNet" technology is available at unos.org/technology/unet. This information is from October 31, 2019.

[10]Indian J Urol [serial online] 2009 [cited2020 Feb 9];25:348-55. Available from: http://www.indianjurol.com/text.asp?2009/25/ 3/348/56203

[11] Zirpe K, Gurav S. Brain Death and Management of Potential Organ Donor: An Indian Perspective. Indian J Crit Care Med 2019;23(Suppl 2):S151– S156.

[12] Organ donation and transplantation-the Chennai experience in India. Shroff S, Rao S, Kurian G, Suresh S. Transplant Proc. 2007; 39:714– 718. [PubMed] [GoogleScholar]

[13]Organ donation in India: scarcity in abundance. Sachdeva S. Indian J Public Health. 2018; 61:299. [PubMed] [GoogleScholar]

[14] Organ donation after brain death in India: a trained intensivist is the key to success. Palaniswamy V, Sadhasivam S, Selvakumaran C, Jayabal P, Ananth SR. Indian J Crit Care Med.2016;20:593–596. [PMC free article] [PubMed] [Google Scholar]