BLOCKCHAIN BASED ORGAN DONATION MANAGEMENT USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

S. SAI ARAVINDH (312420104133)

TEJAS SATISH KUMAR (312420104174)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

St. JOSEPH'S INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

ANNA UNIVERSITY: CHENNAI 600025

MARCH 2024

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**BLOCKCHAIN ORGAN DONATION MANAGEMENT USING MACHINE LEARNING"** is the bonafide work of "**S. SAI ARAVINDH (312420104133) and TEJAS SATISH KUMAR (312420104174)"** who carried out the project under my supervision.

SIGNATURE

Dr. J. DAFNI ROSE M.E., Ph.D.,

PROFESSOR AND HEAD,

Computer Science and Engineering,

St. Joseph's Institute of Technology,

Old Mamallapuram Road,

Chennai - 600 119.

SIGNATURE

Dr. N. MYTHILI M.E., Ph.D.,

SUPERVISOR,

ASSOCIATE PROFESSOR,

Computer Science and Engineering,

St. Joseph's Institute of Technology,

Old Mamallapuram Road,

Chennai – 600 119.

ACKNOWLEDGEMENT

We also take this opportunity to thank our respected and honourable Chairman **Dr. B. Babu Manoharan M.A., M.B.A., Ph.D.,** for the guidance he offered during our tenure in this institution.

We extend our heartfelt gratitude to our respected and honourable Managing Director **Mr. B. Sashi Sekar M.Sc.,** for providing us with the required resources to carry out this project.

We express our deep gratitude to our honourable Executive Director **Mrs. S. Jessie Priya M.Com.,** for the constant guidance and support forour project.

We are indebted to our Principal **Dr. P. Ravichandran M.Tech., Ph.D.,** for granting us permission to undertake this project.

We would like to express our earnest gratitude to our Head of the Department **Dr. J. Dafni Rose M.E., Ph.D.,** for her commendable support and encouragement for the completion of the project with perfection.

We also take the opportunity to express our profound gratitude to our guide **Dr. N. Mythili M.E., Ph.D.,** for her guidance, constant encouragement, immense help and valuable advice for the completion of this project.

We wish to convey our sincere thanks to all the teaching and non-teaching staff of the department of **COMPUTER SCIENCE AND ENGINEERING** without whose cooperation this venture would not have been a success.

CERTIFICATE OF EVALUATION

College Name : St. JOSEPH'S INSTITUTE OF TECHNOLOGY

Branch : COMPUTER SCIENCE AND ENGINEERING

Semester : VIII

Sl.No.	Name of the Students	Title of the Project	Name of the Supervisor with designation
1 2	S. SAI ARAVINDH (312420104133) TEJAS SATISH KUMAR (312420104174)	Blockchain Organ Donation Management Using Machine Learning	Dr. N. MYTHILI M.E., Ph.D., Associate Professor,

The report of the project work submitted by the above students in partial fulfilment for the award of Bachelor of Engineering Degree in Computer Science and Engineering of Anna University were evaluated and confirmed to be report of the work done by above students.

Submitted for project review and viva voce exam held on _____

(INTERNAL EXAMINER) **(EXTERNAL EXAMINER)**

ABSTRACT

Today's organ donation and transplantation systems pose different requirements and challenges in terms of registration, donor-recipient matching, organ removal, organ delivery, and transplantation with legal, clinical, ethical, and technical constraints. Therefore, an end-to-end organ donation system is required to guarantee a fair and efficient process to enhance patient experience and trust. Blockchain offers many features thatcan be used in almost every sphere of life. Features like decentralisation, transparency, privacy makes it an extremely useful technology. Therefore, by making use of all these features, several problems in healthcare sector can be solved like removing complex network of third parties and lack of traceabilityof transactions. In this paper, we propose a private Ethereum blockchain- based web application (Dapp) to enable organ donation and transplantation management in a manner that is fully decentralized, secure, traceable, auditable, private, and trustworthy. In addition, we propose to use a K- means clustering machine learning model to make complex decisions on organ recipient selection when organ donors are of limited numbers. This will remove the need for humans to make these difficult decisions and improve theefficiency of organ donor and recipient selection for all situations and circumstances.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the realm of medical science, the demand for organ transplants to save lives is ever-increasing, while the supply of available organs remains critically limited. The shortage of viable organs not only leads to prolonged suffering for patients awaiting transplants but also significantly reduces their chances of survival. Addressing this pressing global issue requires innovative approaches that optimize the organ donation process, improve transparency, and enhance the efficiency of matching donors with recipients.

1.1. OVERVIEW

In recent years, the fusion of blockchain technology and machine learning has shown immense promise in revolutionizing various sectors, and the domain of organ donation management is no exception. This report delves into the concept and implementation of a groundbreaking Blockchain Organ Donation Management System (BODMS) empowered by Machine Learning (ML). Blockchain's inherent transparency, immutability, and decentralized nature make it an ideal platform to securely store and share medical records, while machine learning algorithms can facilitate the rapid and precise matching of donors with recipients, increasing the chances of successful transplantations.

This project will primarily focus on building an end-to-end decentralized block chain solution that uses machine learning to automaticallymatch potential organ donors to patients that require transplantation of said organ. It will not focus on the logistics and the planning required for transporting an organ and tracking its journey from donor to recipient.

1.2. PROBLEM STATEMENT

Today's organ donation and transplantation systems pose different requirements and challenges in terms of registration, donor-recipient matching, organ removal, organ delivery, and transplantation with legal, clinical, ethical, and technical constraints. Additionally, the bribing of officialsto green-light certain organs transfers regardless of compatibility and other checks have become common place.

The problem statement for our project is to design and develop an efficient and impartial system for organ donation and transplantation that ensures transparency and fairness, eliminating any and all instances of partial or unjust treatment towards any patient. This involves developing smart contracts to facilitate transactions with a private Ethereum blockchain, so that medical data can be securely stored. Additionally, a machine learning model must be trained using the K-means clustering algorithm.

1.3. EXISTING SYSTEM

As the use of Blockchain Organ Donation Management System (BODMS) has become common place in the Blockchain and medical industry, there exists a large body of research on the most efficient ways for BODMS to operate. Some previous works delve into the most efficient ways to perform bipartite matching (whether these operations should be performed on-chain or off-chain), whereas others compared the efficiency and securityof different algorithms.

Most existing systems are implemented in a similar manner, architecture and therefore can be generalized into a single system. This system is a simple fullstack application that uses a browser UI to allow users to trigger/call the smart contract's functions using web3 object providers such as MetaMask and the Mist browser. The smart contract will complete the transaction requesting

a gas fee. The medical records such as patient data and organ details will all be stored on-chain. This solution also provided users a way to view all transactions made to the chain, making the entire process transparent to all authorised individuals.

The above system was implemented by Alandjani, G. [1], Shreya Khatal, Jayant Rane, Dhiren Patel, Pearl Patel, Yann Busne [25], Dajim, L. A., Al-Farras, S. A., AlShahrani, B. S., AlZuraib, A. A., & Merlin Mathew, R. [7], P.L. Wijayathilaka,P.H.P. Gamage, K.H.B. De Silva, A.P.P.S. Athukorala, K.A.D.C.P. Kahandawaarachchi and K.N. Pulasinghe [27], P. Ranjan, S. Srivastava, V. Gupta, S. Tapaswi and N. Kumar [21] and U. Jain [16] in different architectures and approaches which had their own advantages and disadvantages.

1.4. PROPOSED SYSTEM

The proposed system stands apart from the existing system as it implements an automated organ recipient to organ donor matching with the help a K-Means Clustering algorithm to train the ML model. Here is an overview of the K-means Clustering algorithm for organ matching:

1. Data Collection: Gather the dataset containing the relevant information for organ donation, including columns such as age, blood type, size,and tissue type.

2. Data Cleaning: Check for missing values, duplicates, and outliers in the dataset. Handle any missing values by either imputing them or removing the corresponding records if they are negligible. Outliers can be treated using methods like z-score normalization or interquartile range (IQR) trimming, depending on the distribution of the data.

3. Feature Selection: Assess the significance of each feature (age, blood type, size, and tissue type) with respect to the clustering task. Remove any irrelevant or redundant features that may not contribute meaningfully to the clustering process.

4. Data Transformation: For the k-means algorithm, which uses distance-based calculations, it is essential to scale the features to the same range. Apply standardization or normalization to ensure that all features have similar scales, preventing any single feature from dominating the clustering process.

5. Encoding Categorical Variables: If blood type and tissue type are categorical variables (e.g., A, B, AB, O), convert them into numerical representations using techniques like one-hot encoding or label encoding. This step is necessary to handle non-numeric data in the k-means algorithm.

6. Dimensionality Reduction (optional): If the dataset has high dimensionality, consider employing dimensionality reductiontechniques like Principal Component Analysis (PCA) to reduce the number of features while preserving the most relevant information.

7. Choosing K: In this case, we have k=3, which means we want to create four clusters. The selection of the optimal value of K can be done using methods like the Elbow Method or Silhouette Score to find the optimal number of clusters that yield the most compact and well-separated clusters.

8. Training the K-Means Model: Utilize the preprocessed dataset and feed it into the K-Means clustering algorithm. The algorithm will iteratively assign each data point to one of the four clusters based on the similarity of their features and update the cluster centroids accordingly.

9. Evaluating the Clustering Results: Assess the quality of the clustering results using metrics like the Within-Cluster Sum of Squares (WCSS) or the Silhouette Score. These metrics help to quantify the compactness and separation of the clusters, ensuring that the clustering is meaningful.
10. Cluster Visualization: Visualize the clusters using scatter plots or other relevant visualizations to gain insights into the distribution of the data points and the effectiveness of the clustering process.
11. Cluster Analysis: Analyze the characteristics of each cluster based on the original features (age, blood type, size, and tissue type) to interpret the patterns and differences between the identified clusters.
12. Utilizing the Clusters: After the k-means clustering, you can use the clusters to group similar data points together, which can be valuable for tasks like personalized organ donation recommendations or targeted medical interventions.

By following these preprocessing and training steps, we can leverage the power of the k-means clustering algorithm to effectively group the organ donors into three distinct clusters based on age, blood type, size, and tissue type.

One major issue that is commonly faced in medical research papers, is the availability of real-time datasets. This is due privacy and security concerns as patient health information, is sensitive and subject to strict privacy regulations to protect patient confidentiality, which can make sharing and accessing real-time medical data more difficult. In addition, Hospitals, healthcare institutions, and research organizations often have their own data sharing policies. They may be reluctant to share real-time data due to concerns about liability, patient consent, or competition.

Due to the lack of readily available public datasets, in this project we will utilise a synthetic dataset. We acknowledge that the use of a synthetic dataset may introduce potential biases in the training. This may lead to discrepancies when comparing the results of the machine learning model to a real world scenario.

To generate our synthetic dataset, we use GANs (Generative Adversarial networks) and other generative models that use LSTM for synthetic data generation. The two frameworks we used are Synthea (a medical dataset generator) and Gretel which is a general purpose generative model which analyses any given sample dataset and generates synthetic data that matches the patterns found in the sample dataset.

The user interacts with blockchain application through the web browser. The smart contract's functions are called/triggered by user actions. The smart contract will perform asynchronous transaction to the blockchain. When a new donor is added to the system, the donor will be added to list of donors in the blockchain. When a new recipient is added, all donors present inthe blockchain are retrieved and the most appropriate donor is selected by the machine learning model.

The machine learning model will be running on an application server to which the smart contract will feed query points to. The model will receive the query points as a JSON request, which is then parsed. Each query point is then assigned to a cluster, and their respective labels are returned to the user via the smart contract.

CHAPTER 2

LITERATURE SURVEY

Alandjani, G. (2019) [1] created a POC Dapp that tracks allmedical transactions without compromising secrecy of data by keeping its integrity intact legal requirements and prevent risk of fraud. This system is a simple fullstack application that uses a browser UI to allow users to trigger/call the smart contract's functions using web3 object providers such asMetaMask and the Mist browser. The smart contract will complete the transaction requesting a gas fee. The medical records such as patient data and organ details will all be stored on-chain. This solution also provided users a way to view all transactions made to the chain, making the entire process transparent to all authorised individuals. Its main advantage was that the Blockchain recorded every transmission in communication, allowing the saved data to be viewed publicly with no alteration risk by accomplishing decentralized agreements. A major disadvantage with his proposed solution was that, although system was completely transparent, it was implemented on a private blockchain, meaning if all the authorised parties were complicit, then its purpose of employing blockchain is defeated.

Arigela. S. S. D and P. Voola (2023) [2] explores and compares the two prominent blockchain open-source tools, Ethereum and Hyperledger Fabric. The authors likely delve into the technical aspects, functionalities, and features of these blockchain platforms, providing insights into their respective strengths, weaknesses, and potential use cases. The paper may contribute to the understanding of practitioners, researchers, and developers seeking information on the selection and implementation of blockchain tools for various applications.

Bajoudah, S., Dong, C., Missier, P. (2019) [3] created the development of a decentralized marketplace for trading Internet of Things (IoT) data by

integrating blockchain technology. The authors advocate for a trust-less environment facilitated by blockchain, eliminating the need for centralized authorities and enhancing security and transparency in IoT data transactions. The proposed system likely employs smart contracts to automate and enforce transaction terms, ensuring immutability and transparency. Security considerations, marketplace dynamics, and challenges associated with decentralization may also be discussed, providing insights into the potential benefits and complexities of implementing a decentralized, blockchain-based IoT data trading platform.

Bates. M, [4] addresses the complexities and hurdles within the field of organ transplantation. Bates likely discusses various challenges such as organ scarcity, logistical issues, and ethical considerations, offering insights into innovative solutions and advancements that aim to overcome these obstacles. The paper may delve into the application of emerging technologies, improved medical practices, and potential policy changes in the context of organ transplantation. By providing a comprehensive overview of the challenges faced in organ transplantation and potential strategies for overcoming them, the article contributes to the ongoing dialogue and efforts to enhance the efficiency and accessibility of organ transplantation procedures.

Bertsimas, Farias, and Trichakis (2013) [5] addresses the complex challenges of organ allocation for kidney transplantation, exploring the intersection of fairness, efficiency, and flexibility in the allocation process. The authors likely propose mathematical models or optimization frameworks to balance the equitable distribution of organs, the efficiency of the allocation system, and the flexibility to adapt to various contextual factors. The paper may delve into the trade-offs involved in designing an allocation system that considers both individual and societal fairness while optimizing the overall efficiency of organ distribution. By employing operations research techniques, the authors likely aim to contribute insights into improving the organ allocation

process to ensure a fair and efficient utilization of available kidneys.

Chaudhary, N., Manvi, S. S., Koul, N. (2022) [6] is based on an innovative concept of an Organ Bank built on blockchain technology. The authors likely explore the application of blockchain in creating a secure, transparent, and decentralized system for organ transplantation. By leveraging the unique features of blockchain, such as immutability and transparency, the proposed Organ Bank aims to address challenges in organ allocation and tracking. The paper may discuss the implementation of smart contracts to automate processes, ensuring trust and efficiency in organ transactions. The emphasis is likely on enhancing the security and traceability of organ-related data while creating a decentralized network that fosters collaboration among stakeholders.

Dajim, L. A., Al-Farras, S. A., AlShahrani, B. S., AlZuraib, A. A., & Merlin Mathew, R. (2019) [7] proposed an Organ Donation Decentralized Application Using Blockchain Technology. It was an organ donation Dapp using blockchain. It would be a web app for patients to register their information. The system would work on a first-in, first-out basis unless apatient is in critical condition. It was a much faster system with improves scalability, as it can handle increased growth in the amount of work. Althoughtransparency in increased, and corruption is still possible. This is due to dependency on the manual selection of donor for a given recipient. It was very dependent on human intervention for the selection process

Davis, Mehrotra, Friedewald, and Ladner (2013) [8] delves into the development and characteristics of a simulation model for the national kidney transplantation system. The authors likely focus on the construction and validation of the simulation model, aiming to provide insights into the dynamics and complexities of the national kidney transplantation process. The simulation may encompass various factors such as organ availability, waiting lists, donor-recipient matching, and other relevant variables. The paper may

discuss how the simulation model contributes to understanding the functioning of the kidney transplantation system, potentially offering a tool for policymakers and healthcare professionals to evaluate and optimize system performance.

Dimitris Bertsimas, Vivek F. Farias, Nikolaos Trichakis (2013) [9] was on the fairness, efficiency and flexibility in organ allocation for kidney transplantation. Their work challenged the existing organ allocation scheme by pointing to its flaws and introducing their own organ allocation scheme, which surpassed the existing scheme in certain aspects. Their work focussed on finding a scalable, data driven method for designing national policies for the allocation of deceased donor kidneys to patients on a waiting list in a fair and efficient way. Their revised policy delivers an 8% relative increase in lifeyears gains, utilizing the same statistical tools and data as the

U.S. policy makers. This played a major factor in their analytical study of the fairness and flexibility of the system. They didn't consider the policies that OPTN policy makers had proposed in which patients and/or organs are categorized into different groups according to some criteria and then specific groups receive priority in the allocation.

El Haddad B. (2012) [10] focuses on the development of an Intelligent Match Making Assistant for use in Human Organ Transplantation Management. The author likely explores the integration of intelligent technologies to enhance the organ matching process. The paper may discuss the design and functionality of the Match Making Assistant, which could involve the utilization of artificial intelligence or machine learning algorithms to optimize donor-recipient matching based on various criteria such as compatibility and urgency. The goal of the system is likely to improve the efficiency and effectiveness of organ transplantation management by providing intelligent decision support during the matching process.

Ferraza. V, G. Oliveira, P. VieraMarques, and R. Cruz-Correia(2011)[11] dicussed on how to improve the organs transplantation — How to improve the process? The solution involved a creating an application using Java, MySQL and Jade to efficiently store organ data. This characterization allowed the development of a multi-agent web system, providing a way to optimize the studied information workflow. The web platform works with all the inpatients simulated digital records.

Frauenthaler, Sigwart, Spanring, Sober, and Schulte (2020) [12] focuses on the development of ETH Relay, a cost-efficient relay solution for Ethereum-based blockchains. The authors likely address the challenges associated with the scalability and performance of Ethereum by proposing ETH Relay as a mechanism to enhance the relay infrastructure. The paper may discuss the technical details and design considerations of ETH Relay, aiming to improve the efficiency of information propagation across the Ethereum network. Key aspects may include its cost-effectiveness, potential impact on transaction throughput, and the overall scalability benefits it offers to Ethereum-based blockchains.

George and Kizhakkethottam (2023) [13] conducts a survey to explore the impact of blockchain technology on effective organ transplantation. The authors likely review and analyze existing literature and implementations to provide insights into how blockchain is influencing and improving various aspects of the organ transplantation process. The survey may cover topics such as data security, transparency, traceability, and automation of processes through blockchain in the context of organ transplantation. The paper could offer a comprehensive overview of the current state of research and practical applications, outlining the benefits and challenges associated with incorporating blockchain technology into the organ transplantation domain.

Hawashin. D, R. Jayaraman, K. Salah, I. Yaqoob, M. C. E. Simsekler

and S. Ellahham (2022) [14] created a POC Dapp that employs 6 different algorithms to allow efficient and secure transactions to take place throughout the entire flow of the system. They created a Dapp solution that can be customised to needs of other systems as required. Few major disadvantages that could be identified from this robust system were its dependency on the manual selection of donor for a given recipient. It wasvery dependent on human intervention for the selection process. It also wouldn't be very scalable when faced with the large yet ever-growing datasets of the medical industry.

Hölbl. M, M. Kompara, A. Kamišalić, and L. N. Zlatolas (2018) [15] a systematic review of the use of blockchain in healthcare. Its attemptsto identify Attempts to review efficient techniques and methodologies used by blockchain solutions in healthcare. It identified scope for blockchain in healthcare industry. It didn't identify new ways to implement and exploit blockchain technology's use in healthcare

Jain. U (2020) [16] proposed using blockchain technology for theorgan procurement and transplant network. The author developed a private Dapp using HyperLedger Fabric and performed empirical study on the performance of the app. It was deduced that organ matching performed outside the chain was more efficient. The organ matching schemes wereprimitive decision trees and couldn't handle complex scenarios.

Khatal, Rane, Patel, Patel, and Busnel (2019) [17] introduces "FileShare," a framework that combines blockchain and IPFS (InterPlanetary File System) to establish a secure and transparent platform for file sharing with a focus on data provenance. The authors likely detail the technical aspects of the framework, discussing how blockchain enhances security and transparency in file sharing, while IPFS is utilized for decentralized and distributed storage. The paper may address challenges related to data provenance and integrity, proposing solutions for maintaining an immutable and traceable record of file

sharing activities. FileShare is anticipated to offer a novel approach to secure and verifiable file sharing, leveraging the strengths of both blockchain and IPFS technologies.

Malik, Manzoor, Ylianttila, and Liyanage (2019) [18] conducts a performance analysis of blockchain-based smart grids, specifically exploring implementations with Ethereum and Hyperledger frameworks. The authors likely delve into the technical intricacies of integrating blockchain technology into smart grids and compare the performance of Ethereum and Hyperledger in this context. The paper may discuss aspects such as transaction throughput, latency, scalability, and resource utilization to evaluate the efficiency and suitability of each blockchain platform for smart grid applications. Insights from this analysis could be valuable for designing robust and effective blockchain solutions in the domain of smart grids, providing a foundation for further advancements in decentralized energy systems.

Mattei. N, A. Saffidine, and T. Walsh (2017) [19] discussed mechanisms for online organ matching. They proposed a new method MIN for organ matching and compared it to the current mechanism BOX. MIN proved to be more efficient with federal treatment in this empirical study.This was an entirely empirical evaluation of post-transplant success could incorporate more complex features like those found in the SRTR.

Pacheco, Pinheiro, Cadeiras, and Menezes (2017) [20] focuses on characterizing organ donation awareness by analyzing social media data. The authors likely employ data engineering techniques to gather and analyze content from social media platforms, aiming to understand the patterns and dynamics of organ donation discussions in online communities. The paper may discuss the methods used to extract relevant information, identify key themes, and assess the overall awareness and sentiment surrounding organ donation. Insights derived from this study could contribute to a better understanding of

public discourse on organ donation and potentially inform strategies to enhance awareness and engagement through social media channels.

Ranjan. P, S. Srivastava, V. Gupta, S. Tapaswi and N. Kumar (2019) [21] was a decentralised and distributed system for organ/tissue donation and transplantation. They built a Dapp that allowed donors to register themselves, and were approved by a hospital. One of the major improvements made was the data was stored efficiently.

Rao, Behara, and Agarwal (2014) [22] focuses on predictive modeling for organ transplantation outcomes, presented at the 2014 IEEE International Conference on Bioinformatics and Bioengineering. The authors likely delve into the application of advanced computational techniques to predict the outcomes of organ transplantation procedures. The paper may discuss the methodologies employed in predictive modeling, potentially incorporating bioinformatics and data-driven approaches to analyze factors influencing transplantation success. Insights derived from this research could contribute to improving preoperative assessments and decision-making processes, ultimately enhancing the overall success rates and efficiency of organ transplantation procedures.

Richard, Surya, and Wibowo (2020) [23] explores the convergence of artificial intelligence (AI) and blockchain technology, specifically focusing on the integration of Oracle Contracts in the Ethereum Blockchain platform. The authors likely investigate the synergies between AI and blockchain, leveraging Oracle Contracts to enhance data connectivity and interoperability. The paper may discuss the technical aspects of integrating these technologies, addressing challenges and proposing solutions for creating a more efficient and secure decentralized system. Insights from this research could contribute to the development of advanced applications that harness the combined capabilities of AI and blockchain, showcasing the potential of such convergence for decentralized and intelligent systems.

Sankar. L. S, M. Sindhu, and M. Sethumadhavan (2017) [24] was a survey of consensus protocols that were used on blockchain applications. They proposed to provide asymptotic security and flexible trust by introducing the concept of quorum slices that ensures more freedom to the users on deciding the participants they need to trust for validating their transactions. This paper identified the difficulties and lack of freedom users that aren't that experienced may run into. It suggests and points to upcoming tech and industry improvements that will take place. It is merely a tutorial to blockchain, and serves as a way to get introduced to blockchain frameworks and technologies. No attempt was made to implement a Dapp using methodologies and technologies mentioned.

Shreya Khatal, Jayant Rane, Dhiren Patel, Pearl Patel, Yann Busnel (2019) [25] was that on a different blockchain application. They proposed an application 'Fileshare' a secure, tamper proof model for sharing files in a distributed file system. It used IPFS framework for Secure File Sharing and Data Provenance. Advantages to this Dapp solution was that the data can be used to obtain analytical information. The file owners can obtain analytical information about number of people who viewed the file. The Dapp was developed simply for the purposed of file sharing. The scalability of Dapp systems with IPFS wasn't exploited to its fullest extent. Their work shows the importance the IPFS framework plays in the scalability of a blockchain application.

Soni, A., Kumar, D. S. G. (2021) [26] focused on creating an Organ Donation System with Blockchain Technology. It was a Web-based Application which uses FIFO approach to select an organ donor for each genuine patient requiring a transplant and if there is an emergency case then the priority is given to that patient. It provides an efficient platform for potential organ donors and those who need the organs to connect. It uses Blockchain as its underlying Technology. Emergency cases were always given priority. Its

advantage was that it didn't allow any third party access. It eliminated any chance for corruption in the organ donation department. Few areas it fell short in were that it was unable to store large and ever-growing data of the health industry in a secure, cost efficient and scalable manner. Also, that it required human intervention for organ donor selection for a givenrecipient.

Wijayathilaka. P. L., P. H. P. Gamage, K. H. B. De Silva, A. P. P. S. Athukorala, K. A. D. C. P. Kahandawaarachchi and K. N. Pulasinghe (2020) [27] proposed a Secured, Intelligent Blood and Organ Donation Management System - "LifeShare". It is a nation-wide blood and organ bank which can share within any blood bank connects through the general hospitals as well as for private hospitals by including their relevant databases. It was very dependent on human intervention for organ donor selection for a given recipient. This helped set right and changed discrepancies and malpractices in existing procedures. One major drawback tothis proposed system was that there were no specific systems are in place in the Sri Lankan healthcare sector, to map blood groups volumes required and location tracking.

Wang, Cui, and Hou (2022) [28] presents a dynamic load balancing scheme based on network sharding in a private Ethereum blockchain, as discussed at the 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC). The authors likely address the challenges of load distribution within private Ethereum blockchain networks, proposing a dynamic sharding mechanism to optimize resource allocation and improve overall network performance. The paper may discuss the technical details of the proposed scheme, highlighting its effectiveness in enhancing scalability and efficiency in private Ethereum blockchain environments. Insights from this research could contribute to the development of more robust and scalable private blockchain systems, offering potential applications in various domains.

X. Ren, Fu, and Marcus (2023) [29] explores sensitivity analysis for stopping criteria with a specific application to organ transplantations. Presented at the 2023 Winter Simulation Conference (WSC), the authors likely investigate the impact of various stopping criteria on simulation outcomes, particularly in the context of organ transplantations. The paper may discuss how different criteria for determining when to stop a simulation affect the results and reliability of the simulation model. By addressing the sensitivity of stopping criteria, the authors aim to enhance the robustness and accuracy of simulation-based studies in the field of organ transplantations. The insights gained from this research may contribute to refining simulation methodologies and improving decision-making processes in organ transplantation scenarios.

Zheng, Zibin & Xie, Shaoan & Dai, HongNing & Chen, Xiangping & Wang, Huaimin (2017) [30] was one that served as an overview of blockchain technology as a whole. They discussed and analysed new architectures and the general consensus on them. They also discussed the future scope and trends that will come to be in the blockchain domain. it served as a comprehensive overview on blockchain. The authors didn’t implement any application, only analysed the theoretical strengths and weaknesses of any given algorithm. However, despite its comprehensive nature, Zheng et al.'s work (2017) doesn't delve into the practical implementation aspects of blockchain technology. While it explores the theoretical underpinnings of various consensus algorithms and architectures, further research is needed to bridge the gap between theory and application. This could involve analyzing real-world case studies of blockchain deployments, evaluating their performance metrics, and identifying areas for optimization. Such practical insights would be valuable for developers and businesses seeking to leverage the potential of blockchain technology in various domains

CHAPTER 3
SYSTEM DESIGN

In this chapter, the various UML diagrams for the Organ Donation and Management system using blockchain and machine learning is represented and the various functionalities areexplained.

3.1 UNIFIED MODELING LANGUAGE

Unified Modeling Language (UML) is a standard modeling language used in software engineering to visualize, design, and document software systems. It is a graphical language that consists of a set of diagrams and symbols used to represent the various components and interactions within a software system.

The UML was developed by a consortium of software companies and experts in the field of software engineering to provide a standard way of representing software systems. It is widely used in the software development industry as a means of communicating system designs and requirements to various stakeholders, including developers, business analysts, and project managers.

The UML includes various types of diagrams, each of which represents a different aspect of the software system. Some of the most commonly used UML diagrams include:

In conclusion, UML is a powerful tool that helps software engineers to design, document, and communicate complex software systems. Its standardized notation and variety of diagrams make it an effective way to represent different aspects of a software system and ensure that all stakeholders have a common understanding of the system.

3.1.1 USE CASE DIAGRAM

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So it can be said that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actorscan be defined as something that interacts with the system. The actors can be human user, some internal applications or may be some external applications.

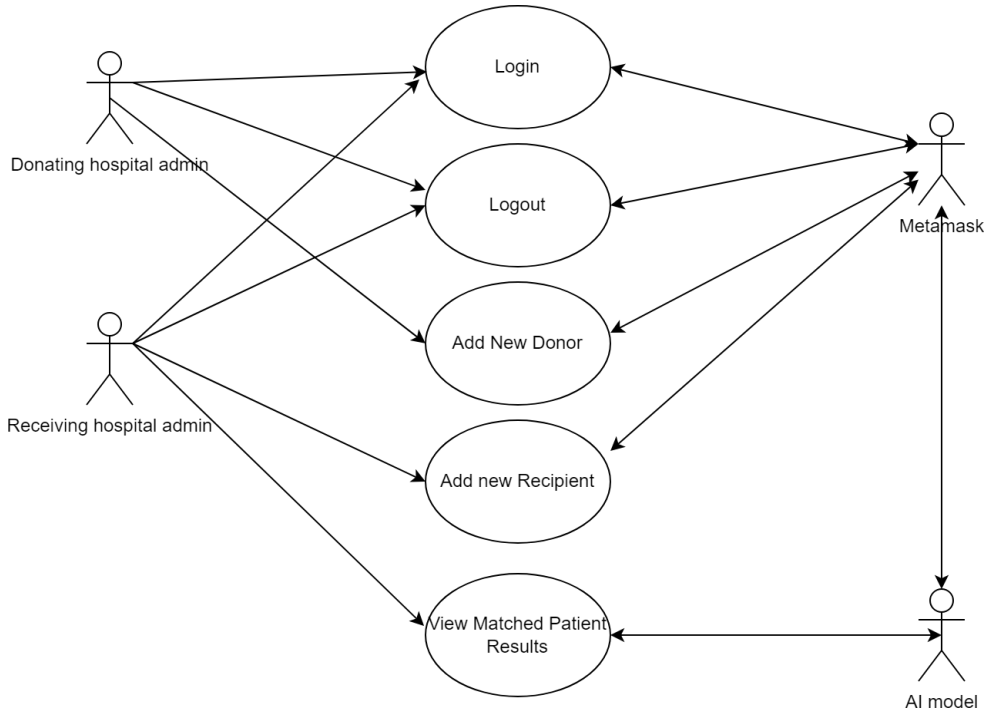

Figure 3.1 Use Case Diagram

Figure 3.1 shows the primary functionalities provided to users. There are 2 primary actors the donating hospital admin and receiving hospital admin, their roles are reversible. There are also 2 supporting actors who are the smart contract and the machine learning model. These two actors are quintessential to the completion of each use case. Every use case has to interact with the smart contract, as the smart contract is used to execute transactions on the blockchain.

3.1.2 SEQUENCE DIAGRAM

Sequence diagrams are graphical representations used in software engineering to illustrate the interactions between various components or objects within a system over time, showcasing the flow of messages exchanged to accomplish specific tasks or scenarios. They are essential for understanding system behavior, aiding in system architecture design, communicating functionality to stakeholders, identifying use cases and requirements, and facilitating testing and debugging processes.

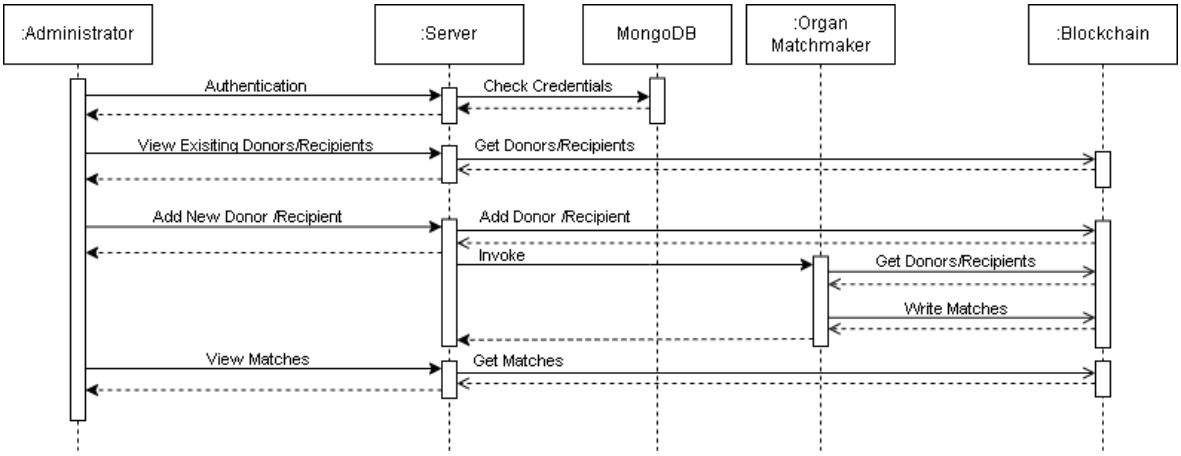

Figure 3.2 Sequence Diagram of the BODM System

Figure 3.2 shows the Sequence diagram of the BODM System. To effectively use the BODMS the administrator must first authenticate themselves with the system with the correct credentials. When the administrator views the existing donors or patients, the application server asynchronously makes a read request to the private blockchain. The response is synchronously displayed to the user by the application. When a new patient is to be added to the system, the server receives the details from the user and makes an asynchronous write request to the blockchain. The server then invokes the organ matchmaker to label the new donor/ recipient based on existing data on the blockchain. The respective matches are then written on the blockchain. When the administrator chooses to view matches made, an asynchronous read call is made to the blockchain. The response is then displayed by the browser application.

3.1.3 COLLABORATION DIAGRAM

Collaboration diagrams, also referred to as communication diagrams, are graphical representations used in software engineering to depict interactions between objects or components within a system or software application Each object is represented as a labeled box, with arrows indicating the flow of messages or communications between them. Collaboration diagrams are valuable for visualizing the relationships and dependencies between objects, facilitating system design, communication among development teams, and comprehension of system behavior. They offer a high-level overview of how objects interact, making them particularly useful during the early stages of system design and architecture planning.

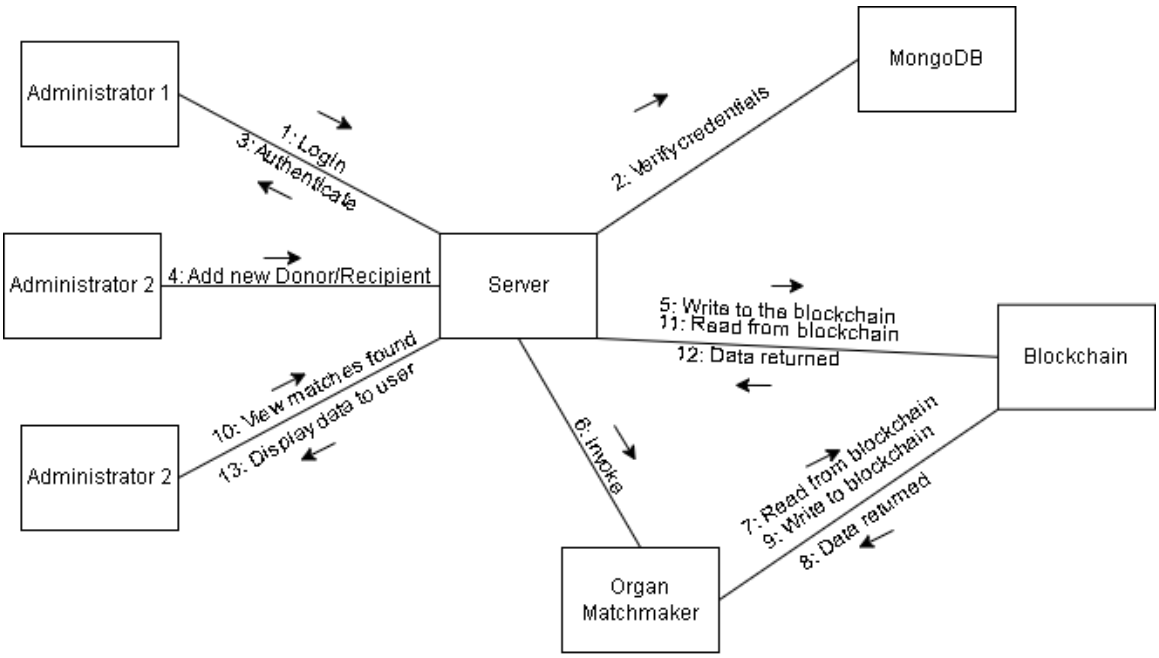

Figure 3.3 Collaboration Diagram of the BODM System

Figure 3.3 shows the collaboration diagram of the BODM System. It visualizes the relationships and dependencies between objects, entities and actors within the system. Additionally, collaboration diagrams serve as effective documentation tools, aiding in the understanding and maintenance of complex software systems throughout their lifecycle. It highlights the communications made between different objects within the system.

3.1.4 ACTIVITY FLOW DIAGRAM

Activity flow in Unified Modeling Language (UML) is a powerful representation that depicts the dynamic behavior of a system through a series of interconnected activities. It serves as an essential tool for visualizing and understanding complex processes within software or business systems. At its core, an activity flow diagram consists of activities, which represent specific actions or tasks, and control flow elements, such as decision points and transitions, that guide the sequence of activities. Activity flow diagrams are particularly useful for modeling business processes, software workflows, and system behaviors, aiding in analysis, design, and communication among stakeholders.

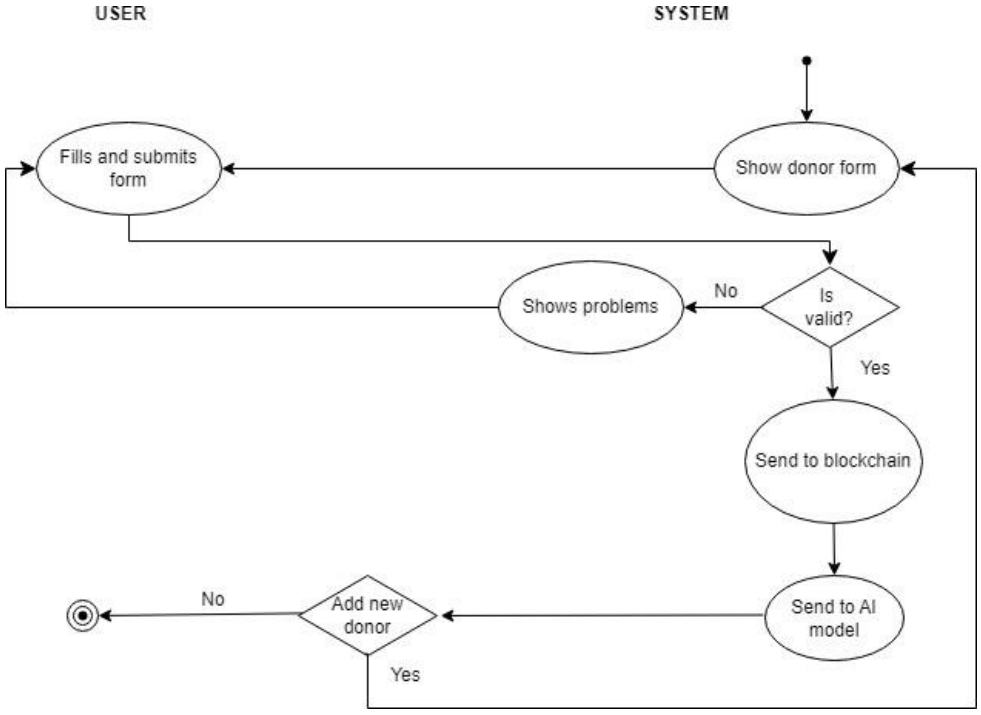

Figure 3.4 Activity Flow Diagram of Creating New Donor

Figure 3.4 provides a clear visualization of activity sequences and decision logic, they facilitate understanding, optimization, and improvement of complex processes and systems. It shows the states that system will be in when the donationflow takes place. First, the user is displayed a donor form which is filled and submitted.

This data is sent to the blockchain, which in turn sends it to the ML model. The user is then given and option to add another donor. In the case that the filled details of the donor are wrong or invalid, then the invalid forms are highlighted, allowing the user to re submit details.

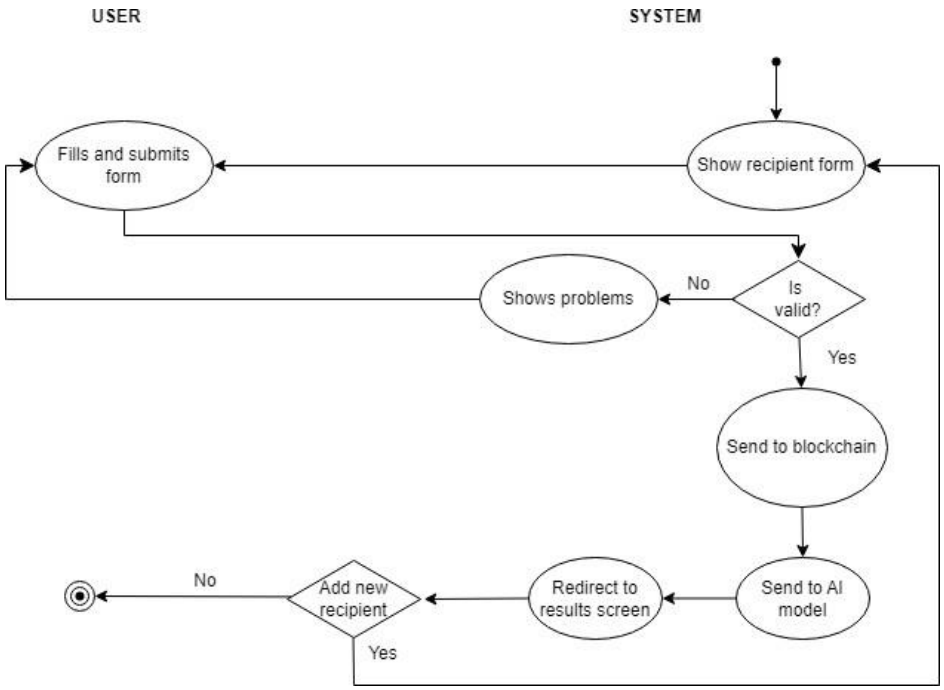

Figure 3.5 Activity Flow Diagram of Creating New Recipient

Figure 3.5 shows that the user is displayed a donor form which is filled and submitted. This data is sent to the blockchain, which in turn sends it to the ML model. The user is then given and option to add another donor. In the case that the filled details of the donor are wrong or invalid, then the invalid forms are highlighted, allowing the user to re submit details.

In conclusion, the Activity Flow diagram in UML is a dynamic and flexible tool that empowers analysts, developers, and stakeholders to capture, visualize, and refine the intricate activities and interactions within a system By illustrating the sequence of activities, decisions, and transitions, they help identify bottlenecks, streamline operations, and improve efficiency, ultimately leading to more effective and productive outcomes.

CHAPTER 4

SYSTEM ARCHITECTURE

In this chapter, the System Architecture for the Blockchain Organ Donation Management using Machine Learning is represented and the modules are explained

4.1 SYSTEM ARCHITECTURE DIAGRAM

The “Blockchain based Organ Donation Management using Machine Learning” is a decentralised application that aids the hospital administration in finding the right organ donor match for an organ recipient. A system architecture provides a high-level overview of the system's design, including hardware, software, networks, and interfaces

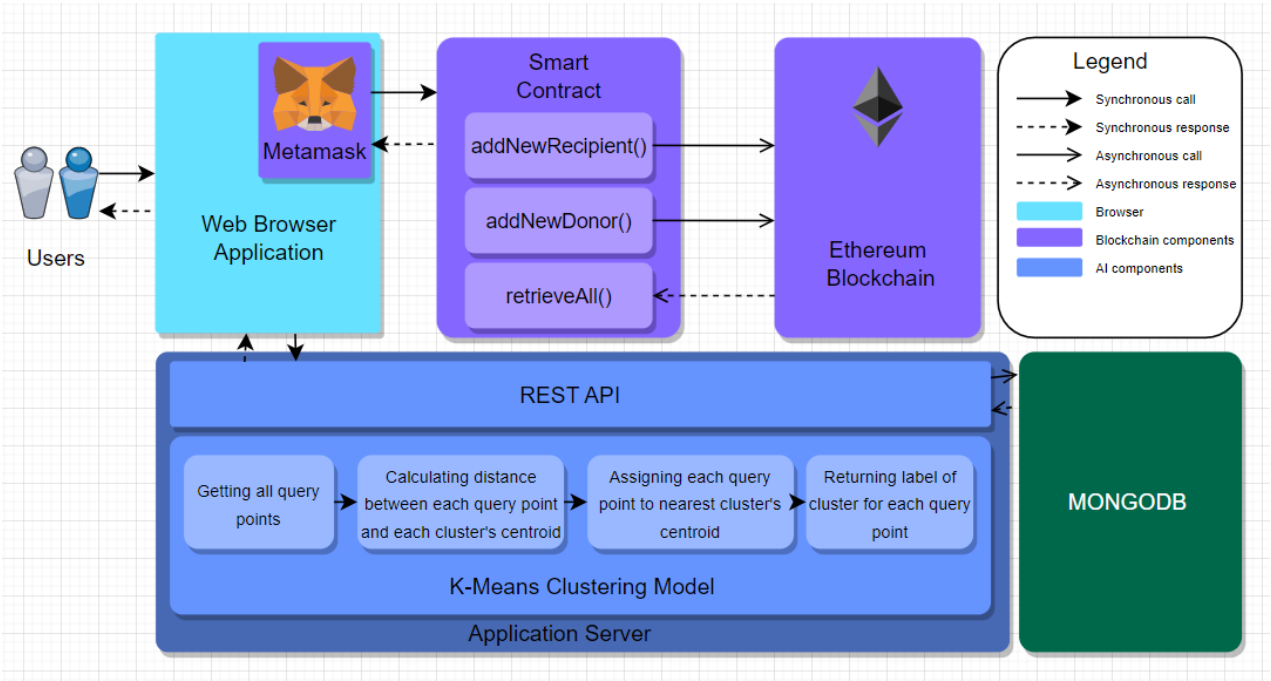

Figure 4.1 System Architecture Diagram

Figure 4.1 serves as a visual representation of the components and structure of a system, illustrating how various elements interact to achieve specific functionalities or goals. This diagram helps stakeholders, such as developers, designers, and project managers, to understand the system's overall layout,

dependencies, and flow of data or processes. By depicting the relationships between different components, it facilitates communication, decision-making, and troubleshooting throughout the development lifecycle.

4.2 ARCHITECTURE DESCRIPTION

The “Blockchain based Organ Donation Management using Machine Learning” is a decentralised application that aids the hospital administration in finding the right organ donor match for an organ recipient. The administrator can add organ donors or recipients by entering their details. The system identifies the best match for each of the recipients, with no need of human intervention. If two recipients have same donor as a match, then the recipients are matched in FIFO (First In First Out) order. If no match is found for a recipient, then they are placed in a waiting list. When new donors are added to the system, the recipients in waiting list are checked with the newly added donors.

The system follows MVC (Model-View-Controller) a typical architecture. Figure 4.2 shows the architecture diagram. The ‘View’ displays the system’s interface and allows user to interact with the system. The ‘Controller’ serves as a business logic layer that facilitates the system’s operations based on user input and interactions. The ‘Model’ layer stores and retrieves the system’s data, this includes user input data, user credentials, derived and resultant data.

4.2.1 MODEL

The Model layer is accessed whenever any form of data storage or retrieval is necessary. This layer consists of 2 data sources: [1] MongoDB instance to store user credentials [2] Solidity contract on the Ethereum blockchain’s Sepolia testnet, that stores relevant donor and recipient data on the

blockchain securely. The MongoDB instance is accessed only for creating, storing and retrieving hospital administrator credentials. When new donors or recipients are added to the system, they are stored and retrieved from the smart contract deployed on the Sepolia testnet.

4.2.2 VIEW

The view component in the Model-View-Controller (MVC) pattern holds significant importance in software development as it represents the user interface of the application. It's responsible for presenting data from the model to the user in a visually appealing and understandable format. The view plays a crucial role in enhancing user experience by providing an interactive and intuitive interface through which users can interact with the underlying system.

The application's view, i.e., is developed using React.js. The View can be broken down into many modules. These modules are:

a) User Login: This module is responsible for handling user authentication. The hospital administrator enters their user credentials to enter the system. The details entered by the user are captured and sent to Controller to verify and authenticate the user. The View sends the user's login details to the Controller, who cross verifies the login credentials with the Model (MongoDB).

b) User Registration: The Registration module handles the registration of new Hospital Administrators into the system. Each hospital administrator and set their own usernames and passwords. Additionally, they must provide their organisation's name. The organisation name will be appended to all donors and recipients created by that hospital administrator. The View captures all the user input and sends it to the Controller. Necessary actions and calls are made to create accounts for hospital administrators.

c) Add New Donor: This module allows users, i.e., Hospital

administrators to add new donors to the system. The hospital administrator must enter donor details such as Name, Age, Gender, Blood type, Tissue type, organ to be donated, organ size, organ lifetime. Once all details have been entered the captured details are sent from the view to the controller. The Controller then performs necessary operations to store the details in the Model (deployed smart contract).

d) View Existing Donor: The View existing donor module allows the user to see all the registered donors in the system. The view here only displays data retrieved from the model (deployed smart contract). The controller simply fetches data from the model and sends it to the view.

e) Add New Recipient: This View module enables the Hospital administrator to add/register potential recipients in need of and organ transplantation. The hospital administrator must enter required recipient details such as Name, Age, Gender, Blood type, Tissue type, organ in need, organ size. When user has entered all the required details, the data is then sent to the Controller so that the recipient data passed may be securely stored in the Model (deploy smart contract).

f) View Existing Recipient: This module allows users to view/see all the recipients registered in the system. The module simply displays all of the recipients along with their respective registered details. The controller simply retrieves the recipient details in the Model (deployed smart contract).

g) View Recipient Matches: Once a recipient has been registered in the system, the ideal match available from the system's organ bank is selected. The matches for each recipient is retrieved and displayed by this view module.

4.2.3 CONTROLLER

The controller facilitates the separation of concerns within the application by decoupling user interactions from the underlying data and

business logic. By intercepting and processing user input, such as button clicks or form submissions, the controller determines the appropriate actions to be taken on the model and updates the view accordingly.

It enhances the reusability of application logic by centralizing it within a single component. Business logic related to user interactions, data validation, and application workflows can be encapsulated within the controller, reducing code duplication and promoting a more consistent and maintainable codebase. This reusability simplifies testing and debugging efforts, as well as the implementation of changes or updates to the application's behavior.

The controller layer utilizes Node.js, Express.js and Solidity to facilitate the many operations of the system. The controller layer consists mainly of 2 components: Application Server and Smart Contract

a) Application Server The application server handles all application logic, model layer communication, request handling, etc. It serves as a central hub for processing and managing requests from clients, enabling efficient communication between clients and backend services. It ensures scalability, reliability, and security in distributing application functionality to multiple users simultaneously. The application server has 2 subcomponents that make up its core functionality: the REST API (Representational State Transfer Application Programming Interface) and the machine learning model.

1) REST API: The REST API acts as a means for the server to handle HTTP(S) (Hypertext Transfer Protocol) requests from the View, i.e., the application’s frontend. The application’s Rest API mostly only works with GET/POST HTTP requests. When a user performs an action that involves submitting data to the application, a POST request is sent to the REST API. Based on which route the request was sent to the operations performed by the REST API differ.

2) Machine Learning Model: The Machine Learning model chosen to train the model is K – Means Clustering algorithm. It is a flexible and popular unsupervised machine learning method that divides a dataset into identifiable groups or clusters. The method works by dividing data points into clusters iteratively according to how close they are to the centroids—the clusters' core locations. As 'K,' or the number of clusters, is predefined, it is essential to the algorithm's performance. K-means optimizes cluster cohesiveness by minimizing the sum of squares within a cluster.

The K- Means model was trained with a synthesized dataset of 500 entries. The CSV dataset had the features Name, Age, Gender, Blood type, Tissue type, organ size.

A synthetic dataset was used as the medical data required is not made publicly available by medical organizations as it serves as a breach of privacy. The synthetic dataset was generated with the help of a GAN (Generative Adversarial Network). The GAN used to synthesize our dataset was made using ‘gretel.ai’.

b) Smart Contract The smart contract is written in Solidity, and it acts as a set of rules for that must be followed when a transaction takes place in the application. Here, a transaction would be the addition of an organ donor (or) the addition of an organ recipient. The rules set in the smart contract are automatically executed when all fields required by the contract’s ABI (Application Binary Interface). The ABI serves as a means of communication with the deployed smart contract, from outside and within the contract.

The smart contract contains the following methods: (1) Add New Donor (2) Add New Recipient (3) View All Donors (4) View All Recipients (5) Delete Donor (6) Delete Recipient.

1) Add New Donor: This method is invoked when a new organ donor is

added to the system. It takes the donor's Name, Age, Gender, Blood type, Tissue type, Organ, Organ Size as input parameters. The new Donor will then be added to the local array stored in the block's memory. After a donor is successfully added, a 'donor added' event is emitted by the contract.

2) Add New Recipient: This method is similar to the add new donor method. It is invoked when a new organ recipient is added to the system. It takes the recipient's Name, Age, Gender, Blood type, Tissue type, Organ, Organ Size as input parameters. The new recipient will then be added to the local array stored in the block's memory. After a donor is successfully added, a 'recipient added' event is emitted by the contract.

3) View All Donors: This method returns all the donors stored in the block's memory. It is invoked by the application's frontend when user views existing donors in the application.

4)View All Recipients: This method returns all the recipients stored in the block's memory. It is invoked by the application's frontend when user views existing recipients in the application.

5) Delete Donor: This method deletes an individual donor from the list of donors. It accepts an input parameter of Id, which specifies the ID of the donor to be deleted. It is invoked by the application's backend when a match has been found.

6) Delete Recipient: This method deletes an individual recipient from the list of recipients. It is invoked by the application's backend when a match has been found

When the user creates a new recipient in the system, the smart contract will retrieve all available donors list along with the new recipient created. All this data will be sent to an application server in which the Machine learning model will be running.

CHAPTER 5

SYSTEM IMPLEMENTATION

The system was implemented with a modular approach. The application's features were each selected and broken down into individual modules. These modules themselves are fullstack modular components, in the sense that each module encapsulates the user interface and the server-side logic necessary to fulfill a specific functionality or feature.

A full-stack module in a web application might include the frontend code responsible for rendering a user interface and collecting user inputs, as well as the backend code handling data processing, storage, and communication with databases or external services. This integrated approach allows for better organization and encapsulation of related functionalities within a single module, simplifying development, maintenance, and scalability.

5.1 MODULE DESCRIPTION

The modules used in the blockchain based organ donation and transfer management each perform a functionality from start to finish. Each of the modules contain a combination of reusable submodules that have been used on other modules of the application system. prioritizes the reuse and distribution of components, emphasizing a design philosophy centered around efficiency and collaboration. The application system was built with component reusability and sharing in mind. Monolithic code blocks were avoided to prevent redundant repetition of code. The modules used in the application system are as follows:

- User Authentication
- Wallet Integration
- Donor Creation
- Recipient Creation
- Donor – Recipient Matching

5.1.1 USER AUTHENTICATION

The user authentication module takes care of the entire lifecycle of a user login session. It consists of:

a. Frontend (React.js):

- LoginForm Component: Create a React component with fields for username and password.
- API Calls: Use Axios to send login credentials to the backend for authentication.
- State Management: Implement state management to handle user input and authentication status.

b. Backend (Node.js with Express.js and MongoDB):

- MongoDB Schema: Define a simple user schema with fields like username and password in MongoDB using Mongoose.
- Express Routes: Set up a route for handling login requests (e.g., "/api/login").
- Controller: Create a controller function that queries the database to check if a user with the provided username and password exists.
- Authentication Logic: If a matching user is found, generate a simple authentication token or set a session cookie indicating successful login.
- Middleware: Implement middleware to check the authentication status for protected routes.

c. Database (MongoDB):

- User Schema: Define a user schema with fields like username and password.
- Database Connection: Establish a connection to the MongoDB database using Mongoose.
- Queries: Implement queries to find a user.

5.1.2 WALLET INTEGRATION

This module is strictly frontend only. It plays a crucial role in the functioning of the application as it facilitates connecting to any chain and perform transactions. It is a digital tool used to securely store, manage, and interact with cryptocurrency assets on a blockchain network. It consists of:

1. Frontend (React.js):
    - Metamask Check: Begin by checking if Metamask is installed in the user's browser. You can do this by looking for the window.ethereum object.
    - Connect Button: Create a "Connect Wallet" button that, when clicked, triggers the Metamask connection process.
    - Connection Logic: Implement logic to request user permission to connect the DApp to their Metamask account. Use the ethereum.enable() method or the new ethereum.request({ method: 'eth_requestAccounts' }) for the user to grant permission.
    - Account Display: Display the user's Ethereum address after a successful connection.

The connected wallet is used by every other module in a part of their functioning.

5.1.3 DONOR CREATION

The donor creation module handles the instantiation of new donors, and their storage on the chain. This module involves a little bit of frontend, backend and the smart contract itself. It is as follows:

a. Solidity Smart Contract:
    - Define a Solidity smart contract with a struct, Donor, containing fields like id, name, age, blood type, etc.
    - Implement a function addDonor, that creates a new instance of Donor and adds it to a public array or mapping.

b. Backend (Node.js with Express.js and MongoDB):

- Create a Node.js server using Express.js to handle HTTP requests.
- Use the web3.js library to connect to the Ethereum network and interact with the deployed Solidity smart contract.
- Set up endpoints for adding a donor (/donate/add) and fetching the donors (/donate/view).
- When a request to add a donor is received, use the web3.js library to call the addDonor function on the smart contract.
- For fetching the donors, call the retrieveDonors function.

c. Frontend (React.js):

- Develop a React.js frontend with a user interface to trigger the addition of a new donor.
- Use Axios to send requests to the backend API.
- Provide a button or user input to initiate the creation and addition of a new donor.
- Display relevant information fetched from the backend.

5.1.4 RECIPIENT CREATION

The recipient creation module handles the instantiation of new recipients, and their storage on the chain. This module involves a little bit of frontend, backend and the smart contract itself. It is as follows:

a. Solidity Smart Contract:

- Define a Solidity smart contract with a struct, Recipient, containing fields like id, name, age, blood type, etc.
- Implement a function addRecipient, that creates a new instance of Recipient and adds it to a public array or mapping.

b. Backend (Node.js with Express.js and MongoDB):

- Create a Node.js server using Express.js to handle HTTP requests.

- Use the web3.js library to connect to the Ethereum network and interact with the deployed Solidity smart contract.
- Set up endpoints for adding a donor (/receive/add) and fetching the recipients (/receive/view).
- When a request to add a recipient is received, use the web3.js library to call the addRecipient function on the smart contract.
- For fetching the recipients, call the retrieveRecipients function.

c. Frontend (React.js):

- Develop a React.js frontend with a user interface to trigger the addition of a new recipient.
- Use Axios to send requests to the backend API.
- Provide a button or user input to initiate the creation and addition of a new recipient.
- Display relevant information fetched from the backend.

5.1.5 DONOR – RECIPIENT MATCHING

This is the most crucial module in the application. It consists of many submodules.

a. Node.js Backend (Express.js):

- Set up an Express.js server to handle HTTP requests from the frontend.
- Use the web3.js library to connect to the Ethereum network and interact with the deployed Solidity smart contract.
- Create an endpoint to fetch data from the Solidity contract array.
- Upon receiving a request, call the appropriate Solidity contract function and return the data to the frontend.

b. Python Script:

- Develop a Python script that uses the K-Means clustering model to perform cluster labelling on candidates. This script can be used to

analyze, transform, or process data from the Solidity contract.

- Accept input parameters are passed to the script by the calling process from the Node.js backend.

c. Node.js Backend (Express.js) Integration with Python Script:

- Integrate the Python script into the Express.js backend using child processes or a suitable method for running external scripts.
- Define an endpoint in Express.js that triggers the execution of the Python script with relevant input parameters.
- Capture the output or results from the Python script and send them back to the frontend or store them in the database.

d. React.js Frontend:

- Develop a React.js frontend with a user interface to initiate the data processing task by interacting with the backend.
- Implement a button or user input to trigger the execution of the Python script via the Express.js backend.
- Display the results or updates received from the backend, providing a user-friendly interface.

The machine learning model is a pre-trained model stored as a ‘pickle’ file. The Model is loaded into memory and run by the application server as and when required. The K-Means clustering model used is has a K value of 150. This K value was selected using Elbow Point method. The K value wasn’t chosen at any value higher than 150, as it leads to overfitting of the model. The K- Means model was trained with a synthesized dataset of 500 entries. The CSV dataset had the features Name, Age, Gender, Blood type, Tissue type, organ size. A synthetic dataset was used as the medical data required is not made publicly available by medical organizations as it serves as a breach of privacy. The synthetic dataset was generated with the help of a GAN (Generative Adversarial Network). The GAN used to synthesize our dataset was made using ‘gretel.ai’

CHAPTER 6

RESULTS AND CODING

The Organ donation and transfer management system using blockchain and machine learning is a cutting-edge application system that seamlessly integrates the power of the MERN (MongoDB, Express.js, React.js, Node.js) stack, Solidity, and Python to deliver a robust and efficient user experience. This innovative system combines the strengths of various technologies to cater to diverse functionalities.

The MERN stack forms the backbone of the application, providing a full-stack JavaScript framework for building dynamic web applications. MongoDB ensures flexible and scalable data storage, while Express.js simplifies server-side development. React.js delivers a responsive and interactive user interface, and Node.js facilitates server-side execution.The application also makes use of the Material UI component library for open source, free to use UI components. Ethers.js is a JavaScript library simplifying Ethereum blockchain interaction. With a clean API, it eases tasks like contract deployment and transaction handling. Moralis is a blockchain development platform streamlining decentralized app (DApp) creation. Offering backend infrastructure, it simplifies tasks like user authentication, real-time updates, and blockchain integration.

Solidity, a language designed for smart contracts on the Ethereum blockchain, adds a decentralized layer to the application. This enables secure and transparent transactions.

Python, a versatile and widely-used programming language, was used to train the machine learning model used to cluster donors and recipients. It proves critical to the functioning of the system by offering additional backend layer to smoothly access the trained model and giving other quality of life functionalities and enhancing overall system efficiency.

6.1 SAMPLE CODE

server.js

```
require('dotenv').config()
const express = require(`express`)
const cors = require(`cors`)
const fs = require('fs');
const mongoose = require(`mongoose`)
var ObjId = require('mongoose').mongo.BSON.ObjectId;
const User = require('./userModel.js');
const app = express()
const path= require('path');
const { error } = require('console');

app.use(cors())
app.use(express.json())

app.use(express.static(path.resolve(__dirname, '../frontend/build')));
const port =3001

mongoose.connect(process.env.MONGODB_ACCESS_URL,{useNewUrlPar
ser: true, useUnifiedTopology: true})
.then(()=>{
    console.log("connected to mongodb atlas")
    app.listen(port,()=>{
```

```
        console.log(`Node server is running on port ${port}`)
    })
})
.catch((e)=>{
    console.error(e)
})

app.get("/*",(req,res)=>{

    res.sendFile(
        path.resolve(__dirname,'../frontend/build/index.html'),
        function (err){
            if(err){
                res.status(500).send(err)
            }
        }
    )
})


app.post(`/login`,async (req,res)=>{
    try{
        const userEntry = req.body
        const dbEntry= await User.find({username:`${userEntry.username}`})
```

```
        console.log('received data from user')
        if(dbEntry==[]){
            console.log('no such username exists')
            return res.status(404).json({"error": "wrong username"})
        }
        if(userEntry.username=== dbEntry[0].username && userEntry.password
=== dbEntry[0].password){
            console.log("login success");
            return
res.status(200).json({"organisationName":`${dbEntry[0].organisationName}`
});
        }else{
            console.log('wrong password')
            return res.status(400).json({"error":"wrong password"})
        }
    }catch(err){
        console.log(err)
    }
})

app.post(`/register`,async (req,res)=>{
    try{
        if(await User.findOne({username:`${req.body.username}`})===null){
            console.log('no such username exists, can be registered')
            const user = await User.create(req.body)
```

```
            return res.status(200).json(user)
        }else{
            console.log(                                    await
User.find({username:`${req.body.username}`}).length)
            console.log('username already taken')
            return res.status(400).json({"error":"username taken"})
        }


    }catch(err){
        console.log(err)
    }
})


const spawn = require("child_process").spawn;
const          windowsPath='D:\\Tejas\\SEM7\\final-year-project\\final-org-
app\\backend\\scripts\\Scripts\\python'
const ubuntuPath='/home/wsdev88/t/final-org-app/backend/scripts/bin/python'

app.post(`/donate/add`,async (req,res)=>{
    try{
        let id=req.body.id
        let name=req.body.name
        let age=req.body.age
```

```
let gender=req.body.gender
let locality=req.body.locality
let bloodtype=req.body.bloodtype
let tissuetype=req.body.tissuetype
let organ=req.body.organ
let size=req.body.organSize

const donor={
   'id':id,
   'name':name,
   'age':age,
   'gender':gender,
   'locality':locality,
   'bloodtype':bloodtype,
   'tissuetype':tissuetype,
   'organ':organ,
   'size':size
}
//const matchJson=

//const pythonScript = spawn(windowsPath,["./scripts/classifier.py",id,name,organ,locality,bloodtype,tissuetype,gender,age,size,"recipient"]);
const pythonScript = spawn(windowsPath,["./scripts/classifier.py",id,name,organ,locality,bloodtyp
```

```
e,tissuetype,gender,age,size,"donor"],{cwd: __dirname});
        pythonScript.stdout.on('data', (data) => {
            console.log(data.toString());
        });

        return res.send("successfully processed by script, and cluster identified")

    }catch(err){
        console.log(err.toString())
    }
})

app.post(`/receive/add`,async (req,res)=>{
    try{
        let id=req.body.id
        let name=req.body.name
        let age=req.body.age
        let gender=req.body.gender
        let locality=req.body.locality
        let bloodtype=req.body.bloodtype
        let tissuetype=req.body.tissuetype
        let organ=req.body.organ
        let size=req.body.organSize
```

```
const recipient={
   'id':id,
   'name':name,
   'age':age,
   'gender':gender,
   'locality':locality,
   'bloodtype':bloodtype,
   'tissuetype':tissuetype,
   'organ':organ,
   'size':size
}
//const matchJson=
const pythonScript = spawn(windowsPath,["./scripts/classifier.py",id,name,organ,locality,bloodtype,tissuetype,gender,age,size,"recipient"],{cwd: __dirname});

//gets result json from python script
const queryResultJson=JSON.parse(fs.readFileSync("./assets/match.json", 'utf8'));

recipient['matches']=queryResultJson
// opens results.json and appends new recipient as key with results as value
fs.readFile('./results.json', (err,data)=> {
   let json = JSON.parse(data)
```

```
            json.table.push(recipient)


            fs.writeFile("results.json", JSON.stringify(json),'utf8',()=>{})
        })
        // const matchson=JSON.parse(fs.readFileSync("./assets/match.json",
'utf8'));
        return res.send("successfully processed by py script, and results found")


    }catch(err){
        console.log(err.toString())
    }
})


app.get(`/receive/matches`, async (req,res)=>{
    try{
        fs.readFile('./results.json', (err,data)=> {
            let allMatches = JSON.parse(data)
            res.status(200).json(allMatches)
        })
    }catch(e){
        console.error(e)
        console.log(typeof(db))
    }
})
```

SimpleStorage.sol

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.9;

// Uncomment this line to use console.log
// import "hardhat/console.sol";
contract SimpleStorage{
    //mappings work like dictionaries, there is no need for them in our project

    struct OrganDonor{
        uint256 id;
        string name;
        uint104 age;
        string gender;
        string locality;
        string bloodType;
        string tissueType;
        string organ;
        string organSize;
        string organLife; //how long the organ is good for
        string hospitalName;
    }
```

```
struct OrganRecipient{
    uint256 id;
    string name;
    uint104 age;
    string gender;
    string locality;
    string bloodType;
    string tissueType;
    string organ;
    string organSize;
    string hospitalName;
}

// Event to log when an item is added.
event DonorAdded(uint256 id,string name,uint104 age,string gender,string locality,string bloodType,string tissueType,string organ,string organSize,string organLife,string hospitalName);
event RecipientAdded(uint256 id,string name,uint104 age,string gender,string locality,string bloodType,string tissueType,string organ,string organSize,string hospitalName);

// Event to log when an item is removed.
event DonorRemoved(uint256 id,string name,uint104 age,string gender,string locality,string bloodType,string tissueType,string organ,string
```

organSize,string organLife,string hospitalName);

event RecipientRemoved(uint256 id,string name,uint104 age,string gender,string locality,string bloodType,string tissueType,string organ,string organSize,string hospitalName);

// Event to log when arrays are cleared.

event DonorsCleared();

event RecipientsCleared();

OrganDonor[] public donors;

OrganRecipient[] public recipients;

function createNewDonor(uint256 _id,string memory _name, uint104 _age,string memory _gender, string memory _locality,string memory _bloodtype,string memory _tissueType, string memory _organ, string memory _organSize,string memory _organLife, string memory _hospitalName)public{

donors.push(OrganDonor(_id,_name,_age,_gender,_locality,_bloodtype,_tissueType,_organ,_organSize,_organLife,_hospitalName)); //creating a donor object and pushing to array

emit DonorAdded(_id,_name,_age,_gender,_locality,_bloodtype,_tissueType,_organ,_organSize,_organLife,_hospitalName);

}

```
    function createNewRecipient(uint256 _id,string memory _name, uint104 _age,string memory _gender, string memory _locality,string memory _bloodtype,string memory _tissueType, string memory _organ,string memory _organSize, string memory _hospitalName)public{


recipients.push(OrganRecipient(_id,_name,_age,_gender,_locality,_bloodtype,_tissueType,_organ,_organSize,_hospitalName)); //creating a donor object and pushing to array
        emit RecipientAdded(_id,_name,_age,_gender,_locality,_bloodtype,_tissueType,_organ,_organSize,_hospitalName);

    }

    function removeDonor(uint256 _id) public{
        uint256 indexToRemove = findDonorIndexById(_id);
        require(indexToRemove < donors.length, "Struct not found");

        OrganDonor memory donorToRemove = donors[indexToRemove];
        OrganDonor memory lastDonor = donors[donors.length - 1];

        donors[indexToRemove] = lastDonor;
        donors.pop();
        emit DonorRemoved(donorToRemove.id, donorToRemove.name, donorToRemove.age,donorToRemove.gender, donorToRemove.locality,
```

```
donorToRemove.bloodType,
donorToRemove.tissueType,donorToRemove.organ,donorToRemove.organSi
ze, donorToRemove.organLife, donorToRemove.hospitalName);
    }

    function findDonorIndexById(uint256 _id) internal view returns (uint256) {
        for (uint256 i = 0; i < donors.length; i++) {
            if (donors[i].id == _id) {
                return i;
            }
        }
        return type(uint).max; // Not found
    }

    function removeRecipient(uint256 _id) public{
        uint256 indexToRemove = findRecipientIndexById(_id);
        require(indexToRemove < recipients.length, "Struct not found");

        OrganRecipient memory recipientToRemove =
recipients[indexToRemove];
        OrganRecipient memory lastRecipient = recipients[recipients.length - 1];

        recipients[indexToRemove] = lastRecipient;
        recipients.pop();
        emit RecipientRemoved(recipientToRemove.id,
```

```
recipientToRemove.name, recipientToRemove.age,
recipientToRemove.gender, recipientToRemove.locality,
recipientToRemove.bloodType,recipientToRemove.tissueType,
recipientToRemove.organ, recipientToRemove.organSize,
recipientToRemove.hospitalName);
    }
    function retrieveDonors() public view returns (OrganDonor[] memory){
        return donors;
    }
    function retrieveRecipients() public view returns (OrganRecipient[]
memory){
        return recipients;
    }

}
```

UserModel.js

```
const mongoose = require(`mongoose`)

const userSchema =mongoose.Schema({organisationName:{ type:
String,required: [true, "Please enter organisation name"]},username:{type:
String,required: [true, "Please enter username"]}, password:{type:
String,required: [true, "Please enter a password"]},},{ timestamps: true})
const User=mongoose.model('User', userSchema)
```

```
module.exports=User
```

App.js

```
import { BrowserRouter as Router, Routes, Route} from "react-router-dom"
import Appbar from "./components/appbar.jsx"
import Login from "./pages/login/login.jsx"
import Register from "./pages/register/register.jsx"
import Donate from "./pages/donate/donate.jsx"
import Receive from "./pages/receive/receive.jsx"
import Dashboard from "./pages/dashboard/dashboard.jsx"
import {Box, Grid} from "@mui/material"
import { MoralisProvider } from 'react-moralis';
import AddNewDonor from "./pages/donate/addNewDonor"
import ViewExistingDonor from "./pages/donate/viewExistingDonor"
import AddNewRecipient from "./pages/receive/addNewRecipient"
import ViewExistingRecipient from "./pages/receive/viewExistingRecipient"
import ViewMatches from "./pages/receive/viewMatches"
import Sidebar from "./components/sidebar.jsx"
function App() {
  const SidebarList=[
    {option : 'DONATE',
    icon:",
    linkTo:" },
    {option : 'Add New Donor',
    icon:'PersonAddIcon',
```

```
    linkTo:'/donate/add' },
    {option : 'View Existing Donor',
    icon: 'ViewListIcon',
    linkTo:'/donate/view' },
    {option : 'RECEIVE',
    icon:'',
    linkTo:'' },
    {option : 'Add New Recipient',
    icon:'PersonAddIcon',
    linkTo:'/receive/add' },
    {option : 'View Existing Recipients',
    icon: 'ViewListIcon',
    linkTo:'/receive/view' },
    {option : 'View Recipient Matches',
    icon:'PreviewIcon',
    linkTo:'/receive/matches' },
    {option : 'DASHBOARD',
    icon:'',
    linkTo:'' },
]
  return (
    <MoralisProvider initializeOnMount={false}>
      <div>
        <Box display={"flex"}  justifyItems={"center"}xs={12}>
```

```
<Router>
 <Appbar/>
 <Grid xs={10} style={{width:"100%"}}container direction="row" >
  <Grid>
   <Sidebar xs={1}tabList={SidebarList}/>
  </Grid>
  <Grid item xs={10} style={{ position: "relative", top: "120px", left:"20%"}}>
   <Routes>
    <Route path="/" element={ <Login/> } ></Route>
    <Route path="/register" element={ <Register/> } ></Route>
    <Route path="/donate/add" element={<AddNewDonor/>}></Route>
    <Route path="/donate/view" element={<ViewExistingDonor/>}></Route>
    <Route path="/receive/add" element={<AddNewRecipient/>}></Route>
    <Route path="/receive/view" element={<ViewExistingRecipient/>}></Route>
    <Route path="/receive/matches" element={<ViewMatches/>}></Route>
    <Route path="/dashboard" element={ <Dashboard/> } ></Route>
   </Routes>
  </Grid>
  <Grid item xs={1}/>
```

```
            </Grid>
          </Router>
        </Box>
      </div>
    </MoralisProvider>
  );
}

export default App;
```

6.2 SAMPLE SCREENSHOTS

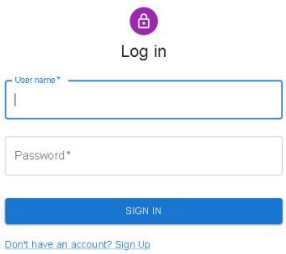

Figure 6.1 Login Screen

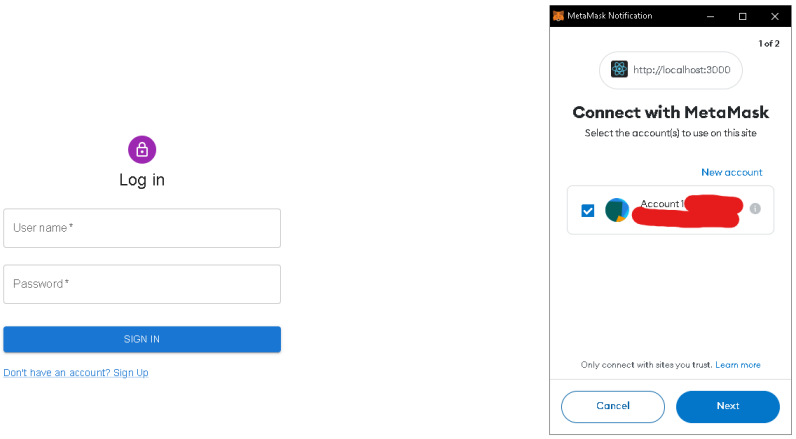

Figure 6.2 Metamask integration

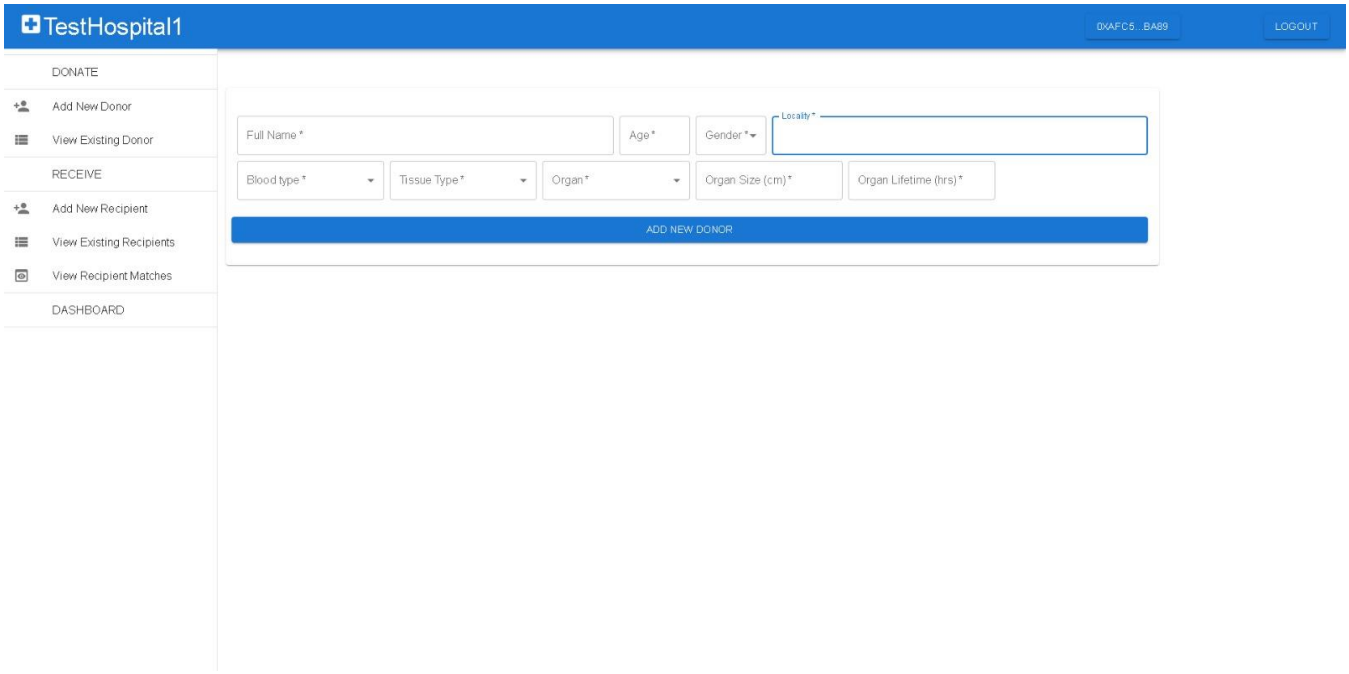

Figure 6.3 Create New Donor

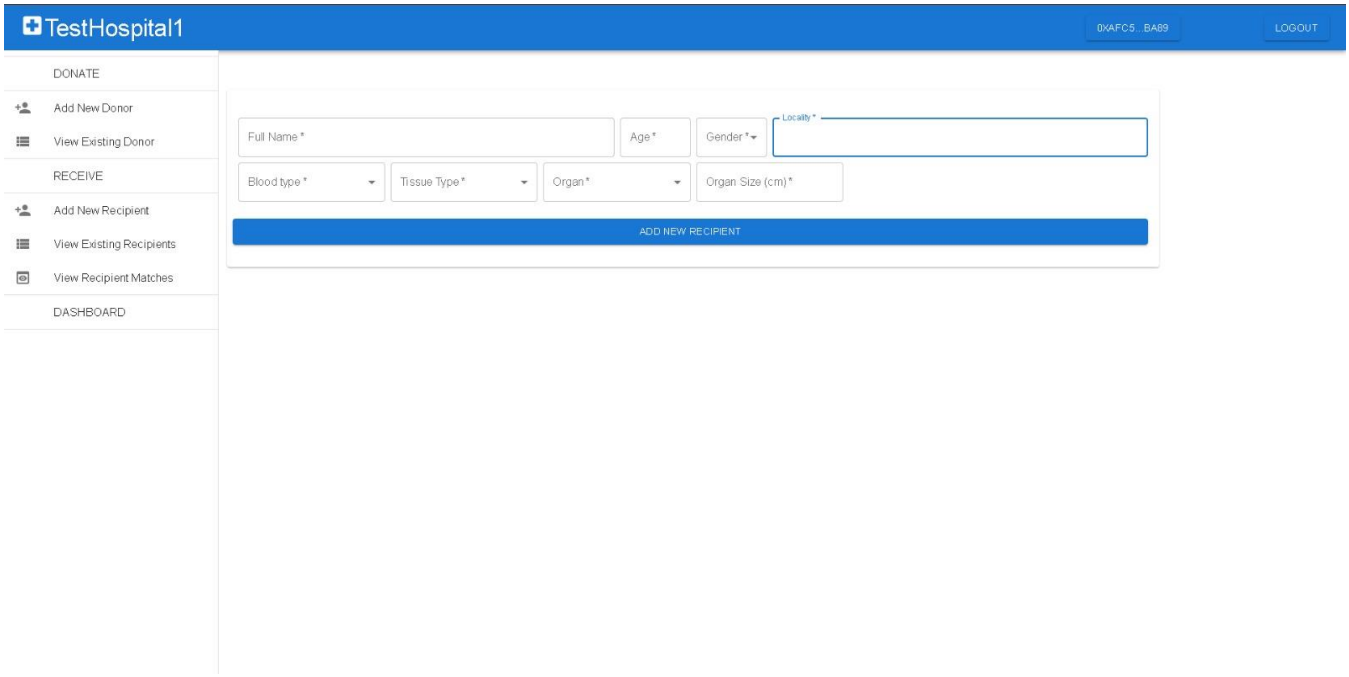

Figure 6.4 Create New Recipient

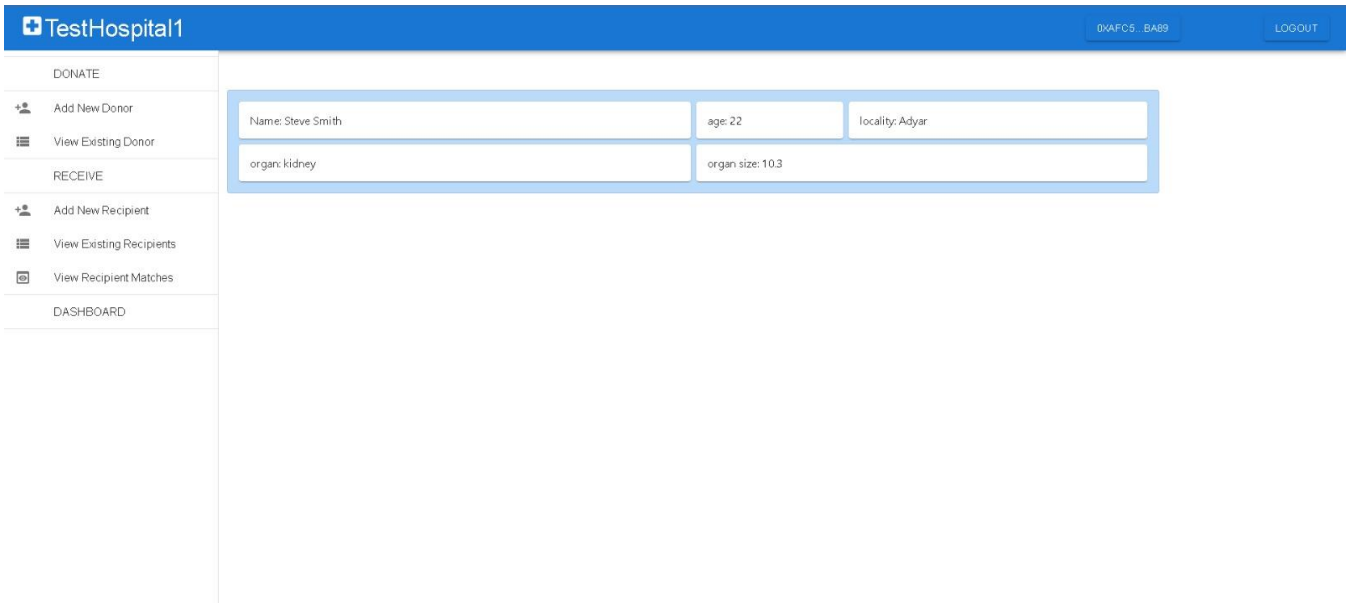

Figure 6.5 View Existing Donors

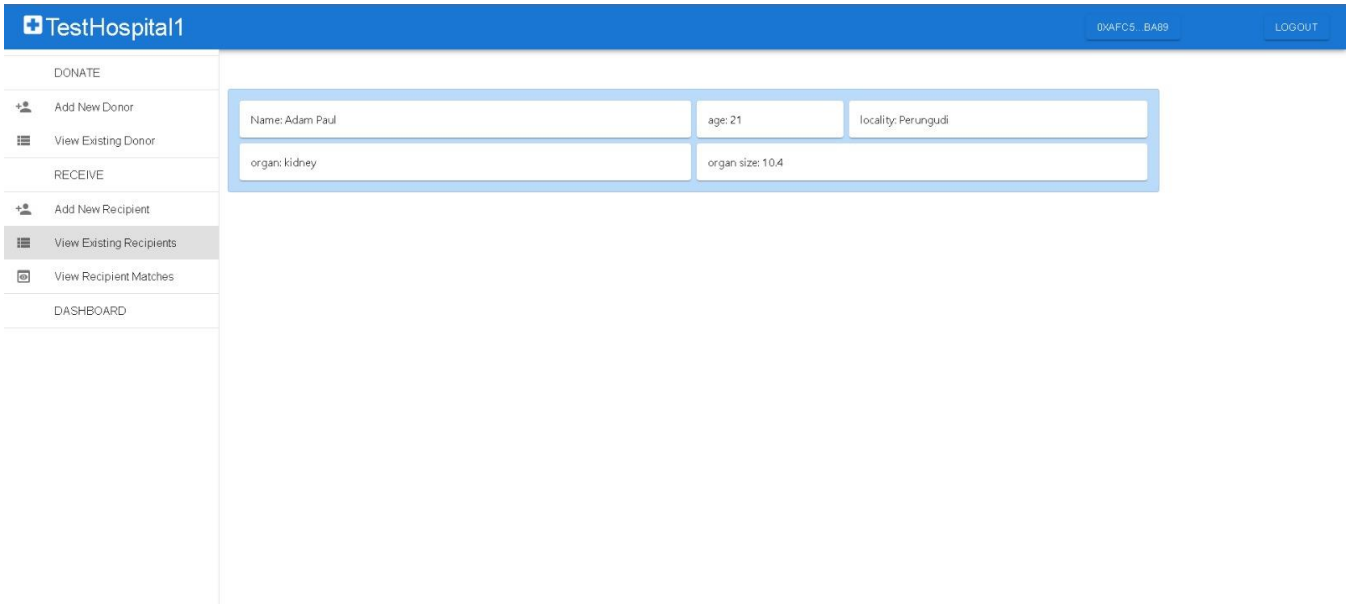

Figure 6.6 View Existing Recipients

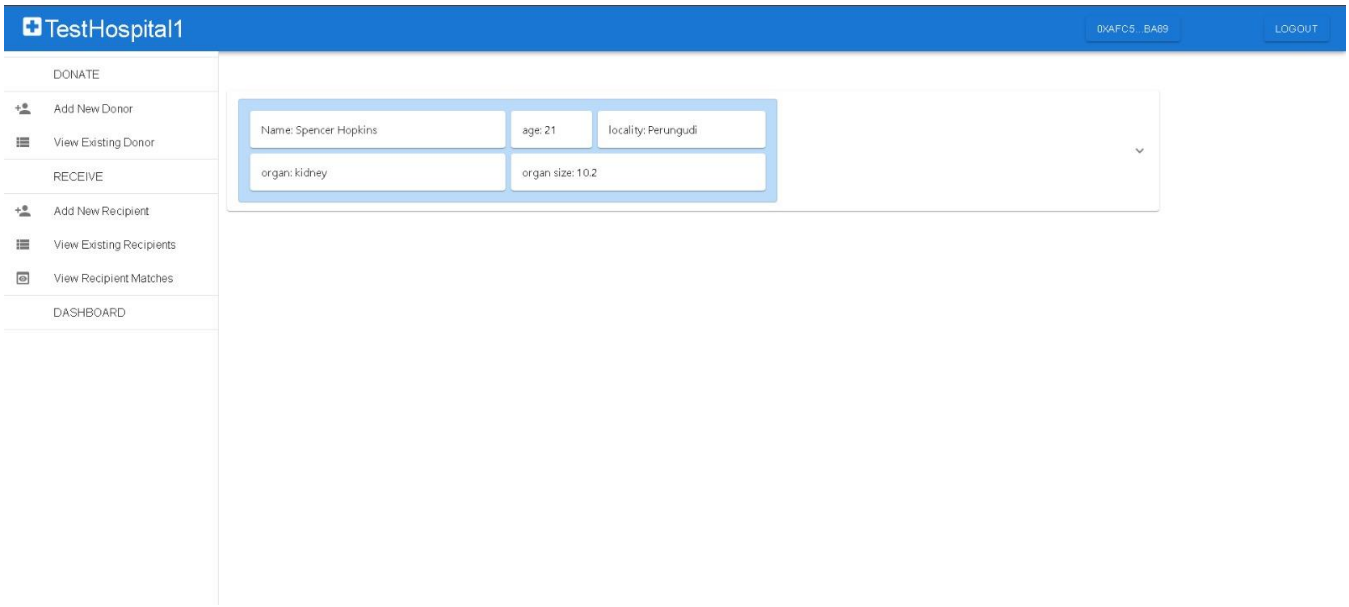

Figure 6.7 Recipient Matches

6.3 RESULTS

The implemented blockchain based application system has saved a significant time period from the organ donation process. The data used to arrive at this conclusion was taken from firsthand sources and internet resources. There are many edge cases to be handled in this analysis, so we are explicitly focusing on the case where both appropriate organ donor and recipient are ready and waiting to be matched.

On average the organ donor matching process takes a few hours to several days. The majority of the cases where organ donor and recipient are ready and waiting to matched are completed within the time period of 8 hours. This also involves the manual effort required by the administrator to find the suitable match and also accurately determine the conditions for a matching donor.

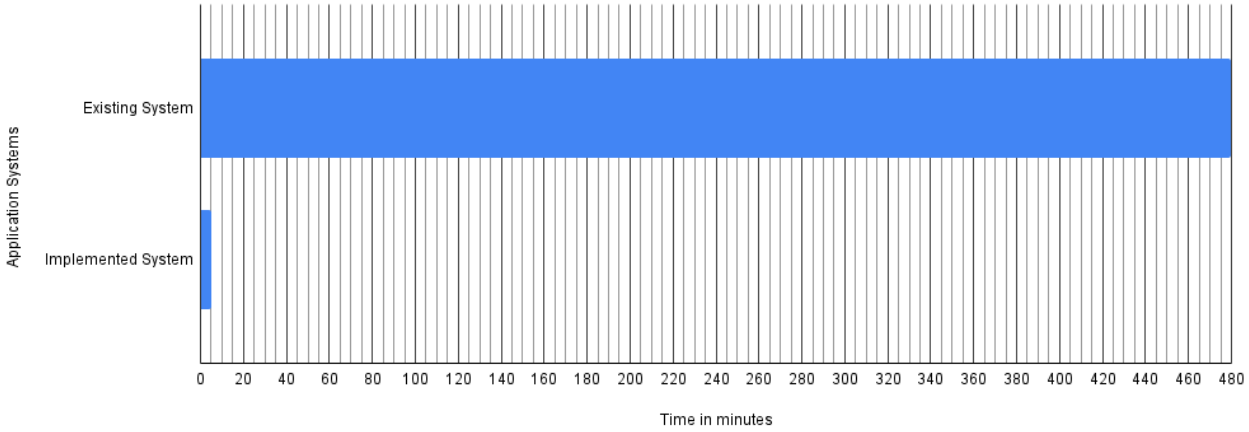

Figure 6.8 Turnaround Time Graph

Figure 6.8 compares the turnaround time to match an available donor to a waiting recipient. The existing organ donation system for the same case has a turnaround time of ~8 hours or ~480 minutes. The implemented blockchain based application system has a turnaround time of ~ 0.08 hours or ~5 minutes. This shows that there is about a 96% turnaround time decrease. This turnaround time decrease will result in lesser manual intervention and save time for the administrator. The implications of the 96% turnaround time savings can lead to fewer lost lives due to slow organ procurement and matching. The entire organ donation and procurement process' turnaround time can be greatly reduced with this application system. The blockchain based application for organ donation management using machine learning was developed and implemented successfully. It served as an effective Proof of Concept for future applications that wish to implement the same and improve upon.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In conclusion, the implementation of "Blockchain-based organ donation management using machine learning" marks a noteworthy stride in the realm of medical diagnostics and digital healthcare. The use of blockchain technology to store organ donor and recipient information allowed for complete transparency of all transactions that take place within the organ donation management system. The use of machine learning to remove the need for human intervention was implemented. Due to the lack of a diverse and large dataset, the effective viability of the decision making model stands to be tested in a real world situation. This groundbreaking approach furnishes healthcare institutions with an organized and efficient organ donation platform capable of accurately assessing and matching potential organ donors with recipients. The utilization of machine learning empowers the platform to analyze patients' health records, identify patterns, and make predictions with commendable precision, thereby streamlining and enhancing the organ donation process for improved healthcare outcomes. Looking forward, the future development roadmap envisions an enhancement of the current model's accuracy and an expansion of its capabilities to discern clusters among donors. In this evolution, the machine learning model will incorporate location data for each donor and recipient, leveraging geographical information as a crucial factor in match selection. This can be achieved through the integration of a mapping API, such as Google Maps, to facilitate the spatial analysis of potential matches. Furthermore, the clustering algorithm may extend its considerations to additional variables, enhancing the precision of donor-recipient pairings. Simultaneously, attention will be directed towards optimizing the smart contract for greater gas efficiency, ensuring a more economical utilization of resources when interacting with the blockchain.

REFERENCES

[1] Alandjani, G. (2019). “Blockchain-based auditable medical transaction scheme for organ transplant services,” 3C Tecnología. Glosas de innovación aplicadas a la pyme. Edición Especial, Noviembre 2019, 41-63. doi: http://dx.doi.org/10.17993/3ctecno.2019.specialissue3.41-63

[2] Arigela. S. S. D and P. Voola, "Blockchain Open Source Tools: Ethereum and Hyperledger Fabric", 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India, 2023, pp. 1-8, doi: 10.1109/ICECONF57129.2023.10084256.

[3] Bajoudah, S., Dong, C., Missier, P. (2019). “Toward a Decentralized, Trust-Less Marketplace for Brokered IoT Data Trading Using Blockchain”, 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, pp. 339-346. doi: 10.1109/Blockchain.2019.00053

[4] Bates. M, "Overcoming Challenges in Organ Transplantation," in IEEE Pulse, vol. 11, no. 6, pp. 25-28, Nov.-Dec. 2020, doi: 10.1109/MPULS.2020.3036148.

[5] Bertsimas, D., Farias, V. F., Trichakis, N. (2013). “Fairness, Efficiency, and Flexibility in Organ Allocation for Kidney Transplantation.” Operations Research, 61(1), 73-87. https://doi.org/10.1287/opre.1120.1138

[6] Chaudhary, N., Manvi, S. S., Koul, N. (2022). “Organ Bank Based on Blockchain”. 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, pp. 1-5. doi: 10.1109/CONECCT55679.2022.9865787

[7] Dajim, L. A., Al-Farras, S. A., Al-Shahrani, B. S., Al-Zuraib, A. A., & Merlin Mathew, R. (2019). “Organ Donation Decentralized Application Using Blockchain Technology”. 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS). doi:10.1109/cais.2019.8769459

[8] Davis, A., Mehrotra, S., Friedewald, J., Ladner, D. (2013). “Characteristics of a simulation model of the national kidney transplantation system”. 2013 Winter Simulations Conference (WSC), Washington, DC, USA, pp. 2320-2329. doi: 10.1109/WSC.2013.6721607

[9] Dimitris Bertsimas, Vivek F. Farias, Nikolaos Trichakis, (2013) "Fairness, Efficiency, and Flexibility in Organ Allocation for Kidney Transplantation". Operations Research 61(1):73-87. https://doi.org/10.1287/opre.1120.1138.

[10] El Haddad, B. (2012). “Towards developing an Intelligent Match Making Assistant to be used during the Human Organ Transplantation Management”. 2012 7th International Conference on Computing and Convergence Technology (ICCCT), Seoul, Korea (South), pp. 970-975.

[11] Ferraza, V., Oliveira, G., Viera-Marques, P., Cruz-Correia, R. (2011). “Organs transplantation—How to improve the process?”. Eur. Fed. Med. Inform., Cardiff, U.K., Tech. Rep. doi: 10.3233/978-1-60750-806-9-300

[12] Frauenthaler, P., Sigwart, M., Spanring, C., Sober, M., Schulte, S. (2020). “ETH Relay: A Cost-efficient Relay for Ethereum-based Blockchains”. 2020 IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, pp. 204-213. doi: 10.1109/Blockchain50366.2020.00032

[13] George, L., Kizhakkethottam, J. J. (2023). “A Survey on the Impact of Blockchain in Effective Organ Transplantation”. 2023 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE), Kerala, India, pp. 1-6. doi: 10.1109/RASSE60029.2023.10363481

[14] Hawashin, D., Jayaraman, R., Salah, K., Yaqoob, I., Simsekler, M. C. E., Ellahham, S. (2022). “Blockchain-Based Management for Organ Donation and Transplantation”. IEEE Access, 10, 59013-59025. doi: 10.1109/ACCESS.2022.3180008

[15] Hölbl, M., Kompara, M., Kamišalić, A., Zlatolas, L. N. (2018). “A systematic review of the use of blockchain in healthcare”. Symmetry, 10(10), 470. doi:

10.3390/sym10100470

[16] Jain, U. (2020). “Using blockchain technology for the organ procurement and transplant network”. San Jose State Univ., San Jose, CA, USA, Tech. Rep. doi: 10.31979/etd.g45p-jtuy

[17] Khatal, S., Rane, J., Patel, D., Patel, P., Busnel, Y. (2019). “FileShare: A Blockchain and IPFS framework for Secure File Sharing and Data Provenance”. ICMLCI 2019 : International Conference on Machine Learning and Computational Intelligence, Dec 2019, Bhubaneswar, Kantabada, India. ff10.1007/978-981-15-5243-4_79ff. ffhal-02451022f

[18] Malik, H., Manzoor, A., Ylianttila, M., Liyanage, M. (2019). “Performance Analysis of Blockchain-based Smart Grids with Ethereum and Hyperledger Implementations”. 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Goa, India, pp. 1-5. doi: 10.1109/ANTS47819.2019.9118072

[19] Mattei, N., Saffidine, A., Walsh, T. (2017). “Mechanisms for online organ matching. Proc. 26th Int. Joint Conf. Artif. Intell.”, Aug. 2017, pp. 345–351. doi: 10.24963/ijcai.2017/49

[20] Pacheco, D. F., Pinheiro, D., Cadeiras, M., Menezes, R. (2017). “Characterizing Organ Donation Awareness from Social Media”. 2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA, USA, pp. 1541-1548. doi: 10.1109/ICDE.2017.225

[21] Ranjan, P., Srivastava, S., Gupta, V., Tapaswi, S., Kumar, N. (2019). “Decentralised and Distributed System for Organ/Tissue Donation and Transplantation”. 2019 IEEE Conference on Information and Communication Technology, Allahabad, India, pp. 1-6. doi: 10.1109/CICT48419.2019.9066225

[22] Rao, V., Behara, R. S., Agarwal, A. (2014). “Predictive Modeling for Organ Transplantation Outcomes”. 2014 IEEE International Conference on Bioinformatics and Bioengineering, Boca Raton, FL, USA, pp. 405-408. doi:

10.1109/BIBE.2014.58

[23] Richard, M. M. Surya, A. C. Wibowo. (2020). “Converging Artificial Intelligence and Blockchain Technology using Oracle Contract in Ethereum Blockchain Platform”. 2020 Fifth International Conference on Informatics and Computing (ICIC), Gorontalo, Indonesia, pp. 1-5. doi: 10.1109/ICIC50835.2020.9288611

[24] Sankar, L. S., Sindhu, M., Sethumadhavan, M. (2017). “Survey of consensus protocols on blockchain applications”. 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Jan 2017, pp. 1–5. doi: 10.1109/ICACCS.2017.1

[25] Shreya Khatal, Jayant Rane, Dhiren Patel, Pearl Patel, Yann Busnel. (2019). “FileShare: A Blockchain and IPFS framework for Secure File Sharing and Data Provenance”. ICMLCI 2019 : International Conference on Machine Learning and Computational Intelligence, Dec 2019, Bhubaneswar, Kantabada, India. ff10.1007/978-981-15- 5243-4_79ff. ffhal-02451022f

[26] Soni, A., Kumar, D. S. G. (2021). “Creating Organ Donation System with Blockchain Technology”. European Journal of Molecular & Clinical Medicine, 8(3), pp. 2387-2395.

[27] Wijayathilaka, P. L., Gamage, P. H. P., De Silva, K. H. B., Athukorala, A. P. P. S., Kahandawaarachchi, K. A. D. C. P., Pulasinghe, K. N. (2020). “Secured, Intelligent Blood and Organ Donation Management System – ‘LifeShare’ ”. 2020 2nd International Conference on Advancements in Computing (ICAC), Malabe, Sri Lanka, pp. 374-379. doi: 10.1109/ICAC51239.2020.9357211

[28] Wang, Z., Cui, B., Hou, W. (2022). “A Dynamic Load Balancing Scheme Based on Network Sharding in Private Ethereum Blockchain”. 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, pp. 362-367. doi: 10.1109/COMPSAC54236.2022.00057 [24]

[29] X. Ren, M. C. Fu and S. I. Marcus, "Sensitivity Analysis for Stopping Criteria

with Application to Organ Transplantations," 2023 Winter Simulation Conference (WSC), San Antonio, TX, USA, 2023, pp. 504-515, doi: 10.1109/WSC60868.2023.10408603.

[30] Zheng, Z., Xie, S., Dai, H.-N., Chen, X., Wang, H. (2017). “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends”. 2017 IEEE Congress on Services (SERVICES), Honolulu, HI, USA, pp. 219-226. doi: 10.1109/SERVICES.2017.36