JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Decentral Chat: Redefining Conversations for the Modern Age

Deepak Gaikar[1], Karsh Dhanga[2], Ritesh Khurana [3], Shreyash Jadhav [4], and Chetan Bhole[5]

[1]Asst. Prof. Computer, Dept. of Computer Engineering, MCT Rajiv Gandhi institute of technology, Mumbai, India.
[2]B.E. Computer, Dept. of Computer Engineering, MCT Rajiv Gandhi institute of technology, Mumbai, India.
[3]B.E. Computer, Dept. of Computer Engineering, MCT Rajiv Gandhi institute of technology, Mumbai, India.
[4]B.E. Computer, Dept. of Computer Engineering, MCT Rajiv Gandhi institute of technology, Mumbai, India.
[5]B.E. Computer, Dept. of Computer Engineering, MCT Rajiv Gandhi institute of technology, Mumbai, India.

Abstract: In today's interconnected world, advanced communication in here has ended up an necessarily portion of our day by day lives. From individual discussions to commerce exchanges, they require for secure and dependable informing stages has never been more prominent. Be that as it may, conventional centralized chat applications regularly raise concerns with respect to protection, security, and control over individual information. To address these challenges, the concept of decentralization, enabled by blockchain innovation, has risen as a promising arrangement. The point of this venture is to create a blockchain-based decentralized chat application that rethinks the way we communicate online. By leveraging the standards of blockchain, this application looks for to offer clients a secure, private, and censorship-resistant informing stage. Not at all like centralized options, where information is put away on servers controlled by a single substance, a decentralized chat application disperses information over a organize of hubs, guaranteeing no single point of disappointment and improving security. In this initial stage, we are going dive into the basis behind the extend, investigating the inadequacies of centralized chat applications and the potential benefits of decentralization. We'll too figure the key goals and highlights of the proposed decentralized chat application, setting the arrange for a more profound investigation of its design, usefulness, and usage. Through this extend, we try to contribute to the advancement of computerized communication, advertising clients a more secure, straightforward, and user-centric informing encounter.

IndexTerms **- Decentralized, Distributed, peer-to-peer, blockchain, user-centric, Etherium.**

I. INTRODUCTION

In an era dominated by digital communication, the need for secure, private, and reliable messaging platforms has become increasingly apparent. However, traditional centralized chat applications often fall short in meeting these demands, leaving users vulnerable to privacy breaches, data manipulation, and censorship. Recognizing these limitations, the concept of decentralization, powered by blockchain technology, emerges as a transformative solution. This project sets out to pioneer a new paradigm in digital messaging through the development of a blockchain-based decentralized chat application. By harnessing the inherent properties of blockchain, such as immutability, transparency, and cryptographic security, this application aims to revolutionize the way individuals communicate in the digital realm. Gone are the days of relying on centralized intermediaries to facilitate our conversations; instead, we embrace a distributed network architecture that empowers users with unparalleled control over their data and communication channels.

In this introductory exploration, we embark on a journey to understand the rationale behind decentralized chat applications and the transformative potential they hold. We delve into the shortcomings of centralized counterparts, examining the vulnerabilities they present to users' privacy and security. Moreover, we highlight the fundamental principles of blockchain technology that underpin our decentralized chat application, laying the groundwork for a comprehensive understanding of its architecture and functionality. Through this project, we not only seek to address the pressing challenges of digital communication but also pave the way for a future where individuals have sovereignty over their online interactions. Join us as we embark on this innovative endeavor to redefine the landscape of messaging platforms and usher in a new era of decentralized communication. In this extended exploration, we delve into the motivations driving the development of a blockchain-based decentralized chat application. We examine the shortcomings of centralized messaging platforms, including their susceptibility to hacking and data monetization.

II. EXISTING SYSTEM

2.1Literature Review

Shikhar Vashishth et al. [1] proposed a peer to peer (p2p) architecture to exchange messages asynchronously and facilitate new peer joins via existing peers in the network, thus reducing the dependency on the bootstrap server. The data channel of WebRTC was used to implement the browser compatible framework. It is a modified version of Chord (a distributed lookup protocol for p2p networks). Multiple STUN servers were used to balance load of ICE candidate requests on them. A TURN server was used to route traffic via the relay server. Further, periodically existing connections were checked for whether or not they were required and if not, destroyed.

Sourabh et al. [2] proposed an implementation of a decentralized chat application on the Ethereum blockchain network with ReactJS. Infura was also used which is a hosted Ethereum node cluster that lets users run the application without requiring to set up their own Ethereum node or wallet. Solidity was used which is an object-oriented, high-level language for implementing smart

contracts (programs which govern the behavior of accounts within the Ethereum state). Also used Etherscan which allows to explore and search the Ethereum blockchain for transactions, addresses, tokens, prices and other activities. The AES-256 algorithm was used for encryption and decryption using CryptoJS.

Faraz Khan et al. [3] proposed a Dchat that uses IPFS technology for decentralized anonymous and tamper-proof cross platform communication systems with user friendly interface. IPFS establishes a permanent distributed network by using content-based reference. It uses the hash of the file itself to be used as a reference. It also gives the users an advantage to be anonymous during the communication. The system uses IPFS hashes for security which are 32 bytes with SHA256 algorithm. LibP2P protocol was used to listen for incoming messages and broadcasts and directly messaging to peers on the network.

2.2 Limitation Existing system or research gap

1. A central point of control, like a server, is a vulnerability. If it fails, the entire system can grind to a halt. Decentralized systems spread data and tasks across multiple locations, making them more resilient.
2. A centralized system stores all the data in one place, making it a prime target for hackers. A breach can expose a vast amount of information. Decentralized systems distribute data, making it harder for attackers to steal everything.
3. As the number of users or devices in a centralized system grows, it can become overloaded. Upgrading the central server can be expensive and disruptive. Decentralized systems can typically add more nodes to handle increased demand.
4. Centralized authorities can control access to information and functionality within a centralized system. This can lead to censorship or manipulation of data. Decentralized systems are generally more resistant to such control.

III. PROPOSED SYSTEM

3.1 Proposed System Architecture

Our proposed system consists of**:** Peer-to-Peer (P2P) Network to eliminate the need for a central server, allowing users to connect directly with each other. We can leverage protocols like libp2p or WebRTC for peer discovery and communication . Distributed Ledger Technology (DLT) that could be a blockchain or a similar technology that acts as a tamper-proof record of messages and user identities. Options include Ethereum with Whisper protocol or custom blockchain solutions designed for scalability. Cryptography so that the messages would be encrypted before transmission, ensuring privacy and confidentiality. Public-key cryptography can be used for secure user authentication and message encryption.The proposed system uses different technologies like Cryptography, Blockchain and various other concepts like encryption, consistency, availability and techniques to improve network stability to create a robust peer to peer decentralized application .Some of them are:

1. Blockchain Technology**:**
   Here we familiarize you with the basic concepts of new blockchain technology, including distributed ledgers, consensus mechanisms, smart contracts and password.. Gain insights into how blockchain can be utilized to create decentralized applications (dApps) and its potential benefits for messaging platforms.

2. Cryptography and Secure Communication:
   Acquire knowledge of cryptographic principles and protocols essential for ensuring secure communication in a decentralized chat application. This includes understanding symmetric and asymmetric encryption, hashing algorithms, digital signatures, and key exchange mechanisms.

3. Protocols for Decentralized Messaging:
   Study existing protocols and standards for decentralized messaging, such as the InterPlanetary File System (IPFS), Whisper, and Matrix. Evaluate their suitability for building a decentralized chat application and consider interoperability with other messaging platforms.

4. Smart Contract Development:
   Learn how to develop smart contracts, self-executing code deployed on the blockchain, using programming languages like Solidity (for Ethereum) or similar languages for other blockchain platforms. Explore the design patterns and best practices for implementing smart contracts that govern messaging functionalities, user interactions, and data storage.

5. Blockchain Development Platforms:
   Familiarize yourself with blockchain development platforms and tools, such as Ethereum, Hyperledger Fabric, or EOSIO, depending on the requirements and objectives of your decentralized chat application. Understand how to configure development environments, deploy smart contracts, and interact with the blockchain network.

6. Decentralized Identity and Authentication:
   Explore decentralized identity solutions, such as Decentralized Identifiers (DIDs) and verifiable credentials, for user authentication and access control in the chat application. Investigate methods for managing user identities and ensuring privacy while maintaining accountability and trust.

7. User Interface (UI) and User Experience (UX) Design using react:
   Develop skills in UI/UX design to create an intuitive and engaging user interface for the decentralized chat application. This uses factors such as as ease of use, accessibility, responsiveness, and aesthetics to enhance the user experience and encourage adoption.

3.2 Data and Sources of Data

The Data Sources for this application are going to be build on the NPM which include these main components. These are used to carry out the data from:

1. Blockchain
   Blockchain is a decentralized digital ledger that records transactions securely and transparently.It is essentially a decentralized database that allows multiple parties to access the same data without the need for a central authority. Blockchain is basically known for its use in cryptocurrency such as Bitcoin, but it can also be applied to many other use cases such as supply chain management, voting systems, and portfolio management.
2. Ethereum Smart Contracts
   Ethereum is platform based on blockchain that enables the development and deployment of distributed applications (dApps) through smart contracts. Smart contracts are self-executing contracts that are kept on the Ethereum blockchain and automatically execute the terms of the contract. They can facilitate transactions, automate business processes, and build trust between parties without the need for middleman.
3. Hardhat
   Hardhat is a development environment for Ethereum that permits developers to build, test, and deploy smart contracts and decentralized applications (dApps). It offers a wide range of features designed to streamline the Ethereum development process.
4. Metamask
   MetaMask is a wallet and gateway of cryptocurrency to the Ethereum blockchain that enables users to interact with decentralized applications (dApps) directly from their web browsers. Metamask basically acts as a software that connects the IP address of the user and the blockchain network.
5. Solidity
   Solidity is a language used to program and to write smart contracts on the Etherium blockchain. It is a carefully typed, contract-oriented language designed to be both safe and efficient. Solidity is same as in syntax to JavaScript, which makes it easy for developers to learn and use for programming and deploying.
6. AES
   A cryptographic algorithm used for generating digital signatures. In the context of your chat app, EDcsa can be used to ensure message integrity and authenticity. When a user sends a message, it can be signed with their private key using EDcsa. The recipient can then verify the signature using the user's public key, confirming the message originated from the intended sender and hasn't been tampered with during transmission.
7. Web3.js
   Web3.js is a library of JavaScript that allows developers to connect with the Ethereum blockchain through a web browser. It provides a simple API for interacting with smart contracts, sending events, and managing user accounts, making it a popular tool for designing various decentralized applications on the Ethereum blockchain.
8. Next.js
   A React framework offering features like server-side rendering and static site generation. This improves your app's Search Engine Optimization (SEO) and provides faster initial page loads, enhancing user experience.
9. IPFS
   A decentralized storage network for storing data in a distributed and censorship-resistant manner. This allows you to store chat history, profile pictures, or other user data outside of a central server, enhancing privacy and security.

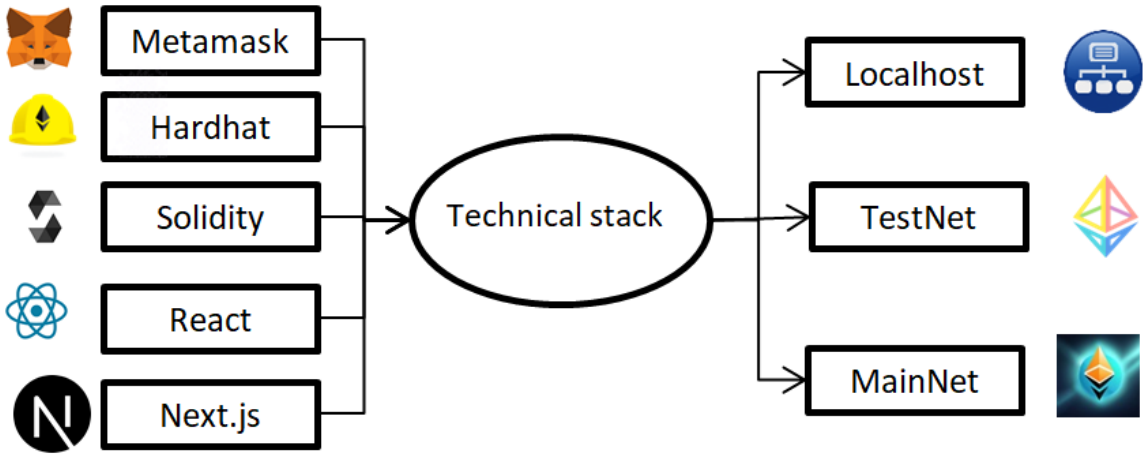

Fig 1: Technical Stack

The figure 1 is a technical stack that illustrates the various components of a system. The figure consists of a block figure that represents the technical stack of a React app, as well as another figure displaying the technical stack of a Next.js app. The technical stack includes tools such as Metamask, Localhost, Hardhat, Solidity, TestNet, React, and Next.js. Metamask is a bride between the browser and the network which is used as wallet in the network , while Localhost is a loopback interface used for 127.0.0.1 (localhost) and is used for testing purposes. Hardhat is an Ethereum development environment, and Solidity is a statically-typed programming language designed for developing smart contracts that run on the Ethereum Virtual Machine (EVM). TestNet is a network used for testing purposes before deploying to the MainNet. React and Next.js are JavaScript libraries for building user interfaces and web applications, respectively.

IV. SYSTEM ARCHITECTURE

4.1 System Design

At its core, the application enables users to engage in private conversations through encrypted messages, which are securely transmitted and stored on the Ethereum blockchain. The frontend, built with React.js, provides an intuitive user interface for accessing chat rooms, sending messages, and managing user profiles. MetaMask integration makes users to connect their Ethereum wallets, authenticate transactions, and connect with smart contracts easily. Smart contracts, developed using Solidity and deployed with Hardhat, govern the messaging functionalities, including message encryption, storage, and retrieval. The Ethereum blockchain serves as the underlying infrastructure, with the Ethereum Mainnet supporting production deployment and the Sepolia Testnet facilitating testing and development. Additionally, a Node.js backend handles user authentication, authorization, and other server-side functionalities, while a database stores non-sensitive metadata associated with user accounts and chat rooms. Real-time communication protocols, such as WebSocket, enable instant messaging between users, ensuring a responsive and interactive messaging experience. Overall, the system architecture provides a comprehensive solution for decentralized messaging, offering users a secure, transparent, and user-centric platform for digital communication in today's interconnected world.

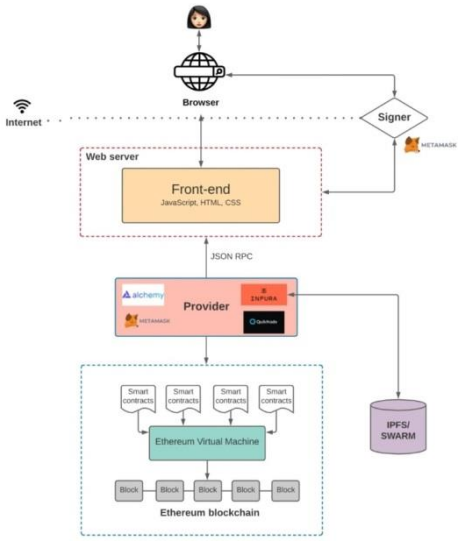

Fig 2: System Architecture

The figure 2 is the system arcitecture that illustrates the various components of a system. The figure consists of a block figure that represents the technical stack of a React app, as well as another figure displaying the technical stack of a Next.js app. The technical stack includes tools such as Metamask, Localhost, Hardhat, Solidity, TestNet, React, and Next.js. Metamask is a browser extension that allows users to interact with the Ethereum blockchain, while Localhost is a loopback interface used for 127.0.0.1 (localhost) and is used for testing purposes. Hardhat is an Ethereum development environment, and Solidity is a statically-typed programming language designed for developing smart contracts that run on the Ethereum Virtual Machine (EVM). TestNet is a network used for testing purposes before deploying to the MainNet. React and Next.js are JavaScript libraries for building user interfaces and web applications, respectively.

The frontend of the decentralized chat application is developed using React.js, a popular library of JavaScript for creating user interfaces. Components of React are used to create various elements of the user interface, including chat rooms, message input fields, user profiles, and authentication screens. User interactions and events are handled using React's component-based architecture, providing a responsive and intuitive user experience. Smart contracts, which govern the messaging functionalities and interactions on the blockchain, are developed using Solidity that is a programming language used for writing Ethereum smart contracts. Hardhat, a development environment for Ethereum smart contracts, is used to compile, deploy, and test the smart contracts locally. Smart contracts define the logic for message encryption, storage, retrieval, and access control, ensuring that messages are securely transmitted and stored on the blockchain. The decentralized chat application interacts with the Ethereum blockchain, leveraging the Ethereum Mainnet for production deployment and the Sepolia Testnet for testing and development purposes. Sepolia Testnet provides a simulated Ethereum blockchain environment for deploying and testing smart contracts without using real Ether (ETH) or incurring transaction fees.

The figure depicts a Web 3.0 application architecture. In this setup, a web browser acts as the user's entry point, connecting to the application through the internet. To interact with the blockchain, the user utilizes a signer, like MetaMask, which holds their crypto keys. A web server delivers the user interface and acts as a middleman between the user and the blockchain. The user interface itself is built with familiar technologies like Javascript, HTML, and CSS. Communication between the front-end and the server is facilitated by JSON RPC. Providers, such as Infura, bridge the gap between the application and the blockchain. Smart contracts, residing on the blockchain (like Ethereum in this example), automate tasks and agreements. The Ethereum Virtual Machine (EVM) executes these smart contracts. Finally, decentralized storage solutions like IPFS/Swarm can store application data in a distributed and secure manner. This architecture showcases how Web 3.0 applications leverage blockchain technology to create a more secure and transparent user experience. This is the structure of the system we are making as an application.

4.2 System Flow

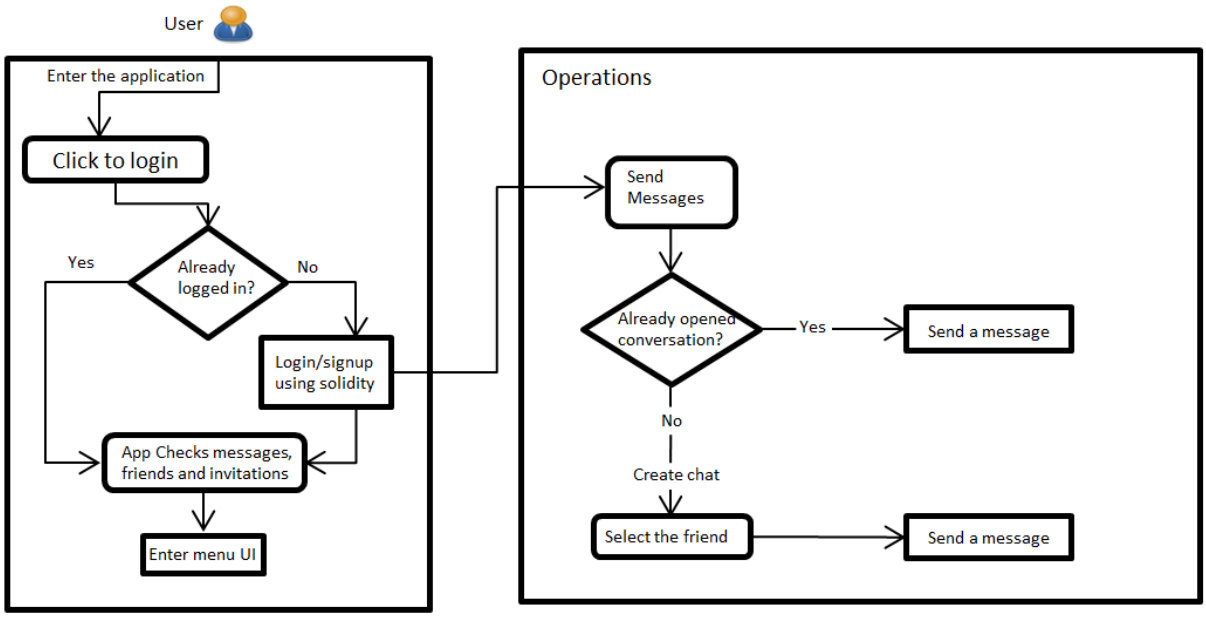

Fig 3: User flow

The figure 3 is a technical figure that illustrates the various components of a system that utilizes both React and Next.js for building user interfaces and web applications. The technical stack includes tools such as Metamask, Localhost, Hardhat, Solidity, TestNet, React, and Next.js. Metamask is a browser extension that allows users to interact with the Ethereum blockchain, making it possible to build decentralized applications (dApps) that can interact with smart contracts. Localhost is a loopback interface used for testing purposes, allowing developers to test their applications on their local machines before deploying them to a live network. Hardhat is an Ethereum development environment that provides tools for compiling, deploying, testing, and debugging smart contracts. Solidity is a statically-typed programming language used for developing smart contracts that run on the Ethereum Virtual Machine (EVM). TestNet is a network used for testing purposes before deploying to the MainNet, allowing developers to test their applications in a production-like environment without the need for real funds. Overall, the figure provides a

comprehensive overview of the technical stack used for building a web application that leverages the power of blockchain technology and smart contracts.

Web3.js, a JavaScript library for interacting with Ethereum smart contracts, is used to connect the frontend application with the blockchain network, enabling users to send and receive messages securely.MetaMask, a browser extension and mobile app for managing Ethereum wallets, is integrated into the decentralized chat application to enable users to sign transactions, authenticate, and interact with the blockchain securely. Users can connect their MetaMask wallets to the application, allowing them to access their Ethereum accounts, view their balances, and sign messages using their private keys. MetaMask's built-in Ethereum provider is utilized to facilitate communication between the frontend application and the Ethereum blockchain, enabling seamless interaction with smart contracts and blockchain transactions. A lightweight backend server built with Node.js and Express.js handles user authentication, authorization, and other server-side functionalities .The backend server interfaces with the Ethereum blockchain and smart contracts to retrieve and store encrypted messages, manage user accounts, and handle other application logic that cannot be executed on the client-side.

The process starts with the user entering the application. The user is then asked if they have already logged in. This involves the user clicking a "compose message" button, selecting a recipient from a contact list, or triggering an automated message based on an event. The user creates the message content. This involves typing text, attaching files, recording audio/video, or selecting content from other sources. If yes, then the process proceeds to step 4 where the app checks for messages, friends and invitations. The system checks if the message content meets pre-defined criteria. This could involve checking for character limits, ensuring valid recipient addresses, or filtering out prohibited content. If no, the user is directed to login or signup using solidity. Once logged in (or if the user was already logged in), the user enters the menu UI. From the menu UI, the user selects a friend to chat with. After selecting a friend, the flowchart checks if there is already an opened conversation with that friend. If there is an opened conversation, the user can proceed to send a message (step 6). If there is no opened conversation, the user can create a new chat by selecting “Create chat” (step 6). Finally, the user sends a message. Overall, this flowchart illustrates a messaging system where users can login, select friends, create chats and send messages. The system receives confirmation (if available) that the message was successfully delivered to the recipient's server or queue. This might not always be available depending on the delivery method. The system updates its internal records with the message delivery status (e.g., sent, delivered, failed). This information might be displayed to the user or used for further processing. The process ends here, indicating the message sending attempt is complete. This is a more detailed breakdown of a our decentralized blockchain based message sending system. The specific steps and functionalities might vary depending on the actual system design and its features.

IV. RESULTS AND DISCUSSIONS

Fig 4: Login Page

The Fig. 4 shows the starting page of our application in which user tries to create his own profile that he handles according to his accord. The user tries to open application from a wallet provider like Metamask through which it access the account number i.e. the private address of the user. By clicking the submit button the user accesses the mainnet of the project. The mainnet we using would be Sepolia mainnet . it will ask the user to accept the transaction fees i.e. the gas fees to enter the blockchain network then after that he wwill be able to gain control over his account and tries to chat with his friends or contracts.

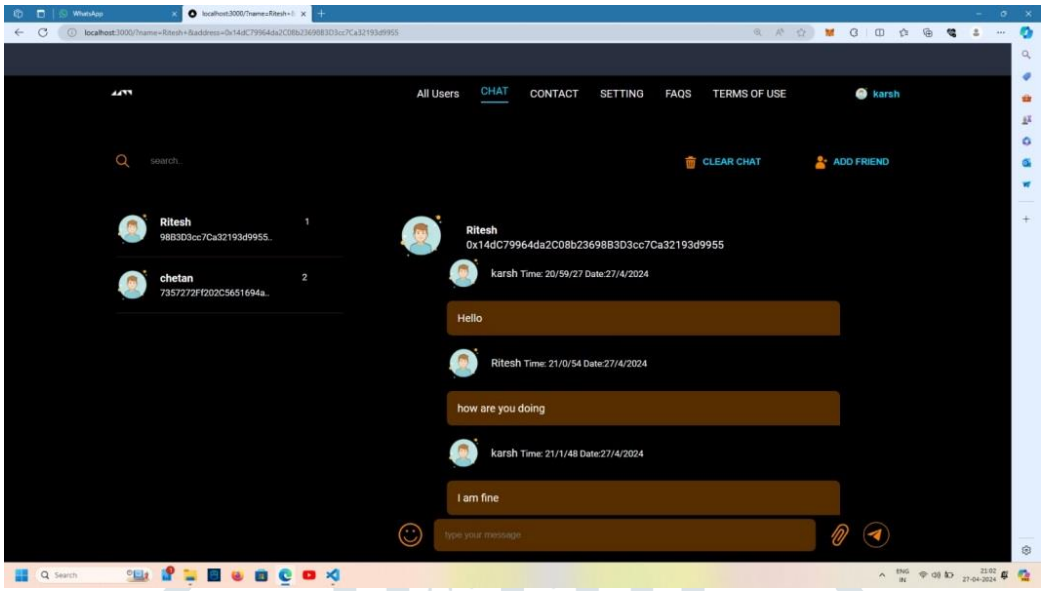

Fig 5: Web chat page

The figure 5 shows the cat that can be don't with friends i.e. in the current stance the user (Karsh) is trying to chat with his friends Ritesh and Chetan that he is in contract with . the application allows user to connect to the group through the mainnet through decentralized server without risk of getting hacked due to the structures transparency. By using this system he/she can try to contact his contracts through our application. This paper has presented the design of a decentralized chat application prioritizing user privacy and security. By leveraging blockchain technology, secure communication protocols, and a user-friendly interface, the application empowers users to communicate freely and securely in a decentralized environment. Hence these are the images produced by our project which shows two things. First it is an interface that lets us login into our chat application , second shows the chat that's successfully don't through our chat application.

V. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The development of the decentralized blockchain-based chat application represents an important step towards redefining the digital media landscape. By leveraging blockchain technology, React.js interface, Hardhat for smart contract deployment, MetaMask wallet integration and interoperability with Ethereum Mainnet and Sepolia Testnet, this project aims to fill the shortcomings of the platform Centralized messaging platform that also prioritizes user privacy, security, and control. Through the creation of an intuitive user interface, secure messaging protocol, and seamless integration with the blockchain network, the decentralized chat application provides users with a transparent, flexible, and people-centric platform used as a hub to exchange messages and engage in private conversations.By decentralizing data storage, eliminating single points of failure, and giving users full control over their communication channels, the application improves reliability, accountability, and resiliency. feedback in digital communication.

5.2 FUTURE WORK

In the future, we plan to expand this project by updating the user interface to the latest available model. Explore the integration of decentralized identity solutions, such as decentralized identifiers (DIDs) and verifiable credentials, to improve user authenticity and privacy . Implement features that allow users to securely manage their identities, verify the identities of other users, and establish trust relationships in a decentralized chat application. Research advanced privacy-preserving techniques, such as zero-knowledge proofs and secure multi-party computation, to further improve the privacy of messages exchanged in the app decentralized chat application. Implement selective disclosure mechanisms for message content and metadata, allowing users to control the visibility of their communication history. Expands the functionality of decentralized chat applications to support cross-platform compatibility, allowing seamless communication between different devices and operating systems. Develop native mobile applications for iOS and Android platforms, ensuring consistent user experience and accessibility across mobile communication devices.

REFERENCES

[1]] Shikhar Vashishth, Yash Sinha, K Hari Babu, "Addressing Challenges in Browser Based P2P Content Sharing Framework Using WebRTC", IEEE 30th International Conference on Advanced Information Networking and Applications, 2016.

[2] Sourabh, Deepanker Rawat, Karan Kapkoti, Sourabh Aggarwal, Anshul Khanna, "bChat: A Decentralized Chat Application", International Research Journal of Engineering and Technology (IRJET), May 2020.

[3] Faraz Khan, Niraj Mantri, Sagar Rajput, Dhananjay Dhakane, Puja Padiya, "Anonymous Decentralized Ephemeral Chat Application using Interplanetary File System", ITM Web of Conferences 32, 02004, 2020.

[4] George Suciu, Stefan Stefanescu, Cristian Beceanu, Marian Ceaparu, "WebRTC role in real-time communication and video conferencing", Global Internet of Things Summit (GIoTS), 2020.

[5] Kahina Khacef, Guy Pujolle, "Secure Peer-to-Peer communication based on Blockchain", 33rd International Conference on Advanced Information Networking and Applications (AINA2019), Matsue, Japan, Mar 2019

[6] Nileshkumar Pandey, Doina Bein, "Web Application for Social Networking using RTC", IEEE Annual Computing and Communication Workshop and Conference (CCWC), 2018.

[7] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, Edward W. Felten. "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies" 2015 IEEE Symposium on Security and Privacy. DOI 10.1109/SP.2015.14.

[8] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang, ―An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends‖, 2017 IEEE 6th International Congress on Big Data.

[9] H. Tewari, A. Hughes, S. Weber and T. Barry, "X509Cloud — Framework for a ubiquitous PKI" MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 2017, pp. 225-230.

[10] P. Rosler, C. Mainka and J. Schwenk, "More is less: On the end-to- ¨ end security of group chats in Signal, WhatsApp, and Threema", IEEE European Symposium on Security and Privacy (EuroS&P), pp. 415-429.