Decentralized Chat System Using Blockchain

Ann Sarah Babu
S3-A, Roll.No:29

Relevance of topic

- A decentralized application for communication and resource sharing is need in today's world, where keeping data on a centralized server can be risky and costly experience.
- By implementing Blockchain technology, we can create a secure and reliable messaging application that overcomes the drawbacks of traditional messaging applications.
- As the name suggests, a decentralized application does not have a centralized server, control is distributed between participants in the system.
- In our application all the user data is stored on a block which is connected to other blocks forming a chain.
- Also the data that is stored in block is almost impossible to view as a very secure encryption and hashing functions (256 bits) are used, if a hacker tries to make changes to the information in block then, he/she will have to make changes to all the copies of that block on whole blockchain network and that can be quite impossible.

Description

- Decentralized application make use of peer-to-peer networks, this ensures that no network failure can occur.
- Blockchain serves as an immutable ledger(the ability to remain unchanged).
- The decentralized application is implemented on Ethereum blockchain network.
- Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority.
- Ganache is used for setting up a personal Ethereum Blockchain for testing your Solidity contracts.
- MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications.

- Ethereum platform allows you to send encrypted messages through smart contract.
- Only you and the recipient of a message can decrypt it.
- Every Ethereum account has a private key and a public key associated with it.
- The private key is what you have to keep in secret (or your wallet software will keep it in secret for you).
- If someone got your private key, they will be able to control your Ethereum account and able to decrypt all your messages.

Objectives

- To provide more secure environment for chatting and resource sharing.
- To provide more efficient system that works even if a node in the network fails.

Existing System and Proposed System

- WhatsApp, WeChat, etc, these traditional applications have taken all over the internet. There is a centralized server which stores all the information including identity to chats. Generally, these chat applications based on the following:

- Centralized Management: In this management system, entire correspondence goes through the company's server which can govern its rules.
- Centralized Architecture: In this architecture, there is only single server which is maintaining all the services.
- Confidentiality: Confidentiality of a user can be compromised on the request of government.
- Single Point of Failure (SPF): If a single node fails then whole application can be compromised.

- The above encouraged us to build an application where, we can have all the features like: Decentralized storage, Data security and Data immutability.

- In our application, we are using the approach of decentralized application (DApp).
- All the user data is stored on a block which is connected to other blocks forming a chain. It is a peer-to-peer network.
- And, tampering the data which is stored on the blockchain is quite impossible because, of the encryption algorithm.
- If malicious user tries to make changes to the information in block then, he/she will have to make changes to all the copies of that block on whole blockchain network and that can be quite impossible.
- Though blocks are on all nodes, they cannot access the information in it, only the person for whom the information is concerned, they can only access.

Modules/Sub-tasks

1. Environment setup

   We have to install all the needed dependencies and environements.

   Here is a list of the dependencies we need to install:

   • node.js

   • metamask

   • truffle

   • ganache

2. Deploy smart contract

   2.1: Connect metamask to the browser

   2.2: Create the smart contract

   2.3: Deploy the smart contract

3. Send message

   3.1: connect to all the available wallet addresses available in Ganache

   3.2: send messages between these addresses

   3.3: monitor the state of the blockchain in real time when the transactions are executed

Code

- Method used to connect the browser to metamask

```
class App extends Component {

  async componentWillMount() {
    await this.loadWeb3()
    console.log(window.web3)
    this.loadBlockchainData()
  }

  async loadWeb3() {
    if (window.ethereum) { //check if Metamask is installed
      try {
          const address = await window.ethereum.enable(); //connect Metamask
          console.log({
            connectedStatus: true,
            status: "",
            address: address
          })
      } catch (error) {
          console.log({
            connectedStatus: false,
            status: "🦊 Connect to Metamask using the button on the top right."
          })
      }
    } else {
          console.log({
              connectedStatus: false
          })
       }
    }
```

- Basic contract used to test the connection between truffle and our app

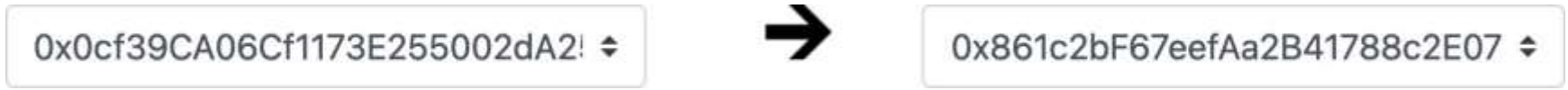
- Migration file used by the blockchain to deploy our smart contract

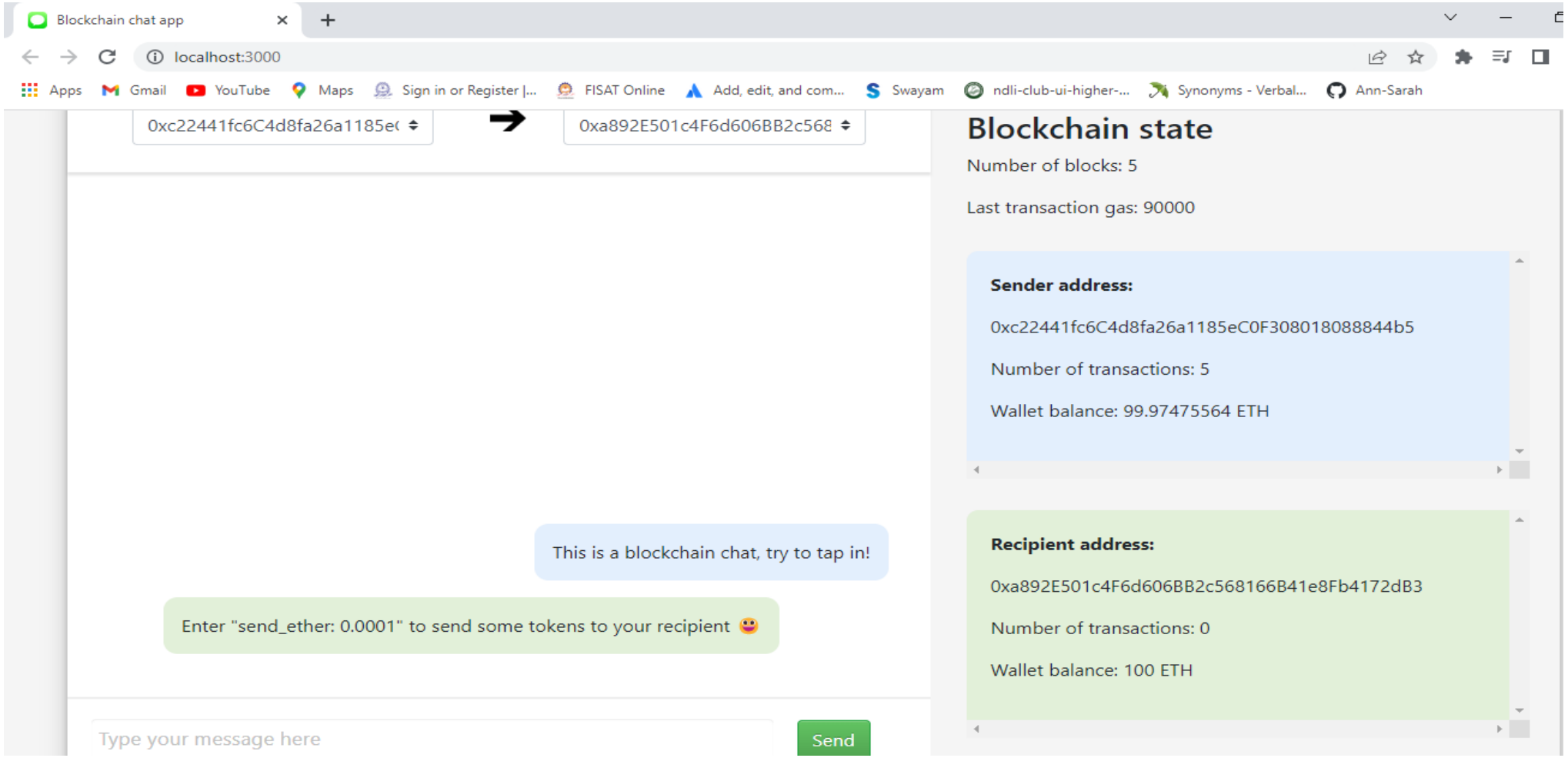
- code used to send a message

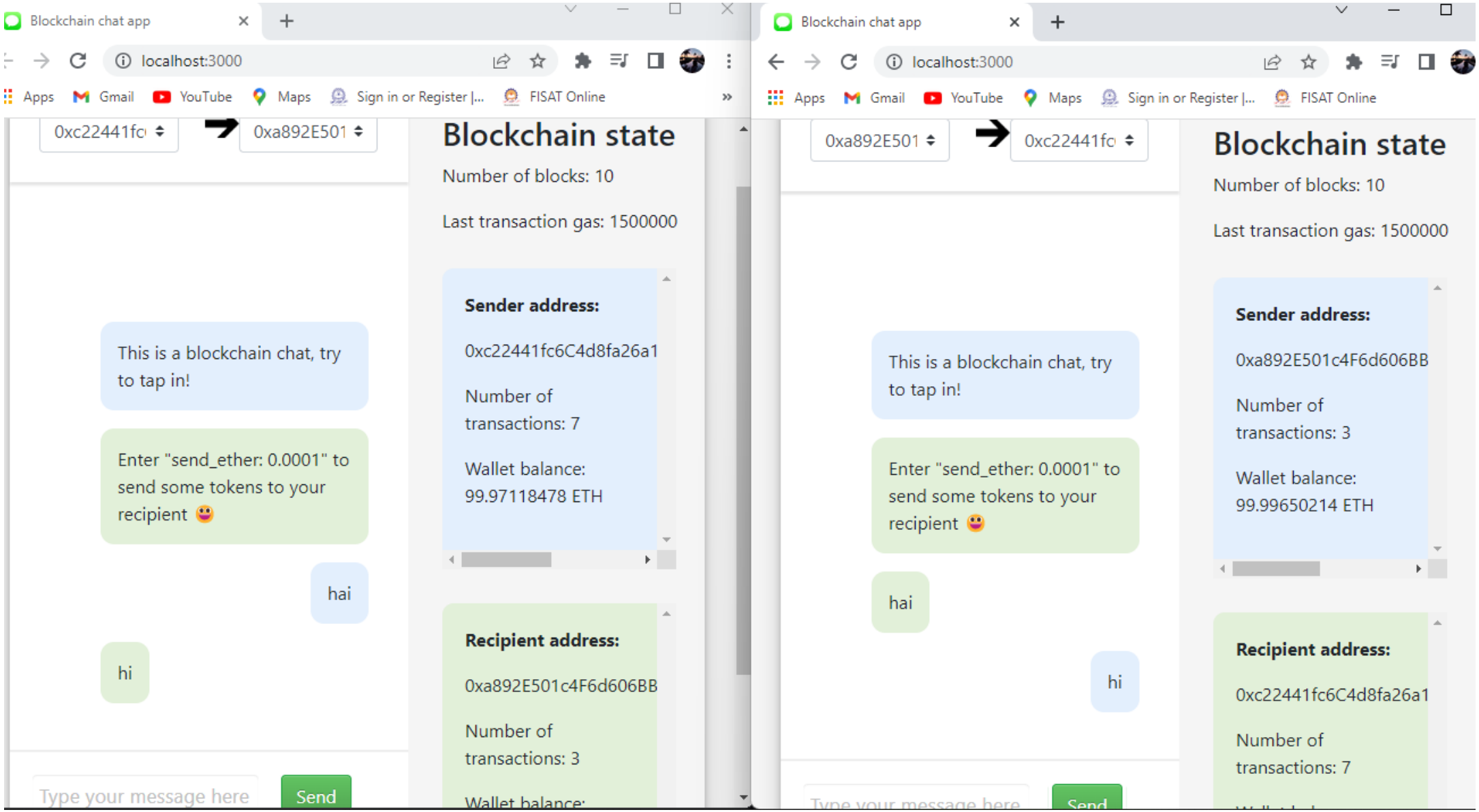
```
async didSendMessage(message) {
    await this.state.chatContract.methods.sendMsg(this.state.otherAccount, message).send({ from: this.state.account, gas: 1500000 })
}
```

Screenshot

- Address selection: sender and recipient




Blockchain chat app
localhost:3000
Apps
Gmail
YouTube
Maps
Sign in or Register |...
FISAT Online
Add, edit, and com...
Swayam
ndli-club-ui-higher-...
Synonyms - Verbal...
Ann-Sarah
0xc22441fc6C4d8fa26a1185e(
0xa892E501c4F6d606BB2c568
This is a blockchain chat, try to tap in!
Enter "send_ether: 0.0001" to send some tokens to your recipient
Type your message here
Send
Blockchain state
Number of blocks: 5
Last transaction gas: 90000
Sender address:
0xc22441fc6C4d8fa26a1185eC0F308018088844b5
Number of transactions: 5
Wallet balance: 99.97475564 ETH
Recipient address:
0xa892E501c4F6d606BB2c568166B41e8Fb4172dB3
Number of transactions: 0
Wallet balance: 100 ETH


Blockchain chat app
localhost:3000
Apps
Gmail
YouTube
Maps
Sign in or Register |...
FISAT Online
0xc22441fc
0xa892E501
Blockchain state
Number of blocks: 10
Last transaction gas: 1500000
Sender address:
0xc22441fc6C4d8fa26a1
Number of transactions: 7
Wallet balance: 99.97118478 ETH
This is a blockchain chat, try to tap in!
Enter "send_ether: 0.0001" to send some tokens to your recipient
hai
hi
Recipient address:
0xa892E501c4F6d606BB
Number of transactions: 3
Type your message here
Send
Blockchain chat app
localhost:3000
Apps
Gmail
YouTube
Maps
Sign in or Register |...
FISAT Online
0xa892E501
0xc22441fc
Blockchain state
Number of blocks: 10
Last transaction gas: 1500000
Sender address:
0xa892E501c4F6d606BB
Number of transactions: 3
Wallet balance: 99.99650214 ETH
This is a blockchain chat, try to tap in!
Enter "send_ether: 0.0001" to send some tokens to your recipient
hai
hi
Recipient address:
0xc22441fc6C4d8fa26a1
Number of transactions: 7
Send