Chapter 4

Implementation & Testing

4.1 Implementation Approach

The implementation approach for the "Decentralized Transaction Application" involves leveraging blockchain technology, specifically the Ethereum blockchain, to develop a decentralized system for financial transactions. The approach consists of the following key steps:

1. Requirements Analysis and Feasibility Study:

- Conduct a detailed analysis of project requirements, including functional, non-functional, and technical requirements.
- Evaluate the feasibility of implementing a decentralized transaction system using blockchain technology, considering factors such as scalability, security, and user experience.

2. Technology Selection:

- Choose appropriate technologies and frameworks aligned with the project's objectives and requirements.
- Select Ethereum as the primary blockchain platform due to its support for smart contracts and widespread adoption in decentralized applications.

3. Architecture Design:

- Design the system architecture, including frontend and backend components, smart contracts, and blockchain integration.

- Define the data model and data flow within the application, ensuring seamless interaction between different modules.

4. Development Iterations:

- Adopt an iterative development approach, dividing the development process into manageable iterations or sprints.
- Implement core functionalities incrementally, starting with user authentication, transaction processing, and smart contract integration.

5. Code Quality Assurance:

- Enforce coding standards, best practices, and code reviews to maintain code quality and consistency.
- Implement automated testing practices, including unit testing, integration testing, and end-to-end testing, to ensure the reliability and robustness of the application.

6. Security Measures:

- Implement security measures such as encryption, authentication, and authorization to protect user data and transactions.
- Conduct security audits and vulnerability assessments to identify and address potential security threats.

7. Deployment and Continuous Integration:

- Deploy the application in stages, starting with a test environment for internal testing and validation.
- Utilize continuous integration/continuous deployment (CI/CD) pipelines to automate the deployment process and ensure smooth deployment of updates and enhancements.

4.2 Coding Details

4.2.1 Client Folder

1. App.jsx

```
import {
  Welcome,
  EthereumChart,
  Footer,
  AboutCard,
  Transactions,
  Cursor,
} from "./components";

const App = () => (
  <div className="min-h-screen">
    <div className="gradient-bg-welcome">
      <Cursor />
      <Welcome />
      <EthereumChart />
    </div>
    <AboutCard />
    <Transactions />
    <Footer />
  </div>
);

export default App;
```

2. main.jsx

```
import React from "react";
import ReactDOM from "react-dom";

import App from "./App";
import { TransactionsProvider } from "./context/TransactionContext";
import "./index.css";

ReactDOM.render(
  <TransactionsProvider>
    <App />
  </TransactionsProvider>,
  document.getElementById("root"),
);
```

3. index.css

```
@import url("https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;400;500;600;700&display=swap");

:root {
  --scrollbar-primary: #000000;
  --scrollbar-secondary: #f6e27a;
}

* {
  padding: 0;
  margin: 0;
  scroll-behavior: smooth;
  cursor: none;
  box-sizing: border-box;
}
```

```
/* Firefox */
* {
  scrollbar-width: thin;
  scrollbar-color: var(--scrollbar-secondary) var(--scrollbar-primary);
  cursor: none;
}

/* Chrome, Edge, and Safari */
*::-webkit-scrollbar {
  width: 15px;
}

*::-webkit-scrollbar-track {
  background: var(--scrollbar-primary);
  border-radius: 0px;
}

*::-webkit-scrollbar-thumb {
  background-color: var(--scrollbar-secondary);
  border-radius: 14px;
  border: 3px solid var(--scrollbar-primary);
}

body {
  margin: 0;
  font-family: "Open Sans", sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.gradient-bg-welcome {
  background-color: #0f0e13;
  background-image: radial-gradient(
      at 0% 0%,
      rgb(109, 58, 17) 0,
      transparent 50%
    ),
    radial-gradient(at 50% 0%, rgba(255, 255, 255, 0.575) 0, transparent 50%),
    radial-gradient(at 100% 0%, rgba(66, 6, 71, 0.493) 0, transparent 50%);
}

.gradient-bg-aboutcard {
  background-color: #0f0e13;
  background-image: radial-gradient(
      at 0% 0%,
      hsla(253, 16%, 7%, 1) 0,
      transparent 50%
    ),
    radial-gradient(at 50% 100%, rgba(253, 238, 102, 0.459) 0, transparent 50%);
}

.gradient-bg-transactions {
  background-color: #0f0e13;
  background-image: radial-gradient(
      at 0% 100%,
      hsla(253, 16%, 7%, 1) 0,
      transparent 50%
    ),
    radial-gradient(at 50% 0%, rgba(253, 238, 102, 0.459) 0, transparent 50%);
}

.gradient-bg-footer {
  background-color: #0f0e13;
  background-image: radial-gradient(
      at 0% 100%,
      hsla(253, 16%, 7%, 1) 0,
      transparent 53%
    ),
    radial-gradient(at 50% 150%, rgba(255, 0, 0, 0.555) 0, transparent 50%);
}

.blue-glassmorphism {
  background: rgba(0, 0, 0, 0.178);
  border-radius: 16px;
  box-shadow: 0px 0px 30px rgba(255, 255, 255, 0.493);
  backdrop-filter: blur(10px);
  -webkit-backdrop-filter: blur(5px);
  border: 1px solid rgba(240, 233, 233, 0.3);

  opacity: 0;
  animation: slideLeft 1s ease forwards;
  animation-delay: 1.3s;
}

.features {
  opacity: 0;
  animation: zoomIn 1s ease forwards;
  animation-delay: 2s;
}

span {
  transform: translate(-50%, -50%);
  top: 50%;
  left: 50%;
  font-family: "Times New Roman", serif;
  letter-spacing: 5px;
  font-size: 130px;
  font-weight: bold;
  background: linear-gradient(
    to right,
    #462523bd 0,
    #cb9b51 22%,
    #f6e27a 45%,
    #f5ed7f 50%,
    #f6e27a 55%,
    #cb9b51 78%,
    #462523bd 100%
  );
  color: transparent;
  -webkit-background-clip: text;
}

.Ether-back {
  text-shadow: 5px 2px 10px #000000;
}

.border-clr {
  border-color: #cb9b51;
}

.btn-box {
  opacity: 0;
  animation: zoomIn 1s ease forwards;
  animation-delay: 0.6s;
```

```
}

.btn-box:hover {
  box-shadow: 0 0 5px #daa455e8, 0 0 25px #daa455e8, 0 0 50px #daa455e8,
    0 0 100px #daa455e8, 0 0 200px #daa455e8;
}

.dec-ani {
  opacity: 0;
  animation: slideRight 1s ease forwards;
  animation-delay: 0.7s;
}

.tag-ani {
  opacity: 0;
  animation: slideLeft 1s ease forwards;
  animation-delay: 0.7s;
}

/* white glassmorphism */
.white-glassmorphism {
  background: rgba(255, 255, 255, 0.05);
  border-radius: 16px;
  backdrop-filter: blur(5px);
  -webkit-backdrop-filter: blur(5px);
  border: 1px solid rgba(255, 255, 255, 0.3);
  width: 90%;
  height: auto;

  opacity: 0;
  animation: zoomIn 1s ease forwards;
  animation-delay: 1.3s;
}

.eth-card {
  background-image: linear-gradient(
    to right,
    #462523bd 0,
    #cb9b51 22%,
    #f6e27a 45%,
    #f5ed7f 50%,
    #f6e27a 55%,
    #cb9b51 78%,
    #462523bd 100%
  );
  color: transparent;

  opacity: 0;
  animation: zoomIn 1s ease forwards, floatImage 4s ease-in-out infinite;
  animation-delay: 0.6s, 2.1s;
}

.text-gradient {
  background-color: #fff;
  background-image: radial-gradient(
      at 4% 36%,
      rgb(255, 255, 255) 0,
      transparent 53%
    ),
    radial-gradient(at 100% 60%, rgb(253, 255, 114) 0, transparent 50%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
```

```
}

.border-company {
  border-color: #cb9b51;
}

.rotate-3d {
  animation: rotate 3s infinite linear;
  transform-origin: center;
}

.about {
  margin-top: 5px;
  margin-bottom: 5px;
}

.about-img {
  border-width: 1px;
  /* border-color: rgba(0, 0, 0, 0.644); */
  border-color: rgba(240, 233, 233, 0.3);
}

.about:hover {
  transform: scale(6); /* Adjust the scale factor as needed */
  transition: transform 0.3s ease; /* Add a smooth transition effect */
  box-shadow: 0 0 5px rgba(240, 233, 233, 0.3),
    0 0 25px rgba(240, 233, 233, 0.3), 0 0 50px rgba(240, 233, 233, 0.3),
    0 0 100px rgba(240, 233, 233, 0.3), 0 0 200px rgba(240, 233, 233, 0.3);
}

.transaction {
  border-radius: 10px;
  background: #050505c7;
  box-shadow: inset 20px 20px 67px #040404, inset -20px -20px 67px #060606;
}

@keyframes rotate {
  0% {
    transform: rotateY(0deg);
  }
  100% {
    transform: rotateY(360deg);
  }
}

@keyframes slideTop {
  0% {
    opacity: 0;
    transform: translateY(100px);
  }

  100% {
    opacity: 1;
    transform: translateY(0);
  }
}

@keyframes slideRight {
  0% {
    opacity: 0;
    transform: translateX(-100px);
  }
  100% {
    opacity: 1;
    transform: translateX(0);
  }
}

@keyframes slideLeft {
  0% {
    opacity: 0;
    transform: translateX(100px);
  }

  100% {
    opacity: 1;
    transform: translateX(0);
  }
}

@keyframes zoomIn {
  0% {
    opacity: 0;
    transform: scale(0);
  }

  100% {
    opacity: 1;
    transform: scale(1);
  }
}
```

```
@keyframes floatImage {
 0% {
  transform: translateY(0);
 }

 50% {
  transform: translateY(-24px);
 }

 100% {
  transform: translate(0);
 }
}

@keyframes circleRotate {
 0% {
  transform: rotate(0);
 }

 100% {
  transform: rotate(360deg);
 }
}

@keyframes fadeIn {
 0% {
  opacity: 0;
 }
 100% {
  opacity: 1;
 }
}

@tailwind base;
@tailwind components;
@tailwind utilities;
```

- **Client Folder – context**

1. TransactionContext.jsx

```
import React, { useEffect, useState } from "react";
import { ethers } from "ethers";

import { contractABI, contractAddress } from "../utils/constants";

export const TransactionContext = React.createContext();

const { ethereum } = window;

const createEthereumContract = () => {
  const provider = new ethers.providers.Web3Provider(ethereum);
  const signer = provider.getSigner();
  const transactionsContract = new ethers.Contract(contractAddress, contractABI, signer);

  return transactionsContract;
};

export const TransactionsProvider = ({ children }) => {
  const [formData, setformData] = useState({ addressTo: "", amount: "", keyword: "", message:
"" });
  const [currentAccount, setCurrentAccount] = useState("");
  const [isLoading, setIsLoading] = useState(false);
  const [transactionCount, setTransactionCount] = useState(localStorage.getItem("transactionCount"));
  const [transactions, setTransactions] = useState([]);

  const handleChange = (e, name) => {
    setformData((prevState) => ({ ...prevState, [name]: e.target.value }));
  };

  const getAllTransactions = async () => {
    try {
      if (ethereum) {
        const transactionsContract = createEthereumContract();

        const availableTransactions = await transactionsContract.getAllTransactions();

        const structuredTransactions = availableTransactions.map((transaction) => ({
          addressTo: transaction.receiver,
          addressFrom: transaction.sender,
          timestamp: new Date(transaction.timestamp.toNumber() * 1000).toLocaleString(),
          message: transaction.message,
          keyword: transaction.keyword,
          amount: parseInt(transaction.amount._hex) / (10 ** 18)
        }));

        console.log(structuredTransactions);

        setTransactions(structuredTransactions);
      } else {
        console.log("Ethereum is not present");
      }
    } catch (error) {
```

```
    console.log(error);
  }
};

const checkIfWalletIsConnect = async () => {
  try {
    if (!ethereum) return alert("Please install MetaMask.");

    const accounts = await ethereum.request({ method: "eth_accounts" });

    if (accounts.length) {
      setCurrentAccount(accounts[0]);

      getAllTransactions();
    } else {
      console.log("No accounts found");
    }
  } catch (error) {
    console.log(error);
  }
};

const checkIfTransactionsExists = async () => {
  try {
    if (ethereum) {
      const transactionsContract = createEthereumContract();
      const currentTransactionCount = await transactionsContract.getTransactionCount();

      window.localStorage.setItem("transactionCount", currentTransactionCount);
    }
  } catch (error) {
    console.log(error);

    throw new Error("No ethereum object");
  }
};

const connectWallet = async () => {
  try {
    if (!ethereum) return alert("Please install MetaMask.");

    const accounts = await ethereum.request({ method: "eth_requestAccounts", });

    setCurrentAccount(accounts[0]);
    window.location.reload();
  } catch (error) {
    console.log(error);

    throw new Error("No ethereum object");
  }
};

const sendTransaction = async () => {
  try {
    if (ethereum) {
      const { addressTo, amount, keyword, message } = formData;
      const transactionsContract = createEthereumContract();
      const parsedAmount = ethers.utils.parseEther(amount);

      await ethereum.request({
        method: "eth_sendTransaction",
        params: [{
          from: currentAccount,
          to: addressTo,
          gas: "0x5208",
          value: parsedAmount._hex,
        }],
      });

      const transactionHash = await transactionsContract.addToBlockchain(addressTo, parsedAmount, message, keyword);

      setIsLoading(true);
      console.log(`Loading - ${transactionHash.hash}`);
      await transactionHash.wait();
      console.log(`Success - ${transactionHash.hash}`);
      setIsLoading(false);

      const transactionsCount = await transactionsContract.getTransactionCount();

      setTransactionCount(transactionsCount.toNumber());
      window.location.reload();
    } else {
      console.log("No ethereum object");
    }
  } catch (error) {
    console.log(error);

    throw new Error("No ethereum object");
  }
};

useEffect(() => {
    checkIfWalletIsConnect();
    checkIfTransactionsExists();
  }, [transactionCount]);

  return (
    <TransactionContext.Provider
      value={{
        transactionCount,
        connectWallet,
        transactions,
        currentAccount,
        isLoading,
        sendTransaction,
        handleChange,
        formData,
      }}
    >
      {children}
    </TransactionContext.Provider>
  );
};
```

- **Client Folder – hooks**

1. useFetch.jsx

```
import { useEffect, useState } from "react";

const APIKEY = import.meta.env.VITE_GIPHY_API;

const useFetch = ({ keyword }) => {
  const [gifUrl, setGifUrl] = useState("");
  const fetchGifs = async () => {
    try {
      const response = await fetch(`https://api.giphy.com/v1/gifs/search?api_key=${APIKEY}&q=${keyword.split(" ").join("")}&limit=1`);
      const { data } = await response.json();

      setGifUrl(data[0]?.images?.downsized_medium.url);
    } catch (error) {
      setGifUrl("https://metro.co.uk/wp-content/uploads/2015/05/pokemon_crying.gif?quality=90&strip=all&zoom=1&resize=500%2C284");
    }
  };

  useEffect(() => {
    if (keyword) fetchGifs();
  }, [keyword]);

  return gifUrl;
};

export default useFetch;
```

- **Client Folder - components**

1. About.jsx

```
import React from "react";
import { saket, sai, ether } from "../assets";

const About = ({ color, title, image, subtitle1, subtitle2 }) => (
    <div className="flex flex-row justify-start items-start white-glassmorphism p-3 m-2 cursor-
pointer hover:shadow-xl">
    <div
      className={`w-10 h-10 rounded-full flex justify-center items-center ${color} about`}
    >
      {image ? (
        <img
          src={image}
          alt={`Image of ${title}`}
          className="w-10 h-10 about-img rounded-full"
        />
      ) : null}
    </div>
    <div className="ml-5 flex flex-col flex-1">
      <h3 className="mt-2 text-white text-lg">{title}</h3>
      <p className="mt-2 text-white text-sm md:w-9/12">
        {subtitle1}
        <p>{subtitle2}</p>
      </p>
    </div>
  </div>
);

const AboutCard = () => (
  <div className="flex w-full justify-center items-center gradient-bg-aboutcard">
    <div className="flex mf:flex-row flex-col items-center justify-between md:p-20 py-12 px-4">
      <div className="flex-1 flex flex-col justify-start items-start">
        <h1 className="text-white text-3xl sm:text-5xl py-2 text-gradient ">
          Name of the College, Student and Project.
        </h1>
        <p className="text-left my-2 text-white font-light md:w-9/12 w-11/12 text-base">
          The purpose of this crypto exchange project is to enable the exchange
          of Ethereum, offering users a platform to transfer their digital
          assets securely and conveniently.
        </p>
      </div>
      <div className="flex-1 flex flex-col justify-start items-center">
        <About
          color=""
          title="SAKET COLLEGE OF ARTS, SCIENCE & COMMERCE"
          image={saket}
          subtitle=""
        />
        <About
          color=""
          title="TERUKULA SAI"
          image={sai}
          subtitle1="PNR No : 2021016400730531"
          subtitle2="Roll No : 233026 "
        />
        <About
          color=""
          title="Decentralized Transaction Web3 Application"
          image={ether}
          subtitle=""
        />
      </div>
    </div>
  </div>
);
export default AboutCard;
```

2. EthereumChart.jsx

```
import React, { useEffect, useRef, useState } from "react";
import Chart from "chart.js/auto";
import axios from "axios";

const EthereumChart = () => {
  const chartRef = useRef(null);
  const [priceData, setPriceData] = useState([]);
  const [dataToShow, setDataToShow] = useState(7);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await axios.get(
          `https://api.coingecko.com/api/v3/coins/ethereum/market_chart?vs_currency=usd&days=$
{dataToShow}&interval=daily`
        );
        const data = response.data.prices;
        setPriceData(data);
      } catch (error) {
        console.error("Error fetching Ethereum price data:", error);
      }
    };

    fetchData();
  }, [dataToShow]);

  useEffect(() => {
    if (priceData.length === 0 || !chartRef.current) {
      return;
```

```
}

const timestamps = priceData.map((entry) =>
  new Date(entry[0]).toLocaleDateString()
);
const prices = priceData.map((entry) => entry[1]);
const ctx = chartRef.current.getContext("2d");

new Chart(ctx, {
  type: "line",
  data: {
    labels: timestamps,
    datasets: [
      {
        label: "Ethereum Price (USD)",
        data: prices,
        borderColor: "rgba(203, 155, 81, 1)", // Adjust color for a 3D-like effect
        borderWidth: 1.5,
        fill: true,
        backgroundColor: "rgba(203, 155, 81, 0.1)", // Adjust color for a 3D-like effect
        pointRadius: 3,
        pointBackgroundColor: "rgba(203, 155, 81, 1)", // Adjust color for a 3D-like effect
        pointBorderWidth: 2,
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
    scales: {
      x: {
```

```
      display: true,
      title: {
        display: true,
        text: "Date",
        color: "white",
        font: {
          size: 18,
        },
      },
      ticks: {
        color: "white",
        font: {
          size: 14,
        },
        maxRotation: 0,
        maxTicksLimit: 10, // Number of visible ticks
      },
    },
    y: {
      display: true,
      title: {
        display: true,
        text: "Price (USD)",
        color: "white",
        font: {
          size: 18,
        },
      },
      ticks: {
        color: "white",
        font: {
            size: 14,
          },
        },
      },
    },
    plugins: {
      legend: {
        display: true,
        labels: {
          color: "rgba(203, 155, 81, 1)", // Adjust color for a 3D-like effect
          font: {
            size: 18,
          },
        },
      },
      tooltip: {
        enabled: true,
      },
    },
  },
 });
}, [priceData]);

return (
  <div>
    <canvas ref={chartRef} className="ethereum-chart" />
  </div>
);
};
export default EthereumChart;
```

3. Loader.jsx

```
const Loader = () => (
  <div className="flex justify-center items-center py-3">
    <div className="animate-spin rounded-full h-32 w-32 border-b-4 border-clr" />
  </div>
);

export default Loader;
```

4. Transaction.jsx

```
import React, { useContext } from "react";

import { TransactionContext } from "../context/TransactionContext";

import useFetch from "../hooks/useFetch";
import dummyData from "../utils/dummyData";
import { shortenAddress } from "../utils/shortenAddress";

const TransactionsCard = ({
  addressTo,
  addressFrom,
  timestamp,
  message,
  keyword,
  amount,
  url,
}) => {
  const gifUrl = useFetch({ keyword });

  return (
<div
  className="m-4 flex flex-1
  2xl:min-w-[450px]
  2xl:max-w-[500px]
  sm:min-w-[270px]
  sm:max-w-[300px]
  min-w-full
  flex-col p-3 hover:shadow-2xl transaction"
>
  <div className="flex flex-col items-center w-full mt-3">
    <div className="display-flex justify-start w-full mb-6 p-2">
      <a
        href={`https://sepolia.etherscan.io/address/${addressFrom}`}
        target="_blank"
        rel="noreferrer"
      >
        <p className="text-white text-base">
          From: {shortenAddress(addressFrom)}
        </p>
      </a>
      <a
        href={`https://sepolia.etherscan.io/address/${addressTo}`}
        target="_blank"
        rel="noreferrer"
      >
        <p className="text-white text-base">
          To: {shortenAddress(addressTo)}
        </p>
      </a>
      <p className="text-white text-base">Amount: {amount} ETH</p>
      {message && (
          <>
            <br />
            <p className="text-white text-base">Message: {message}</p>
          </>
        )}
      </div>
      <img
        src={gifUrl || url}
        alt="nature"
        className="w-full h-64 2xl:h-96 rounded-md shadow-lg object-cover"
      />
      <div className="bg-black p-3 px-5 w-max rounded-3xl -mt-5 shadow-2xl">
        <p className="text-[#c8a849] font-bold">{timestamp}</p>
      </div>
    </div>
  </div>
 );
};

const Transactions = () => {
  const { transactions, currentAccount } = useContext(TransactionContext);

  return (
    <div className="flex w-full justify-center items-center 2xl:px-20 gradient-bg-transactions">
      <div className="flex flex-col md:p-12 py-12 px-4">
        {currentAccount ? (
          <h3 className="text-white text-3xl text-center my-2">
            Latest Transactions
          </h3>
        ) : (
          <h3 className="text-white text-3xl text-center my-2">
          Connect your account to see the latest transactions
        </h3>
      )}

      <div className="flex flex-wrap justify-center items-center mt-10">
        {[...dummyData, ...transactions].reverse().map((transaction, i) => (
          <TransactionsCard key={i} {...transaction} />
        ))}
      </div>
    </div>
  </div>
 );
};

export default Transactions;
```

5. Welcome.jsx

```
import React, { useContext } from "react";
import {
  AiFillAccountBook,
  AiFillPlayCircle,
  AiOutlineApi,
} from "react-icons/ai";
import { SiEthereum } from "react-icons/si";
import { BsInfoCircle } from "react-icons/bs";

import { TransactionContext } from "../context/TransactionContext";
import { shortenAddress } from "../utils/shortenAddress";
import { Loader } from ".";

const companyCommonStyles =
  "min-h-[70px] sm:px-0 px-2 sm:min-w-[120px] flex justify-center items-center border-[0.2px] border-company text-sm font-light text-white";


const Input = ({ placeholder, name, type, value, handleChange }) => (
  <input
    placeholder={placeholder}
    type={type}
    step="0.000001"
    value={value}
    onChange={(e) => handleChange(e, name)}
    className="my-2 w-full rounded-sm p-2 outline-none bg-transparent text-white border-[1px] border-[#c8a849] text-sm white-glassmorphism"
  />
);


const Welcome = () => {
  const {
    currentAccount,
    connectWallet,
    handleChange,
    sendTransaction,
    formData,
    isLoading,
  } = useContext(TransactionContext);


  const handleSubmit = (e) => {
    const { addressTo, amount, keyword, message } = formData;


    e.preventDefault();


    if (!addressTo || !amount || !keyword || !message) return;
  sendTransaction();
};

return (
  <div className="flex w-full justify-center items-center">
    <div className="flex mf:flex-row flex-col items-start justify-between md:p-20 py-12 px-4">
      <div className="flex flex-1 justify-start items-start flex-col mf:mr-10">
        <h1 className="text-3xl sm:text-5xl text-white text-gradient py-1 dec-ani">
          Connecting you to the <span>Decentralize</span> future.
        </h1>
        <p className="text-left mt-5 text-white font-light md:w-9/12 w-11/12 text-base tag-ani">
          Experience a secure, user-friendly platform for seamless
          cryptocurrency exchange. Unlock the potential of digital assets and
          shape your financial future with confidence.
        </p>
        {!currentAccount && (
          <button
            type="button"
            onClick={connectWallet}
             className="flex flex-row justify-center items-center my-5 bg-[#c0c0c06c] p-3 rounded-
full cursor-pointer hover:bg-[#daa455e8] btn-box"
          >
            <AiOutlineApi className="text-white mr-2" />
            <p className="text-white text-base font-semibold">
              Connect Wallet
            </p>
          </button>
        )}

        <div className="grid sm:grid-cols-3 grid-cols-2 w-full mt-10 features">
        <div className={`rounded-tl-2xl ${companyCommonStyles}`}>
          Web 3.0
        </div>
        <div className={companyCommonStyles}>Security</div>
        <div className={`sm:rounded-tr-2xl ${companyCommonStyles}`}>
          Ethereum
        </div>
        <div className={`sm:rounded-bl-2xl ${companyCommonStyles}`}>
          Reliability
        </div>
        <div className={companyCommonStyles}>Low Fees</div>
        <div className={`rounded-br-2xl ${companyCommonStyles}`}>
          Blockchain
        </div>
      </div>
    </div>


    <div className="flex flex-col flex-1 items-center justify-start w-full mf:mt-0 mt-10">
        <div className="p-3 flex justify-end items-start flex-col rounded-xl h-40 sm:w-72 w-full
my-5 eth-card .white-glassmorphism ">
        <div className="flex justify-between flex-col w-full h-full">
          <div className="flex justify-between items-start">
                <div className="w-10 h-10 rounded-full border-2 border-white flex justify-center
items-center">
              <SiEthereum fontSize={21} color="#fff" />
            </div>
            <BsInfoCircle fontSize={17} color="#fff" />
          </div>
          <div>
            <p className="text-white font-light text-sm">
              {shortenAddress(currentAccount)}
          </p>
          <p className="text-white font-semibold text-lg mt-1 Ether-back">
            Ethereum
          </p>
        </div>
      </div>
    </div>

        <div className="p-5 sm:w-96 w-full flex flex-col justify-start items-center blue-
glassmorphism">
      <Input
        placeholder="Address to Transfer"
        name="addressTo"
        type="text"
        handleChange={handleChange}
      />
      <Input
        placeholder="Amount (ETH)"
        name="amount"
        type="number"
        handleChange={handleChange}
      />
      <Input
        placeholder="Keyword (Gif)"
        name="keyword"
        type="text"
        handleChange={handleChange}
      />
      <Input
        placeholder="Enter Message"
        name="message"
```

```
            type="text"
            handleChange={handleChange}
          />

          <div className="h-[1px] w-full bg-gray-400 my-2" />

          {isLoading ? (
            <Loader />
          ) : (
            <button
              type="button"
              onClick={handleSubmit}
              className="text-white w-full mt-2 border-[1px] p-2 border-[#c8a849] hover:bg-[#c8a849] hover:text-black rounded-full cursor-pointer"
            >
              Transfer
            </button>
          )}
        </div>
      </div>
    </div>
  </div>
  );
};

export default Welcome;
```

4.2.2 Smart Contract Folder

1. Transactions.sol

```
// SPDX-License-Identifier: UNLICENSED

pragma solidity ^0.8.0;

import "hardhat/console.sol";

contract Transactions {
   uint256 transactionCount;

    event Transfer(address from, address receiver, uint amount, string message, uint256 timestamp,
string keyword);

   struct TransferStruct {
       address sender;
       address receiver;
       uint amount;
       string message;
       uint256 timestamp;
       string keyword;
   }

   TransferStruct[] transactions;

      function addToBlockchain(address payable receiver, uint amount, string memory message,
string memory keyword) public {
       transactionCount += 1;
           transactions.push(TransferStruct(msg.sender, receiver, amount, message, block.timestamp,
keyword));
```

```
        emit Transfer(msg.sender, receiver, amount, message, block.timestamp, keyword);
    }

    function getAllTransactions() public view returns (TransferStruct[] memory) {
        return transactions;
    }

    function getTransactionCount() public view returns (uint256) {
        return transactionCount;
    }
}
```

2. hardhat.config.js

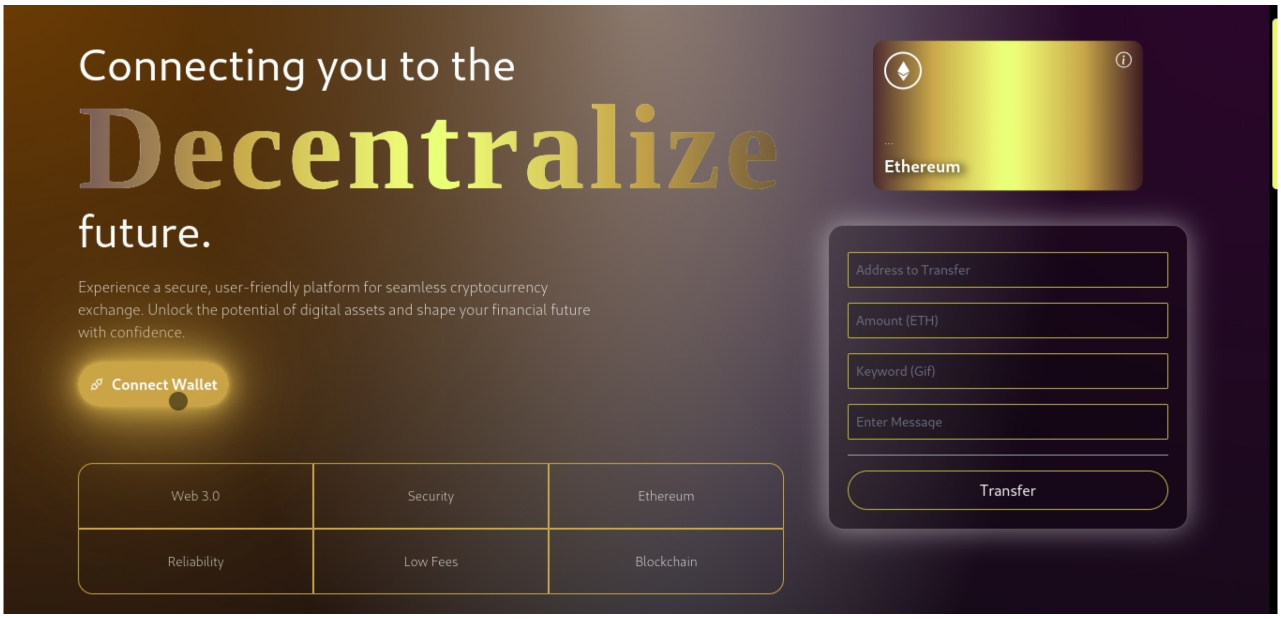
```
require('@nomiclabs/hardhat-waffle');

module.exports = {
  solidity: '0.8.0',
  networks: {
    sepolia: {
      url: 'https://eth-sepolia.g.alchemy.com/v2/mRTQxofIkLI_GF7HZZoqN1fWO6TLS7Ky',
      accounts: ['1a5c0009993bd25d28988d107c71b2ad6a5526db8e856sea9i25393cd9409c62'],
    },
  },
};
```

Chapter 5

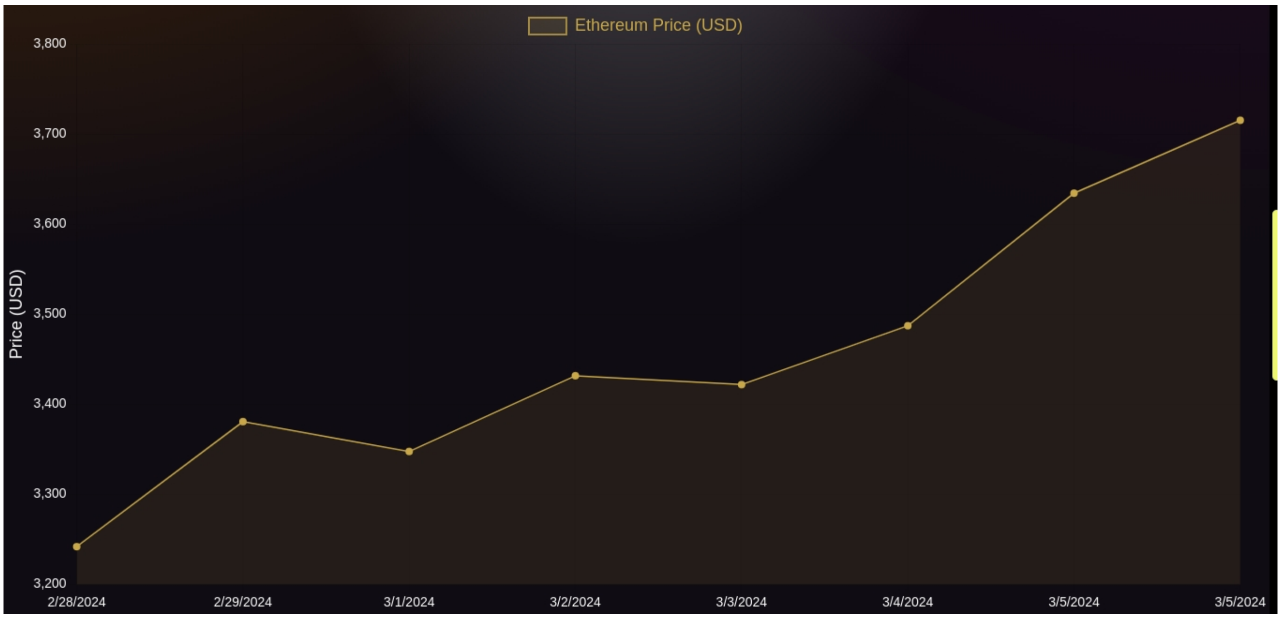
Results & Discussions

5.1 Interface before the connection of wallet

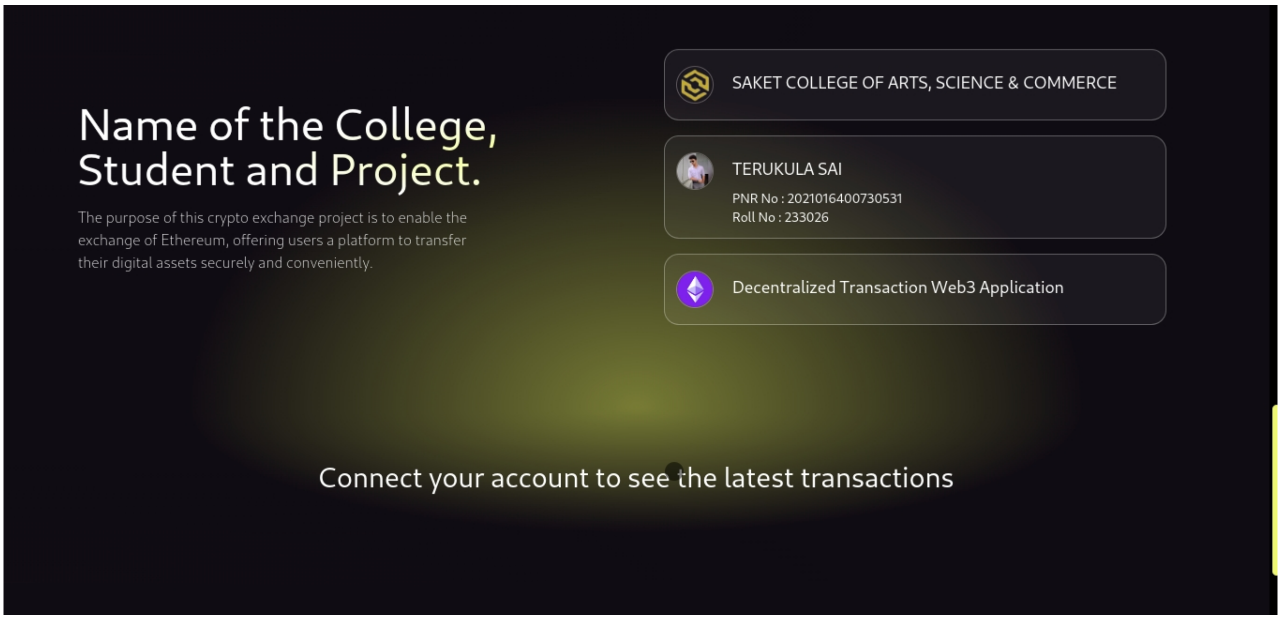
1. Home/Welcome Section



2. Ethereum Chart Section



3. About Section



Name of the College, Student and Project.

The purpose of this crypto exchange project is to enable the exchange of Ethereum, offering users a platform to transfer their digital assets securely and conveniently.

SAKET COLLEGE OF ARTS, SCIENCE & COMMERCE

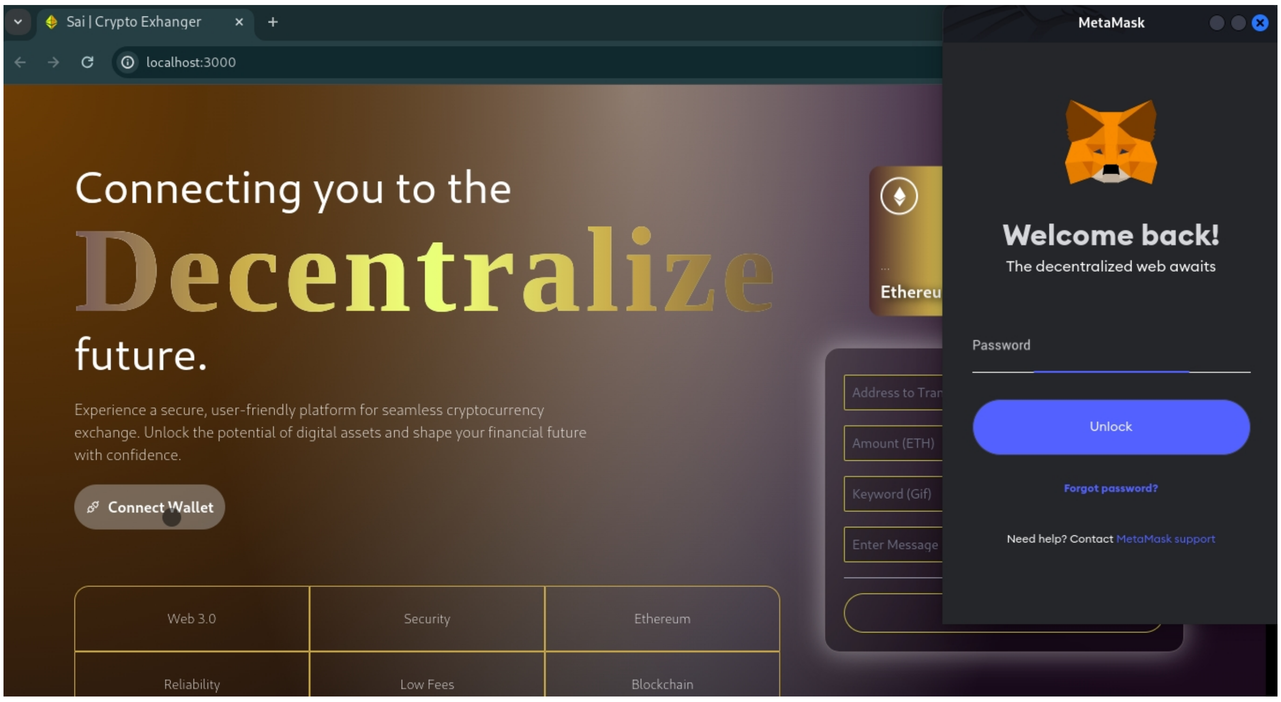
TERUKULA SAI
PNR No : 2021016400730531
Roll No : 233026

Decentralized Transaction Web3 Application

Connect your account to see the latest transactions

5.2 Interface after the connection of wallet

1. Account Login (Metamask Wallet)

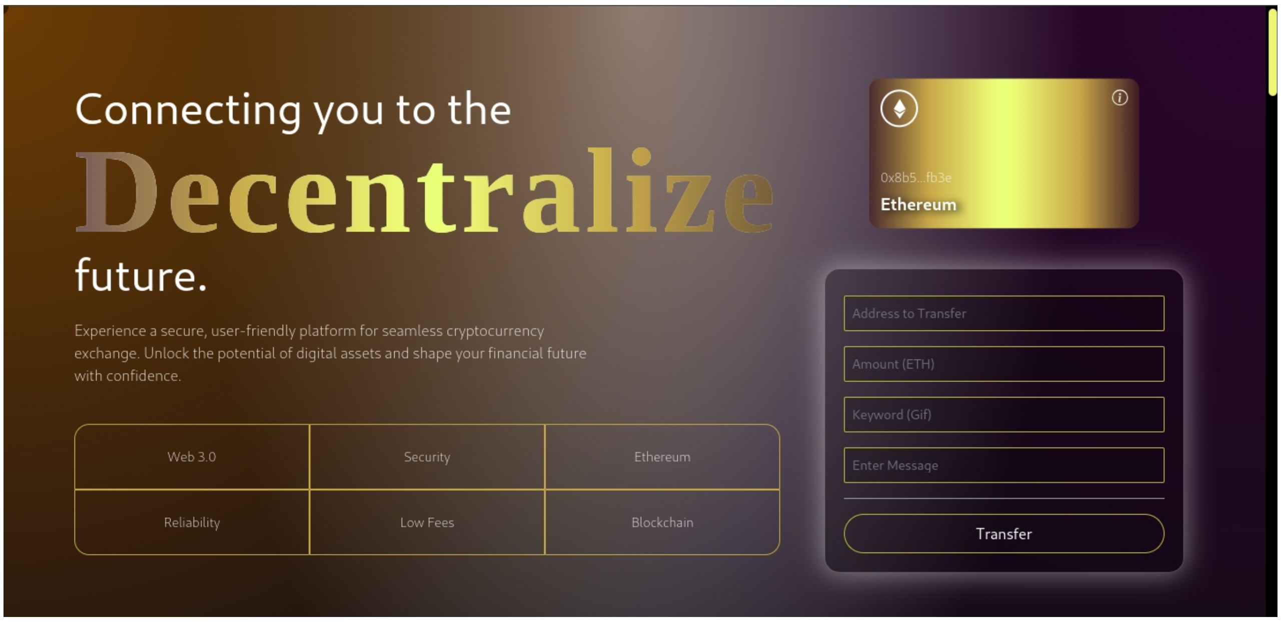


Sai | Crypto Exhanger
localhost:3000

Connecting you to the Decentralize future.

Experience a secure, user-friendly platform for seamless cryptocurrency exchange. Unlock the potential of digital assets and shape your financial future with confidence.

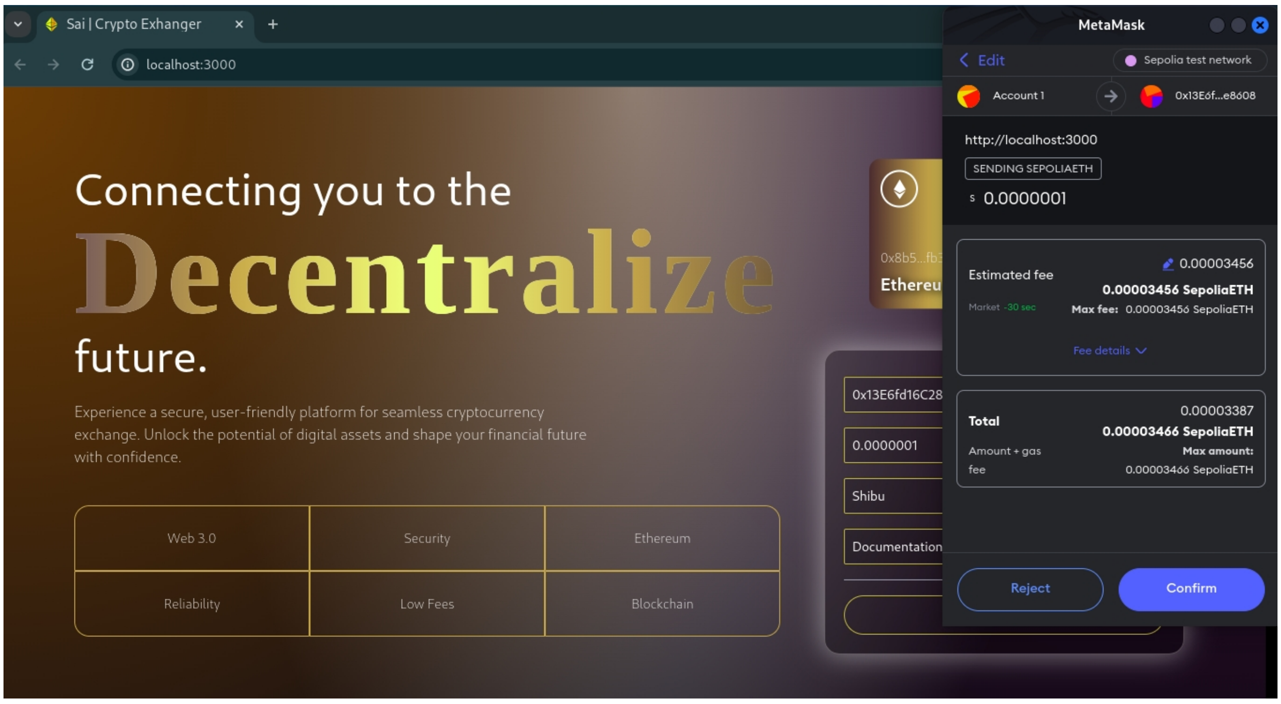
2. Home/Welcome Section



Connecting you to the Decentralize future.

Experience a secure, user-friendly platform for seamless cryptocurrency exchange. Unlock the potential of digital assets and shape your financial future with confidence.

3. Transaction Confirmation



Sai | Crypto Exhanger

localhost:3000

Connecting you to the Decentralize future.

Experience a secure, user-friendly platform for seamless cryptocurrency exchange. Unlock the potential of digital assets and shape your financial future with confidence.

0x13E6fd16C28

0.0000001

Shibu

Documentation

MetaMask

Sepolia test network

Account 1 → 0x13E6f...e8608

http://localhost:3000

SENDING SEPOLIAETH

0.0000001

Estimated fee 0.00003456

0.00003456 SepoliaETH

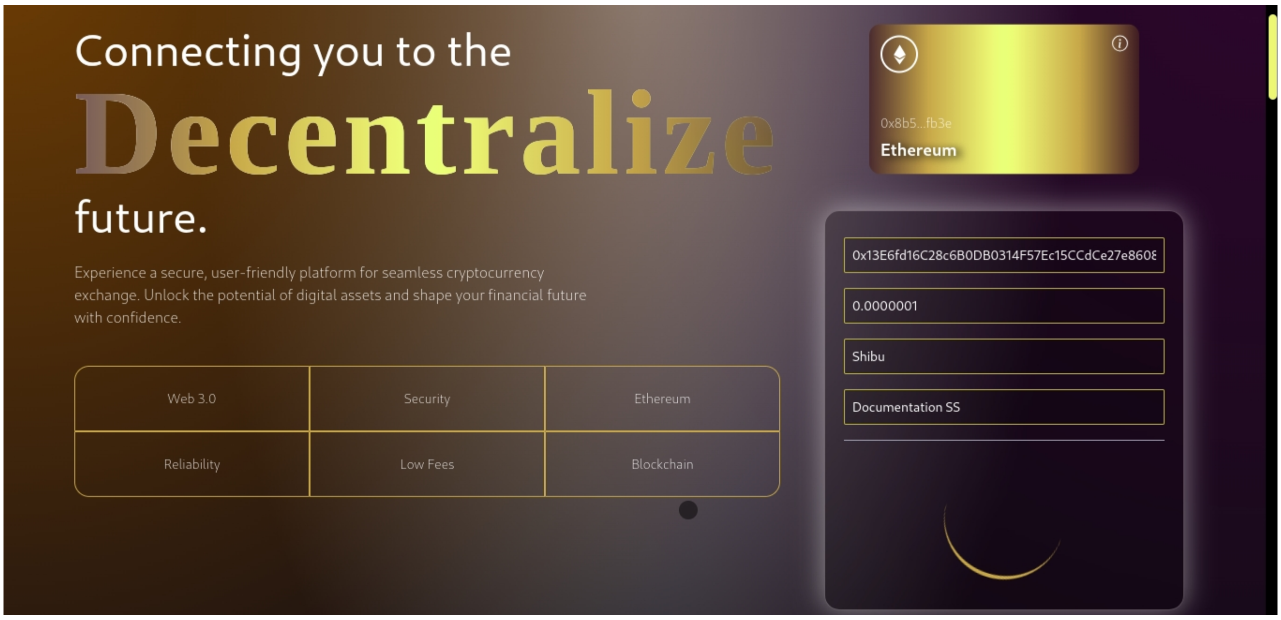
Market ~30 sec Max fee: 0.00003456 SepoliaETH

Total 0.00003387

0.00003466 SepoliaETH

Amount + gas fee Max amount: 0.00003466 SepoliaETH

4. Transaction Loader

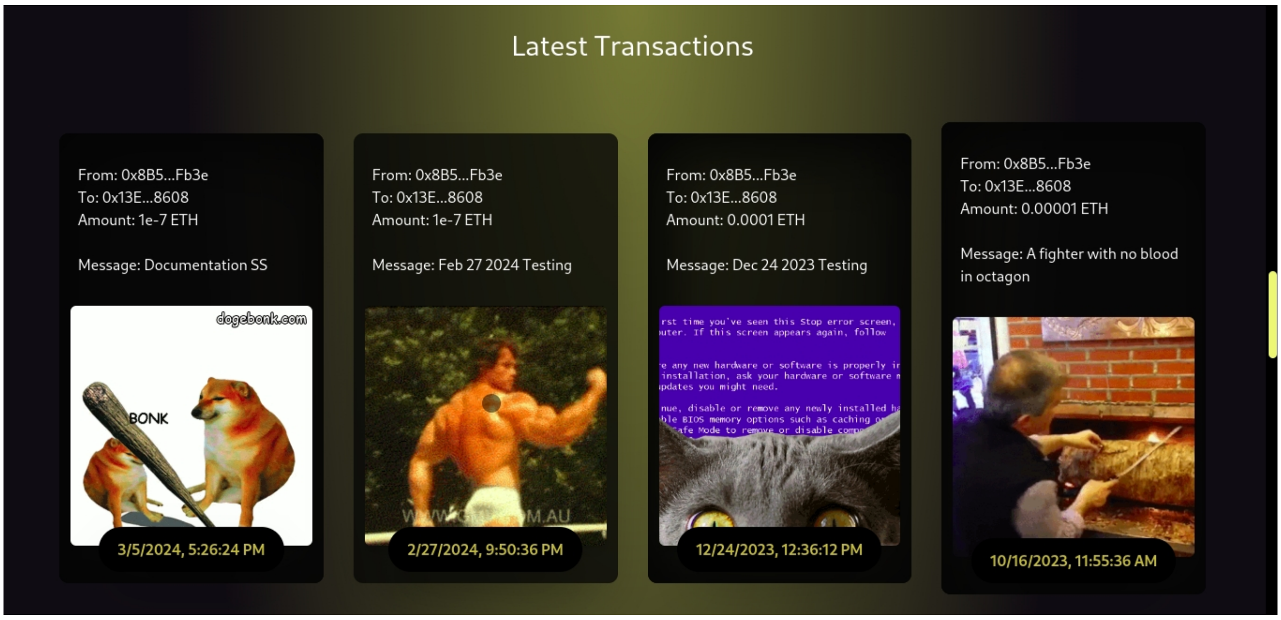


Connecting you to the Decentralize future.

Experience a secure, user-friendly platform for seamless cryptocurrency exchange. Unlock the potential of digital assets and shape your financial future with confidence.

0x13E6fd16C28c6B0DB0314F57Ec15CCdCe27e8608
0.0000001
Shibu
Documentation SS

5. Transaction History



Latest Transactions

From: 0x8B5...Fb3e
To: 0x13E...8608
Amount: 1e-7 ETH

Message: Documentation SS

3/5/2024, 5:26:24 PM

From: 0x8B5...Fb3e
To: 0x13E...8608
Amount: 1e-7 ETH

Message: Feb 27 2024 Testing

2/27/2024, 9:50:36 PM

From: 0x8B5...Fb3e
To: 0x13E...8608
Amount: 0.0001 ETH

Message: Dec 24 2023 Testing

12/24/2023, 12:36:12 PM

From: 0x8B5...Fb3e
To: 0x13E...8608
Amount: 0.00001 ETH

Message: A fighter with no blood in octagon

10/16/2023, 11:55:36 AM

Chapter 6

Conclusion

The "Decentralized Transaction Application" project represents a significant leap forward in the world of blockchain technology and decentralized finance. With a clear focus on empowering individuals, enhancing security, promoting financial inclusion, and exploring emerging technologies, this project aims to reshape traditional financial systems and set an example for industry transformation. By eliminating traditional intermediaries, implementing robust security measures, and providing a user-friendly platform for secure transactions, it addresses the core challenges and limitations associated with centralized financial systems.

The project's versatile applicability extends to a broad audience, from individuals seeking secure and efficient financial transactions to underserved communities in need of accessible financial services. Moreover, it serves as an educational resource for blockchain enthusiasts and developers, contributing to the advancement of blockchain knowledge and capabilities.

By pioneering the adoption of decentralized transactions and applications, the "Decentralized Transaction Application" project serves as a catalyst for change within the financial sector. It leads the way in making financial systems more transparent, efficient, and user-centric, inspiring further innovation and adoption in the blockchain and decentralized finance space. This project represents a promising step towards a more equitable and inclusive financial landscape and has the potential to significantly impact the way financial transactions are conducted in the future.

Chapter 7

Reference

- https://research.csiro.au/blockchainpatterns/general-patterns/deployment-patterns/dapp/
- https://www.alchemy.com/create-web3-dapp
- https://hardhat.org/docs
- https://ethereum.org/pdfs/EthereumWhitePaper.pdf
- https://ethereum.org/en/developers/docs/
- https://web3js.readthedocs.io/en/v1.10.0/
- https://etherscan.io/
- https://consensys.github.io/smart-contract-best-practices/