SAVITRIBAI PHULE PUNE UNIVERSITY

The Mini Project Based On

"Create a dApp (de-centralized app) for e-voting system"

Submitted By:

Siddhi Sachin Diwadkar

Seat No: A-29

Under Guidance of:

Prof. A.P Bhalke

In partial fulfillment of

Laboratory Practice-III (310258)

DEPARTMENT OF COMPUTER ENGINEERING)

SAVITRIBAI PHULE PUNE UNIVERSITY 2024-25

CERTIFICATE

This is to certify that the Mini Project based on,

"Create a dApp (de-centralized app) for e-voting system"

has been successfully completed by,

Name: Siddhi Sachin Diwadkar

Exam seat number: A-29

Towards the partial fulfilment of the Final Year of Computer Engineering as awarded by the Savitribai Phule Pune University, at PDEA's College of Engineering, Manjari Bk," Hadapsar, Pune 412307, during the academic year 2024-25.

Prof. A.P Bhalke	**Dr. M. P. Borawake**
Guide Name	**H.O.D**

Acknowledgement

My first and for most acknowledgment is to my guide Prof. A.P Bhalke During the long journey of this study, she supported me in every aspect. She wasthe one who helped and motivated me to proposer search in this field and inspired me with her enthusiasm on research, her experience, and her lively character.

I express true sense of gratitude to my guide Prof. A.P Bhalke for her perfect valuable guidance, all the time support and encouragement that he gave me.

I would also like to thanks our head of department Dr. M. P. Borawake and Principal Dr. R. V. Patil and management inspiring me and providing all lab and other facilities, which made this mini project very convenient.

I thankful to all those who rendered their valuable help for successful completion on Internship presentation.

Name: Siddhi Sachin Diwadkar

Index

Abstract

The proposed decentralized application (dApp) for an e-voting system aims to revolutionize the electoral process by enhancing transparency, security, and accessibility. Built on blockchain technology, this dApp ensures that every vote is recorded immutably and can be verified by all participants, eliminating concerns about tampering and fraud. Utilizing smart contracts, the system automates the voting process, streamlining voter registration, ballot casting, and result tallying while maintaining anonymity and confidentiality. This innovative solution addresses the challenges of traditional voting methods, making it easier for citizens to engage in the democratic process from anywhere in the world.

In addition to its technical advantages, the e-voting dApp fosters trust among users by providing real-time tracking of votes and results through a user-friendly interface. Voters can verify their participation and ensure their choices are accurately counted without compromising their privacy. By incorporating robust security measures such as cryptographic signatures and decentralized identity verification, the platform protects against unauthorized access and potential cyber threats. Ultimately, this e-voting system not only empowers voters but also promotes higher turnout rates and greater civic engagement, paving the way for a more inclusive and representative democracy

SOFTWARE REQUIREMENT

- ❖ Windows.
- ❖ Blockchain Platform: Ethereum or Hyperledger
- ❖ Frontend Framework: React, Angular, or Vue.js
- ❖ Smart Contracts: Custom voting logic contracts
- ❖ Identity Verification: Decentralized identity solutions
- ❖ Testing Tools: Truffle or Mocha for smart contract testing
- ❖ System: Windows 11.

INTRODUCTION

The advent of digital technology has transformed numerous aspects of our daily lives, and the electoral process is no exception. Traditional voting methods often face challenges such as inefficiency, lack of transparency, and susceptibility to fraud. As societies increasingly embrace digital solutions, the need for a secure and reliable e-voting system becomes paramount. This decentralized application (dApp) leverages blockchain technology to address these issues, providing a modern approach to voting that enhances the integrity and accessibility of elections.

At the core of this e-voting dApp is blockchain technology, which offers a secure and immutable ledger for recording votes. By utilizing smart contracts, the system automates the entire voting process, from voter registration to ballot casting and result tallying. This automation not only reduces the administrative burden associated with traditional voting systems but also minimizes the potential for human error. The decentralized nature of blockchain ensures that no single entity can manipulate the results, fostering trust among voters.

The user experience is a critical component of the dApp's design. The interface is designed to be intuitive and user-friendly, allowing voters to easily navigate the voting process. Additionally, the integration of decentralized identity solutions ensures secure authentication, protecting voter privacy while maintaining accountability. By enabling remote access, the system increases voter participation, particularly for those who may face barriers in traditional voting settings.

In summary, the proposed e-voting dApp represents a significant advancement in electoral technology. By combining blockchain's security and transparency with an emphasis on user accessibility, this system aims to enhance public trust in the electoral process. As we move towards a more digital future, innovative solutions like this are essential for ensuring that democracy remains robust and inclusive.

PROBLEM STATEMENT

Create a dApp (de-centralized app) for e-voting system

Objective: To understand the concept of Mini-project.

Outcome: Implement and Create a dApp (de-centralized app) for e-voting system.

IMPLEMENTATION CODE

Solidity Smart Contract:

```
// SPDX-License-Identifier: MIT pragma solidity 0.8.9;

contract Election { struct Candidate { uint256 id;

string name; uint256 voteCount;

}

address public owner; mapping(uint256 => Candidate) public

candidates; mapping(address => bool) public voters;

uint256 public candidataCount; constructor()

{

owner = msg.sender; addCandidate("Ryle"); addCandidate("Max");

}

function addCandidate(string memory _name) public{ require(msg.sender == owner, "Only owner can add candidates"); candidataCount++;

candidates[candidataCount] = Candidate(candidataCount, _name, 0);

}

function vote(uint256 _candidateId) public { require(! voters[msg.sender], "You have already voted"); require(

_candidateId <= candidataCount && _candidateId >= 1, "Invalid candidate Id"

);

voters[msg.sender] = true; candidates[_candidateId].voteCount++;

}

} 10
```

Index.js File :

```
import Head from "next/head"; import Image from "next/image";

import styles from "../styles/Home.module.css"; import React, { useState, useEffect } from "react"; import { ethers } from "ethers";

import { contractABI, contractAddress } from "./Engine"; import Navbar from "./Navbar";

export default function Home() {

const [candidatesUseState, setCandidatesUseState] = useState([]); const [voters, setVoters] = useState([]);

const [account, setCurrentAccount] = useState();
```

```
const [walletAddress, setWalletAddress] = useState(""); const [votedOrNot, setVotedOrNot] = useState();
const [candidateId, setCandidateId] = useState(); const checkIfWalletIsConnected = async () =>
{ try {
const { ethereum } = window; if (!ethereum) {
console.log("Please install metamask!!!");
} else {
// console.log("We have the ethereum object", ethereum);
}
const accounts = await ethereum.request({ method: "eth_accounts" }); if (accounts.length) {
const account = accounts[0];
// console.log("Authorized account has found", account); setCurrentAccount(account); console.log("Connected"); setCurrentAccount("");
console.log("No authorized account has found!");
}
} catch (error) { console.error(error); 11


}
};
const connectWallet = async () =>
{ try {
const { ethereum } = window; if (!ethereum) { alert("Metamask has found!"); return;
}
const accounts = await ethereum.request({ method: "eth_requestAccounts",
});
// console.log("Connected", accounts[0]);
// setCurrentAccount(accounts[0]);
} catch (err) { console.error(err.message);
}
};
const getCandidates = async (candidateId) => {
const provider = new ethers.providers.Web3Provider(window.ethereum); const connection = new
ethers.Contract( contractAddress, contractABI,
```

```
provider
const candidatesCount = Number(await connection.candidataCount()); console.log(candidatesCount);
for (var i = 1; i <= candidatesCount; i++) {
const candidate = await connection.candidates(i); const id = candidate[0];
const name = candidate[1]; const voteCount = candidate[2]; const item = {
id: Number(id),
name: name.toString(),
voteCount: voteCount.toNumber(),
}; 12


setCandidatesUseState((prev) => [...prev, item]);
}
// console.log(candidatesUseState);
};
const checkVotingStatus = async (voter) =>
{ try {
const provider = new ethers.providers.Web3Provider(window.ethereum); const connection = new
ethers.Contract( contractAddress, contractABI,
provider
);
const hasVoted = await connection.voters(voter); console.log(voter, "hasVoted: ", hasVoted);
setVotedOrNot(hasVoted);
} catch (error) {
}
};
const changeHandler = (e) => { setWalletAddress(e.target.value);
};
const handleButtonClick = async () => { await checkVotingStatus(walletAddress);
};
const vote = async (candidateId) =>
{ try {
const provider = new ethers.providers.Web3Provider(window.ethereum); const signer = await
provider.getSigner();
```

```
const connection = new ethers.Contract( contractAddress, contractABI,
signer
);
const vote = await connection.vote(candidateId);
} catch (error) { 13

console.error(error);
}
};
const handleChange2 = (e) =>
{ setCandidateId(e.target.value
);
};
const buttonClick2 = () =>
{ vote(candidateId);
};
useEffect(() => { checkIfWalletIsConnected(); connectWallet(); getCandidates();
}, []);
return (
<div className="">
<Head>
<title>Voting Dapp ~ heysourin</title>
<meta name="description" content="Generated by create next app" />
<link rel="icon" href="/icon.png" />
</Head>
<main className="">
<Navbar account={account} connectWallet={connectWallet} />
<h1 className="font-bold text-3xl m-5">Candidates:</h1>
<div className="flex flex-row">
<table className="w-full border-collapse mx-10">
<thead>
<tr>
```

```
<th className="py-2 px-4 border">Candidate Id</th>
<th className="py-2 px-4 border">Candidate Name</th>
<th className="py-2 px-4 border">Vote Count</th>
</tr>
</thead>
<tbody>
{candidatesUseState.map((candidate, i) => (
<tr key={i}>
<td className="py-2 px-4 border">{candidate.id}</td> 14

<td className="py-2 px-4 border">{candidate.name}</td>
<td className="py-2 px-4 border">{candidate.voteCount}</td>
</tr>
))}
</tbody>
</table>
</div>
{/* @note CHECK IF VOTED*/}
<div>
<h2 className="font-bold text-3xl mt-10 ml-5">Check Voted or Not:</h2>
<div className="flex flex-row mx-10">
<input type="text"
className="border border-gray-300 px-4 m- 4" placeholder="Enter wallet address" value={walletAddress || ""} onChange={changeHandler}
/>
<button onClick={handleButtonClick}
className="bg-gradient-to-r from-orange-600 to-yellow-500 text-white font-bold py- 2 px-4 rounded m-4"
>
Check if voted
</button>
<div className="mt-6">
{votedOrNot ? (
```

You have already voted, can not vote anymore!

) : (

You have not voted yet!

)}

<h2 className="font-bold text-3xl mt-8 ml-5">Vote: </h2>

<input 15

type="text"

className=" w-24 mx-3 my-4 border border-gray-300 rounded-md focus:outlinenone focus:ring-2 focus:ring-blue-500"

pattern="\d{0,2}" maxLength="2" placeholder="Candidate Id" value={candidateId || ""} onChange={handleChange2} required

/>

<button onClick={buttonClick2}

className="bg-gradient-to-r from-purple-700 to-blue-500 text-white font-bold py-2 px-4

rounded m-4"

>

Vote

</button>

</main>

<footer className="flex items-center justify-between bg-gradient-to-r from-gray-900 to-gray-

700 p-6 text-white">

<a href="https://github.com/heysourin" target="_blank"

className="text-left"

>

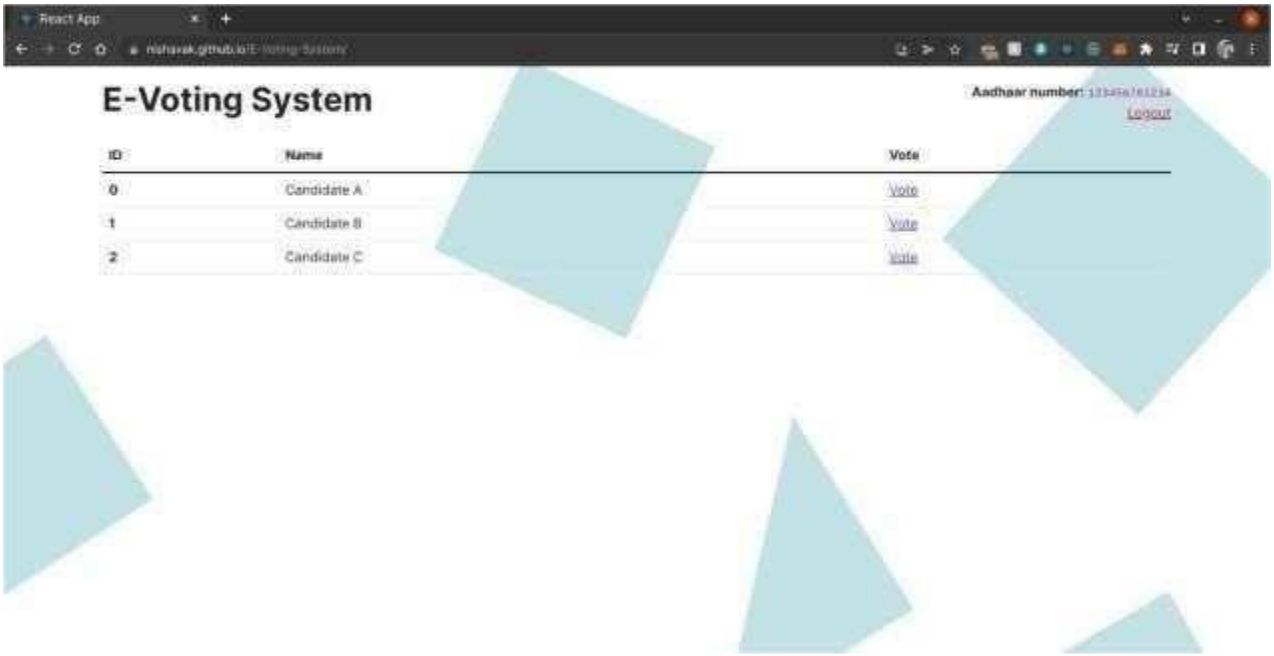
```
My Github
</a>
<a href="https://linkedin.com/in/heysourin" target="_blank"
className="text-left"
> 16


My LinkenIn
</a>
<span className="text-right">
<a
href="https://github.com/heysourin/Voting-dApp-on-Avalanche-Network" className="text-white"
>
Source Code
</a>
</span>
</footer>
</div>
);
}
```

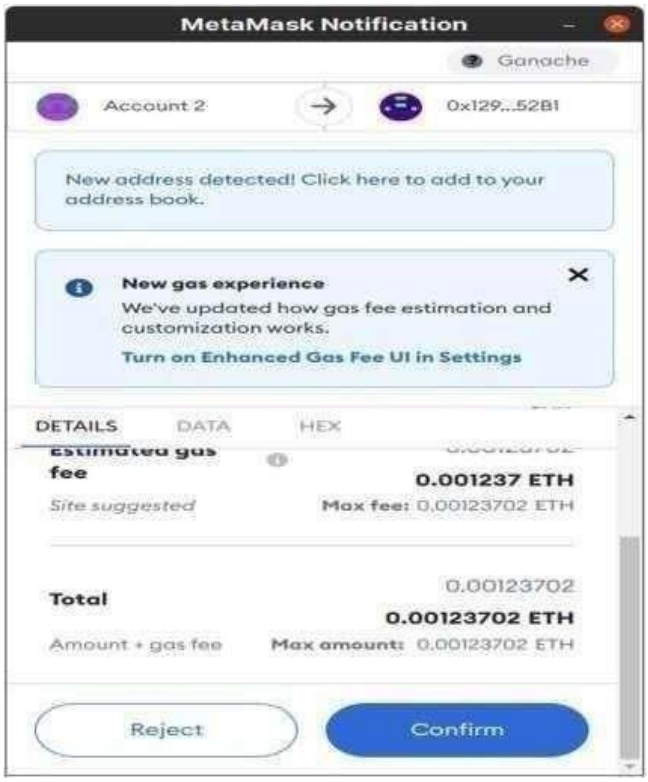
Navbar.jsx:

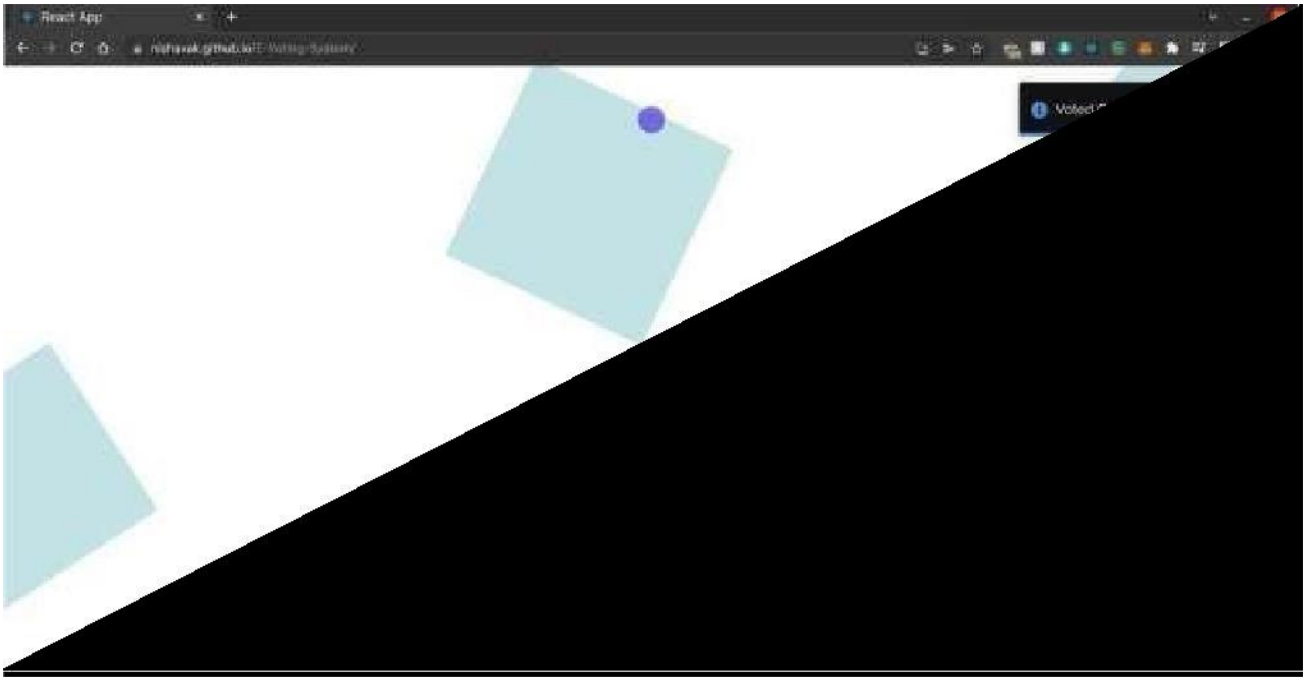
```
import { React } from 'react'
const Navbar = ({ account, connectWallet }) =>
{ return (
<nav className="flex items-center justify-between flex-wrap bg-gradient-to-r from- gray-
900 to-gray-700 p-6">
<div className="flex items-center flex-shrink-0 text-white mr-6">
<span className="font-bold text-3xl tracking-tight"> Web3 Voting DAPP
</span>
</div>
<div className="flex">
{account ? (
<button type="button"
```

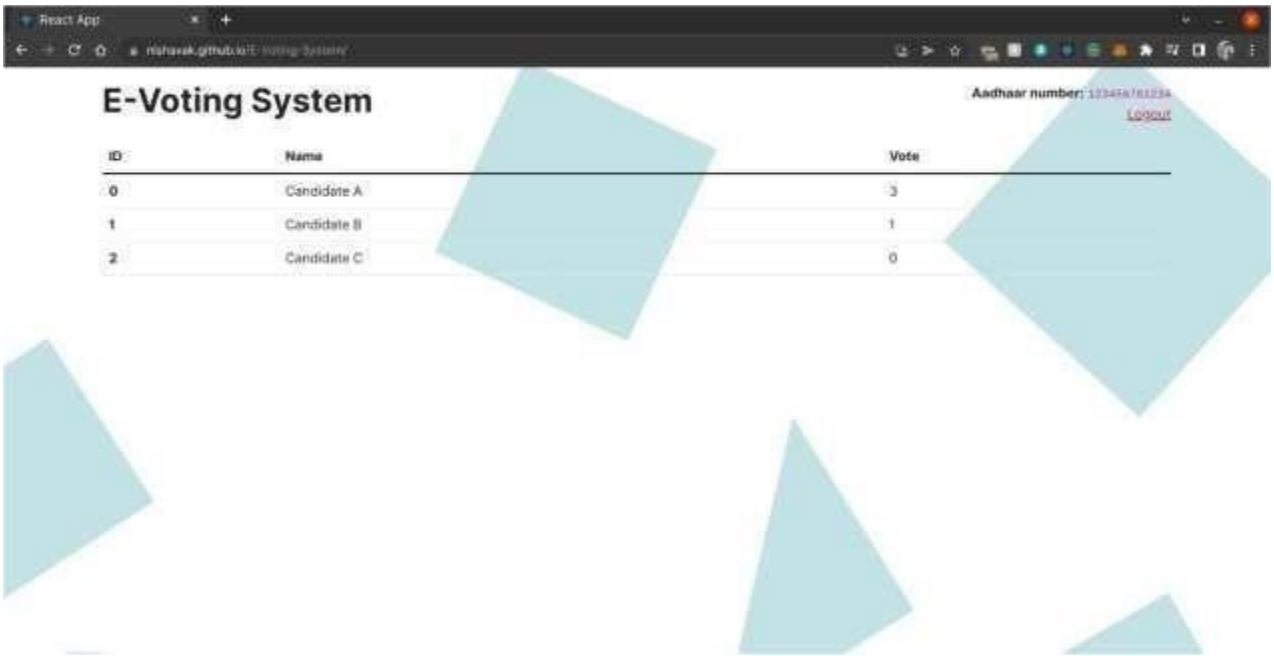
```
className='className="bg-gradient-to-r from-blue-500 to-blue-700 text-white py-2 px-4
rounded-m font-bold'
>
{account.slice(0, 6) + '...' + account.slice(38, 42)} 17

</button>
)
<button type="button"
className='className="bg-blue-500 bg-blue-700 text-white font-bold py-2 px-4 rounded"'
onClick={connectWallet}
>
Connect Wallet
</button>
)}
</div>
</nav>
)
}
export default Navbar
```

Output


E-Voting System
Aadhaar number:
Logout
ID
Name
Vote
0
Candidate A
Vote
1
Candidate B
Vote
2
Candidate C
Vote


MetaMask Notification
Ganache
Account 2
0x129...52B1
New address detected! Click here to add to your address book.
New gas experience
We've updated how gas fee estimation and customization works.
Turn on Enhanced Gas Fee UI in Settings
DETAILS
DATA
HEX
fee
0.001237 ETH
Site suggested
Max fee: 0.00123702 ETH
Total
0.00123702
0.00123702 ETH
Amount + gas fee
Max amount: 0.00123702 ETH
Reject
Confirm


React App


React App
E-Voting System
Aadhaar number:
Logout
ID
Name
Vote
0
Candidate A
3
1
Candidate B
1
2
Candidate C
0

CONCLUSION

The development of a decentralized e-voting dApp represents a significant advancement in modernizing the electoral process. By utilizing blockchain technology, this system addresses critical challenges associated with traditional voting methods, such as security vulnerabilities, lack of transparency, and administrative inefficiencies. The immutability of blockchain ensures secure vote recording, reducing the risk of fraud and fostering voter trust.

Smart contracts further streamline the voting process by automating functions like registration, ballot casting, and result tallying, enhancing accuracy and speed while minimizing human error and administrative overhead. An intuitive interface and secure authentication mechanisms encourage higher voter participation by making the voting process more accessible.

Ultimately, the e-voting dApp exemplifies a commitment to upholding democratic values in the digital age. By integrating security, efficiency, and accessibility, it paves the way for a more inclusive and trustworthy electoral landscape, reinforcing the foundations of democracy for future generations.

Reference

- https://www.github.com
- https://chat.openai.com/
- https://www.python.org/
- https://www/w3schools.com