DECENTRALIZED VOTING APP

A PROJECT REPORT

Submitted by

BENHER BASHEER

VDA23MCA-2015

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Master of Computer Applications

Department of Computer Applications

College of Engineering Vadakara

April 2025

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING VADAKARA

CERTIFICATE

This is to certify that the report entitled **DECENTRALIZED VOTING APP** submitted by **BENHER BASHEER (VDA23MCA-2015)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Master of Computer Applications is a bonafide record of the project work carried out by him under my guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor:	Head of the Department:
Name: Ms. Rajalekshmy K D	Name: Mrs. Fathima A
Signature:	Signature:

External Examiner:

Name:

Signature:

DECLARATION

I undersigned hereby declare that the project report **DECENTRALIZED VOTING APP**, submitted for partial fulfillment of the requirements for the award of Degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by me under the supervision of Ms. Rajalekshmy K D, Assistant Professor, Department of Computer Applications. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any Degree, Diploma or similar title of any other University.

Place: VADAKARA BENHER BASHEER

Date: 02/04/2025 VDA23MCA-2015

ACKNOWLEDGEMENT

My endeavor stands incomplete without dedicating my gratitude to a few people who have contributed towards the successful completion of my project. I pay my gratitude to the Almighty for his invisible help and blessing for the fulfilment of this work. At the outset I express my heartful thanks to Dr. Vinod Pottakulath, Principal College of Engineering Vadakara for giving me an opportunity to complete my project successfully. I extremely grateful to Mrs. Fathima A, Assistant Professor and HOD, Department of Computer Applications for permitting me to do this project. I take this opportunity to express my profound gratitude to Ms. Rajalekshmy K D, Assistant Professor, Department of Computer Applications, my guide, for her valuable support and help in doing my project. I am also grateful to all our teaching and non-teaching staff for their encouragement, guidance and wholehearted support. Last but not least, I am gratefully indebted to my family and friends, who gave me a precious help in doing my project.

BENHER BASHEER

VDA23MCA-2015

ABSTRACT

This project focuses on developing a Decentralized Voting Application to address the challenges faced by organizations, institutions, and communities in conducting secure, transparent, and tamper-proof elections. Traditional voting systems often suffer from issues such as lack of transparency, susceptibility to manipulation, limited accessibility, and centralized control, which can undermine trust in the electoral process.

To overcome these limitations, the proposed system leverages blockchain technology and smart contracts to ensure a transparent, secure, and decentralized voting mechanism. The application facilitates voter and candidate registration, vote casting, real-time result tracking, and tamper-proof audit trails. By eliminating intermediaries and automating core voting functionalities, the system guarantees data integrity, improves voter confidence, and enhances overall efficiency.

This project aims to modernize the election process through a blockchain-powered solution that ensures fairness, boosts participation, and empowers communities with a reliable and transparent voting platform.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 BACKGROUND

Voting is a fundamental aspect of decision-making in democratic systems, organizations, and communities. However, traditional voting methods continue to face significant challenges related to security, transparency, accessibility, and trust. Paper-based and centralized digital voting systems are often vulnerable to fraud, data manipulation, and unauthorized access. Moreover, centralized control over voter data and election processes raises concerns about the integrity and fairness of election outcomes.

Voters, in many cases, face difficulties in verifying the legitimacy of results, ensuring their vote is counted accurately, or even participating due to logistical barriers. Additionally, election organizers struggle to manage voter registration, maintain data security, and conduct elections efficiently without a unified and tamper-proof system in place. These limitations often result in reduced voter confidence, low turnout, and disputes over results.

To address these issues, this project introduces a **Decentralized Voting Application** built using **blockchain technology, smart contracts, and IPFS**. The system decentralizes control, encrypts voting data, and automates core functions such as registration, voting, and result computation. By offering a transparent and secure platform accessible to all stakeholders, the application ensures trust, enhances participation, and provides a tamper-resistant and verifiable voting experience for organizations of all sizes.

1.2 OBJECTIVE

The primary objective of this project is to design and develop a Decentralized Voting Application that ensures secure, transparent, and tamper-proof elections using blockchain technology. The system aims to automate critical voting processes such as voter registration, candidate management, vote casting, and result computation, eliminating the need for centralized control and manual oversight. By decentralizing the voting infrastructure, the application reduces the risks of fraud, data manipulation, and unauthorized access, ultimately enhancing trust in the electoral process.

Another core objective is to improve accessibility, accountability, and user engagement throughout the voting lifecycle. The system provides distinct modules for administrators, candidates, and voters, enabling each group to participate efficiently and securely. Voters can cast their votes from anywhere, verify the integrity of their vote, and access real-time updates, while administrators can monitor election progress. Additionally, the application leverages smart contracts to enforce fair voting rules and uses IPFS for secure document and media storage. Overall, the system is designed to deliver a user-friendly, decentralized solution that supports reliable, verifiable, and democratic voting experiences.

1.3 PROJECT DESCRIPTION

The **Decentralized Voting Application** is a blockchain-based platform designed to streamline and secure the voting process for organizations, institutions, and communities. This system addresses common challenges in traditional voting methods, such as data manipulation, voter fraud, lack of transparency, and limited accessibility. By leveraging blockchain and smart contracts, the application ensures a tamper-proof, transparent, and decentralized election process.

The platform eliminates the need for centralized authorities and manual vote counting by providing an automated, trustless environment that maintains election integrity from start to finish. It empowers voters, candidates, and administrators with clear roles and permissions, making the entire voting process verifiable, efficient, and user-friendly

Key features of the platform includes:

- **User-Friendly Interface:** A responsive frontend built using Next.js, offering a seamless and intuitive experience for voters, candidates, and administrator.
- **Role-Based Access Control**: Distinct modules for Admin and User, each with tailored permissions and dashboard views.
- **Blockchain-Powered Voting System:** All votes are recorded on the Ethereum blockchain via smart contracts, ensuring immutability.
- **Voter and Candidate Registration:** Users can register securely. Admins approve/reject applications, ensuring only verified participants take part in elections.
- **Voting Period Management:** Admins can define specific start and end times for voting periods, after which results can be automatically calculated.
- **Winner Declaration:** Once voting concludes, the system automatically identifies and displays the candidate with the highest votes, enhancing trust in the result.

- **IPFS Integration:** Candidate images, PDFs, and other important files are stored securely and immutably using IPFS (via Pinata), preventing unauthorized modifications.
- **Reset Functionality:** Admin can reset the system to clear all election data, enabling fresh elections to be conducted as needed.

Built with **Solidity** for smart contracts and **Next.js** for the frontend, the Decentralized Voting App provides a modern, secure, and scalable solution for conducting elections in any environment—be it educational institutions, decentralized organizations (DAOs), or corporate boards. The system reduces fraud, enhances accessibility, and restores trust in the voting process through technology-driven transparency and automation.

2. GENERAL BACKGROUND

2.1 EXISTING SYSTEM

In many organizations, communities, and institutions, elections are still conducted using manual methods or basic digital tools such as paper ballots, spreadsheets, emails, and messaging apps. Voter registration, vote collection, and result tallying are often managed separately and manually, leading to disorganized data, increased administrative burden, and potential for human errors. Communication regarding election timelines and results is usually handled through informal channels like group chats or announcements, which can result in missed information and a lack of proper transparency.

These fragmented systems make it difficult to ensure security, accuracy, and trustworthiness in the election process. Voters have no way to verify whether their votes were recorded correctly or whether the results were manipulated. Administrators, meanwhile, face challenges in managing voter eligibility, maintaining records, and ensuring a smooth and fair election experience. Disputes, delays in result announcements, and limited accessibility further affect the credibility and inclusiveness of the process.

Moreover, traditional systems offer little to no transparency or real-time feedback, making it hard to build confidence among voters. The absence of an integrated, tamper-proof platform leads to inconsistencies, inefficiencies, and a lack of accountability in managing elections. As a result, the current voting systems fall short in meeting the evolving demands of secure, scalable, and verifiable democratic processes in today's digital age.

2.2 PROPOSED SYSTEM

To overcome the limitations of traditional and centralized voting methods, this project proposes a Decentralized Voting Application built using blockchain technology, smart contracts, and IPFS (Inter Planetary File System). The system is designed to deliver a secure, transparent, and tamper-proof platform for conducting elections, ensuring integrity, accessibility, and trust for all participants involved.

The proposed system decentralizes the entire voting process—from voter registration and candidate approval to vote casting and result declaration—by leveraging Ethereum smart contract**s**. These contracts automate core functionalities, eliminating the need for manual intervention and preventing unauthorized alterations or fraudulent activities. Once deployed, smart contracts enforce election rules, validate votes, and store results immutably on the blockchain, making them publicly verifiable and resistant to tampering.

A key component of the system is its role-based architecture, offering dedicated interfaces and functionalities for three primary roles: Admin, Voter, and Candidate. Admins can manage the election, approve or reject user requests, define the voting period, and view results. Voters can register, view candidate profiles, and securely cast their votes. Candidates can submit their details—including images and manifesto PDFs—which are stored securely on **IPFS**, ensuring authenticity and accessibility.

Key features will include:

- **Security & Tamper Resistance:** All transactions are recorded on the blockchain, making it impossible to alter or delete any vote once cast.
- **Transparency:** All voters can view the process and outcome on the public ledger, enhancing trust in the election results.
- **Automation:** Smart contracts handle critical functions like vote validation and winner determination, reducing human errors.
- **Real-Time Results**: Once the voting period ends, the system automatically calculates and displays the winning candidate.
- **Accessibility:** Voters can participate from any location.
- **Reset Functionality:** Admin can reset the system to clear old data and prepare for new elections, supporting repeated use.
- **Transfer Ownership:** Admin can securely hand over system control to another address directly through the dashboard.

By combining blockchain with web technologies like **Next.js** for the frontend and using **IPFS for file storage**, the proposed system introduces a modern, user-friendly, and reliable solution for conducting elections in academic institutions, decentralized communities, and private organizations. It ensures trust, efficiency, and fairness**,** addressing the core challenges of existing systems while embracing the power of decentralized technology.

2.3 MODULE DESCRIPTION

The platform consists of three primary user modules: the Admin, Voter, and Candidate modules.

2.3.1. ADMIN MODULE

- **Manage Voter Requests:** Admin can view, approve, or reject voter registration requests to ensure only eligible users participate in the election.
- **Manage Candidate Requests:** Admin can review and approve or reject candidate registration requests, including checking submitted profiles, PDFs, and images.
- **Set Voting Period:** Admin can define the start and end times of the voting period using the dashboard. Once set, voters can only vote within the specified timeframe.
- **Reset Election:** Admin can reset the entire system to remove previous data, preparing the application for a new election cycle.
- **Transfer Ownership:** Admin can securely transfer control of the smart contract to another Ethereum address via the dashboard, ensuring flexible and secure management handovers.
- **Check Voting Status:** Admin can monitor whether voting is active or ended and take actions accordingly.
- **View Winner:** After the voting period ends, the admin can access the winning candidate's details through the dashboard

2.3.2. VOTER MODULE

- **Register as Voter:** Users can submit a voter registration request with necessary details to participate in the election.
- **View Approved Candidates:** Once registered and approved, voters can browse the list of candidates, view their profiles, PDFs, and details.
- **Cast Vote:** During the active voting period, voters can securely cast a single vote. The system automatically prevents duplicate voting using blockchain validation.
- **Verify Vote Status:** Voters can confirm whether their vote has been successfully recorded by checking the transaction hash or vote confirmation.
- **View Winner:** After the election ends, voters can view the winning candidate and relevant vote count information directly through the interface.

2.3.3. CANDIDATE MODULE

- **Register as Candidate:** Users can submit a candidate registration request with full name, image, PDF, and other required details.
- **Profile Visibility:** Approved candidate details, images, and documents (stored via IPFS) become visible to all registered voters.
- **Monitor Election Status:** Candidates can view whether the election is ongoing or concluded and check results after voting ends.
- **View Winner:** After the election ends, candidates can view the winning candidate and relevant vote count information directly through the interface.

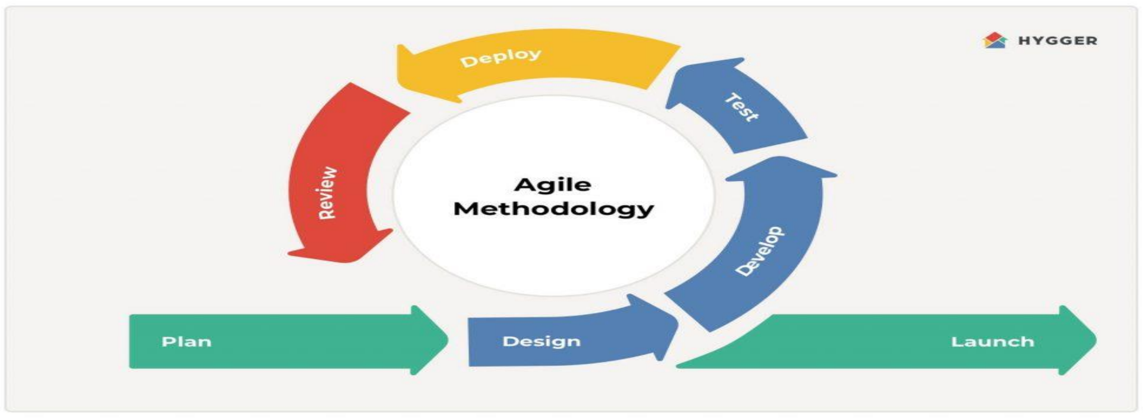
3. METHODOLOGY

3.1 AGILE SOFTWARE DEVELOPMENT

Agile software development was chosen as the most suitable approach for this project due to its focus on client satisfaction through the continuous delivery of functional components. Agile methodology emphasizes delivering small, incremental requirements rather than attempting to complete the entire product at once. This is particularly advantageous for a project with various individual requirements, allowing for partial deliveries and iterative improvements.

One of the primary methods within Agile is the Scrum framework, which provides a structured yet flexible approach to managing complex product development. Rather than dictating specific processes or techniques, Scrum serves as a framework that enables the development team to utilize diverse approaches within each cycle. This framework supports lean software development and emphasizes efficient and effective software building. Agile is known for its adaptability, continuous improvement, rapid delivery, and emphasis on constant feedback, making it well-suited for dynamic project requirements.

In Agile development, testing and integration are ongoing processes rather than sequential stages. This allows users to make adjustments at any point during the development phase, enabling the user to view and provide feedback on their product throughout its production. This adaptability is especially valuable in projects with evolving requirements, as it ensures flexibility and the ability to meet client expectations



3.2 SCRUM FRAMEWORK

The Scrum framework is implemented in this project to facilitate team-based development and streamline the Agile process. Scrum focuses on three main roles, which together form the backbone of the development process:

1. **Product Owner (PO)**: Represents stakeholders and the business. The PO prioritizes the backlog to ensure the most valuable features are developed first.
2. **Scrum Master**: Facilitates the Scrum process, removing any impediments that may slow down the team and ensuring that the team follows Agile principles.
3. **Development Team**: A cross-functional, self-organizing team responsible for analysis, design, development, and testing. This team organizes itself to complete each sprint.

In Scrum, projects are divided into sprints, each representing a short, focused development cycle where specific goals are achieved. Sprints encourage rapid progress, as each sprint cycle requires the team to plan, set goals, and complete work within a defined timeframe. By focusing on one sprint at a time, the team can maintain high productivity and continually deliver increments of the project.

3.3 USER STORY

A user story is a core component of Agile development, focusing on the end-user experience. User stories prioritize people by centering development around user needs and real-world applications. Written in non-technical language, user stories help clarify the purpose of a particular feature and the value it will create for users. For the development team, this provides essential context and a shared understanding of the project's purpose.

In Agile, a user story describes a feature from the end-user's perspective, answering key questions: Who will use it, what do they need, and why? User stories drive collaboration, foster creativity, and ensure that each requirement supports the user's needs and the project's goals.

A Decentralized Voting Application (DApp) serves multiple stakeholders with distinct needs. The following user stories outline how different types of users will interact with the system. The user story for the Decentralized Voting App is given below.

1.Admin User Stories

As an Admin, I want to set up elections, approve candidates and voters, and monitor the voting process transparently, so that I can ensure fair and accountable election outcomes.
Goals:

- Owner can deploy and reset the election smart contract.
- Owner can approve/reject voter and candidate requests.
- Owner can view voting results after the voting period ends.
- Owner cannot alter or manipulate any vote once cast.

2.Voter User Stories

As a voter, I want to register myself on the Decentralized voting app and cast my vote securely, so that I can be assured that my vote is counted correctly and not manipulated by any central authority.
Goals:

- Register using their MetaMask wallet.

- Get verified before voting.
- View candidate profiles and documents.
- Cast only one secure vote.
- Voter can view voting results after the voting period ends.

3.Candidate User Stories

As a candidate, I want to register and upload my profile with a manifesto and documents, so that voters can understand my qualifications and make informed decisions.
Goals:

- Candidate can upload image, PDF, and name.
- Candidate must be approved before appearing on the ballot.
- Candidate's profile is stored using IPFS and is visible to voters.
- Candidates can view voting results after the voting period ends.

3.4 SPRINT

A sprint is a fundamental concept in Scrum, representing a time-boxed period during which a set amount of work is completed. Each sprint typically lasts 1-4 weeks, allowing the team to focus on specific objectives and tasks within that timeframe. Sprint goals, determined during planning sessions, help the team stay focused on priorities and ensure continuous progress. With each sprint, the team delivers incremental project components, gradually building toward the final product.

In summary, Agile methodology and the Scrum framework enable a responsive, user-focused approach to software development. By organizing the project into sprints and using user stories to guide development, the team can ensure a flexible and efficient process, delivering valuable product increments regularly.

Code	As a/an	I want to perform	So that I can
US1	Admin	Log into the app securely	Access and manage the voting system effectively
US2	Admin	Approve or reject voter requests	Ensure only verified users can participate in voting
US3	Admin	Approve or reject candidate requests	Validate legitimate candidates before elections
US4	Admin	Set voting start and end time	Control and automate the voting schedule
US5	Admin	View registered voters and candidates	Monitor system participation and activity
US6	Admin	Reset election data	Start a fresh election cycle when needed
US7	Admin	Transfer contract ownership	Hand over system control securely when required
US8	Admin	Check election status	Monitor whether voting is active or ended
US9	Admin	View winning candidate after voting ends	See the election winner
US10	Voter	Register as voter	Become eligible to vote in the election
US11	Voter	View approved candidates with image and PDF	Make informed voting decisions
US12	Voter	Cast a vote during the active voting period	Participate in the election process
US13	Voter	View election results	See which candidate won after voting ends
US14	Candidate	Register as a candidate	Apply to participate in the election
US15	Candidate	Upload profile image PDF via IPFS	Present a detailed profile to voters
US16	Candidate	Track approval status	Know when their registration is accepted or rejected
US17	Candidate	View election progress and results	Stay updated on voting activity and outcome
US18	User(all)	Use a decentralized, tamper-proof system	Ensure trust, transparency, and data security

Scrum Agile User Stories

Milestones	Completion Date
User Stories Collection	11-01-2025
Product Backlog	17-012025
High level Sprint Planning	20-01-2025
UI and Database Design	24-01-2025
Sprint 1	05-02-2025
Sprint 2	09-02-2025
Sprint 3	22-02-2025
Sprint 4	06-03-2025
Sprint 5	14-03-2025
Sprint 6	19-03-2025

Major Milestones

Sprint Breakdown:

- Sprint 1: User Registration and Authentication.

Goal:

Enable users to register securely on the decentralized platform. Includes voter and candidate registration workflows with basic validations and Ethereum wallet integration

  - Voter and candidate registration forms.
  - MetaMask wallet connection for identity verification.
  - Secure login using wallet signature.
  - Admin login interface setup.

- Sprint 2: Admin Dashboard and Request Management.

Goal:

Allow the admin to manage and approve/reject voter and candidate requests. Set up the admin dashboard with role-based access and core admin controls.

  - Admin panel UI.
  - View and manage voter/candidate registration requests.
  - Approve/reject requests with smart contract interaction.
  - Role-based navigation.

- Sprint 3: Candidate Profile and file Upload

Goal:

Enable candidates to upload their image and ID using IPFS and display candidate information to voters.

- Candidate profile submission with IPFS integration.
- Upload and store PDFs/images.
- Candidate listing view for voters.
- Candidate approval and visibility after admin verification.

- Sprint 4: Voting System Implementation

Goal:

Develop secure voting functionality that allows approved voters to cast a single vote during an active voting period.

- Voting smart contract integration.
- One-vote-per-user enforcement.
- Active voting period checks.
- Vote confirmation UI with transaction hash.

- Sprint 5: Voting Period Control & Result Calculation

Goal:

Allow admin to set voting start and end times, automatically enable/disable voting, and display winner after voting ends.

- Voting period setter (start/end timestamp).
- Winner calculation function.
- Winner display page for admin and users.

- Sprint 6: Ownership Transfer and System Reset

Goal:

Enable admin to transfer contract ownership and reset the voting system for a new election cycle.

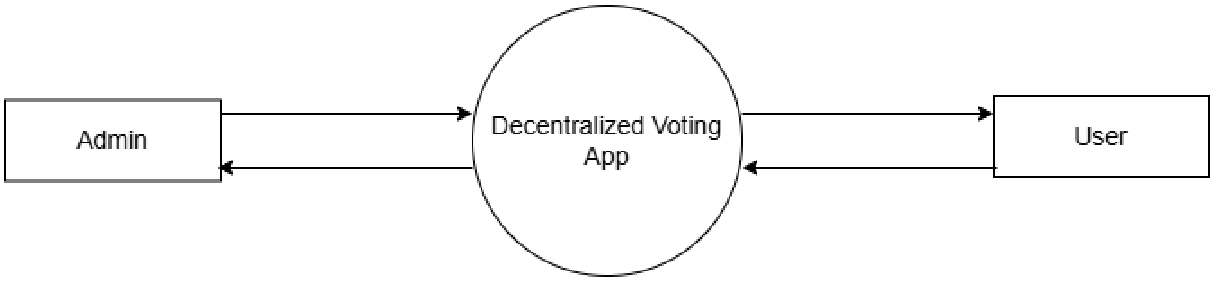
- Ownership transfer function on admin dashboard.
- Reset system functionality (clear candidates, votes, statuses).
- Smart contract functions for reset/ownership transfer.

4 SYSTEM ANALYSIS AND DESIGN

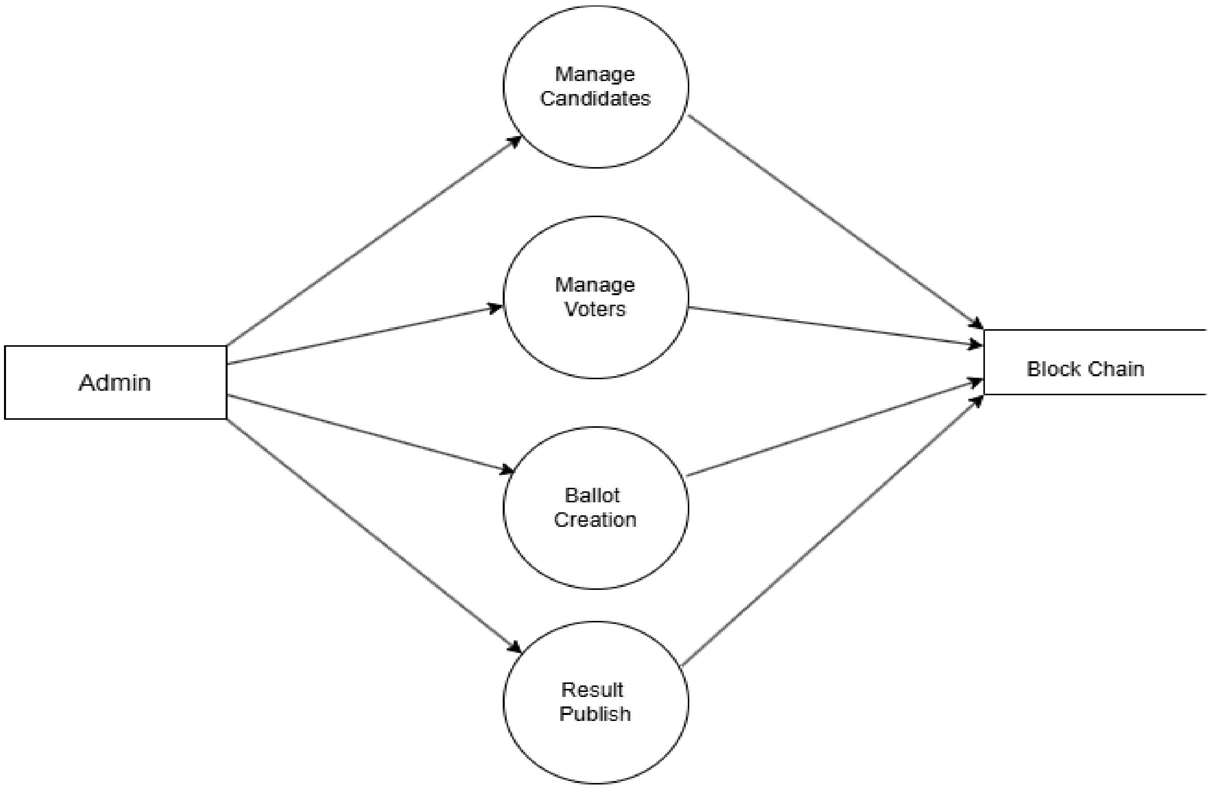
4.1 DATA FLOW DIAGRAM

Data Flow Diagrams are used to graphically represent the flow of data in a decentralized voting system. The DFD describes the processes involved in the system to transfer data from user input through the blockchain, decentralized file storage, and ultimately to result generation and display.

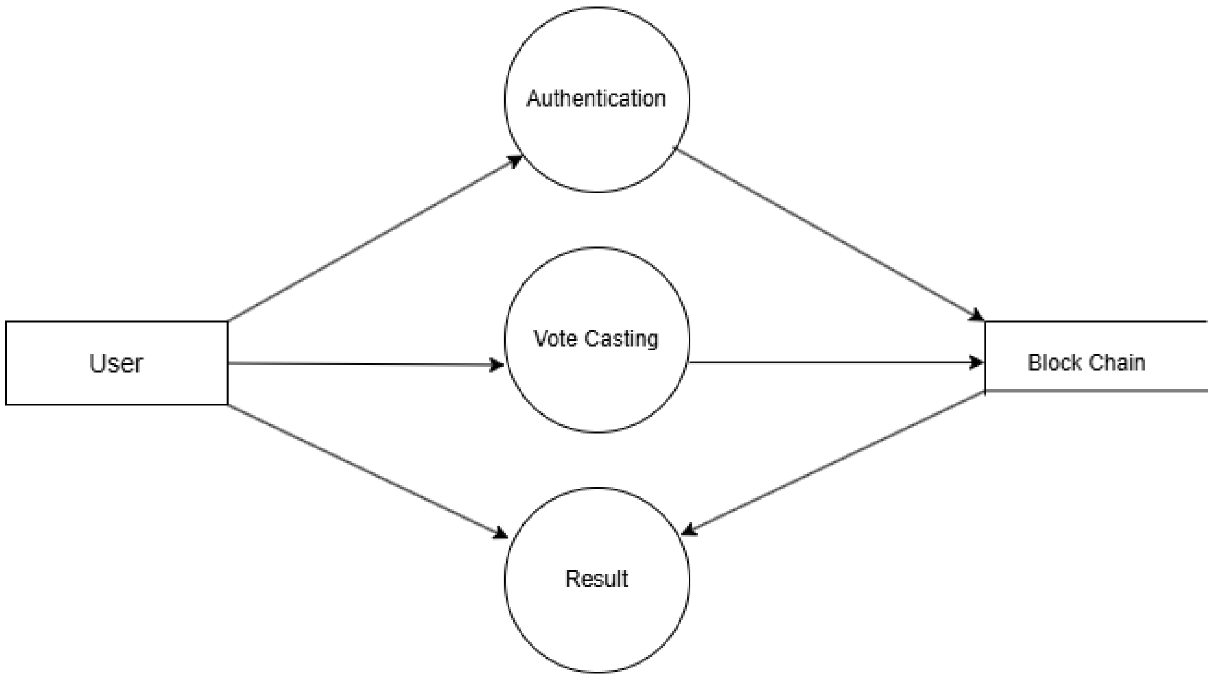
LEVEL 0



LEVEL 1 ADMIN



LEVEL 1 USER



5 TOOLS AND PLATFORM

5.1 BACKEND

5.1.1 SMART CONTRACT (SOLIDITY)

Solidity is a contract-oriented programming language used to write smart contracts on the Ethereum blockchain. It supports inheritance, libraries, and complex user-defined types. In the Decentralized Voting App, Solidity powers the core logic behind candidate registration, voter validation, voting mechanisms, and result computation. It ensures transparency, immutability, and trustless interactions in the election process.

5.1.2 Hardhat

Hardhat is a development environment for compiling, deploying, testing, and debugging Ethereum smart contracts. It provides a local blockchain environment, supports Ethers.js integration, and simplifies contract testing and deployment. Hardhat is used to build, test, and deploy the smart contract for the voting app.

5.2 FRONTEND

5.2.1 NEXT.js

Next.js is a React-based web development framework that enables server-side rendering and static site generation for highly responsive and SEO-friendly apps. It powers the user interface of the voting app, providing dynamic routing, page prefetching, and better performance

5.2.2. HTML (HYPERTEXT MARKUP LANGUAGE)

HTML is the standard language used to create the structure of web pages. It forms the backbone of your website’s frontend, allowing you to define elements like headings, paragraphs, forms, images, and links.

5.2.3. CSS (CASCADING STYLE SHEETS)

CSS is used to style the appearance of HTML elements. It provides layout, design, colors, fonts, and spacing for the website, making the user interface visually appealing.

5.2.4. JAVASCRIPT

JavaScript is a programming language that adds interactivity to the web pages. It is used to create dynamic content like interactive forms, real-time bidding, and notifications on the website, enhancing user experience.

5.3. BLOCKCHAIN INTEGRATION

5.3.1. Ethers.js

Ethers.js is a JavaScript library for interacting with the Ethereum blockchain. It allows the frontend to connect with the smart contract deployed on the blockchain, enabling functionalities like casting votes, fetching candidates, and viewing results.

5.3.2 MetaMask

MetaMask is a browser extension wallet that allows users to interact with Ethereum-based Dapps. It is used for wallet authentication, transaction signing, and interacting with the voting smart contract securely.

5.4. STORAGE

5.4.1 IPFS (Inter Planetary File System)

IPFS is a peer-to-peer decentralized storage protocol. In the Decentralized Voting App, it is used to store candidate images, documents (like PDFs), and other large files off-chain, while storing only their references (hashes) on the blockchain to reduce cost and improve efficiency.

5.5. VERSION CONTROL: GIT AND GITHUB

Git: Git is a distributed version control system used to track changes in the source code during development. It allows multiple developers to collaborate, manage code history, and roll back changes if necessary. Git tracks every change made, making it easier to work on different features or parts of a project simultaneously without conflicts.

GitHub: GitHub is an online platform for hosting Git repositories. It provides a cloud-based interface to share and collaborate on projects. GitHub allows developers to track issues, manage code branches, and handle pull requests, making it an essential tool for team-based development and version control.

5.6. DEVELOPMENT TOOLS: VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a lightweight, open-source code editor developed by Microsoft. It’s highly extensible, with a vast range of extensions and support for different programming languages and frameworks. VS Code provides features like IntelliSense (code suggestions), debugging, version control integration, and built-in terminal, making it a popular choice for web development. It’s ideal for Django development due to its compatibility with Python and Django-specific extensions.

6 IMPLEMENTATION

The implementation is one phase of software development. Implementation is that stage in the project where theoretical design is turned into a working system. Implementation involves placing the complete and tested software system into the actual work environment. Implementation is concerned with translating design specifications into source code. The primary goal of implementation is to write the source code to its specification, which can be achieved by making the source code clear and straightforward as possible. Implementation means the process of converting a new or revised system design into an operational one. The three types of implementations are: i. Implementation of a computerized system to replace a manual system, ii. Implementation of a new system to replace an existing one, and iii. Implementation of a modified system to replace an existing one. The implementation is the final stage, and it is an important phase. It involves the individual programming, system testing, user training, and the operational running of the developed proposed system that constitute the application subsystem. The implementation phase of the software development is concerned with translating design specifications into source code. The user tests the developed system and the changes are made according to the needs. Before implementation, several tests have been conducted to ensure no errors are encountered during the operation. The implementation phase ends with an evaluation of the system after placing it into operation over time.

7 RESULTS AND CONCLUSION

This project aimed to address the limitations and trust issues in traditional voting systems by developing a secure and transparent decentralized voting application.

7.1 CONCLUSION

The **Decentralized Voting App** successfully implements a transparent and tamper-proof voting system using blockchain technology. By leveraging smart contracts, IPFS integration via Pinata, and a user-friendly frontend built with Next.js, the system automates core voting functionalities such as voter registration, candidate approval, vote casting, winner declaration, and ownership transfer.

The application ensures integrity by recording all voting actions immutably on the Ethereum blockchain while using IPFS to store and retrieve media like candidate documents and images securely. Role-based access for admins and voters, real-time updates, and secure handling of voting periods all contribute to a reliable and democratic experience.

This project demonstrates a modern approach to conducting elections, offering transparency, decentralization, and increased voter confidence—essential qualities for both institutional and public elections in the digital age.

7.3 SCOPE OF FUTURE WORK

- **Mobile Application Development:** Build a cross-platform mobile app for increased accessibility, allowing voters and admins to interact with the voting system on the go.
- **Multi-Election Support:** Extend the system to support multiple simultaneous elections with separate voting periods, candidate pools, and results handling.
- **Automated Election Scheduling:** Allow admins to schedule multiple future elections with customizable parameters and automated status transitions.

8 REFERENCES

DOCUMENTATION AND TUTORIALS

- Next.js Documentation: https://nextjs.org/docs
- GitHub Documentation: https://docs.github.com/ - Used for version control and project collaboration.
- Pinata Documentation: https://docs.pinata.cloud/ - For uploading, managing, and retrieving files from IPFS securely using Pinata.
- https://www.google.com
- https://www.chatgpt.com - For generating code snippets, documentation, and debugging help.
- https://www.youtube.com- Tutorials on building DApps, smart contracts, and integrating with Web3 technologies.

9 APPENDIX

9.1 Schema

Voter Schema

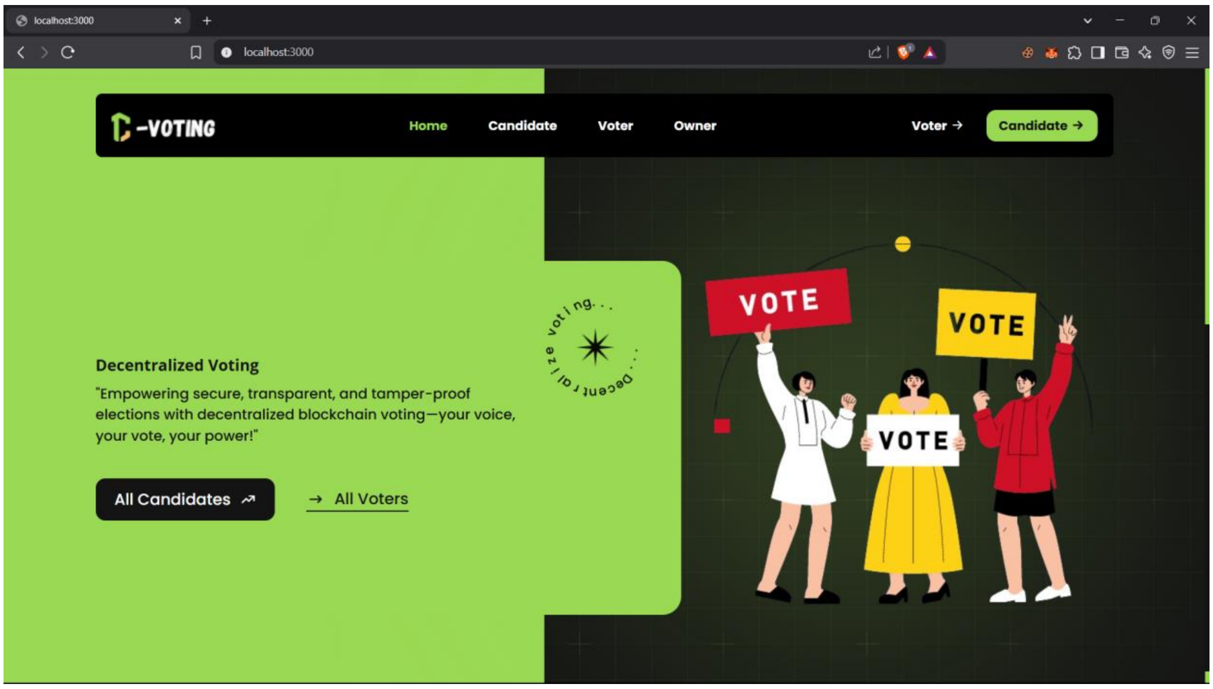
```
struct Voter {
        address voterAddress;
        string name;
        string ipfs;
        uint256 registerId;
        ApprovalStatus status;
        bool hasVoted;
        string message;
    }
```

Candidate Schema

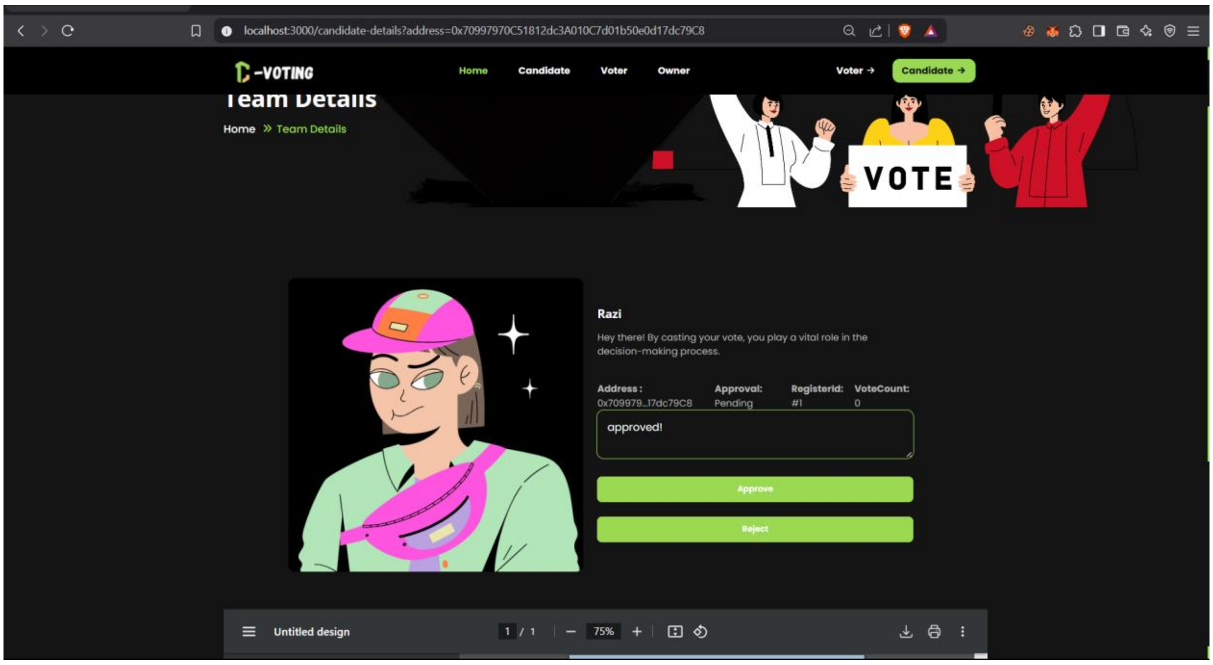
```
struct Candidate {
        address candidateAddress;
        string name;
        string ipfs;
        uint256 registerId;
        ApprovalStatus status;
        uint256 voteCount;
        string message;
    }
```

9.2 PROJECT IMAGES

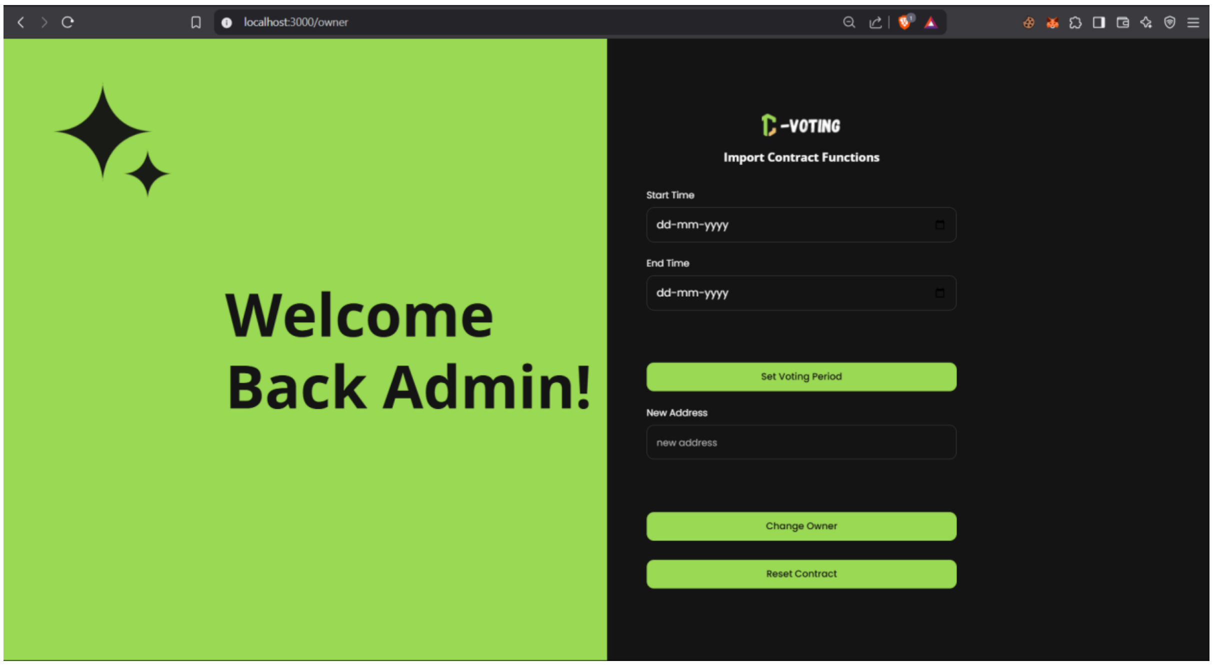
ADMIN DASHBOARD

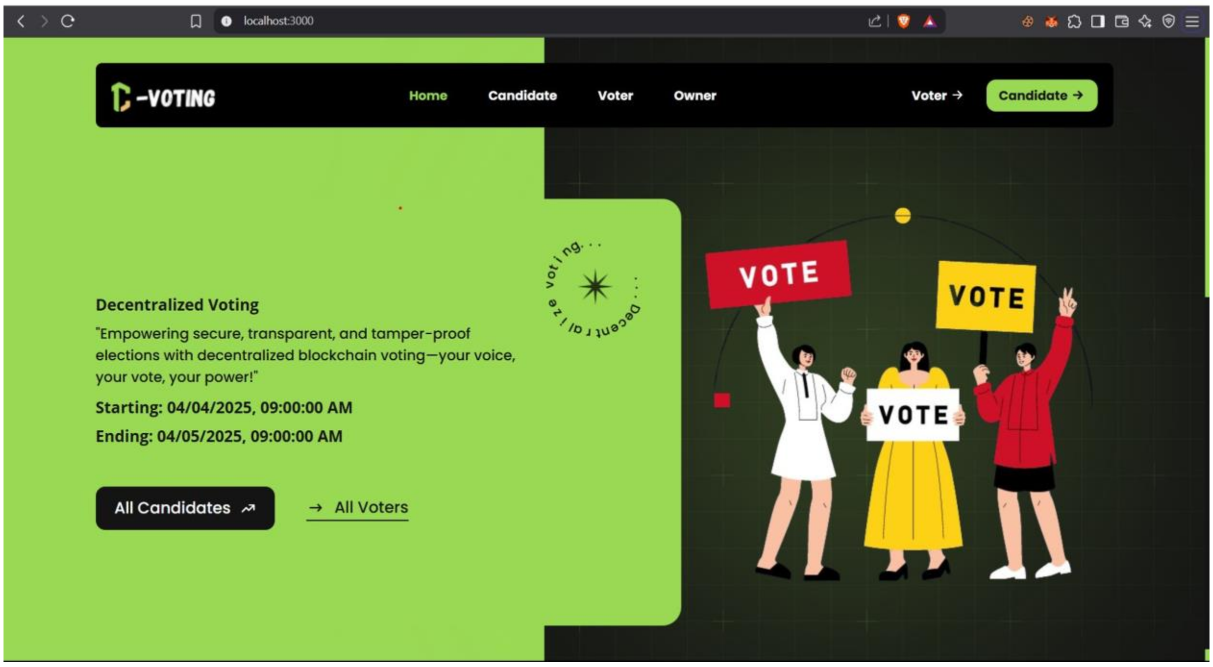


APPROVE CANDIDATE

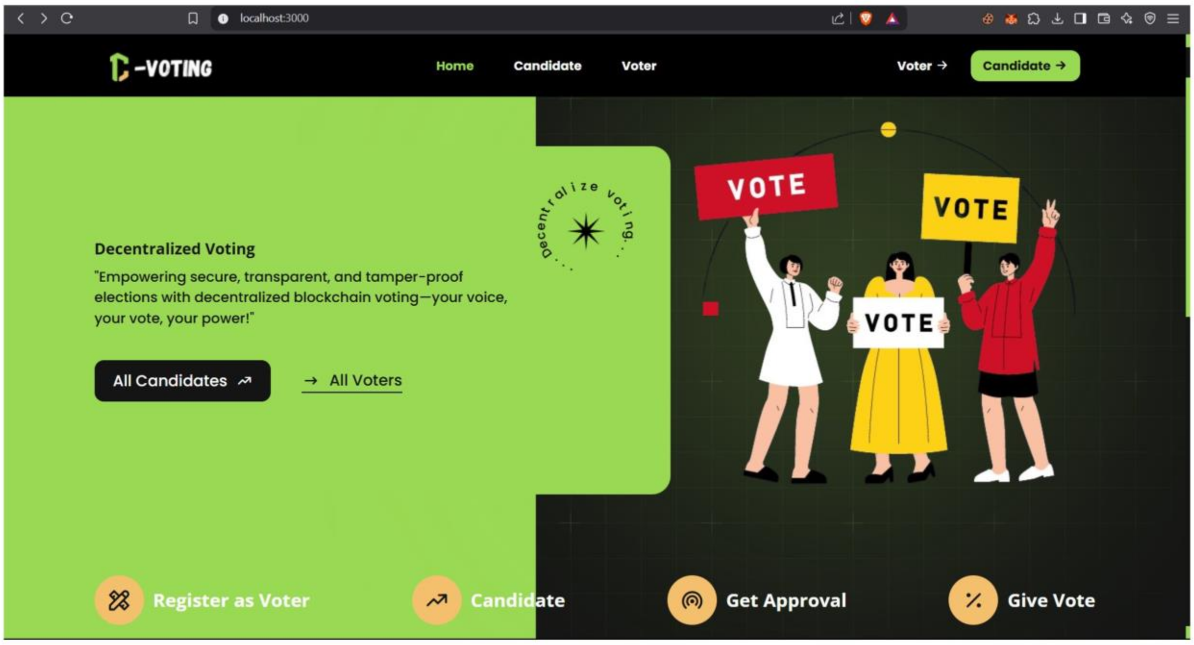

localhost:3000/candidate-details?address=0x70997970C51812dc3A010C7d01b50e0d17dc79C8
C-VOTING
Home
Candidate
Voter
Owner
Voter →
Candidate →
Team Details
Home » Team Details
VOTE
Razi
Hey there! By casting your vote, you play a vital role in the decision-making process.
Address: 0x709979...17dc79C8
Approval: Pending
Registerid: #1
VoteCount: 0
approved!
Approve
Reject
Untitled design
1 / 1
75%

ADMIN FUNCTIONALITIES


localhost:3000/owner
Welcome Back Admin!
C-VOTING
Import Contract Functions
Start Time
dd-mm-yyyy
End Time
dd-mm-yyyy
Set Voting Period
New Address
new address
Change Owner
Reset Contract


localhost:3000
C-VOTING
Home
Candidate
Voter
Owner
Voter →
Candidate →
Decentralized Voting
"Empowering secure, transparent, and tamper-proof elections with decentralized blockchain voting—your voice, your vote, your power!"
Starting: 04/04/2025, 09:00:00 AM
Ending: 04/05/2025, 09:00:00 AM
All Candidates
→ All Voters
Decentralize voting...
VOTE
VOTE
VOTE

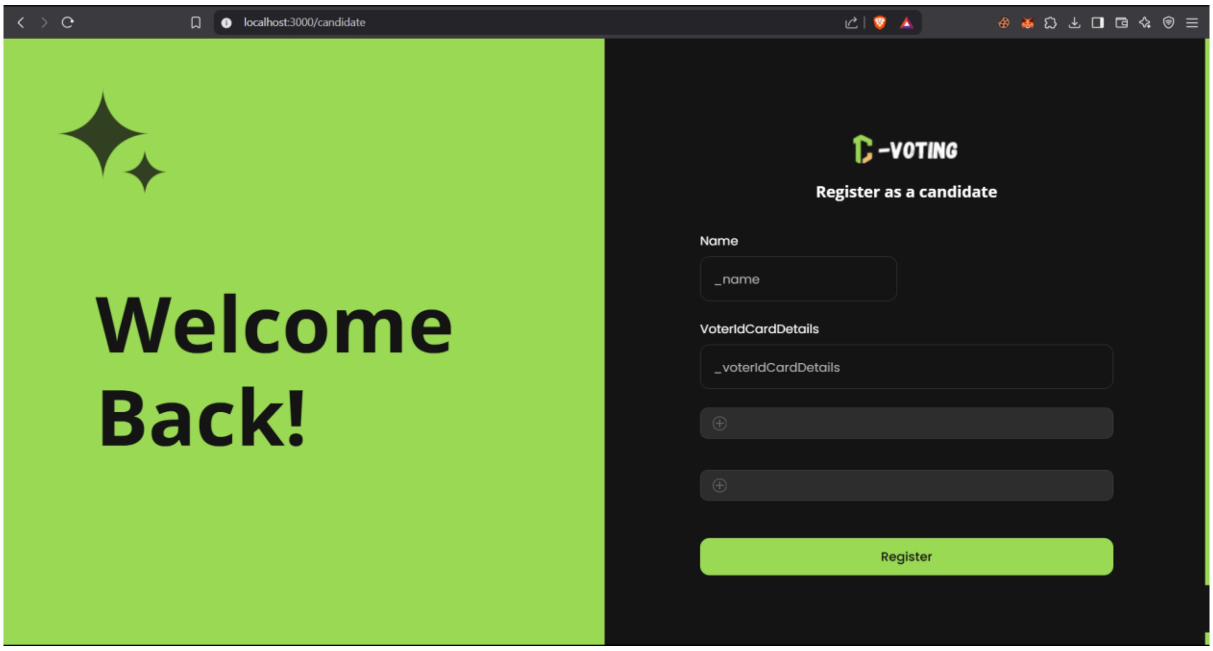
HOME PAGE

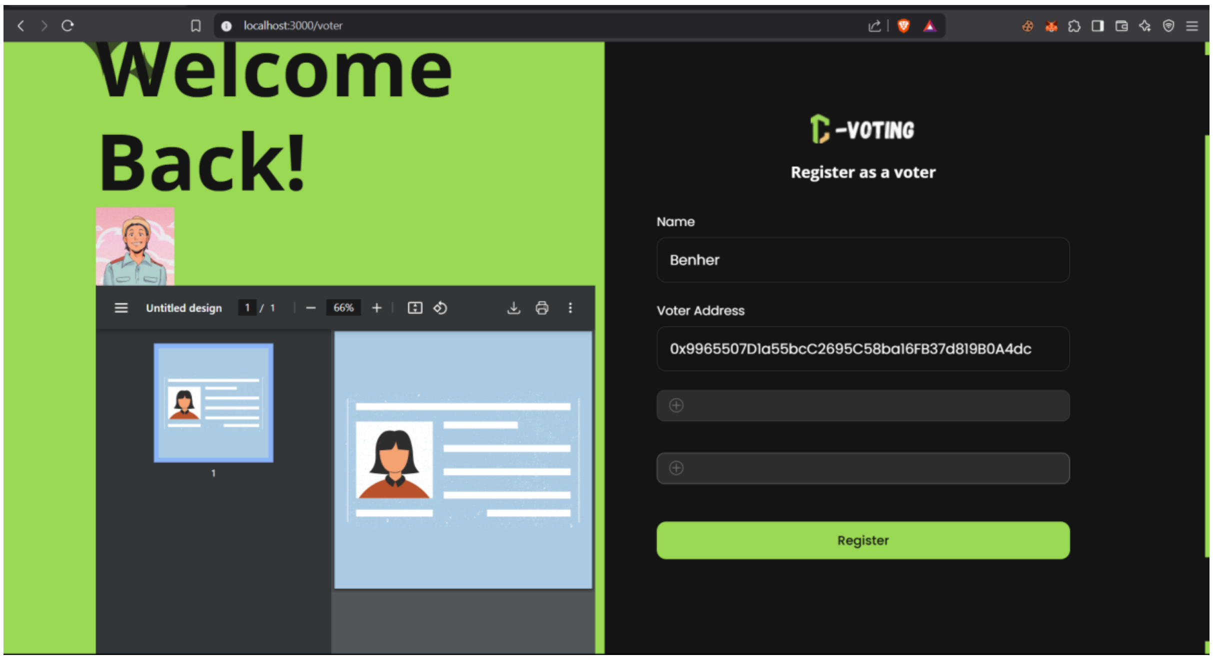


Decentralized Voting

"Empowering secure, transparent, and tamper-proof elections with decentralized blockchain voting—your voice, your vote, your power!"

CANDIDATE REGISTRATION





localhost:3000/voter

Register as a voter

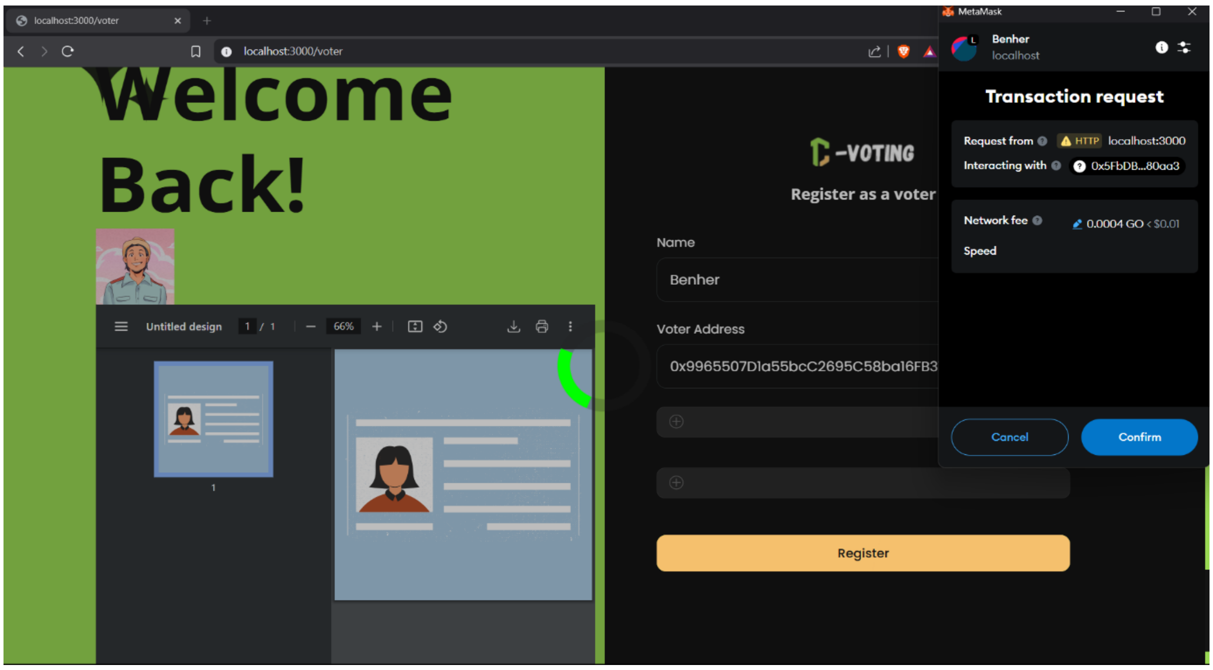
Name

Benher

Voter Address

0x9965507D1a55bcC2695C58ba16FB37d819B0A4dc

Register



localhost:3000/voter

Register as a voter

Name

Benher

Voter Address

0x9965507D1a55bcC2695C58ba16FB3

Register

MetaMask

Benher
localhost

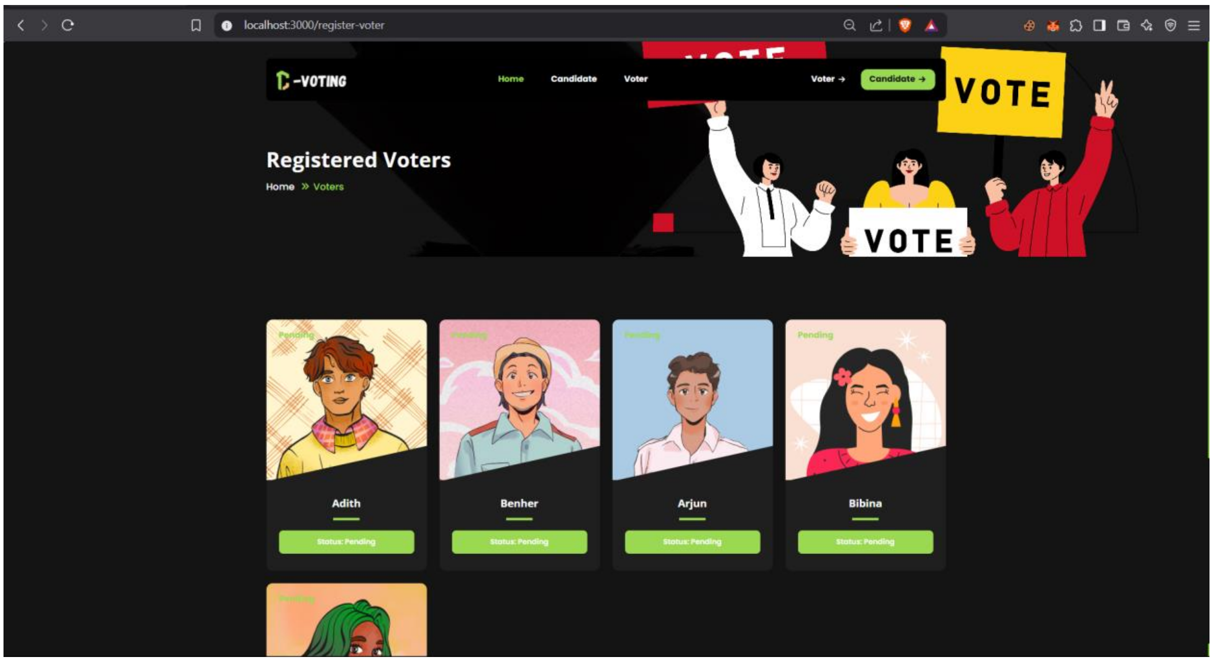
Transaction request

Request from HTTP localhost:3000
Interacting with 0x5FbDB...80aa3

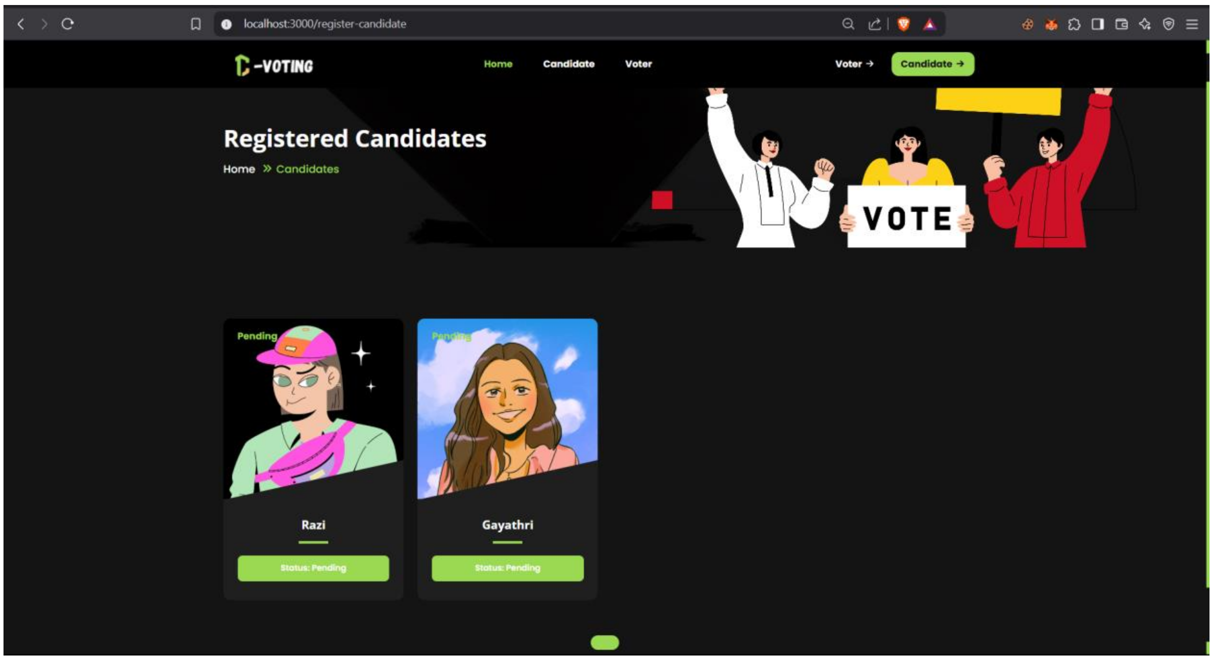
Network fee 0.0004 GO < $0.01
Speed

Cancel
Confirm

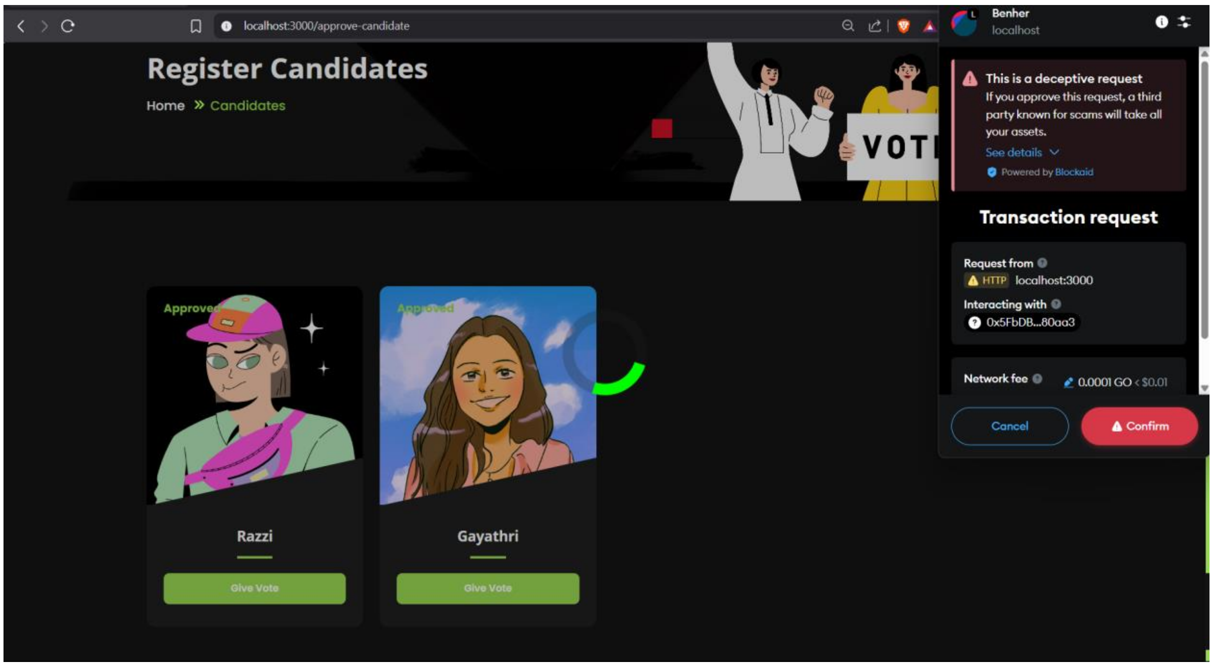
REGISTERED VOTERS



REGISTERED CANDIDATES



CAST VOTE

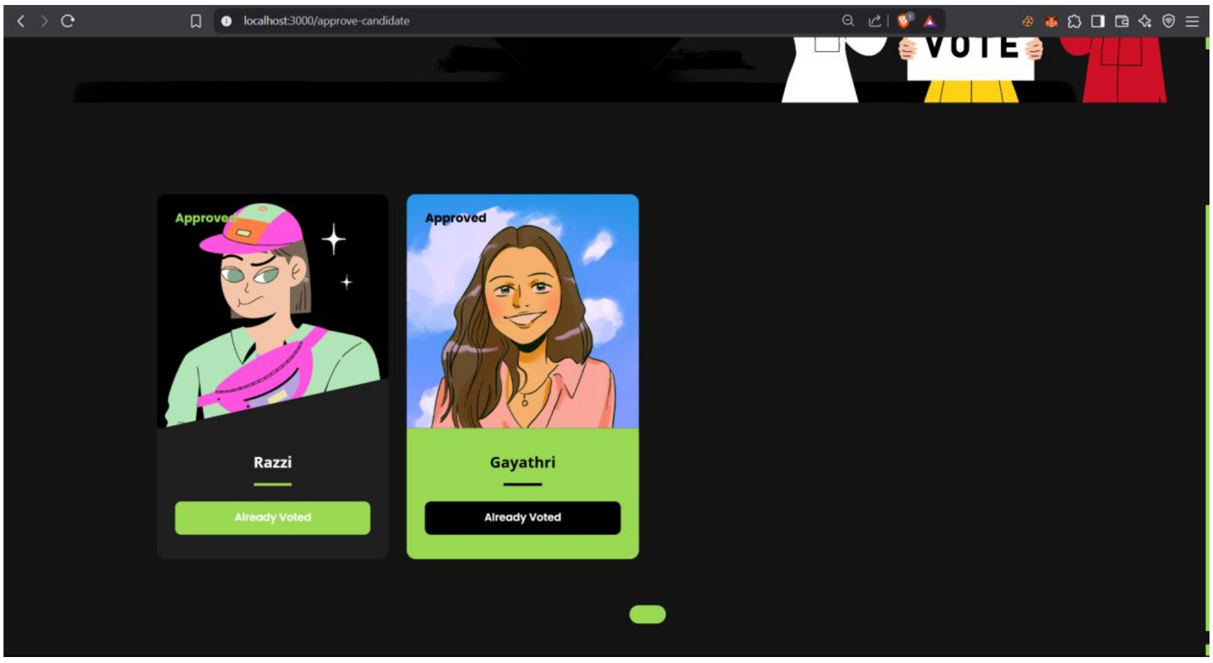


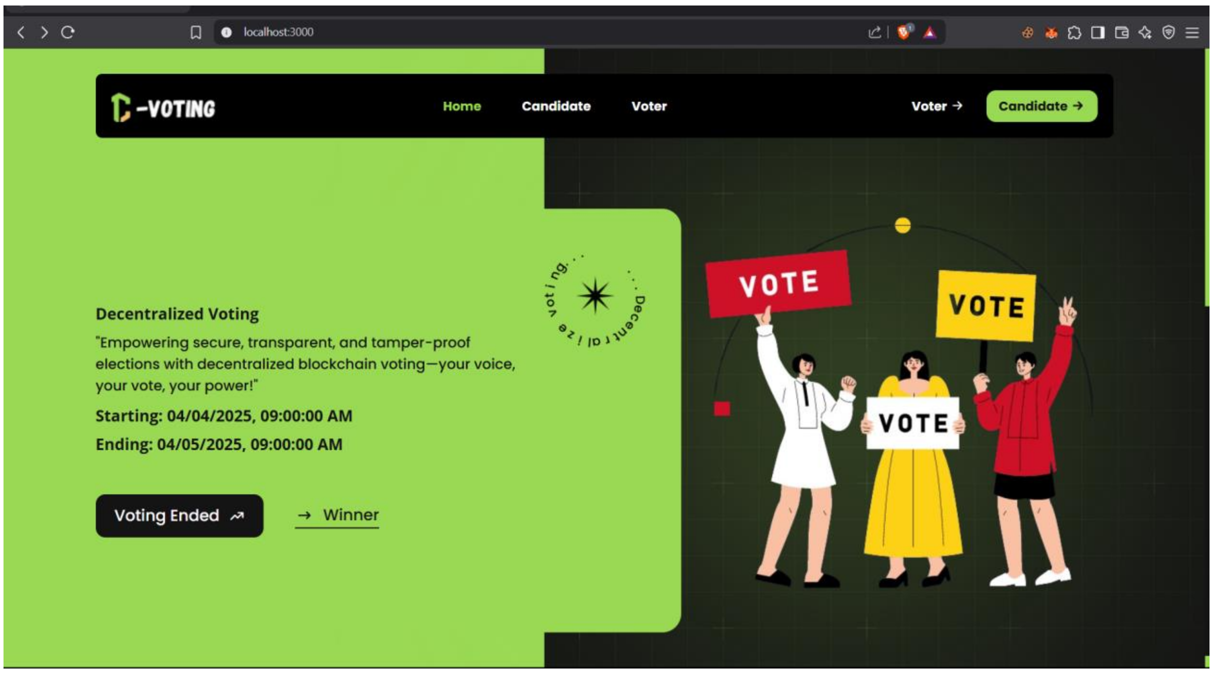
This is a deceptive request
If you approve this request, a third party known for scams will take all your assets.
See details
Powered by Blockaid

Transaction request

Request from
HTTP localhost:3000
Interacting with
0x5FbDB...80aa3

Network fee 0.0001 GO < $0.01





localhost:3000

C-VOTING Home Candidate Voter Voter → Candidate →

Decentralized Voting

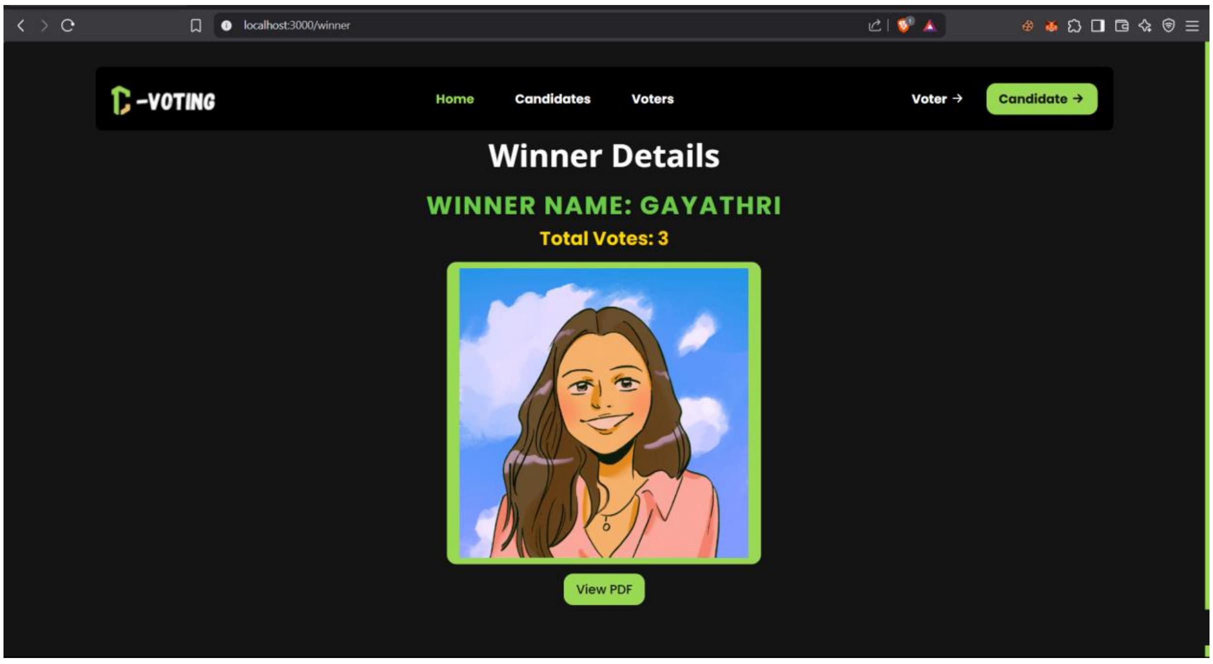
"Empowering secure, transparent, and tamper-proof elections with decentralized blockchain voting—your voice, your vote, your power!"

Starting: 04/04/2025, 09:00:00 AM

Ending: 04/05/2025, 09:00:00 AM

Voting Ended ↗ → Winner

WINNER



localhost:3000/winner

C-VOTING Home Candidates Voters Voter → Candidate →

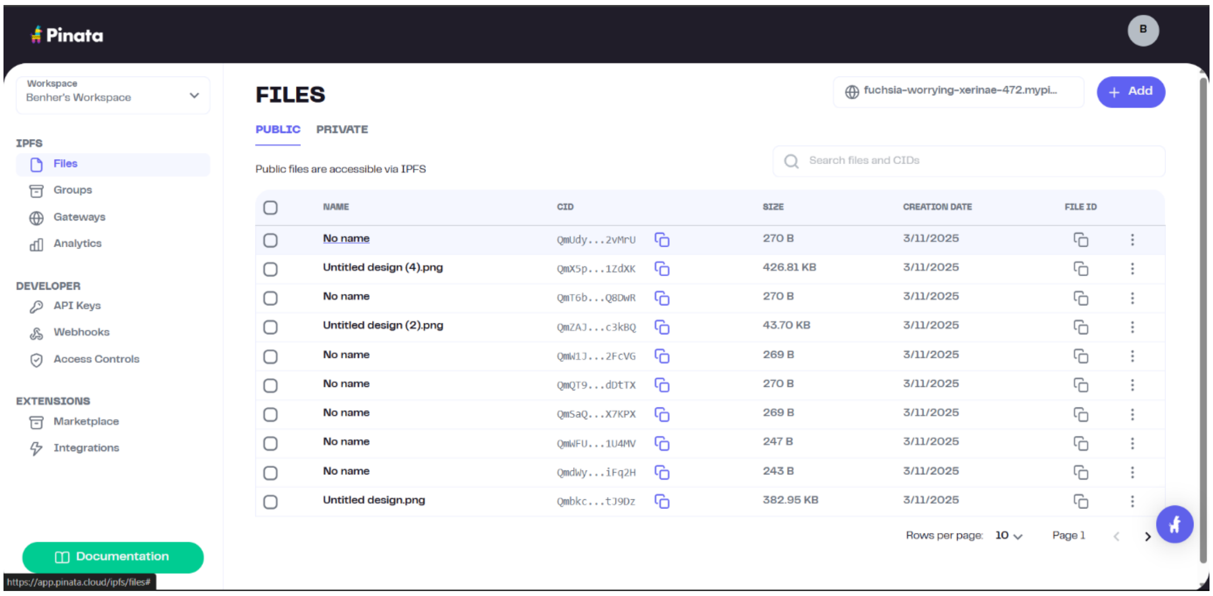
Winner Details

WINNER NAME: GAYATHRI

Total Votes: 3

View PDF

PINATA DASHBOARD



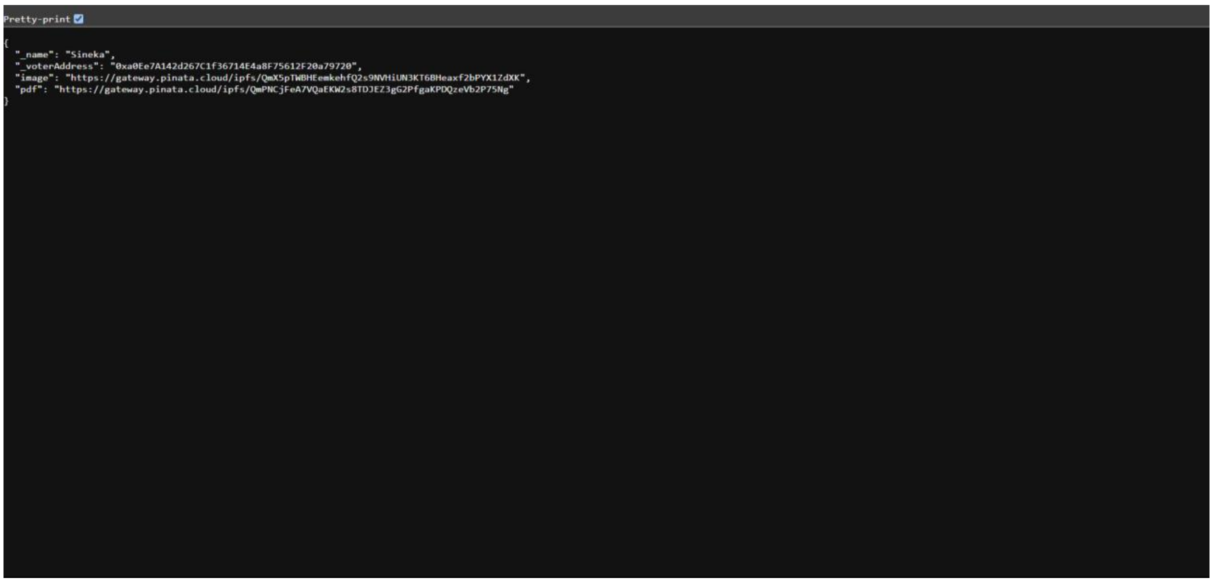
FILES

PUBLIC PRIVATE

Public files are accessible via IPFS

NAME	CID	SIZE	CREATION DATE
No name	QmUdy...2vMrU	270 B	3/11/2025
Untitled design (4).png	QmX5p...1ZdXK	426.81 KB	3/11/2025
No name	QmT6b...Q8DwR	270 B	3/11/2025
Untitled design (2).png	QmZAJ...c3kBQ	43.70 KB	3/11/2025
No name	QmW1J...2FcVG	269 B	3/11/2025
No name	QmQT9...dDtTX	270 B	3/11/2025
No name	QmSaQ...X7KPX	269 B	3/11/2025
No name	QmWFU...1U4MV	247 B	3/11/2025
No name	QmdWy...iFq2H	243 B	3/11/2025
Untitled design.png	Qmbkc...tJ9Dz	382.95 KB	3/11/2025

Rows per page: 10 Page 1



Pretty-print ☑

```
{
  "_name": "Sineka",
  "_voterAddress": "0xa0Ee7A142d267C1f36714E4a8F75612F20a79720",
  "image": "https://gateway.pinata.cloud/ipfs/QmX5pTWBHEemkehfQ2s9NVHiUN3KT6BHeaxf2bPYX1ZdXK",
  "pdf": "https://gateway.pinata.cloud/ipfs/QmPNCjFeA7VQaEKW2s8TDJEZ3gG2PfgaKPDQzeVb2P75Ng"
}
```