Next Generation Decentralised Electronic Voting System (VoteNet)

Integrated Post Graduate (IPG) Master of Technology

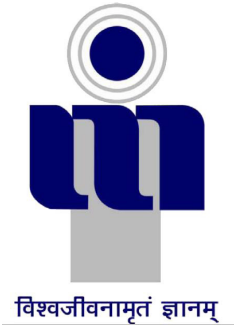
in

Information Technology

Mini Project Report

by

Rushikesh Taksande (2017IMT-074)
Ojaswa Sharma (2017IMT-102)
Vishesh Khandelwal (2017IMT-110)
Yashvant Gond (2017IMT-112)



ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT
GWALIOR-474 015

2019

CANDIDATES DECLARATION

We hereby certify that the work, which is being presented in the report, entitled **Next Generation Decentralised Electronic Voting System (VoteNet)**, in partial fulfillment of the requirement for the award of the Degree of **Integrated Post Graduate (IPG) Master of Technology** and submitted to the institution is an authentic record of our own work carried out during the period *February 2020* to *May 2020* under the supervision of **Dr. Somesh Kumar**. We also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Date: Signatures of the Candidates

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: Signature of the Research Supervisor

ABSTRACT

To be eligible to vote in India, one must first get his name on the Electoral Roll. On the day of the election, a prospective voter must be present in his constituency. The official checks their name on the Elector list and their identity card. Before casting a vote in India, the polling officers take voters' signatures and ink their finger. This inked finger allows the voter to proceed further to the booth where the voter can cast their vote on an Electronic Voter Machine. But the Indian Voting System allows it's voters to vote only from their constituencies. And because of this reason, voters have to rush back to their homes to cast a vote or not vote at all. This results in a decrease in the polling percentage. The polling percentage is important for a country because it depicts how legitimate a country's current system is. Next Generation Decentralised Electronic Voting System proposes a solution that helps voters to vote remotely, only by visiting a booth nearest to them.

Keywords: VoteNet, polling percentage, Retina/Fingerprint scanner, Ethereum, IPFS/IPNS, MERN, Blockchain, Face Recognition, Frontend, backend.

ACKNOWLEDGEMENTS

We are highly indebted to **Dr. Somesh Kumar**, and are obliged for giving us the autonomy of functioning and experimenting with ideas. We would like to take this opportunity to express our profound gratitude to them not only for their academic guidance but also for their personal interest in our project and constant support coupled with confidence boosting and motivating sessions which proved very fruitful and were instrumental in infusing self-assurance and trust within us. The nurturing and blossoming of the present work is mainly due to their valuable guidance, suggestions, astute judgment, constructive criticism and an eye for perfection. Our mentor always answered myriad of our doubts with smiling graciousness and prodigious patience, never letting us feel that we are novices by always lending an ear to our views, appreciating and improving them and by giving us a free hand in our project. It's only because of their overwhelming interest and helpful attitude, the present work has attained the stage it has.

Finally, we are grateful to our Institution and colleagues whose constant encouragement served to renew our spirit, refocus our attention and energy and helped us in carrying out this work.

(Rushikesh Taksande) (Ojaswa Sharma)

(Vishesh Khandelwal) (Yashvant Gond)

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

IPFS	InterPlanetary File System
GPG	GNU Privacy Guard
MERN	MongoDB, Express.js, React.js, and Node.js
SQL	Structured Query Language
FR	Face Recognition
webapp	Web Application
EVM	Electronic Voting Machine
OVM	Online Voting Machine
HTML	Hypertext Markup Language
JSX	JavaScript XML

CHAPTER 1

Introduction

VoteNet is a web app, written using MERN stack, that uses flask and python for Retina/Fingerprint scanner, and IPFS/IPNS and Ethereum for storing voter's data. As a voter enters the VoteNet booth, the Retina/Fingerprint scanner will fetch his/her voter ID, verify it across their physical Voter Identity Card. His data, stored and maintained safely using Blockchain, will be fetched. The voter can then cast a vote, which will remain anonymous and count of the candidate chosen by him/her will be incremented. The data stored on Blockchain is changeless, and therefore the voter's details are resistant to tampering.

1.1 Background

It is a well-known fact that India is the world's largest democracy. Before democracy was introduced in India, it was a country with vast religious diversity, illiteracy, and backward society. There were several doubts about democracy and if a dictatorship was the only best option for India. However, an electoral system was developed taking into consideration the ease of voting for the then citizens of India [1]

There have been a few changes in Indian politics and the election procedure since 1947. But in modern India, especially the one coping with an epidemic, there needs to be a change in the age-old processes being followed. How much money and movement in the country will it save if people don't have to return back to their homes in haste for one day, the voting day. VoteNet will be a support for voters to spare the time to reach the nearest VoteNet center. Thereby,reducing the time to vote by a substantial margin.

1.2 Motivation

Each vote matters and even a single vote can prove to be a decisive factor. How big a loss in polling percentage can it be to have a percentage of the population of India not vote? And 1% of Indian Population as of 1st June 2020 is 13,788,633 [2].

- According to the present Indian voting system, people are only allowed to vote from their own constituency, which makes it difficult for voters to reach back their hometowns, and in most cases, it is impossible for the voter to make a vote.
- This inability to vote significantly reduces the polling percentage.
- The previous work performed in this field lacks transparency and security of data. There is a need of a full fledged system that is user friendly.
- The recent growth in the field of deep learning has helped in creating better image/face recognition models which can help to correctly identify each voter.
- Use of Blockchain with IPFS can help securely store the voter's data.
- We propose a system that allow voters to vote remotely by visiting a VoteNet booth nearest to them.

1.3 Deep Learning for Face Recognition

We have used deep learning for face recognition [3] of a legitimate voter. When a Voter visits the booth installed with our application software his/her Voter ID will be fetched using Face Recognition, which can be verified from his physical Voter ID Card and data stored on IPFS records.

The face recognition algorithm is built using dlib's [4] state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark [5].

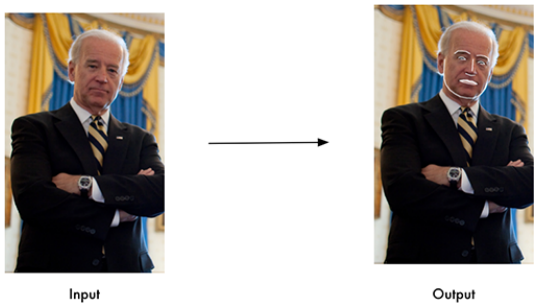


Figure 1.1: Recognition of facial features [3]

Facial recognition is used to map the features with those in the existing IPFS records and gather the Voter's data.

1.4 InterPlanetary File System

The InterPlanetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system.Each node stores a collection of hashed files [6]. A client who wants to retrieve any of those files enjoys access to a nice abstraction layer where it simply needs to call the hash of the file it wants. IPFS then combs through the nodes and supplies the client with the file. We have used IPFS for storing the voter's data which is securely kept on Blockchain on our own IPFS node. The data, being on blockchain, is immutable and hence the voter's details cannot be modified or deleted.

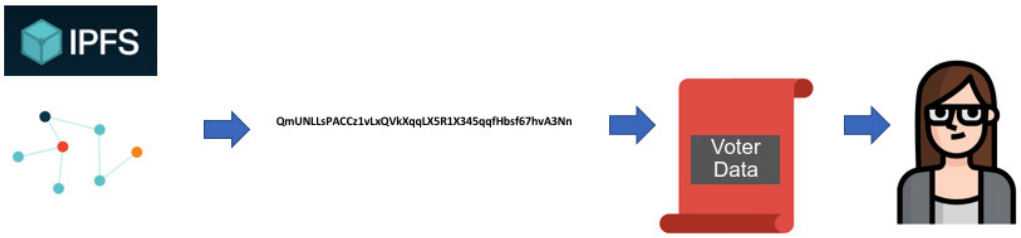


Figure 1.2: Retrieval of Voter's details using IPFS [7]

1.5 GPG Encryption

GPG is a type of asymmetric encryption that allows us to encrypt a file with the public key of the intended recipient so that only they can decrypt it when they retrieve it with IPFS. A malicious party who retrieves the file from IPFS can't do anything with it since they can't decrypt it.

A public and private key each have a specific role when encrypting and decrypting documents. A public key may be thought of as an open safe. When a correspondent encrypts a document using a public key, that document is put in the safe, the safe shut, and the combination lock spun several times. The corresponding private key is the combination that can reopen the safe and retrieve the document. We use GPG Encryption to store the Voter's data on the IPFS. Only the legitimate voter who's facial features map to the key of the record can retreive the data.

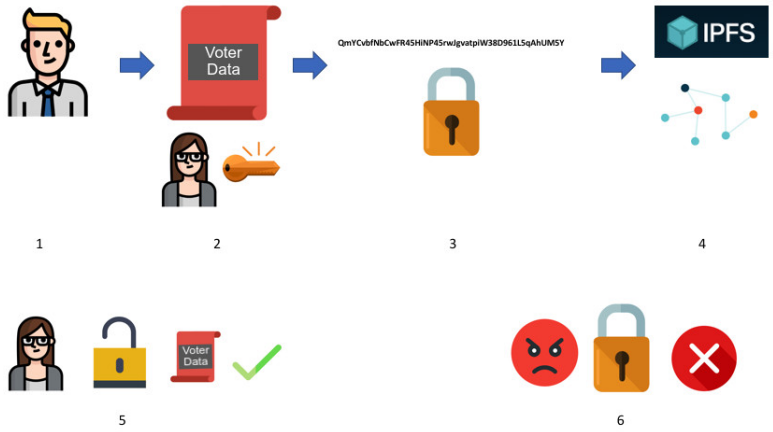


Figure 1.3: GPG Encryption [7]

1.6 Technology Stack

1.6.1 JavaScript

We have used the MERN stack for front-end and back-end of the web app of VoteNet. ReactJs and NodeJs frameworks of JavaScript are used in our project. React is an open-source JavaScript library for building user interfaces. React can be used as a base in the development of single-page or mobile applications. Node.js is an open-source, cross-platform, JavaScript run time environment that executes JavaScript code outside a web browser [8, 9].

1.6.2 Python

We have used Python in our project for Face Recognition and server for FR. Created by Guido van Rossum and first released in 1991,It was released in the year 1991 and designed by Guido van Rossum. Python considers white space significant and it emphasizes code readability by design. We have used deep learning and machine learning libraries for the development of the FR model. Flask is used for the server of FR. Flask is a micro web framework written in Python [10].

1.6.3 SQL

We have used PostgreSQL for the database management. PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures [11].

CHAPTER 2

Literature Survey

Authors in [12], suggest application of Blockchain as a service to implement distributed electronic voting systems. This Blockchain-based electronic voting system that utilizes smart contracts to enable secure and cost efficient election while guaranteeing voters privacy. Wei-Jr Lai, Yung-chen Hsieh, Chih-Wen Hsueh, Ja-Ling Wu in [13] suggest that may ensure the transparency of election by putting all messages on the Ethereum Blockchain; in the meantime, the privacy of individual voter is protected via an effective ring signature mechanism. As these papers suggest, there are several previous works done on Blockchain Based E-Voting Systems. But, for the best of our knowledge, none of them have devised for recognizing the voter to vote remotely. In our web application we have used deep learning for face recognition to recognize the voter as well as VoteNet is a proposal that helps people of modern India to reduce their travel cost and time. Hence, our project adapts features from the existing implementations and incorporating more transparency, security and cost effectiveness for a voter.

CHAPTER 3

System Design, Project Description and Methodology

3.1 System Design

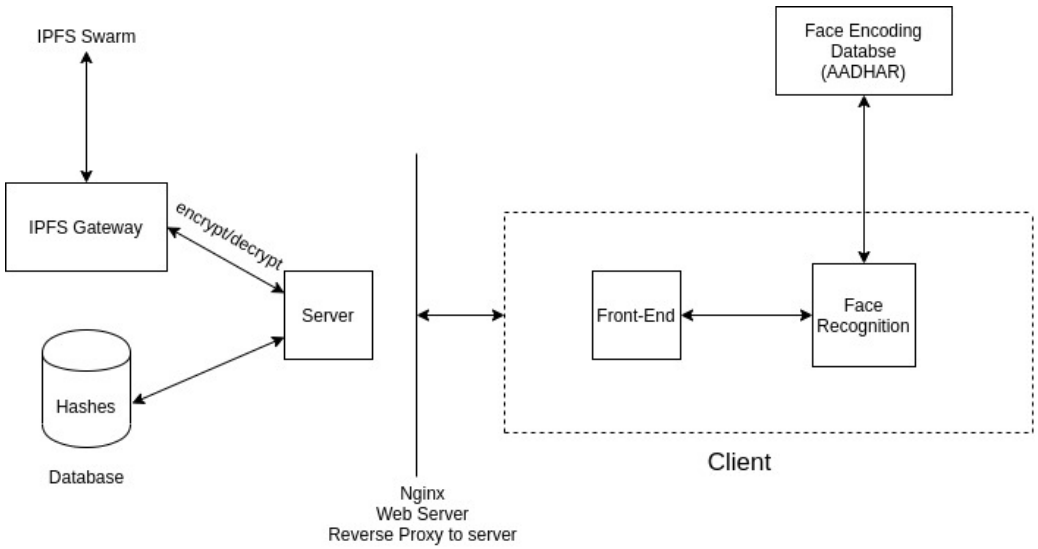


Figure 3.1: System Architecture of VoteNet

3.2 Project Description and Methodology

Whenever a person aligns his/her face in front of the camera for the purpose of voting, the face recognition mechanism is initiated. When the face is recognised, the Voter ID of the concerned person is retrieved from the database. Though in a real-world scenario, the face recognition system will have to be linked with the database of AADHAR maintained by the Government of India. For the sake of the project, we have currently created a testing database with dummy data to showcase the functioning of the system.

After the voter ID is retrieved from the database, it is sent by the Front-End to the Back-End server for initiation (as can be seen in the diagram). Now, at the back-end server, all the related details of the voter are fetched with the help of the voter ID. This includes the voter's constituency, details of the candidates from that constituency, if the voter has already voted or not etc. All these details are then sent back to the Front-End of the system. Once these details are verified, the names of the various participating candidates along with the political parties that they are representing are displayed to the voter, just like in an EVM.

Now the voter can select the candidate he/she wants to vote. All these details are then sent back to the Back-End, where the count of votes for the selected candidate is incremented by 1 and a column named 'didVote' is marked as true for the concerned voter, preventing him from casting his vote again during the ongoing elections.

The system also includes a secure admin panel, where the total count of votes of each candidate from the various constituencies can be seen, and that can be added to the number of physical votes.

Key features include:

- Any form of data from the database is decrypted on retrieval and encrypted before updates.
- All the data on IPFS is base64 encoded and the public keys are encrypted as well.

3.2.1 Data Flow and Methodology

The working and data flow of VoteNet can be broadly divided into 4 steps:

1. When the Vote Process is initiated on the client.

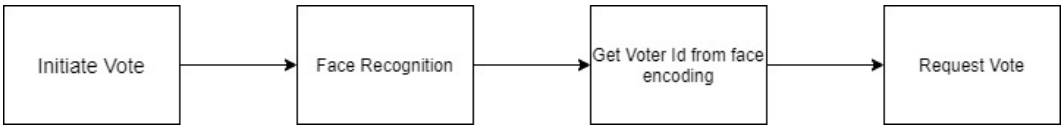


Figure 3.2: Front-end Data Flow

2. When Voter Id has been identified via Face Recognition, the client requests the server to Vote.

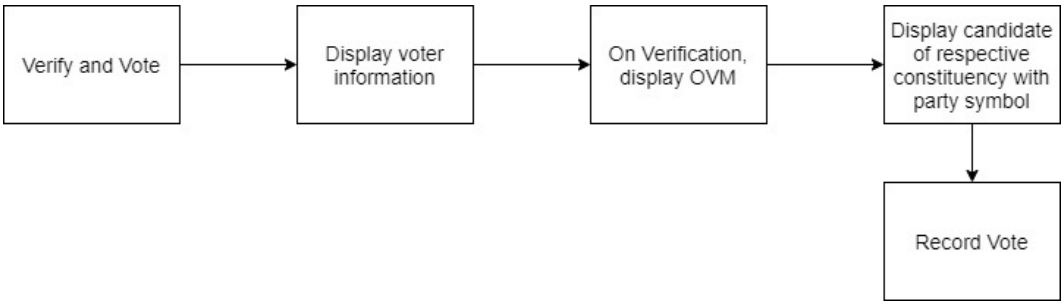


Figure 3.3: Server Data Flow

3. When the Voter and Constituency data is received by client.

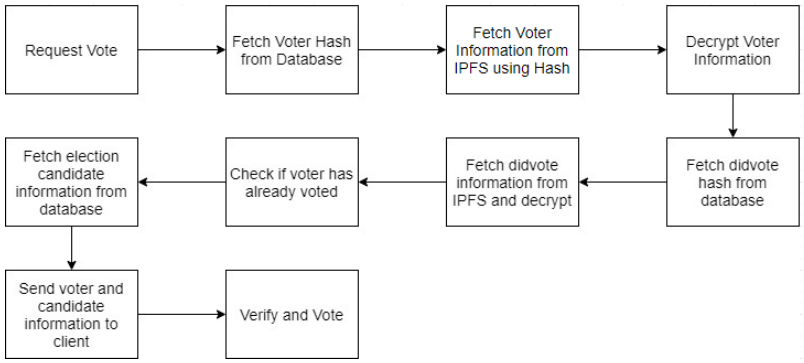


Figure 3.4: Client Data Flow

4. After the Voter casts his vote, the Candidate Id is sent to server to record the votes and complete the voting process.

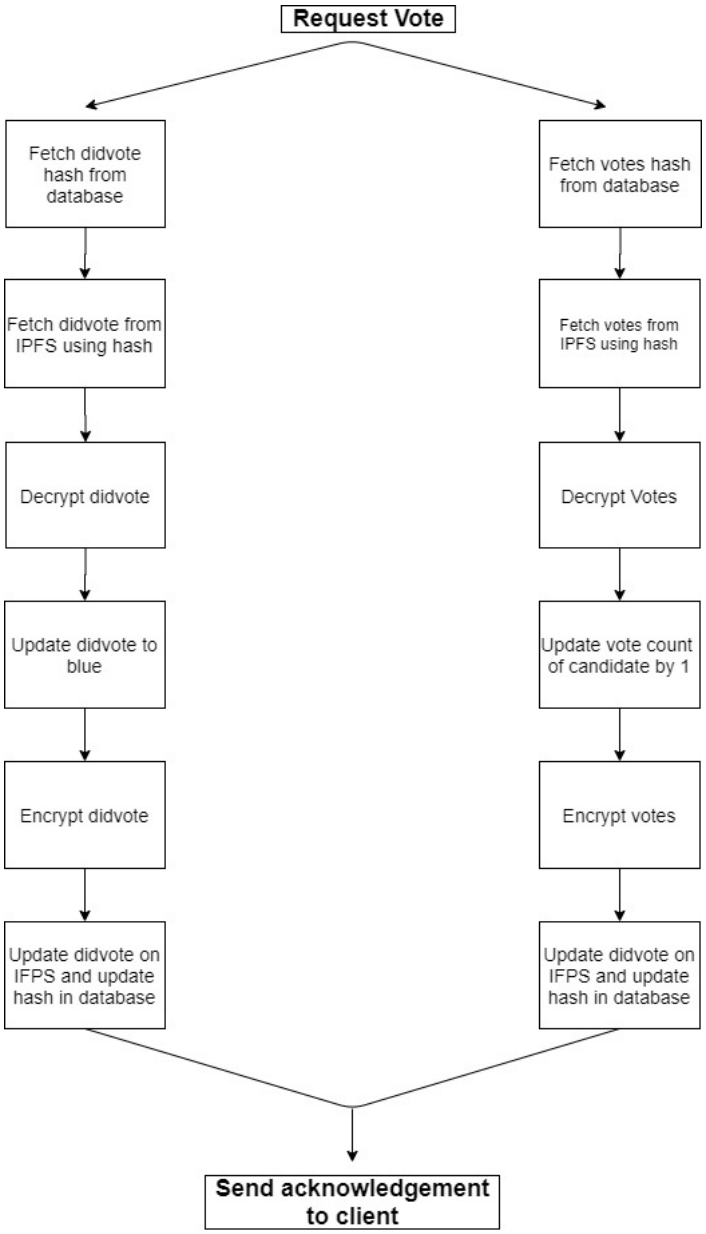


Figure 3.5: Back-end Data Flow

3.2.2 Implementation

3.2.2.1 The Front-end

The front-end is implemented using ReactJs, which is a JavaScript-based web development library. The code is hence written in JavaScript using the latest ECMAScript specifications. The following is a snapshot of front-end dependencies:

```
"dependencies": {
  "react": "^16.13.1",
  "react-awesome-modal": "^2.0.5",
  "react-dom": "^16.13.1",
  "react-rellax": "^1.0.3",
  "react-router-dom": "^5.2.0",
  "react-scripts": "3.4.1",
  "tachyons": "^4.12.0"
},
```

Figure 3.6: Front-end Dependencies

The complete web design is broken into different Components and rendered together conditionally. Here is an example of a Component OVMCard.js:

OVMCard.js

```
import React, { Component } from 'react';
import './OVMCard.css';
import logo from '../../assets/logo.png';
import btn from '../../assets/btn.png'


class OVMCard  extends Component {
    constructor(props){
        super(props);

    }
    render() {
        return(
            <div
                value={this.props.cid}
                onClick={() => {this.props.handleVoteReq(this.props.cid)}}
                className="OVMC-container pointer dim"
            >
                <img class="party_logo" alt="party-logo" src={this.props.symbol}></img>
                <p className="party_name">{this.props.name}</p>
            </div>
        );
    }
}

export default OVMCard;
```

Figure 3.7: OVMCard.js

As we can see in the above code, a Component is similar to a Function which takes some input and returns the HTML equivalent JSX (within JavaScript with React in scope). When the Component is rendered, the output JSX is compiled into an HTML element by React and thus displays the element on the webpage.

Similar to OVMCard, various other components are defined in the project structure, such as CandCard, Form and Dashboard.

After the development is complete, the React JavaScript files are built into normal JS,

CSS and HTML files which are familiar web browsers. The following is the structure of the public folder after build:

Figure 3.8: File structure

3.2.2.2 The Face Recognition Server

The face recognition server is a part of the Client side of the application, and identifies a Voter using his/her Face Encoding and returns their Voter Id for further processing.The web-server is built using Python-based Flask, which triggers the face-recognition model built using dlib's and OpenCV state-of-the-art face recognition built with deep learning.

```python
import face_recognition
from flask import Flask, jsonify, request, redirect
from facerec_from_webcam import startRec
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/', methods=['GET', 'POST'])
def startFaceRec():
    name = startRec()
    return jsonify(name)

if __name__ == "__main__":
    app.run(host='127.0.0.1', port=5001)
```

Figure 3.9: Flask Server

As you can see, the server hosts a / root route which launches the face-recognition model.The face-recognition function matches a face from video stream and matches frame-by-frame for existing face encoding and returns the most trusted value if found, or unknown otherwise.

3.2.2.3 The Server

The server is implemented using Express, which is a minimal and flexible Node.js web application framework that provides a robust set of features for web mobile applications and APIs. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.

```
app.get('/api', (req,res)=>{ res.send('it is working')});
app.post('/api/register', (req,res)=> {register.handleRegister(req, res, db, bcrypt, xss)});
app.post('/api/vote', (req,res)=>{vote.handleVoteRequest(req, res, db)});
app.post('/api/voted', (req,res)=>{vote.handleVoteResponse(req, res, db)});
app.get('/api/admin', (req,res)=>{ admin.getData(req, res, db) });
app.get('/api/profilex', withAuth, (req, res) => {profilex.handleProfile(req, res, db)});
app.get('/api/getusers', withAdmin, (req, res) => {getUsers.returnUsers(req, res, db)});
app.get('/api/checkAdmin', withAdmin, (req, res) => {
  res.sendStatus(200);
});
app.get('/api/checkToken', withAuth, (req, res) => {
  res.sendStatus(200);
});
```

Figure 3.10: Server Endpoints

3.2.2.4 Interplanetary File System (IPFS)

All uploaded files, and all of the blocks within it, are given a unique fingerprint called a cryptographic hash. IPFS powers the creation of diversely resilient networks that enable persistent availability — with or without Internet backbone connectivity. This means better connectivity and file availability even during natural disasters. It is a *peer-to-peer hypermedia protocol* which depends on nodes instead of traditional internet implementation.

For this application, we have set up a self-hosted IPFS peer gateway which allows decentralized storage and ownership of the gateway. However, even *if the gateway goes down, the votes are accessible due to distributed storage.* Following is the output after starting the ipfs daemon:

```
ubuntu@ip-172-26-12-163:/etc/nginx/sites-available$ sudo ipfs daemon
Initializing daemon...
go-ipfs version: 0.5.0
Repo version: 9
System version: amd64/linux
Golang version: go1.13.10
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/172.26.12.163/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/13.234.121.10/tcp/4001
Swarm announcing /ip4/172.26.12.163/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

Figure 3.11: IPFS Daemon

3.2.2.5 The Database

As specified in various sections above, PostgreSQL server is used to maintain a map (table) between storage entities and IPFS Hashes. The following SQL query is used to generate the database structure required for the usage of this application:

```
CREATE TABLE public.storage
(
    name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    hash character varying(200) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT storage_pkey PRIMARY KEY (name)
)
CREATE TABLE public.candidates
(
    cid integer NOT NULL,
    name character varying(40) COLLATE pg_catalog."default" NOT NULL,
    symbol character varying(500) COLLATE pg_catalog."default" NOT NULL,
    con character varying(20) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT candidates_pkey PRIMARY KEY (cid)
)
```

Figure 3.12: Database Structure

The Hashes and the Candidate Data are stored in the following manner:

	name [PK] character varying (30)	hash character varying (200)
1	voters	QmPiv1Mt7Macd6TX8NnTQ2v8yttH8o2QWMErW13J9acePh
2	didvote	QmQULuJWSkNGecMnsSYnhr3VgB5ykGaYs1jPktrTC8U86P

Figure 3.13: Hashes

	cid [PK] integer	name character varying (40)	symbol character varying (500)	con character varying (20)
1	1	Lalu Prasad	https://upload.wikimedia.or...	BKN01
2	2	Lal Pant	https://upload.wikimedia.or...	BKN01
3	3	Ghadi Chand	https://upload.wikimedia.or...	BKN01
4	4	Damru Singh	https://upload.wikimedia.or...	BKN01
5	5	Mahatma gandhi	https://upload.wikimedia.or...	BKN01
6	6	Sonoaaa gandhi	https://upload.wikimedia.or...	GND02
7	7	Honey Singh	https://upload.wikimedia.or...	GND02
8	8	GhadiMan Chandu	https://upload.wikimedia.or...	GND02

Figure 3.14: Candidate Data

All private information such as Votes and Voter Ids are encrypted and stored on IPFS, while all public information such as Candidates are stored in the database.

CHAPTER 4

Implementation Results & Screenshots of the Project

1. The user visits the web application. At the home page of the application, there are tabs on the navigation bar for About Us and VOTE page.

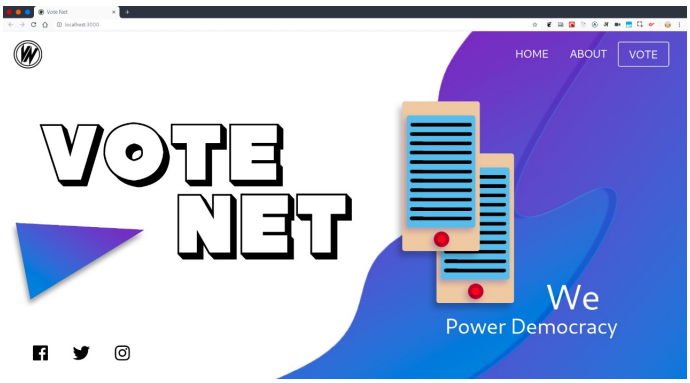


Figure 4.1: Index Page

2. The user visits the About Us page.

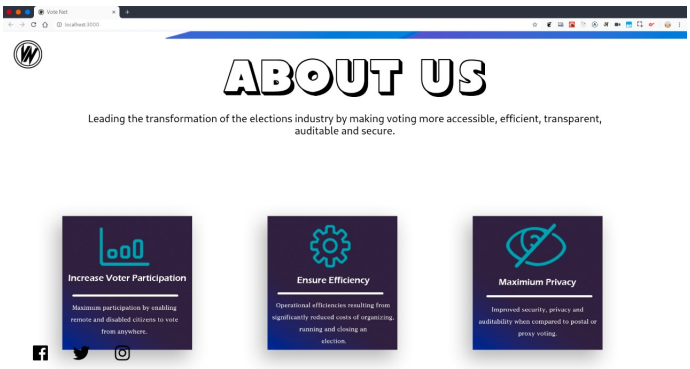


Figure 4.2: About Us Page

3. After clicking on VOTE, this button initializes the voting process by launching face recognition. Face Recognition pops up and uses live video stream to identify face of Voter and displays corresponding name for reference.

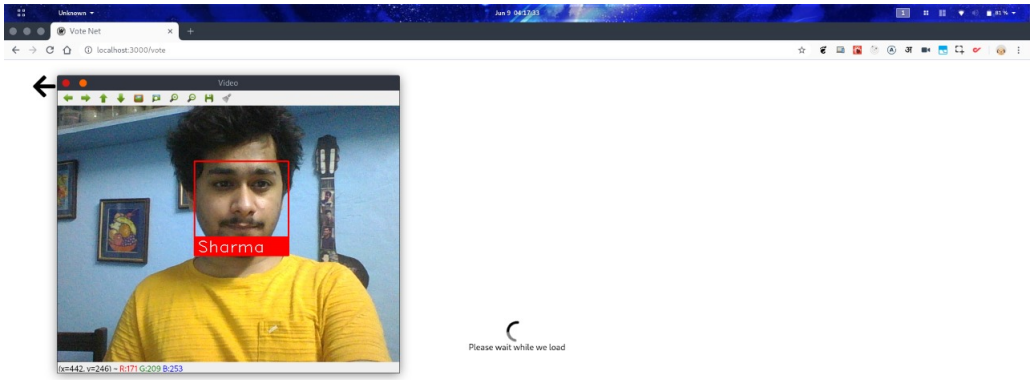


Figure 4.3: Face Recognition

4. After successful recognition, voter details are prompted for verification. These can also be matched with physical Voter Id for any discrepancies.

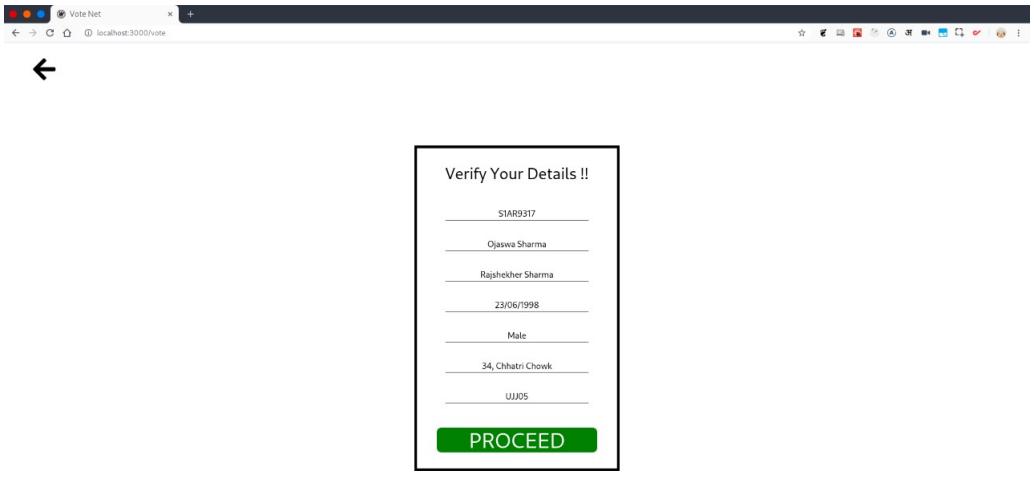


Figure 4.4: Voter details after successful face recognition

5. After details verification, Online Voting Machine (OVM) with candidates from corresponding constituency are displayed.

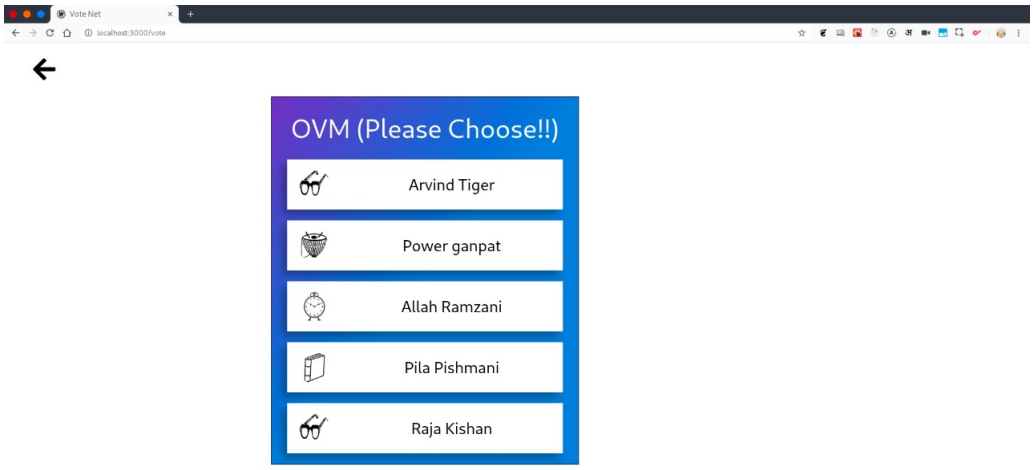


Figure 4.5: Candidates for Election

6. On clicking on a candidate, the vote is casted and user input is blocked. If the same user tries to vote again, he is refused to vote and an error saying "Already Voted" is prompted.

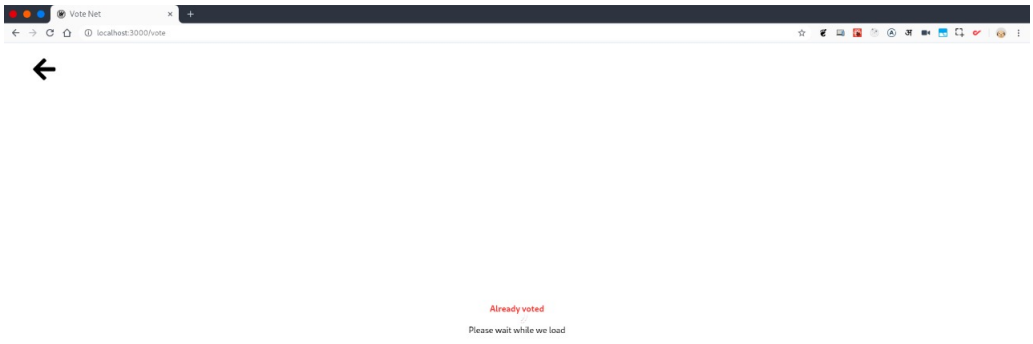


Figure 4.6: Error Prompt

7. After the election, the constituency-wise results can be displayed on the admin dashboard. These results can be merged with the physical voting (if any) for calculating the final results.

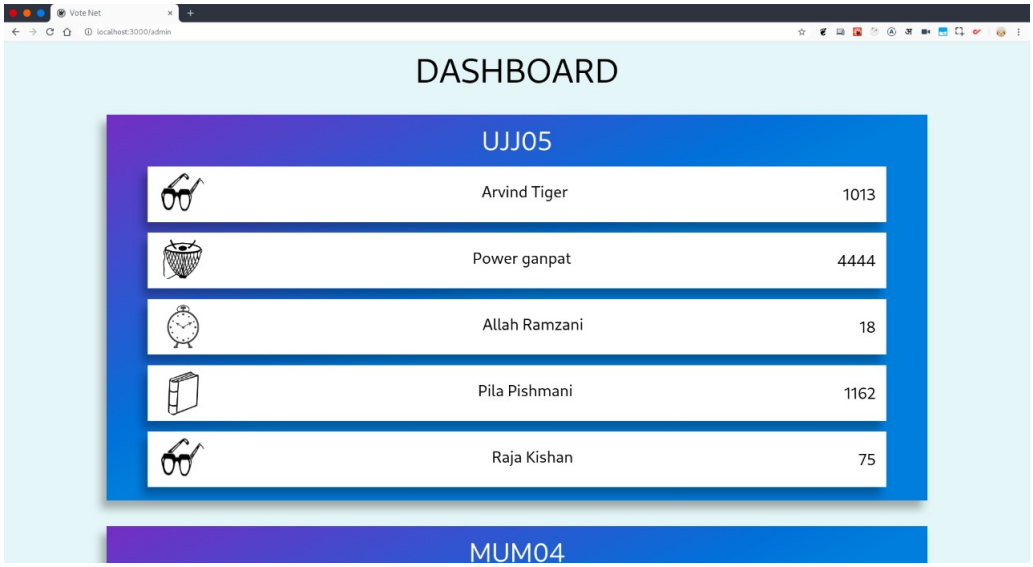


Figure 4.7: Admin Dashboard

CHAPTER 5

Conclusion and Future Scope

5.1 Conclusion

A piece of technology is recognized and accepted when it can lead to a positive change in several lives. VoteNet is a proposal that helps people of modern India to reduce their travel and focus on what matters. They can now vote without having to consider making a travel plan, which for one day of worth forces them to travel back and forth for days. This article suggests a VoteNet booth to be set up in major cities of India. All a VoteNet booth uses is a web-app that takes care of users' data and their voting records and these functionalities are made secure using Blockchain and IPFS records.

The idea of adapting digital voting systems to make the public electoral process cheaper, faster and easier, is a compelling one in modern society. Making the electoral process cheap and quick, normalizes it in the eyes of the voters, removes a certain power barrier between the voter and the elected official. It also opens the door for a more direct form of democracy, allowing voters to express their will on individual bills and propositions.

5.2 Future Scope

Deep learning and face recognition is a very broad territory and therefore the algorithm used is open to a variety of programmable and advanced results. For the sake of the project, we have currently created a testing database with dummy data to showcase the functioning of the system. However, in a real-world scenario, the face recognition system will have to be linked with the database of AADHAR maintained by the Government of India.

While the model used provides 97.5% accuracy, fingerprints and Iris maps can be further used individually or compositely to enhance the overall accuracy of the system and

achieve near 100% accuracy. Each vote can be considered as a Blockchain transaction which leverage Ethereum Smart Contracts to provide Immutable, Deterministic, decentralized and controlled outcome for every action [15]. Kubernetes can be used while deploying the server. It provides framework to run distributed systems resiliently. It takes care of scaling and failover for the application and provides deployment patterns [16].

With the evolution of technology supercomputers in future may be able to break though the security and keeping the database secure in the most primary concern. Opening and setting-up VoteNet booths in a big country such as India and to be able to handle such a large population requires servers which can handle such data in a secure and efficient manner is an area which requires further discussion and necessary permissions from the concerned authorities.

REFERENCES

[1] Election History. [Online]. Available: http://indiansaga.com/history/postindependence/elections.html

[2] Population Survey. [Online]. Available: https://www.worldometers.info/world-population/india-population/

[3] Deep Learning Face Recognition. [Online]. Available: https://github.com/ageitgey/face_recognition

[4] DLib C++ Library. [Online]. Available: http://dlib.net/

[5] Labeled Faces in the Wild. [Online]. Available: http://vis-www.cs.umass.edu/lfw/

[6] IPFS. [Online]. Available: https://ipfs.io/

[7] Blockchain with IPFS [Online]. Available: https://medium.com/@mycoralhealth/learn-to-securely-share-files-on-the-blockchain-with-ipfs

[8] React.js [Online]. Available: https://reactjs.org/

[9] Node.js [Online]. Available: https://nodejs.org/

[10] Python [Online]. Available: https://www.python.org/

[11] PostgreSQL [Online]. Available: https://www.postgresql.org/

[12] Friðrik Þ. Hjálmarsson, Gunnlaugur K. Hreiðarsson, "Blockchain-Based E-Voting System," in *Proceeding of 2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE,2018,pp. 1-10.

[13] Wei-Jr Lai, Yung-chen Hsieh, Chih-Wen Hsueh, Ja-Ling Wu , "DATE: A Decentralized, Anonymous, and Transparent E-voting System," in *Proceeding of 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. IEEE,2019,pp. 341-349.

[14] Gang Han, Yannan Li, Yong Yu, Kim-Kwang Raymond Choo, Nadra Guizani ,"Blockchain-Based Self-Tallying Voting System with Software Updates in Decentralized IoT" in *IEEE Network*. IEEE,2020,p. 1-7.

[15] Ethereum Book [Online]. Available: https://github.com/ethereumbook/ethereumbook/blob/develop/07smart-contracts-solidity.asciidoc#what-is-a-smart-contract

[16] Kubernetes [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#why-you-need-kubernetes-and-what-can-it-do