1. INTRODUCTION

1.1 Project overview

Voting turnout rate in India has steadily but slowly increased over the years but it is still not up to the global standards. This inadequacy can be overcome by making the current voting system more accessible to the general populace while still being secure and reliable. A secure online voting system can be achieved using blockchain technology. Blockchain is a distributed database existing on multiple computers at the same time. It is a constantly growing as new sets of recordings, or 'blocks', are added to it. In layman's terms, a blockchain is a record book which contains the details of transaction data. By design, a blockchain is resistant to modification of the data. Using blockchain technology in voting can increase the voter turnout exponentially by providing a secure, reliable and remote transfer of votes.

A blockchain is a distributed, immutable, incontrovertible, public ledger. This new technology works through four main features:

(i) The ledger exists in many different locations: No single point of failure in the maintenance of the distributed ledger.

(ii) There is distributed control over who can append new transactions to the ledger.

(iii) Any proposed "new block" to the ledger must reference the previous version of the ledger, creating an immutable chain from where the blockchain gets its name, and thus preventing tampering with the integrity of previous entries.

(iv) A majority of the network nodes must reach a consensus before a proposed new block of entries becomes a permanent part of the ledger.

These technological features operate through advanced cryptography, providing a security level equal and/or greater than any previously known database. The blockchain technology is therefore considered by many, including us, to be the ideal tool, to be used to create the new modern democratic voting process.

1.2 Objectives

- To increase voter turnout
- To make voting more accessible
- To increase security in remote voting
- To increase trust between government body and public
- To give real-time updates on results
- To decrease extravagant expenses on booths

1.3 Purpose

The purpose of this project is to study ways to reduce public inconvenience during voting season and make voting accessible to people who are old, sickly and also to people who are forced to skip voting as they live in remote places.

1.4 Scope

The main motivation behind this project is our interest in making the voting process accessible to each and every voter of India. This involves increasing the voter turnout and curbing the rampant corruption that is prevalent during every election season. India being a largest democracy in the world undergoes elections very frequently. This project if implemented correctly can prevent loss of votes. Furthermore it will eliminate human error and heavy expenses from the equation.

1.5 Applicability

- General body elections
- Cross Country elections
- College Student body elections
- Board of Directors election
- Club head election
- Other elections where anonymity is required

2. LITERATURE SURVEY

2.1 Existing System

- The existing system uses Electronic Voting Machines (EVM) in polling booths which are located in every constituency with appropriate security deployed before each voting session.
- Voter has to stand in a queue before bring admitted in to the voting center, where they have to be verified manually by the government employees.
- To be admitted inside a constituency the voter has to carry multiple identity proofs.
- Indelible ink is applied to every voter to prevent repeated votes.
- Remote voting is only available to soldiers deployed to the border.
- Votes are manually counted and displayed after a few days.

2.2 Problems in the existing system

- Provisions are not made for voting from outside the constituency.
- Long waiting times in queues outside the voting constituency.
- Vote counting errors are still prevalent.
- Gerrymandering is used to garner votes.
- Some people are allergic to the indelible ink that is applied after voting.
- Senior citizens face problems while standing in the queues.
- Most constituencies are not handicap accessible.

2.3 Available solutions and their features

2.3.1 Paper Ballots

- The voter is assured that their vote has been cast
- Vote cannot be changed later
- Less prone to mechanical or electronic errors
- In-person campaigning

2.3.2 Mail-in voting

- Voters can mail their vote through post
- This method can be used during pandemic outbreaks
- Voters must implicitly trust their mailing service
- Multiple checks are conducted

2.3.3 Remote/Internet voting

- Can vote from anywhere
- IP tracing system can be deployed
- Quicker election results
- Ease of use is better than current system

2.4 Proposed System

2.4.1 Problem Definition

Developing a secure online voting system using blockchain technology to increase voter turnout exponentially.

2.4.2 Defining Features

- Triple authentication
  - Local finger print authentication
  - Login credentials
  - OTP authentication
- User-friendly interface
- Real-time update of results
- Secure transfer of vote using blockchain
- Ease of use is highly increased

2.5 Advantages of Proposed System

- Increased voter turnout
- Remote voting ensures that everyone can vote
- Easy registration compared to existing system

- Provide employment for maintainers
- Better user interface compared to existing system
- More secure than any other previous systems

2.6 Identification of System need

- The android application requires constant and stable internet connection.
- Registration must be completed before an attempt to login.
- Finger print identification is required as a form of authentication.
- The current version of the system requires manually searching for the results.
- Multitude of details must be furnished for the registration.

2.7 Feasibility Study

Creating a sustainable and adaptable voting application on a national scale is a challenging task due to the varying number of voters in any particular constituency. Our proposed project seeks to eliminate many such problems by using a blockchain instead of a conventional storage system. As to the question, if the proposed model is feasible or not, since it uses a blockchain there doesn't have to be a limit on the number of people / blocks that are added to an existing chain. Since hacking a block chain requires an arsenal of very skilled hackers who must co-ordinate their attacks to the tenth of a second to alter just one of the blocks, it is quite unlikely that such an attack will occur. Depending on what organization handles the registration, it can be used either from the organizers side or voter side, which will reduce meddling of biased and politically backed entities. It increased the ease of use for first time voters who might shy away from voting booths due to long queues and poor facilities. This application can also be used in times of a pandemic to vote from home instead of using a mail-in voting system.

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Problem Definition

Developing a secure online voting system using blockchain technology to increase voter Turnout exponentially.

3.2 Requirements Specification

- **Name of voter:** First name, Middle name and Last name
- **Date of Birth:** Format DD/MM/YY
- **Voter ID:** Unique alphanumeric code
- **Email ID:** Any generic email ID
- **Password:** Any generic password
- **Mobile Number:** 10 digit Indian registered number
- **Gender:** Male, Female and Other
- **Address:** Multiline address matching the one on voter ID

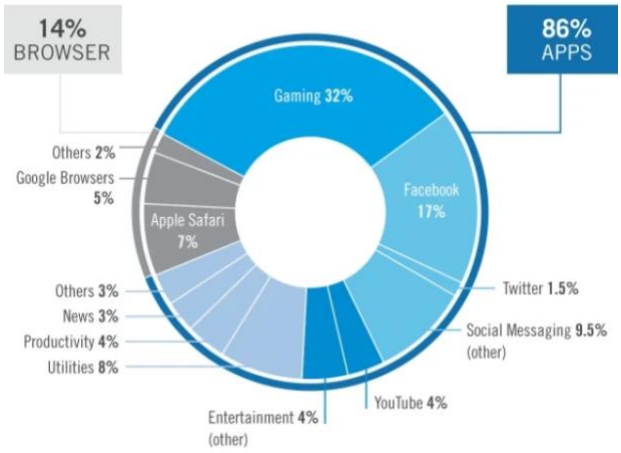
3.3 Planning and Scheduling

Why use Blockchain?

A blockchain is a computer file for storing data. The data is distributed across many computers, and the whole blockchain is entirely decentralized. This means no one person or entity has control over the blockchain; this is a radical departure from the centralized databases that are controlled and administered by businesses and other entities. The file is comprised of blocks of data, with each block being connected to the previous block, forming a chain. As well as the data itself, each block also contains a record of when that block was created or edited, which makes it very useful for maintaining a detailed system of record that cannot be corrupted or lost. There is no central point of control that can be targeted by hackers and it contains a history of activity and not just a snapshot in time.

Why use mobile application instead of a website?

A well-designed mobile app can perform actions much quicker than a mobile website. Mobile websites use JavaScript code to perform most of their functions. And the framework that mobile apps use can run almost five times faster than a JavaScript code.



It is apparent from figure 3.3.1, the general mobile application usage is approximately 5.6 times the browser usage, and the User Interface is easy to understand in a mobile application.

Figure 3.3.1 time spent on iOS and android Connected Devices

3.4 Software and Hardware Requirements

3.4.1 Software Requirements

- **Android 6.0 or above**

  A Smartphone with android version 6.0 or higher is needed to run this application as it requires fingerprint detection technology.

- **Fingerprint scanner**

  The fingerprint scanner is required for authentication to open the voting application.

- **Internet connection**

  Steady internet connection is required for exchange of APIs and to process the votes cast by voters.

- **GSM connectivity**

  GSM connection is required to receive OTP during authentication.

3.4.2Hardware Requirements

- **Android Smartphone**

  Android Smartphone of version 6.0 and higher is necessary to run the voting application.

- **Laptop or Desktop PC**

  A laptop or Desktop PC with an internet connection to act as a sever for the mobile client.

3.5 Preliminary Product Description

This project constituted development of an android application for voting as its major part as well as a web application that is used for registration. Management of any project has several steps in it so, our project can be described under the following steps:

3.5.1 Mapping

In this step, the necessary software and blockchain creating platforms were chosen. Literature survey about similar projects, gathering the information of programming language was carried out. Necessary planning was conducted.

3.5.2 Design

In this phase, we were designing the required models, the basic layout of the system and the features to be included.

3.5.3 Experimentation

In this phase, we experimented with multiple android development environments and database systems.

3.5.4 Development and Testing

In this phase, the development of model was performed. The outliers were identified and removed. Relevant fine tuning is performed by varying the parameters.

3.5.5 Real-World Testing

The system has been tested with a small set of details and needs to be tested on a larger scale.

3.6 Details of Conceptual Modules and their functionalities

3.6.1 Registration

In this phase, the voter has to register either by themselves or by visiting their registered constituency office and register with the help of an official by providing them with the necessary documents, for example their voter ID card, mobile number, login credentials etcetera. They must provide whatever details that they are asked to furnish by the officials with the guarantee of anonymity. The organization must be vary of data breaches and system failure etc.

3.6.2 Pre-Election day

Before the Election Day, details of the upcoming election must be released to every voter and data must be collected on whether the voter will vote on the application or in person.Details of the candidates must be uploaded along with a picture of them and their party symbol to help the voters.

3.6.3 Voting day

On the voting day the voters can use the android application to cast their vote. The voter has to go through three levels of authentication to get access to the voter interface, namely login credentials, fingerprint authentication and a One-time password. The candidates list is released on the previous day so that the voters can become familiar with the voting process. After the vote is cast, the voting interface will display a thank you note and exit. The voter cannot login again after a vote has been cast.

3.6.4 Declaring results

The voting interface is based on a real-time update system. This makes sure that the voters and organizers get a real-time estimate of how the election might proceed. The result interface contains a tabular form which displays the names of the candidates, the constituencies to which they belong and the number of votes they have received. The organizers can set a predetermined time to end the voting day and then declare a final result.

4. SYSTEM DESIGN

4.1 Project Architecture

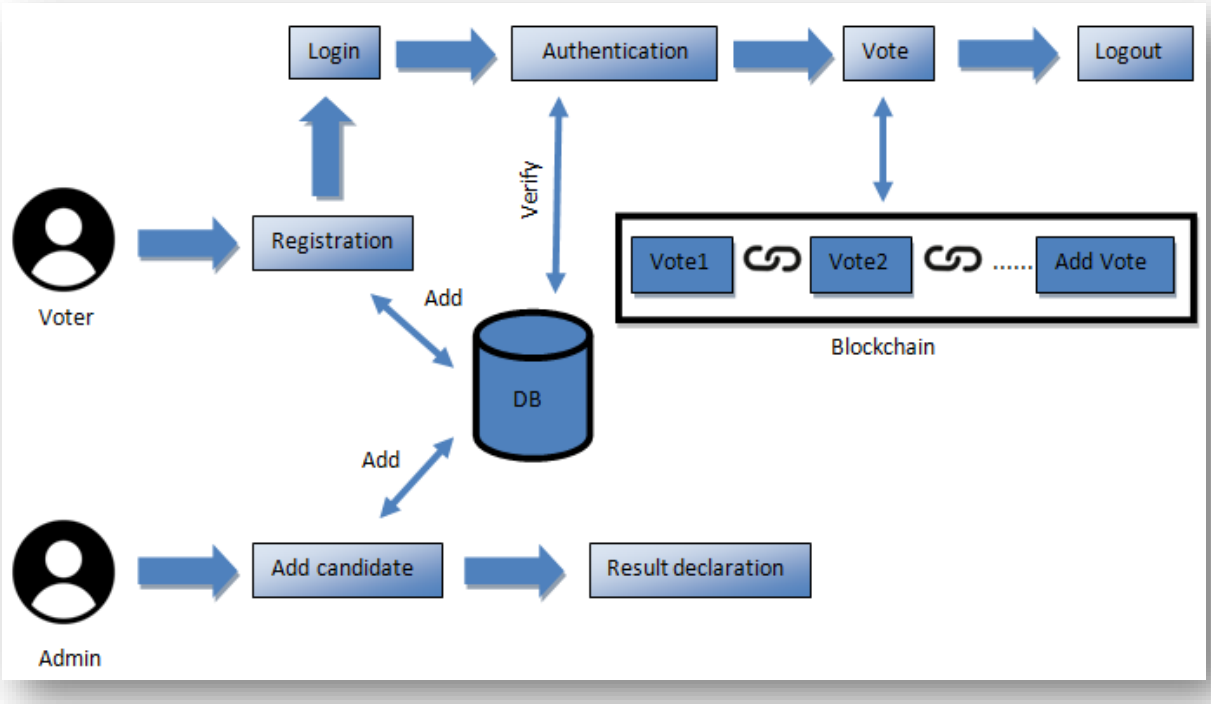


Fig 4.1 Block diagram for voting system

Voting system using blockchain works as shown in block diagram (fig 4.1). Voter has to register to get his/her account for using app. Voter has to create account at the time of registration by providing username and password. Once they login, voter needs to be authenticated at the time of voting so that they can vote. When a voter casts his/her vote, a block will be created and added to the existing blockchain with information about his/her vote which consists of the candidate's code. At the end they can logout. If he/she is found to be a valid voter, they can vote and then further interactions will be blocked.

4.2 Modules Description

4.2.1 Registration

Voter has to register to access voting app. It will be done in nearby centres. He/she need to provide details. He/she can create account by providing email Id and password, these details will be stored in database. This account can be used on the day of election to vote. The registered mobile number is used for OTP authentication.

4.2.2 Login

On the day of election, Voter has to log in using email Id and password. It will be verified against the registered email id and password. If it is not valid, It will show invalid email id or password. For valid user, it will check whether he/she already voted. If yes, he/she will not be able to log in. Otherwise it will go to the next activity. User can click on forgot password to reset the password.

4.2.3 Authentication

- **Fingerprint authentication**

Voter has to be authenticated using fingerprint. Fingerprint authentication is against mobile lock authentication which uses fingerprint. It is local to the mobile phone, to ensure the owner of mobile phone is using this app. Mobile phone fingerprint sensor will be verified at the time of registration. If authenticated user, it will move to the next activity.

- **OTP authentication**

In this activity, One-Time-Password will be sent to the registered mobile number. First user id will be fetched from the database then mobile number will be searched for that user in the database and then OTP will be sent. Once OTP is entered by the user, it will be verified. If valid it will move to next activity. User can click on resend OTP if not received OTP.

4.2.4 Vote

In this activity, voter can vote. Once he/she click on submit button, It will check whether blockchain is created or not. If not, It will create first block then it will create block with vote of voter and then it will be linked to the blockchain. Otherwise, it will create block with vote of voter and then that will be linked to existing block.

4.2.5 Logout

This is the last activity in app. The voter can Log out by clicking sign out button. When this activity opens, It will block the voter for this app, so that he/she should not be able to log in again and vote.

4.2.6 Result

At the end of election, everyone can view the election results in the result form. This form can be viewed on the day of election to know real time changes in the vote. It is also possible to view blockchain in this section.

4.3 Use case diagram

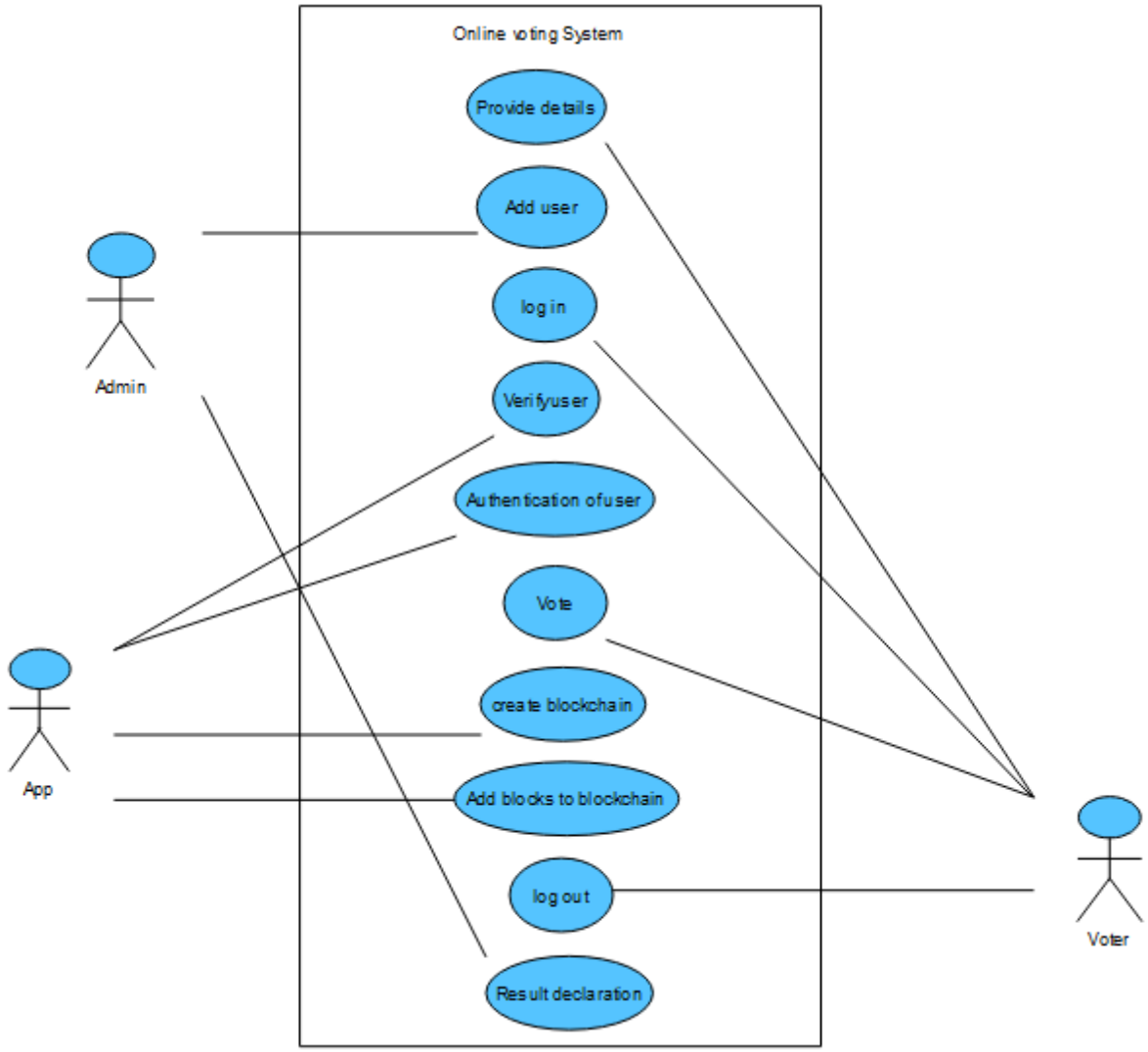


Fig 4.3 Use case diagram of online voting system using blockchain

4.4 Detailed Design

4.4.1 High Level Design

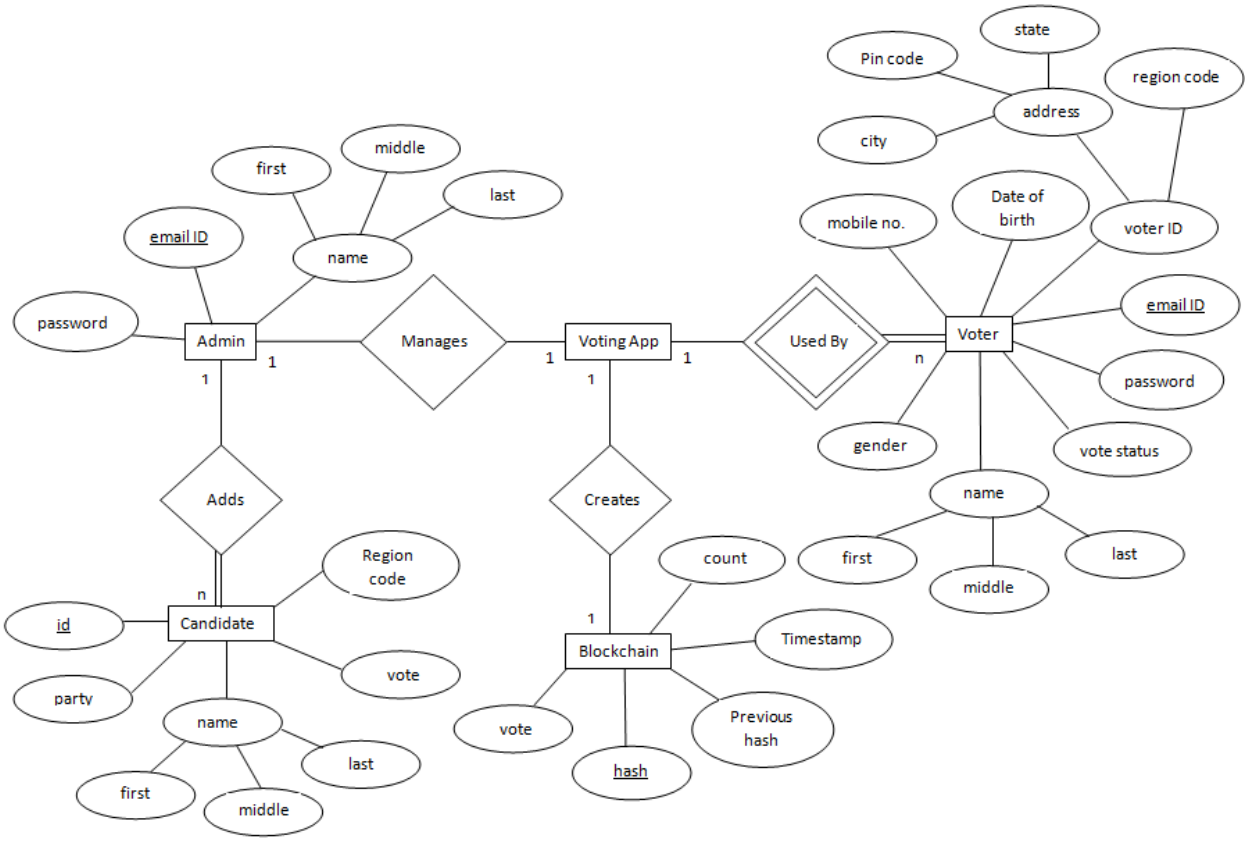


Fig 4.4.1.ER diagram of online voting system using blockchain

4.4.2 Low Level Design

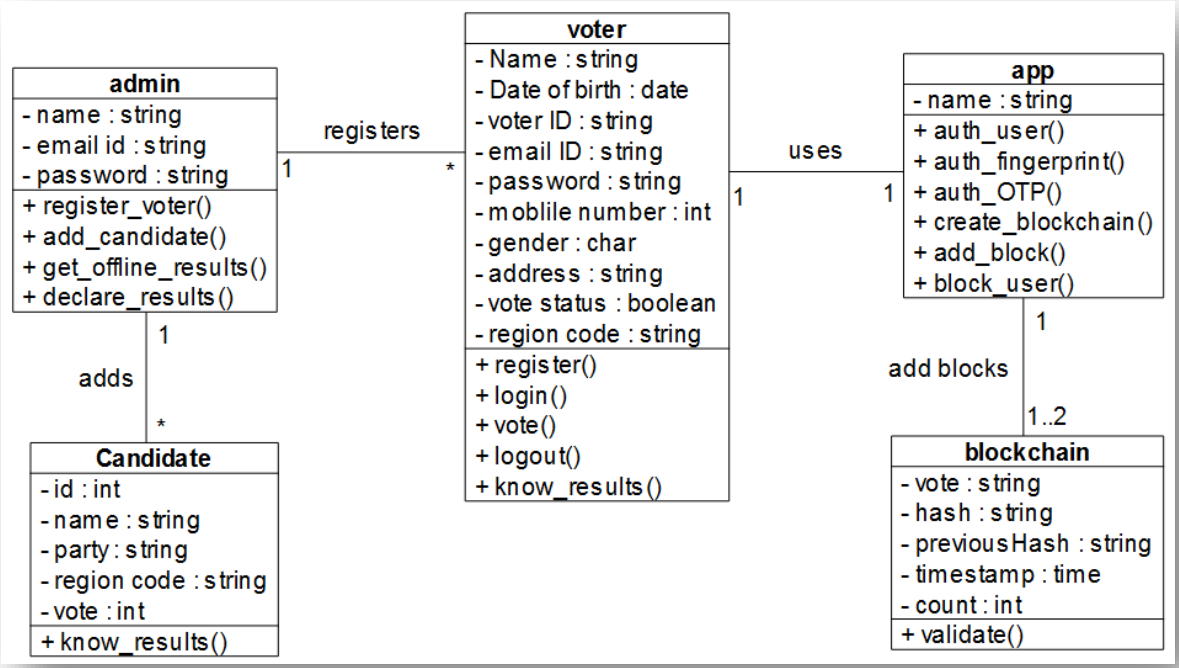


Fig 4.4.2.a class diagram of online voting system using blockchain

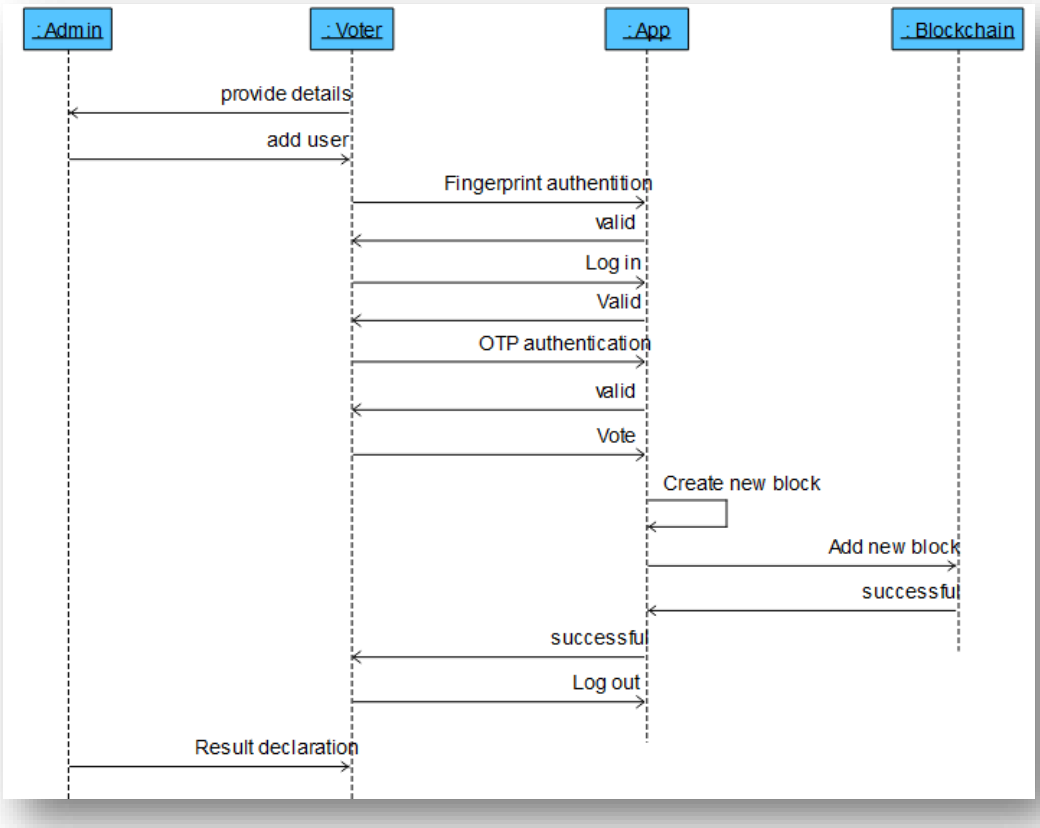


Fig 4.4.2.b sequence diagram of online voting system using

4.5 Pseudo code

- **Registration form**

```
import pyrebase  #import statements
#configuring connecting to database
config= {
   "apiKey": "AIzaSyBi_ISmnHBud-wtv2Nit4PZafDOzianN7g",
   "authDomain": "flaskk-4389a.firebaseapp.com",
   "databaseURL": "https://flaskk-4389a.firebaseio.com",
   "projectId": "flaskk-4389a",
   "storageBucket": "flaskk-4389a.appspot.com",
   "messagingSenderId": "499576188020",
   "appId": "1:499576188020:web:074057a6721ea9957a3521",
   "measurementId": "G-DS2BGNMNDV"}
#creating connection to data base
firebase= pyrebase.initialize_app(config)
#creating connection object
db=firebase.database()
from flask import *
from flask import render_template
import re
#app = Flask(__name__)
#connecting the html template folder link
app = Flask(__name__,
template_folder='C:\\Users\\PREDATOR\\Desktop\\Project\\Reg\\templates')
app._static_folder = 'C:\\Users\\PREDATOR\\static'
#start page after website gets hosted
@app.route('/')
def start():
   return render_template('start.html')
```

```
#HTML registration page
#collecting the data from HTML page n storing it to the cloud
@app.route('/register',methods=['GET','POST'])
def register():
    if request.method=='POST':
        fname= request.form['First_Name']
        mname= request.form['Middle_Name']
        lname= request.form['Last_Name']
        b_day= request.form['Birthday_day']
        email= request.form['Email_Id']
        password= request.form['Password']
        re_password= request.form['Re-Password']
        mob= request.form['Mobile_Number']
        gender= request.form['Gender']
        address= request.form['Address']
        voterid= request.form['Voter_Id']
        region= request.form['Region']
        city= request.form['City']
        pincode= request.form['Pin_Code']
        state= request.form['State']
        country= request.form['Country']
        db.child(fname+" "+lname).push({'First Name':fname,'Middle Name':mname,'Last Name':lname,'Date of birth':b_day,'Email':email,'Password':password, 're-entered password':re_password,'Mobile':mob,'Gender':gender,'Voter Id':voterid,'Regional code':region,'Address':address,'City':city,'Pincode': pincode,'City':city,'Country':country})
    return render_template('reg2.html')
#page after registration is successful
@app.route('/reg_suc',methods=['GET'])
def reg_suc():
    return render_template('reg_suc.html')
```

❖ **Fingerprint authentication on android app**

```
public class FingerprintActivity1 extends AppCompatActivity {
    ImageView fingerprintImage;
    TextView headingLabel,paraLabel;
    FingerprintManager fingerprintManager;
    KeyguardManager keyguardManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ……
        headingLabel = (TextView) findViewById(R.id.headingLabel);
        fingerprintImage = (ImageView) findViewById(R.id.fingerprintImage);
        paraLabel = (TextView) findViewById(R.id.paraLabel);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
     fingerprintManager = (FingerprintManager) getSystemService(FINGERPRINT_SERVICE);
            keyguardManager= (KeyguardManager) getSystemService(KEYGUARD_SERVICE);
            if (!fingerprintManager.isHardwareDetected()) {
                paraLabel.setText("Fingerprint Scanner not detected in device");
            }
else if (ContextCompat.checkSelfPermission(this, Manifest.permission.USE_FINGERPRINT) !=
PackageManager.PERMISSION_GRANTED) {
                paraLabel.setText("permission not granted to use Fingerprint Scanner");
            } else if (!keyguardManager.isKeyguardSecure()) {
                paraLabel.setText("Add Lock to your Phone in settings");
            } else if (!fingerprintManager.hasEnrolledFingerprints()) {
                paraLabel.setText("You should add atleast 1 fingerprint to access this app");
            } else {
                paraLabel.setText("Place your finger on Scanner to Access the App");
                FingerprintHandler fingerprintHandler = new FingerprintHandler(this);
                fingerprintHandler.startAuth(fingerprintManager,null);
            }
        }
```

```
    }
}
```

❖ **Login activity on android app**

```
public class LoginActivity2 extends AppCompatActivity {
    ……
    private FirebaseAuth mAuth;
    private FirebaseAuth.AuthStateListener mAuthStateListener;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ……..
        mAuth = FirebaseAuth.getInstance();
        e = (EditText) findViewById(R.id.editText);
        p = (EditText) findViewById(R.id.editText2);
        view.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                final progressButton progressButton=new progressButton(LoginActivity2.this,view);
                progressButton.buttonActivated();
                email = e.getText().toString();
                password = p.getText().toString();
                if(email.isEmpty())
                {
                    e.setError("Please enter email id");
                    e.requestFocus();
                    progressButton.buttonFinished();
                }
                else if(password.isEmpty(){ ……}
                else if(email.isEmpty() && password.isEmpty())
                {
                    Message.message(getApplicationContext(), "Feilds are empty");
                    progressButton.buttonFinished();
                }
```

```
 else if(!(email.isEmpty() && password.isEmpty())) {
    mAuth.signInWithEmailAndPassword(email, password)
.addOnCompleteListener(LoginActivity2.this, new OnCompleteListener<AuthResult>()
 {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
              if (task.isSuccessful()) {
                String reg = "[@]";
                String[] res = email.split(reg);
           DatabaseReference database = FirebaseDatabase.getInstance().getReference();
                DatabaseReference myRef = database.child("users").child(res[0]);
                myRef.addValueEventListener(new ValueEventListener()
                {
                  @Override
                  public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
             int n = Integer.parseInt(dataSnapshot.child("status").getValue().toString());
                    if(n==0){
               Message.message(getApplicationContext(), "signInWithEmail:success");
                   Intent intent = new Intent(LoginActivity2.this, OTPActivity3.class);
                      startActivity(intent);
                      textview.setText("");
                    }
                    else
                    {
               textview.setText("You can't login because you have voted successfully");
                      FirebaseAuth.getInstance().signOut();
                      progressButton.buttonFinished();
                    }
                  }
                  @Override
                  public void onCancelled(DatabaseError databaseError) {
```

```
                        }
                    });
                }
                else{
                    Message.message(getApplicationContext(), "signInWithEmail:failure");
                    textview.setText("Invalid username or password");
                    e.setText("");
                    p.setText("");
                    progressButton.buttonFinished();
                }
            }
        });
    }
  }
});
forgotpass=(TextView) findViewById(R.id.textView13);
forgotpass.setOnClickListener(new View.OnClickListener() {
  public void onClick(View v) {
    email = e.getText().toString();
    if(email.isEmpty())
    {
      e.setError("Please enter email id");
      e.requestFocus();
    }
    else
    mAuth.sendPasswordResetEmail(email)
      .addOnCompleteListener(new OnCompleteListener<Void>() {
      @Override
      public void onComplete(@NonNull Task<Void> task) {
        if (task.isSuccessful()) {
```

```
                    Message.message(getApplicationContext(), "We have sent you instructions to reset your password!");
                } else {
                    Message.message(getApplicationContext(),"Failed to send reset email!");
                }
            }
        });
    }
});
}
@Override
public void onBackPressed()
{
    Message.message(getApplicationContext(),"You can't go back");
}
}
```

❖ **OTP authentication activity on android app**

```
public class OTPActivity3 extends AppCompatActivity {
……
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ……..
        textview = (TextView) findViewById(R.id.textView9);
        progressBar = findViewById(R.id.progressbar);
        mAuth = FirebaseAuth.getInstance();
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        if (user != null) {
            email = user.getEmail();
        }
        String reg = "[@]";
        String[] res = email.split(reg);
```

```
e=res[0];
DatabaseReference database = FirebaseDatabase.getInstance().getReference();
DatabaseReference myRef = database.child("users").child(res[0]);
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        n = dataSnapshot.child("no").getValue().toString();
        sendVerificationCode("+" + n);
        textview.setText("OTP sent successfully " );
        view=(TextView) findViewById(R.id.textView8);
        textview.setSelected(true);
        view.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (i < 4) {
                    sendVerificationCode("+" + n);
                     textview.setText("OTP sent successfully " );i++;
                } else {
                    textview.setText("You have reached maximum number of attempts ");
                    FirebaseAuth.getInstance().signOut();
                    Intent intent = new Intent(OTPActivity3.this, FingerprintActivity1.class);
                    startActivity(intent);
                }
            }
        });
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {
    }
});
Button button = (Button) findViewById(R.id.button3);
```

```
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                EditText e = (EditText) findViewById(R.id.editText3);
                String code = e.getText().toString().trim();
                if ((code.isEmpty() || code.length() < 6)) {
                    e.setError("Enter code...");
                    e.requestFocus();
                    return;
                }
                verifyCode(code);
            }
        });
    }
    …..
    private void verifyCode(String code) {
        PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationid, code);
        signInWithCredential(credential);
    }
    private void signInWithCredential(PhoneAuthCredential credential) {
        mAuth.signInWithCredential(credential)
                .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            Intent intent = new Intent(OTPActivity3.this, VoteActivity4.class);
                            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                            startActivity(intent);
                        } else {
                            if(!connectionManager.checkConnection(getBaseContext()))
```

```
                    {
                        FirebaseAuth.getInstance().signOut();
                        Intent i = new Intent(OTPActivity3.this, noconnection.class);
                        startActivity(i);
                    }
                    Toast.makeText(OTPActivity3.this, task.getException().getMessage(),
Toast.LENGTH_LONG).show();
                }
            }
        });
    }
    private void sendVerificationCode(String number){
        PhoneAuthProvider.getInstance().verifyPhoneNumber(
                number,
                60,
                TimeUnit.SECONDS,
                TaskExecutors.MAIN_THREAD,
                mCallBack
        );
    }
    private PhoneAuthProvider.OnVerificationStateChangedCallbacks
            mCallBack = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
        @Override
        public void onCodeSent(String s, PhoneAuthProvider.ForceResendingToken
forceResendingToken) {
            super.onCodeSent(s, forceResendingToken);
            verificationid = s;
        }
        @Override
        public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential) {
            String code = phoneAuthCredential.getSmsCode();
```

```
        if (code != null){
            progressBar.setVisibility(View.VISIBLE);
            verifyCode(code);
        }
    }
    @Override
    public void onVerificationFailed(FirebaseException e) {
        Toast.makeText(OTPActivity3.this, e.getMessage(),Toast.LENGTH_LONG).show();
    }
  };
}
```

- **Vote activity on android app**

```
public class VoteActivity4 extends AppCompatActivity {
  RadioGroup vote;
  ……
  Blockchain blockchain = new Blockchain(4);
  @Override
  protected void onCreate(Bundle savedInstanceState) {
……
    vote=findViewById(R.id.vote);
    email=OTPActivity3.e;
    database = FirebaseDatabase.getInstance();
    myRef= database.getReference("users").child(email).child("status");
    Button button = (Button) findViewById(R.id.button4);
    button.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        if(!connectionManager.checkConnection(getBaseContext()))
        {
          FirebaseAuth.getInstance().signOut();
          Intent i = new Intent(VoteActivity4.this, noconnection.class);
          startActivity(i); }
```

```
            int checkedId=vote.getCheckedRadioButtonId();
            if(checkedId == -1){
                Message.message(getApplicationContext(),"Please cast your vote");
            }
            else{
                findRadioButton(checkedId);
                myRef.setValue(Integer.toString(1));
                Intent intent = new Intent(VoteActivity4.this, LogoutActivity5.class);
                startActivity(intent);
            }
        }
    });
    vote.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){
        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            switch(checkedId)
            {
                case R.id.radioButton1:Message.message(getApplicationContext(),"you have
selected P1 to vote");
                    break;
                case R.id.radioButton2:Message.message(getApplicationContext(),"you have
selected P2 to vote");
                    break;
                case R.id.radioButton3:Message.message(getApplicationContext(),"you have
selected P3 to vote");
                    break;
            } }
    });}
    private void findRadioButton(int checkedId) {
        switch(checkedId)
        {
```

```
        case R.id.radioButton1:Message.message(getApplicationContext(),"you have successfully
voted to P1");
            blockchain.addBlock(blockchain.newBlock("Vote:P1"));
            break;
        case R.id.radioButton2:Message.message(getApplicationContext(),"you have successfully
voted to P2");
            blockchain.addBlock(blockchain.newBlock("Vote:P2"));
            break;
        case R.id.radioButton3:Message.message(getApplicationContext(),"you have successfully
voted to P3");
            blockchain.addBlock(blockchain.newBlock("Vote:P3"));
            break;
    }
 }……
}
```

- **Blockchain implementation**

```
public class Blockchain {
   private int difficulty;
   FirebaseDatabase database;
   DatabaseReference myRef;
   int count;
   String ph;
   public Blockchain(final int difficulty) {
      this.difficulty = difficulty;
      database = FirebaseDatabase.getInstance();
      myRef = database.getReference("blockchain").child("count");
      myRef.addValueEventListener(new ValueEventListener() {
         @Override
         public void onDataChange(DataSnapshot dataSnapshot) {
            count = Integer.parseInt(dataSnapshot.getValue().toString());
            if(count==0)
```

```
            { // create the first block
                Block b = new Block(count, System.currentTimeMillis(), null, "First Block");
                b.mineBlock(difficulty);
            }
        }
        @Override
        public void onCancelled(DatabaseError error) {
        }
    });
}
public int getDifficulty() { return difficulty; }
public Block setPreviousHash(int n, final String data)
{
    final Block[] b = new Block[1];
    database = FirebaseDatabase.getInstance();
    myRef = database.getReference("blockchain").child("block "+n);
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            ph = dataSnapshot.child("hash").getValue().toString();
           b[0] =new Block(count+ 1, System.currentTimeMillis(),
                    ph, data);
        }
        @Override
        public void onCancelled(DatabaseError error) {
        }
    });
    return b[0];
}
public Block newBlock(String data) {
    Block b=setPreviousHash(count,data);
```

```
        return b;
    }
    public void addBlock(Block b) {
        if (b != null) {
            b.mineBlock(difficulty);
        }
    }
}
public class Block {
    private int index;
    private Long timestamp;
    private String hash;
    private String previousHash;
    private String data;
    private int nonce;
    FirebaseDatabase database;
    DatabaseReference myRef;
    public Block(int index, Long timestamp, String previousHash, String data) {
        this.index = index;
        this.timestamp = timestamp;
        this.previousHash = previousHash;
        this.data = data;
        nonce = 0;
        hash = Block.calculateHash(this);
        database = FirebaseDatabase.getInstance();
        myRef = database.getReference("blockchain").child("block " + index);
        if(previousHash==null){
        myRef.child("previousHash").setValue("null");}
        else
            myRef.child("previousHash").setValue(previousHash);
        myRef.child("timestamp").setValue(String.valueOf(new Date(timestamp)));
```

```
        myRef.child("data").setValue(data);
        myRef.child("hash").setValue(hash);
        myRef = database.getReference("blockchain").child("count");
        myRef.setValue(Integer.toString(index));
    }
    public int getIndex() { return index;}
    public Long getTimestamp() { return timestamp;}
    public String getHash() { return hash;}
    public String getPreviousHash(int n) {return previousHash;}
    public String getData() { return data;}
    public String str() {return index + timestamp + previousHash + data + nonce; }
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("Block #").append(index).append(" [previousHash : ").append(previousHash).append(", ").
                append("timestamp : ").append(new Date(timestamp)).append(", ").append("data : ").append(data).append(", ").
                append("hash : ").append(hash).append("]");
        return builder.toString();
    }
    public static String calculateHash(Block block) {
        if (block != null) {
            MessageDigest digest = null;
            try {
                digest = MessageDigest.getInstance("SHA-256");
            } catch (NoSuchAlgorithmException e) {
                return null;
            }
            String txt = block.str();
            final byte bytes[] = digest.digest(txt.getBytes());
            final StringBuilder builder = new StringBuilder();
```

```
        for (final byte b : bytes) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) {
                builder.append('0');
            }
            builder.append(hex);
        }
        return builder.toString();
    }
    return null;
  }
  public void mineBlock(int difficulty) {
    nonce = 0;
    while (!getHash().substring(0,  difficulty).equals(Utils.zeros(difficulty))) {
        nonce++;
        hash = Block.calculateHash(this);
    }
  }
}
```

- **Logout activity on android app**

```
public class LogoutActivity5 extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_logout5);
    Button button = (Button) findViewById(R.id.button5);
    button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            FirebaseAuth.getInstance().signOut();
            Intent intent = new Intent(LogoutActivity5.this, FingerprintActivity1.class);
            startActivity(intent);
```

```
        }
      });
   }
   @Override
   public void onBackPressed()
   {
      Message.message(getApplicationContext(),"You can't go back");
   }
}
```

- **Result page**

```
#HTML page to display the result in raw format
@app.route('/result',methods=['GET'])
def result():
   res_con= {
   "apiKey": "AIzaSyCxxmQZ5-icpXiI_R3eBq05hOewzjESc3o",
   "authDomain": "my-application-fc670.firebaseapp.com",
   "databaseURL": "https://my-application-fc670.firebaseio.com",
   "projectId": "my-application-fc670",
   "storageBucket": "my-application-fc670.appspot.com",
   "messagingSenderId": "740838641986",
   "appId": "1:740838641986:web:1edb74c8ac061dc690b8db"}
   firebase2= pyrebase.initialize_app(res_con)
   db2=firebase2.database()
   if request.method == 'GET':
      res = db2.child("blockchain").get()
      result = res.val()
      return render_template('result.html', t=result.values())
   return render_template('result.html')
#HTML page to display the result in tabular form
@app.route('/restab',methods=['GET'])
def restab():
```

```
    res_con= {
    "apiKey": "AIzaSyCxxmQZ5-icpXiI_R3eBq05hOewzjESc3o",
    "authDomain": "my-application-fc670.firebaseapp.com",
    "databaseURL": "https://my-application-fc670.firebaseio.com",
    "projectId": "my-application-fc670",
    "storageBucket": "my-application-fc670.appspot.com",
    "messagingSenderId": "740838641986",
    "appId": "1:740838641986:web:1edb74c8ac061dc690b8db"}
    firebase2= pyrebase.initialize_app(res_con)
    db2=firebase2.database()
    if request.method=='GET':
        res = db2.child("blockchain").get()
        z = res.val()
        vp1=0
        vp2=0
        vp3=0
        for j in z.values():
            x = re.search("P1", str(j))
            if(x):
                vp1+=1
            y = re.search("P2", str(j))
            if(y):
                vp2+=1
            z = re.search("P3", str(j))
            if(z):
                vp3+=1
    print(vp1,vp2,vp3)
    return render_template('restab.html',p1=vp1,p2=vp2,p3=vp3)
if __name__=="__main__":
    app.run(debug=True,use_reloader=False)
```

5. SYSTEM IMPLEMENTATION AND TESTING

5.1 Implementation Approaches of Tools and Technologies

The technologies used in this project are Python, Web Development and Android. Tools used in Python are

- **Pyrebase**: This package is used to connect directly to the Google Firebase and can communicate with it.
- **Flask**: This package is used to create a web localhost and displays out HTML and CSS file. The python backend acts as the server. Backend python server can be used to store, process and display the output.

Tools used in Web Development are

- **HTML**: The skeleton form of our web page.
- **CSS**: The make the web page more colorful and attractive.
- **Javascript**: Used to validate the input which user gives.

Tools used in Android are

- **Fingerprint Auth**: This package is used for the fingerprint Authentication.
- **Blockchain**: The blockchain code is used to create a block. It needs a strong processor to compute the block.

5.2 Coding Details and Code Efficiency

5.2.1 Class name:FingerprintActivity1

- **Methods:**
  - onCreate(): It will be invoked when FingerprintActivity1 is created.
    - Input: fingerprint data
    - Output: valid user or not.
- **Functionality:** This class will authenticate the user by checking fingerprint with local fingerprint lock of mobile phone.

5.2.2 Class name: FingerprintHandler

❖ **Methods:**

- FingerprintHandler(): It will handle context of the activity.
  - Input: context
  - Output: It will assign context
- startAuth(): It will start authentication of the user.
  - Input: fingerprintManager, cryptoObject
  - Output: It will call fingerprintManager.authenticate()
- onAuthenticationError(): It will be invoked whenever error occurs.
  - Input: errorCode, errString
  - Output: It will print why error occurred
- onAuthenticationFailed(): It will be invoked when authentication fails.
  - Output: Authentication Failed
- onAuthenticationHelp(): It will be invoked when there is need of help.
  - Input: helpCode, helpString
  - Output: help message
- onAuthenticationSucceeded():It will be invoked when authentication is successful.
  - Input: result
  - Output: You can access the app
- update(): It will be invoked when above function calls.
  - Input: s(string message),b(Boolean value)
  - Output: update text on textView

❖ **Functionality:** This class will handle the output message based on fingerprint authentication.

5.2.3 Class name: LoginActivity2

❖ **Methods:**

- onCreate():It will be invoked when LoginActivity2 is created.
  - Input: email Id and password
  - Output: valid user or not.

- onBackPressed(): It will be invoked when user presses back button.
    - Output: You can't go back

❖ **Functionality:** This class with authenticate user with email Id and password.

5.2.4 Class Name: OTPActivity3

❖ **Methods:**

- onCreate():It will be invoked when OTPActivity3 is created.
    - Input: OTP
    - Output: valid user or not.
- onBackPressed(): It will be invoked when user presses back button.
    - Output: You can't go back
- verifyCode(): It will be invoked to verify OTP code.
    - Input: code
    - Output: invoking signInWithCredential()
- signInWithCredential(): It will be invoked to sign in with registered mobile number.
    - Input: credential
    - Output: sign in
- sendVerificationCode(): It will be invoked to send OTP to user.
    - Input: number
    - Output: OTP will be sent to registered mobile number
- onVerificationFailed(): It will be invoked when authentication fails.
    - Input: e
    - Output: exception message

❖ **Functionality:** This class with authenticate user with OTP.

5.2.5 Class Name: VoteActivity4

❖ **Methods:**

- onCreate():It will be invoked when VoteActivity4 is created.
    - Input: vote
    - Output: adding vote to blockchain
- findRadioButton(): It will be invoked to find the radio button checked.

- Input: checkedId
- Output: checked radio button

- onBackPressed():It will be invoked when user presses back button.
  - Output: You can't go back

❖ **Functionality:** This class will get vote from the user and add that to blockchain.

5.2.6 Class Name: Blockchain

❖ **Methods:**

- Blockchain(): It will be invoked when object of Blockchain class is created.
  - Input: difficulty
  - Output: initialize default values to class attributes
- getDifficulty(): It will be invoked when there is a need of Difficulty.
  - Output: difficulty
- setPreviousHash():It will be invoked to set previous Hash value.
  - Input: n, data
  - Output: block
- newBlock(): It will be invoked to create new block.
  - Input: data
  - Output: block
- addBlock(): It will mine the block.
  - Input: b

❖ **Functionality:** This class will create blockchain by using block class.

5.2.7 Class Name: Block

❖ **Methods:**

- Block(): It will be invoked when object of Block class is created.
  - Input: index, timestamp, previousHash, data
  - Output: initialize values to class attributes
- getIndex(): It will be invoked to get Index.
  - Output: Index
- getTimestamp():It will be invoked to get Index.
  - Output: Timestamp

- getHash(): It will be invoked to get Index.
    - Output: Hash
- getPreviousHash(): It will be invoked to get Index.
    - Output: previousHash
- getData(): It will be invoked to get Index.
    - Output: Data
- str(): It will be invoked to merge the attributes.
    - Output: index + timestamp + previousHash + data + nonce
- calculateHash(): It will be invoked to calculate Hash value.
    - Input: block
    - Output: Hash(hex value)
- mineBlock(): It will be invoked to mine the block.
    - Input: difficulty
    - Output: hash

❖ **Functionality:** This class will create block for blockchain with vote of the user.

5.2.8 Class Name: LogoutActivity5

❖ **Methods:**

- onCreate():It will be invoked when LogoutActivity5 is created.
    - Input: email Id
    - Output: signout
- onBackPressed():It will be invoked when user presses back button.
    - Output: You can't go back

❖ **Functionality:** This class will logout user from the voting app.

6. EXPERIMENTATION AND RESULT ANALYSIS

6.1 Experimental Results

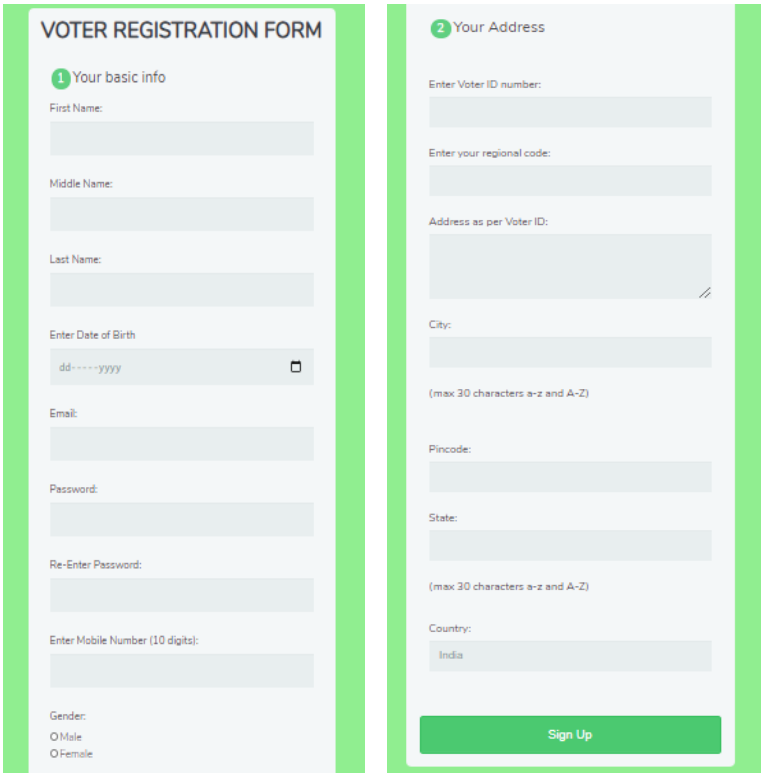


Fig 6.1.1 Registration form

Fig 6.1.2 Fingerprint auth. activity

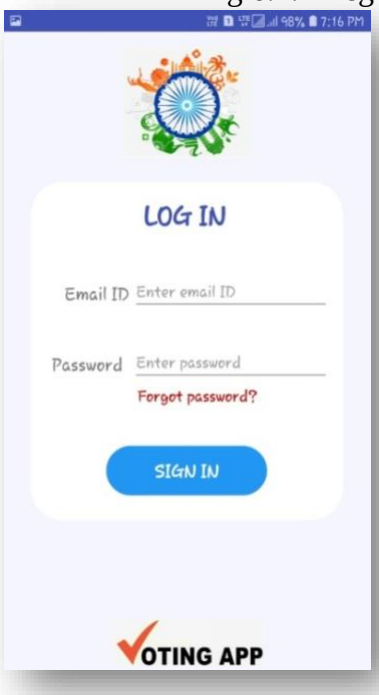


Fig 6.1.3 Login activity

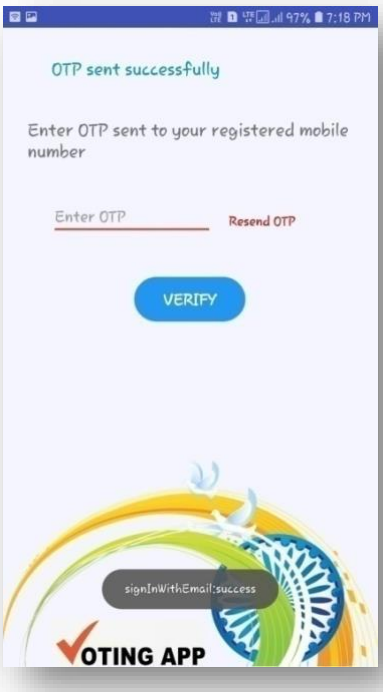


Fig 6.1.4 OTP auth. activity

Fig 6.1.5 Vote activity

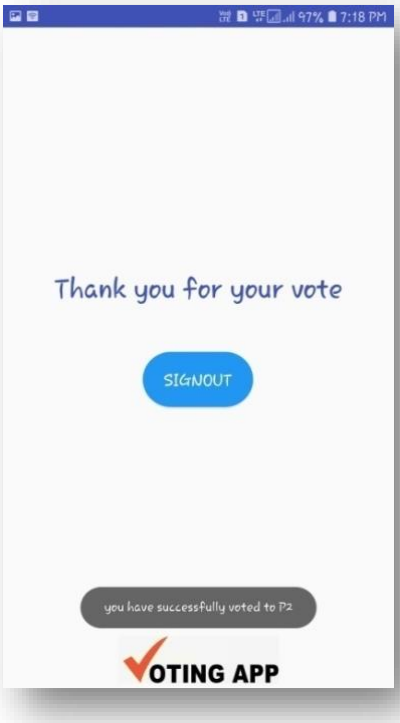


Fig 6.1.6 Logout activity

Fig 6.1.7 Raw Blockchain

Fig 6.1.8 Election result form

6.2 Analysis of the Result with the help of Graphs and Plots

The result is as we've expected.

The analysis of the result with the help of graphs and plots are not applicable.

7. CONCLUSION AND FUTURE SCOPE

The idea of adapting digital voting system 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 and puts a certain amount of pressure on the elected official.

Blockchain based electronic voting system utilizes smart contracts to enable secure and cost-efficient election while guaranteeing voters privacy. There is a need of future work on quality standards available tools for resource allotment and determining legal framework of blockchain implementation on governance.

BIBLIOGRAPHY

[1] https://en.wikipedia.org/wiki/Blockchain

[2] Hye Ri Kim, Kyoungsik Min and Seng-philHong , "A Study on Ways to Apply the Blockchain-based Online Voting System"(International Journal of Control and Automation 10(12):121-130 ,2017).

[3] G Bhavani , "Survey on Blockchain Based E-Voting Recording System Design"(IJIRSET Volume 7, Issue 11, November 2018)

[4] HaiboYi, "Securing e-voting based on blockchain in P2P network"(EURASIP Journal on Wireless Communications and Networking 2019(1) · December 2019)

[5] Kashif Mehboob Khan, Junaid Arshad, Muhammad Mubashir Khan, "Secure Digital Voting System based on Blockchain Technology (JournalInternational Journal of Electronic Government ResearchVolume 14 Issue 1,Pages 53-62, January 2018).

[6] Snehal Kadam, Khushaboo Chavan, Ishita Kulkarni, Prof. Amrut Patil, "Survey on Digital E-Voting System by using Blockchain Technology"(Volume 4 || Issue 2 || February 2019 || ISSN (Online) 2456-0774Journal International Journal of Electronic Government Research)

[7] Ganaraj K, "Advanced E-Voting application using android platform"(International Journal of Computer- Aided Technologies (IJCAx) Vol.4, No.1/2, April 2017)

[8] Ahmed Ben Ayed, "A conceptual secure blockchain-based electronic voting system"(International Journal of Network Security & Its Applications (IJNSA) Vol.9, No.3, May 2017)

[9] SmitaKhairnar, Reena Kharat, "Survey on Secure Online Voting System"(International Journal of Computer Applications (0975 – 8887) Volume 134 – No.13, January 2016)

[10] Octavian DOSPINESCU, Iulian POPA, "Face Detection and Face Recognition in Android Mobile Applications"(Informatica Economică vol. 20, no. 1/2016)

[11] CharalamposDoukas, IliasMaglogiannis, "A Fast Mobile Face Recognition System for Android OS Based on Eigenfaces Decomposition"(International Conference on Artificial Intelligence Applications and Innovations (AIAI), Oct 2010)

[12] Shruti Thakkar, Nisha Pawar, Nikita Sarang, Prof.VijaylaxmiKadrolli, "Online Voting System (Android Application)"(IOSR-JCE, Volume 18, Issue 2, Ver. II, Mar-Apr. 2016)

[13] Himika Parmar, Nancy Nainan and SumaiyaThaseen, "Generation of secure one-time password based on image authentication"(ITCSE, CS & IT 07, pp. 195–206,2012)

[14] NoorazlinaMohamid Salih, Muhamad ZulfikreNorsid, ShareenAdlina Shamsuddin, "A Design of Voting System by Using an Android Apps for Fisherman"(ARPN Journal of Engineering and Applied Sciences, VOL. 12, NO. 6, MARCH 2017)

[15] Ms. Anagha Hegde, Ms. Chetana Anand, Ms. Jyothi B,"Mobile Voting System"(International Journal of Science, Engineering and Technology Research, Volume 6, Issue 4, April 2017, ISSN: 2278 -7798 2017)

[16] Sreerag M, Subash R, Vishnu C Babu, Sonia Mathew, Reni K Cherian, "Aadhaar Based Voting System Using Android Application"(International Research Journal of Engineering and Technology, Volume: 05 Issue: 04 | Apr-2018)

[17] Vaibhav Anasune, Pradeep Choudhari, Madhura Kelapure, PranaliShirke Prasad, "Literature survey- Online Voting: Voting System Using Blockchain"(International Research Journal of Engineering and Technology, Volume: 06 Issue: 06 | Apr-2018)

[18] Faaiz Ahmad, Mubeen Rafay, Kumail Haider, "Mobile Voting System"(, International Journal of Computer Science and Mobile Computing, Vol.7 Issue.1, January- 2018, pg. 13-17)

[19] Sagar Shah, QaishKaish, Huann Mi, "Block Chain Voting System"(IEEE 2018)

[20] Asad Razzaq, Muhammad Murad Khan, Ramzan Talibm,"Use of Blockchain in Governance: A SystematiLiteratureReview"(International Journal of Advanced Computer Science and Applications, Vol. 10, No. 5, 2019)

[21] Friðrik Þ. Hjálmarsson, Gunnlaugur K. Hreiðarsson, "Blockchain-Based E-Voting System"(IEEE 2017)

[22] Monica Chillaron, Larisa Dunai, Guillermo Peris Fajarnes, Ismael LenguaLengua, "Face detection and recognition application for Android Blockchain-Based E-Voting System"(IECON2015-Yokohama ,November 2015)

[23] Andrew Barnes, Christopher Brake and Thomas Perry","Digital Voting with the use of Blockchain Technology"(International Research Journal of Engineering and Technology, Volume: 07 Issue: 07 | March-2018)

[24] Gaby G. Dagher, PraneethBabu Marella, MateaMilojkovic and Jordan Mohler, "Bronco Vote: Secure Voting System using Ethereum's Blockchain"(4th International Conference on Information Systems Security and Privacy (ICISSP 2018), pages 96-107,2018)

[25] Sayantan Sarkar, Vishal M. Patel and Rama Chellappa,"Deep Feature-based Face Detection on Mobile Devices"(IEEE November 2016)

[26] Aesha Shah, Kavin Shah, Vidhi Shah, Chintan Shah,"Built-in Face Recognition for Smart Phone Devices" International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 04 | Apr -2017)