A Project Report on

Android Voting App

Submitted in partial fulfilment for the award of the degree of

B.Tech (Computer Science and Engineering)
By

Rimjhim Singh, 18BCI0150
Ishan Rathore, 18BCI0144
Aayush Mishra, 17BCE2322

Under the guidance of
Dr. Sweta Bhattacharya
Assistant Professor (Senior)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
November 2020

Table of Contents

Abstract

Voting is a process of choosing or electing someone from a list of candidates. The main goal of voting is to come up with the leader of people's choice.
With the increase in use of digital technology in every sector we are aim to bring the same to the voting sector too. As everyone has mobile-phone these days, we plan to make the voting system available to all using android programming by developing a mobile application to facilitate the voting. This will also allow critical analysis of the logical and functional aspects of the design before any commitment is made to actual code. Online Android voting system tool is a tool designed as a prototype to demonstrate the functionality of Pailler Threshold Cryptosystem (PTC). We have also considered some additional security concerns during the design process.

Acknowledgement

We would like to express our special thanks of gratitude to our faculty Dr. Sweta Bhattacharya for giving us an opportunity to work on a project that helped us understand better and apply the concepts that we have been learning. It further helped us explore novel concepts beyond the traditional curriculum. We would like to thank her for the continuous and constant support for helping us with the project throughout the semester as well as our Chancellor Dr. G Viswanathan who gave us the golden opportunity of doing this wonderful project on the topic Voting App, which also helped us in doing a lot of research and we came to know about so many new things we am really thankful to them.

Also, we would like to thank our respected Dean for the opportunity to carry out this interesting and educational project.

We would also like to thank Vellore Institute of Technology for giving us an opportunity to carry out this project.

Secondly we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

We hope that this project lives up to the professors' expectations.

1. Introduction

1.1. Motivation

The software produced will be an online voting system. Voting App is a system which enables all citizens to cast their vote online. The purpose is to increase the voting percentage across the country, as in the present system people have to visit the booth to cast their vote and those people who live out of their home town are not able to cast vote during the elections. So due to this the voting percentage across the country is very less. Through this software those people who live out of their home town will also be able to cast their votes as this system is online.

1.2. Aim of the proposed work

The main aim behind building this portal is to make the whole voting process more simple and transparent. This will also come as a help to the people who have moved out of their voting constituency because of work and can't cast votes now.

1.3. Objective of the proposed work

The main objective of this software is to increase the overall voting percentage. As the voting databases contain highly sensitive information, special care will be given to the security part. The voter's credentials will be properly anonymized and hashed. Also the database containing the results will be properly encrypted using different encryption algorithms. We also see the cases of booth capturing and vote tampering across the country during the general elections .Online voting will reduce such number of cases to zero. Today everything is becoming digital and there is a great scope of it because it helps to cast your vote digitally. Its objective is to provide a simple and clean system which is able to give result on the same day, the benefit of this system is not countable because it save your money, time and a huge manpower and our goal is to spread it all over the country.

1.4. Report Organization

Section 2 contains Literature Survey with respect to our project, Section 3 contains

2. Literature Survey

2.1. Survey of the Existing Models/Work

2.1.1. Bounded Verification of Voting Software

We present a case-study in which vote-tallying software is analyzed using a bounded verification technique, whereby all executions of a procedure are exhaustively examined within a finite space given by a bound on the size of the heap and the number of loop unrollings. The technique involves an encoding of the procedure in an intermediate relational programming language, a translation of that language to relational logic, and an analysis of the logic that exploits recent advances in finite model-finding. Our technique yields concrete counterexamples – traces of the procedure that violate the specification. The vote-tallying software, used for public elections in the Netherlands, had previously been annotated with specifications in the Java Modeling Language and analyzed with ESC/Java2. Our analysis found counterexamples to the JML contracts, indicating bugs in the code and errors in the specifications that evaded prior analysis.

2.1.2. Analysis of an Electronic Voting System

Recent election problems have sparked great interest in managing the election process through the use of electronic voting systems. While computer scientists, for the most part, have been warning of the perils of such action, vendors have forged ahead with their products, claiming increased security and reliability. Many municipalities have adopted electronic systems, and the number of deployed systems is rising. For these new computerized voting systems, neither source code nor the results of any third-party certification analyses have been available for the general population to study, because vendors claim that secrecy is a necessary requirement to keep their systems secure. Recently, however, the source code purporting to be the software for a voting system from a major manufacturer appeared on the Internet. This manufacturer’s

systems were used in Georgia's state-wide elections in 2002, and the company just announced that the state of Maryland awarded them an order valued at up to $55.6 million to deliver touch screen voting systems.

2.1.3. On the notion of 'software independence' in voting systems

This paper defines and explores the notion of 'software independence' in voting systems: 'A voting system is software independent if an (undetected) change or error in its software cannot cause an undetectable change or error in an election outcome'. For example, optical scan and some cryptographically based voting systems are software independent. Variations and implications of this definition are explored. It is proposed that software-independent voting systems should be preferred, and software-dependent voting systems should be avoided. An initial version of this paper was prepared for use by the Technical Guidelines Development Committee in their development of the Voluntary Voting System Guidelines, which will specify the requirements that the USA voting systems must meet to receive certification.

2.1.4. Application of Firebase in Android App Development-A Study

The web application has become more and more reliant upon large amount of database and unorganized data such as videos, images, audio, text, files and other arbitrary types. It is difficult for Relational Database Management System (RDBMS) to handle the unstructured data. Firebase is a relatively new technology for handling large amount of unstructured data. It is very fast as compared to RDBMS. This paper focuses on the application of Firebase with Android and aims at familiarizing its concepts, related terminologies, advantages and limitations. The paper also tries to demonstrate some of the features of Firebase by developing an Android app.

2.1.5. Real-time Communication Application Based on Android Using Google Firebase

: In today's world, communication is extremely vital and keeping this communication real-time is essential as our lives have become more fast paced. Keeping this is mind, a communication application should be able to transfer files and messages instantly without or with minimal delay,

depending on the broadcast medium. For such a system to be functional, there must be a database which will update in real-time so as to keep track of all the data being transferred. Google Firebase is a service which provides such a real-time database server, along with a host of other features and Firebase enables us to develop communication-based applications with relative ease. In this paper, we propose a system which will be capable of sending text-based messages and files such as images, audio, videos, texts over through the internet between two users on the network in real-time. We make use of the Android operating system and Google Firebase to handle the backend of the communication operation, highlighting the various features of both the operating system and the service.

2.2. Summary/Gaps identified in the Survey

A software-independent approach to voting systems will provide voters with an assurance that errors or fraud in election results can reliably be detected. Testing costs to prove the correctness of the software can be held somewhat in check if, fundamentally, the correctness of the election results does not rely on the correctness of the software.

For quite some time, voting equipment vendors have maintained that their systems are secure, and that the closed-source nature makes them even more secure. Our glimpse into the code of such a system reveals that there is little difference in the way code is developed for voting machines relative to other commercial endeavors. In fact, we believe that an open process would result in more careful development, as more scientists, software engineers, political activists, and others who value their democracy would be paying attention to the quality of the software that is used for their elections. (Of course, open source would not solve all of the problems with electronic elections.

This paper highlights on the study about google provided Firebase API and its unique features. This paper helps in studying how to use Firebase in the Android application according to the developer requirement. This also helps in making android apps faster and efficient as no PHP is required as a third party language to communicate with the database. It provides a secure channel to communicate with the database directly from JAVA. The study

material is based on the data provided online and referring to the examples given. Google has been updating Firebase on regular basis, AdSense is the beta phase of Firebase. It can not only be used in Android but also to connect cross platform .The work can be further extended by adding new features and exploring new possibilities in Android applications.

Firebase is a step forward in the right direction in the context of application development in the sense that it provides an allround service for developers. Many companies will follow in its footsteps and hence the development of the applications will be much smoother and efficient. As for our application, it has numerous upgrade paths from implementing call functionality or any other enhanced future thanks to the flexible nature of the application and the application will continue to upgrade as long as the technology upgrades. So, the application is extremely future-proof.

3. Proposed System Requirements Analysis and Design

3.1. Introduction

We are going to present an android voting app that would contain credenrials of voterID of a citizen. A citizen will use the same credentials to login into the app and vote for his favourite candidate.Proper emphasis will be given to encryption of data as security being the main aspect of the project.

3.2. Reuqirement Analysis

3.2.1. Stakeholder Identification

- Voters
- The decision-making process to introduce New Voting Technologies(NVT)
- The electoral system and political parties
- Civil society
- Vendors

3.2.2. Functional Requirements

- User details such as Name, Email, Phone, Address
- VoterID
- Finger Print
- Candidate speeh video/campaign video

3.2.3. Non-Functional Requirements

- The software is expected to have reasonably short response time. It should be able to log-in and feed the voter with new pages on request with a response time of the order of a few seconds.
- In order to prevent data loss in case of system failure, the result of votes that were polled till then have to be saved in the database, for thesystem to resume the counting process on reboot.
- The system should provide basic security features like password authentication and encrypted transactions
- All the passwords generated and communicated to the users should be stored in the server only in an encrypted form for login management to prevent misuse.

3.2.4. System Requirements

3.2.4.1. Hardware Requirements

- Minimum 2 GB RAM
- Finger Print Sensor
- 1080x1920 pixels
- Qualcomm MSM8974 Snapdragon 800
- Quad-core 2.3 GHz Krait 400

3.2.4.2. Software Requirements

- Android 4.0 and above

3.2.5. Software Requirement Specification Document

3.2.5.1. Purpose

With the increase in use of digital technology in every sector we are aiming to bring the same to the voting sector too. As everyone has mobile-phone these days, we plan to make the voting system available to all using android programming by developing a mobile application to facilitate the voting. This will also allow critical analysis of the logical and functional aspects of the design before any commitment is made to actual code. Online Android voting system tool is a tool designed as a prototype to demonstrate the functionality of Pailler Threshold Crytptosystem (PTC). We will

also consider some additional security concerns during the design process.

3.2.5.2. Intended Audience and Reading Suggestions

Our intended audience is the domain of adults who are older than 18 years and having legal document to caste vote .The document may also serve as a reference guide to the developers of the system.

3.2.5.3. Product Scope

The software produced will be an online voting system. The main objective of this software is to increase the overall voting percentage. As the voting databases contain highly sensitive information, special care will be given to the security part. The voter's credentials will be properly anonymized and hashed. Also the database containing the results will be properly encrypted using different encryption algorithms. We also see the cases of booth capturing and vote tampering across the country during the general elections .Online voting will reduce such number of cases to zero. Today everything is becoming digital and there is a great scope of it because it helps to caste your vote digitally. Its objective is to provide a simple and clean system which is able to give result on the same day, the benefit of this system is not countable because it save your money, time and a huge manpower and our goal is to spread it all over the country.

3.2.5.4. Product Perspective

With the increase in use of digital technology in every sector we are aiming to bring the same to the voting sector too. The current voting system using EVM requires people to go to a particular location to cast their vote. As everyone has mobile-phone these days, we plan to make the voting system available to all using android programming by developing a mobile application to facilitate the voting. This will also come to the help of the people who have shifted to some other place and have their names on the electoral roll in some other place.

3.2.5.5. Product Functions

On the Admin side, the system can be used to create/update/delete the election details (posts, candidates, electoral rolls etc). The Admin should be able to specify the different attributes it wants for posts/candidates of a particular election instance and voters. From the voters perspective, the system is used to help them cast their votes and after the elections are over, allow them to view the results, which are automatically posted on the same site after the election duration is over.

3.2.5.6. User Classes and Characteristics

The users can be divided into two main classes: –

The Admin: They manage the entire Voting System Software and Conduct the Elections. They act as the Election Authority.

The Voters: The voters should have a basic knowledge of how to use a basic app and navigate through apps. The voters should be aware that they have to keep their user-id and password confidential.

3.2.5.7. Operating Environment

The app runs on Android 4.0+ and uses firebase to store data which requires net connectivity while running the app.

3.2.5.8. Design and implementation constraints

- GUI is only in English.
- This system is working for single server
- User should have basic knowledge of computer.
- The app is only designed for android users and won't run on iOS.

3.2.5.9. User Documentation

- The app will contain a help section for any guidance needed.
- Certifications of government of India will also be displayed in the app.
- Furthermore upon logging in, the user will get a quick tour about how to use the app.

3.2.5.10. Assumptions and Dependencies

- The end user should have a basic knowledge of English and computer usage.

- Administrator is created in the system already.
- The voting results will be managed and calculated by the admin.
- Roles and tasks are predefined.

3.2.5.11. User Interfaces

Voters: The citizens of the country who are eligible for casting vote.

Register for Online Voting System – Those who already have voter id, they will register themselves for online voting system and they will use their voter id as their user name and separate password will be used for secure authentication.

Cast vote – The citizens will cast their votes for their favorite candidates online through a secure system.

View own details – The voters will view their own details which they filled up at the time of their registration.

3.2.5.12. Hardware Interfaces

There are no hardware interfaces to this software system. The only interfaces are through a computer system.

3.2.5.13. Software Interfaces

The app runs on Android 4.0+ and uses firebase to store data which requires net connectivity while running the app.

3.2.6. Work Breakdown Structure

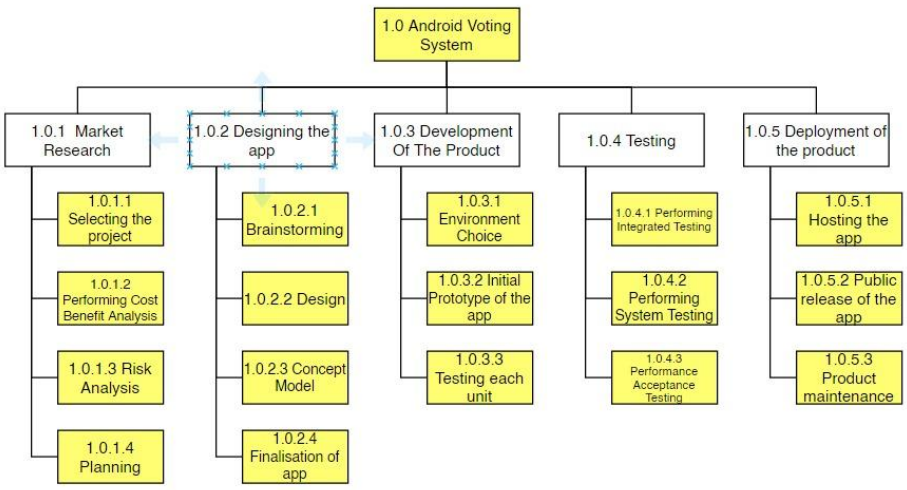


Fig : Verb Based WBS

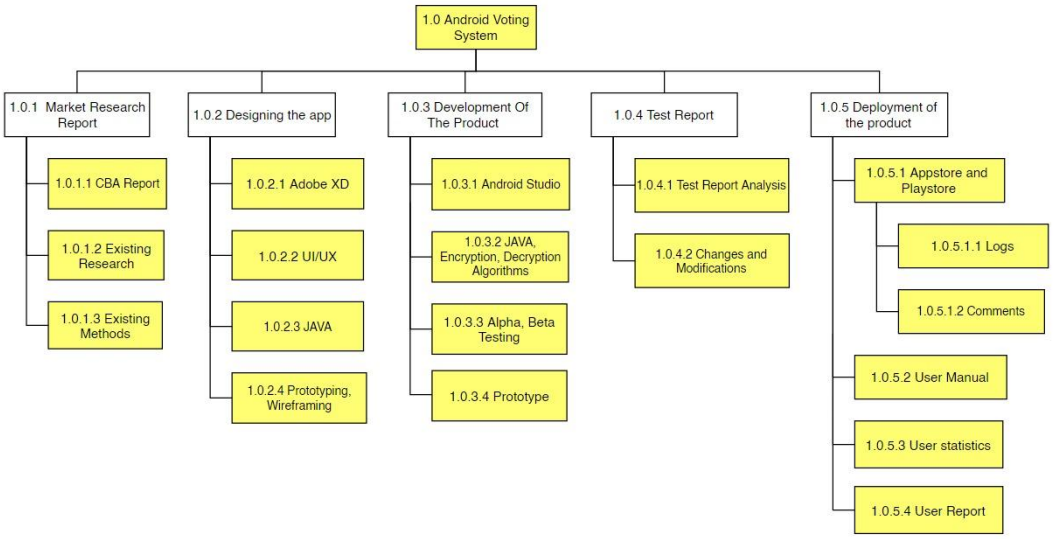


Fig : Noun Based WBS

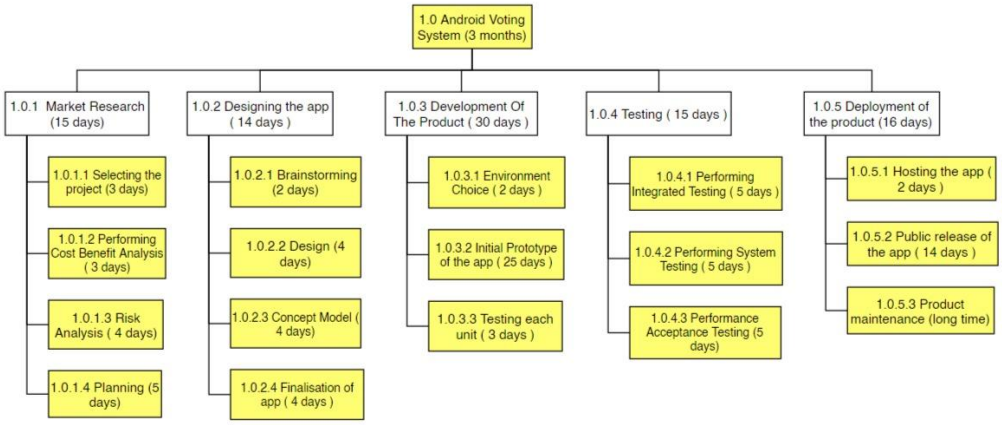
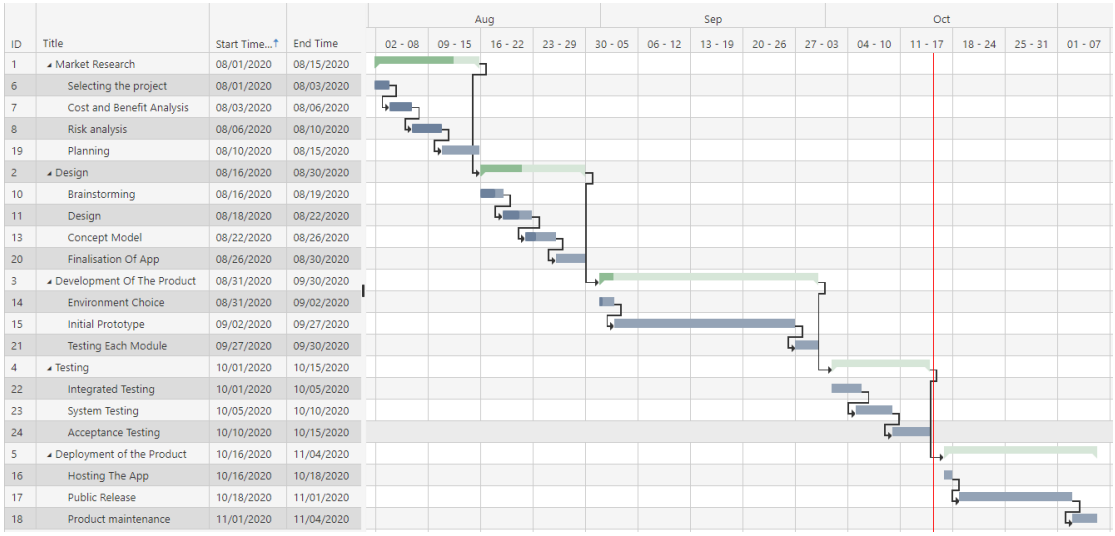


Fig : Time Based WBS

3.2.7. Gantt Chart



ID	Title	Start Time	End Time
1	Market Research	08/01/2020	08/15/2020
6	Selecting the project	08/01/2020	08/03/2020
7	Cost and Benefit Analysis	08/03/2020	08/06/2020
8	Risk analysis	08/06/2020	08/10/2020
19	Planning	08/10/2020	08/15/2020
2	Design	08/16/2020	08/30/2020
10	Brainstorming	08/16/2020	08/19/2020
11	Design	08/18/2020	08/22/2020
13	Concept Model	08/22/2020	08/26/2020
20	Finalisation Of App	08/26/2020	08/30/2020
3	Development Of The Product	08/31/2020	09/30/2020
14	Environment Choice	08/31/2020	09/02/2020
15	Initial Prototype	09/02/2020	09/27/2020
21	Testing Each Module	09/27/2020	09/30/2020
4	Testing	10/01/2020	10/15/2020
22	Integrated Testing	10/01/2020	10/05/2020
23	System Testing	10/05/2020	10/10/2020
24	Acceptance Testing	10/10/2020	10/15/2020
5	Deployment of the Product	10/16/2020	11/04/2020
16	Hosting The App	10/16/2020	10/18/2020
17	Public Release	10/18/2020	11/01/2020
18	Product maintenance	11/01/2020	11/04/2020

Fig : Gantt Chart

4. Design of the Proposed System

4.1. Introduction

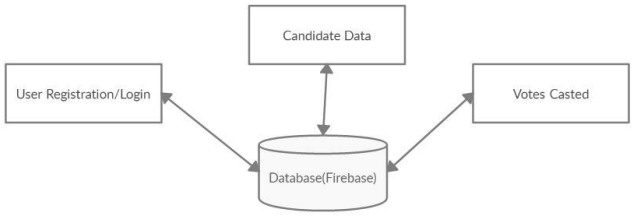
The system is developed using JAVA, Android Studio and Firebase. Firebase is used both as a storage system which can contain the various files the user uploads and also as a database to store all the details of the voters, candidates as well as the votes.

4.2. High Level Design

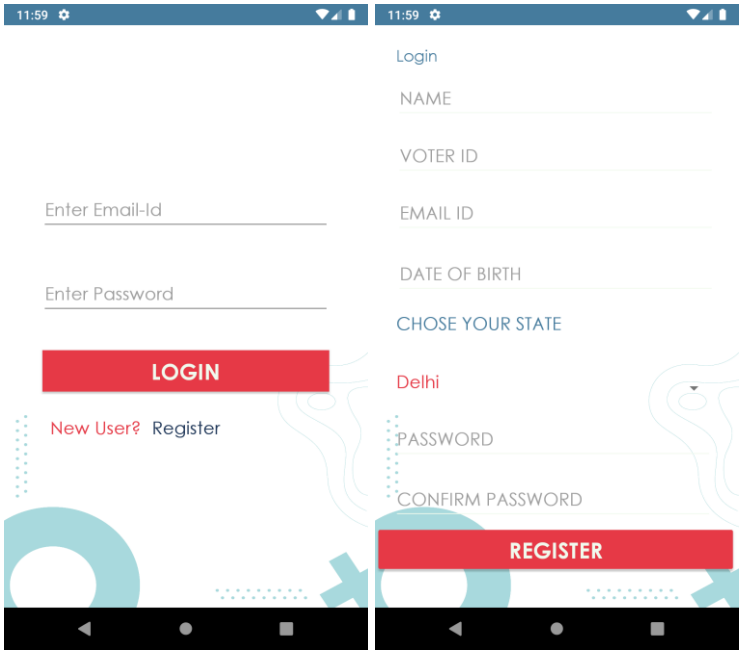
4.2.1. Architecture Design

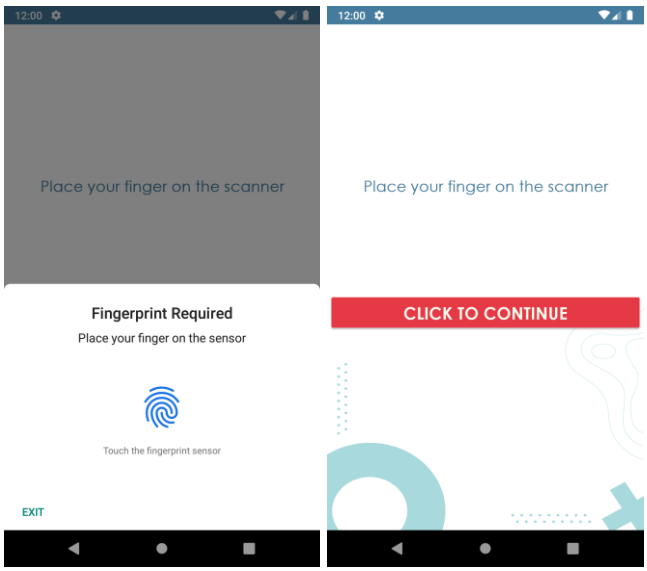
Upon entering the app for the first time you are asked to register into the database. If you already are a user, you need to login into the app. After registering or logging in you are asked to choose between a voter, a candidate or a political party. The explicit identifiers like the Voter ID and the votes are thoroughly encrypted by AES and MD5 respectively. The Quasi identifiers like the name and address are fully masked to maintain the anonymity of the data. A voter can get access to all the details of the candidate of his constituency along with watching his speeches till 2 days of the voting. Also they can see the results 5 hours after the voting ends. A candidate can apply from any political party of his choice. They can also edit his profile and upload their speeches and portfolios. He/She can edit his profile one week and apply for the candidature one month prior to the elections A political party has to approve whether a particular candidate belongs to the party or not. It has to do the same one month prior to the elections.

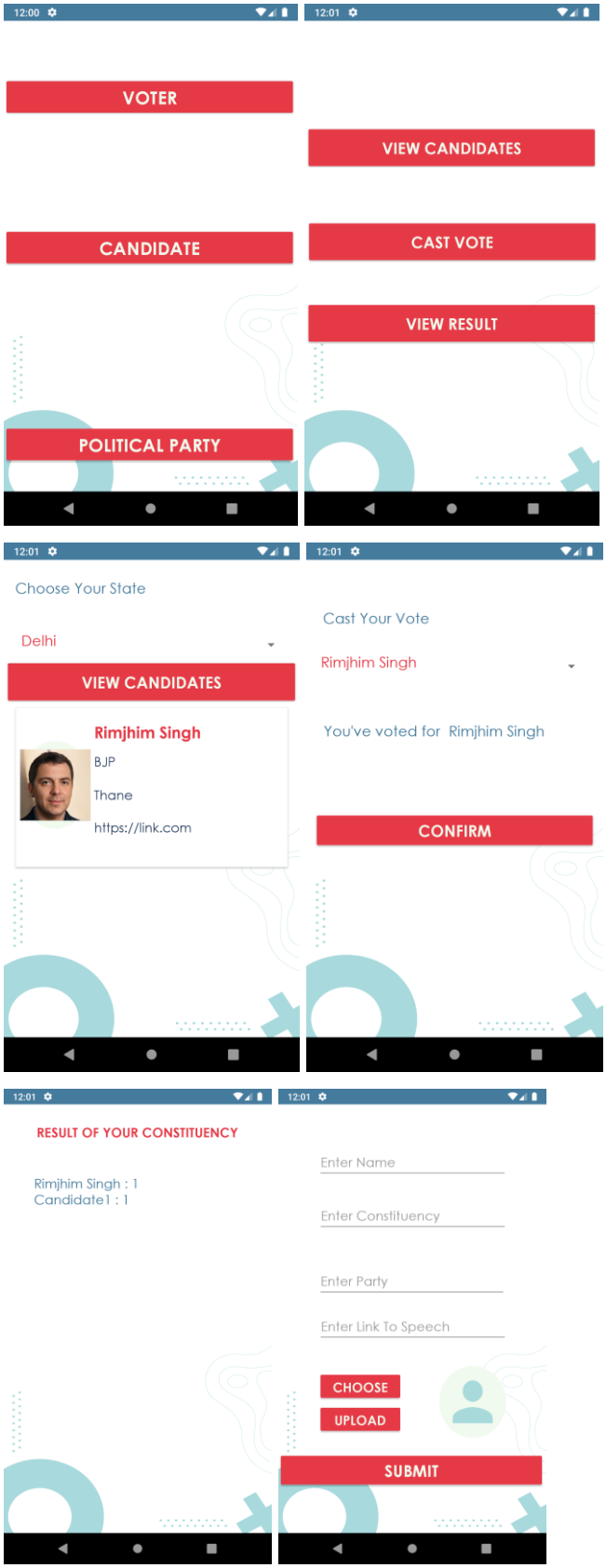
4.2.2. Architecture Diagram



4.2.3. UI Design







4.3. Detailed Design

4.3.1. UML Diagram

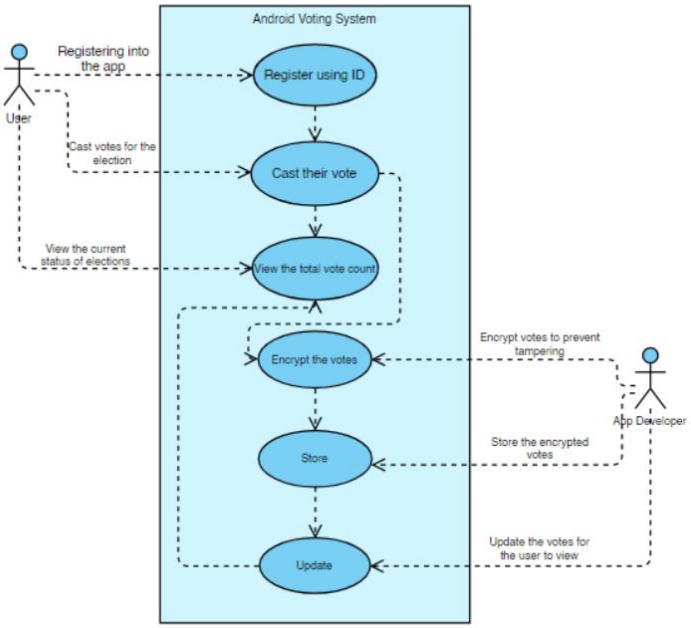


Fig: Use Case Diagram

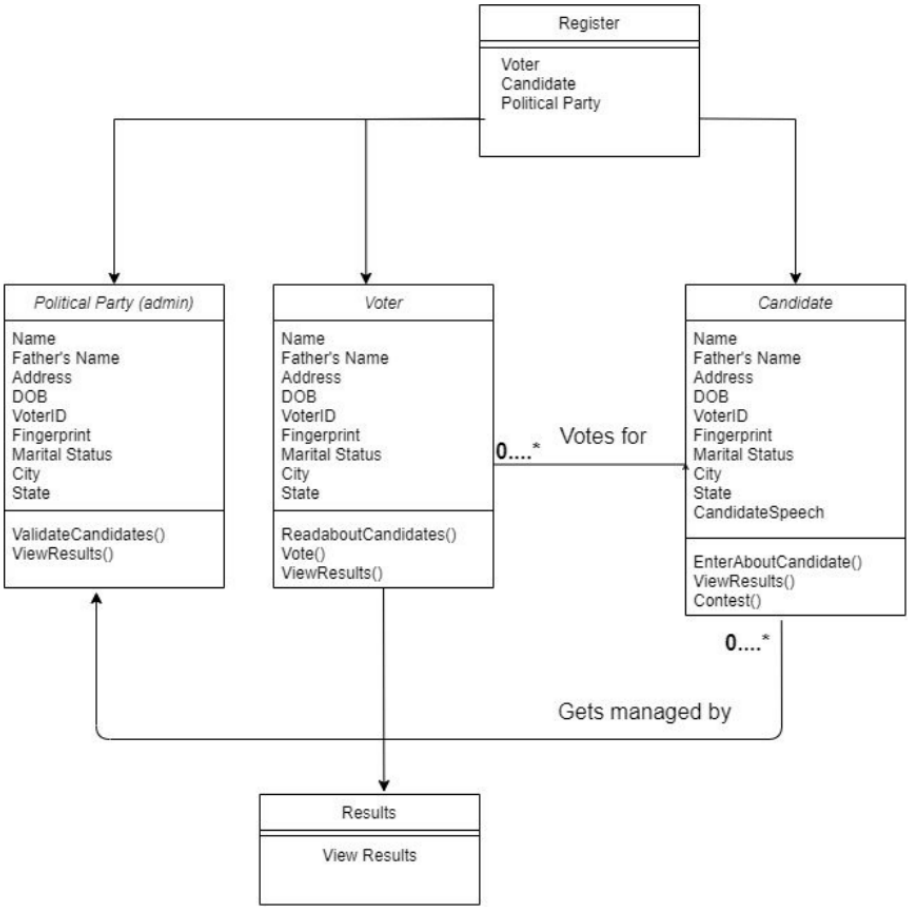


Fig: Class Diagram

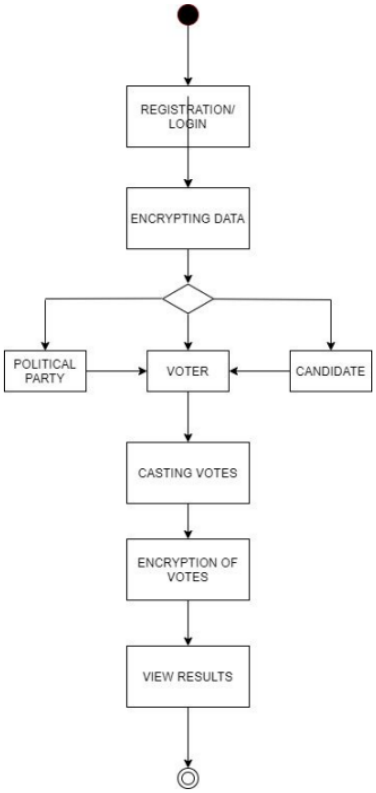


Fig: Activity Diagram

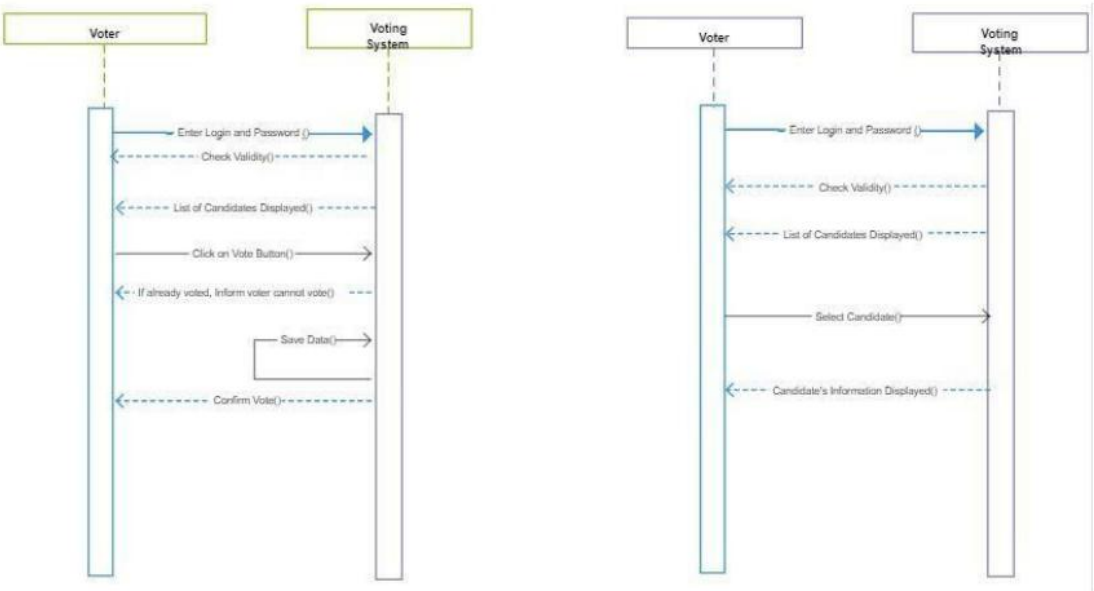


Fig: Sequence Diagram

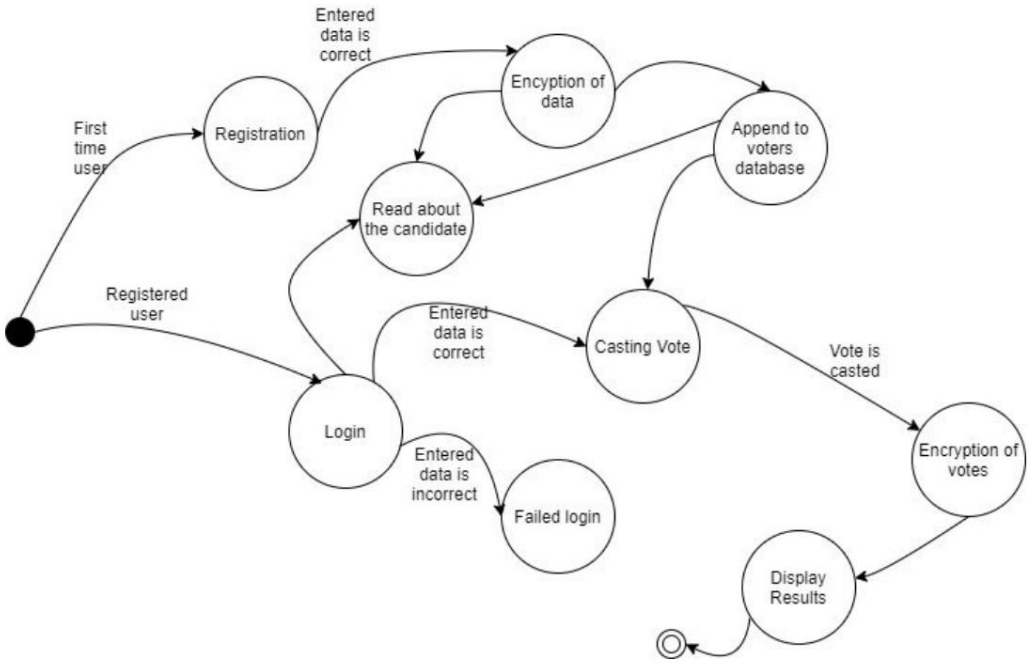


Fig: State Transition Diagram

5. Implementation and Testing (Snap shots with description)

5.1. Implementation Details

```
public class biometric_verification extends AppCompatActivity {

    Executor executor;
    BiometricPrompt biometricPrompt;
    BiometricPrompt.PromptInfo promptInfo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_biometric_verification);
        executor= ContextCompat.getMainExecutor(this);
        biometricPrompt=new BiometricPrompt(this,executor,new BiometricPrompt.AuthenticationCallback(){
            @Override
            public void onAuthenticationSucceeded(@NonNull BiometricPrompt.AuthenticationResult result){
                super.onAuthenticationSucceeded(result);
                Toast.makeText(biometric_verification.this,"Successful Verification",Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onAuthenticationError(int errorCode, @NonNull CharSequence errString){
                super.onAuthenticationError(errorCode,errString);
                Toast.makeText(biometric_verification.this,errString,Toast.LENGTH_SHORT).show();
                biometric_verification.this.finish();
                startActivity(new Intent(getApplicationContext(),LoginOrRegister.class));
            }

            @Override
            public void onAuthenticationFailed(){
                super.onAuthenticationFailed();
                Toast.makeText(biometric_verification.this,"Verfication Failed",Toast.LENGTH_SHORT).show();
            }
        });
```

```java
public class Candidates extends AppCompatActivity {

    EditText t1,t2,t3,t4;
    String name,cons,pp,file;
    Button btn,btnChoose,btnUp;
    Candidate c=new Candidate();
    private ImageView imageView;
    DatabaseReference mDbRef;
    FirebaseAuth mAuth;
    FirebaseUser mUser;

    private Uri filePath;
    private final int PICK_IMAGE_REQUEST = 71;
    FirebaseStorage storage;
    StorageReference storageReference;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_candidates);
        t1=(EditText)findViewById(R.id.t1);
        t2=(EditText)findViewById(R.id.t2);
        t3=(EditText)findViewById(R.id.t3);
        t4=(EditText)findViewById(R.id.t4);
        btn=(Button)findViewById(R.id.submit);
        btnChoose=(Button)findViewById(R.id.btnChoose);
        btnUp=(Button)findViewById(R.id.btnUpload);
        imageView=(ImageView)findViewById(R.id.imgView);
        storage = FirebaseStorage.getInstance();
        storageReference = storage.getReference();
        mAuth=FirebaseAuth.getInstance();
        mUser=mAuth.getCurrentUser();
```

```java
private void chooseImage() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(Intent.createChooser(intent, title: "Select Picture"), PICK_IMAGE_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK
            && data != null && data.getData() != null )
    {
        filePath = data.getData();
        try {
            Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), filePath);
            imageView.setImageBitmap(bitmap);
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}
```

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    n=(EditText)findViewById(R.id.name);
    uid=(EditText)findViewById(R.id.voterid);
    eid=(EditText)findViewById(R.id.emailid);
    birth=(EditText)findViewById(R.id.birth);
    pd=(EditText)findViewById(R.id.password);
    pC=(EditText)findViewById(R.id.passwordC);
    mAuth=FirebaseAuth.getInstance();
    user=new User();

    Button btn1=(Button)findViewById(R.id.register);
    btn1.setOnClickListener((view) -> {
            name=n.getText().toString();
            voterid=uid.getText().toString();
            pword=pd.getText().toString();
            pwordC=pC.getText().toString();
            emailid=eid.getText().toString();
            dob=birth.getText().toString();

            fUser=mAuth.getCurrentUser();
            user.setUserId(fUser.getUid());

            Voters vote=new Voters();
            vote.setName(name);
            vote.setEmailId(emailid);
            vote.setVoterId(voterid);
            vote.setDOB(dob);
            vote.setState(v_state);
            DatabaseReference mDbRef = FirebaseDatabase.getInstance().getReference();
            mDbRef.child("Users").child(fUser.getUid()).setValue(user);
```

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login_or_register);
    t1= (TextView)findViewById(R.id.t1);
    tname= (TextView)findViewById(R.id.tname);
    tpd= (TextView)findViewById(R.id.tpd);
    btn1=(Button)findViewById(R.id.btn1);
    mAuth=FirebaseAuth.getInstance();
    t1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(getApplicationContext(),MainActivity.class));
        }
    });

    btn1.setOnClickListener((view) -> {
            String username=tname.getText().toString();
            String password=tpd.getText().toString();
            mAuth.signInWithEmailAndPassword(username,password).addOnCompleteListener( activity: LoginOrRegister.this, (task) -> {
                    if(task.isSuccessful()){
                        Toast.makeText(getApplicationContext(), text: "Logged In",Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(getApplicationContext(),biometric_verification.class));
                    }
                    else{
                        Toast.makeText(getApplicationContext(), text: "Login Failed",Toast.LENGTH_SHORT).show();
                    }
            });
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_political_party);
    rv=(RecyclerView)findViewById(R.id.rView1);
    c2= new Candidate();
    myref1= FirebaseDatabase.getInstance().getReference( path: "Candidate");

    rv.setLayoutManager(new LinearLayoutManager( context: this));
    FirebaseRecyclerOptions<Candidate> options =
                    new FirebaseRecyclerOptions.Builder<Candidate>()
                            .setQuery(FirebaseDatabase.getInstance().getReference().child("Candidate"), Candidate.class)
                            .build();
            adapt = new PartyAdapter(options);
            adapt.startListening();
            rv.setAdapter(adapt);
            new ItemTouchHelper(new ItemTouchHelper.SimpleCallback( dragDirs: 0,ItemTouchHelper.RIGHT) {
                @Override
                public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
                    return false;
                }

                @Override
                public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
                    adapt.deleteItem(viewHolder.getAdapterPosition());
                    Toast.makeText( context: PoliticalParty.this, text: "Deleted",Toast.LENGTH_SHORT).show();
                }
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_result);
    //Initializing arraylists and hashmap
    userid=new ArrayList<String>();
    uservote=new ArrayList<String>();
    decvotes=new ArrayList<String>();
    hm = new HashMap<String, Integer>();
    tv=findViewById(R.id.res);

    database  = FirebaseDatabase.getInstance();
    ref = database.getReference();
    ref.child("Votes").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
                for (DataSnapshot ds : snapshot.getChildren()) {
                    userid.add(ds.child("userID").getValue().toString());
                    uservote.add(ds.child("vote").getValue().toString());
                }
                Log.d(TAG, userid.get(0));
                Log.d(TAG, uservote.get(0));
                for (String item : uservote) {
                    Integer j = hm.get(item);
                    hm.put(item, (j == null) ? 1 : j + 1);
                }
                Log.d(TAG, String.valueOf(hm.get("Rimjhim Singh")));
                for(Map.Entry<String, Integer> val : hm.entrySet()){
                    tv.append(val.getKey()+" : "+val.getValue()+"\n");
                }
        }
```

```java
package com.example.votingapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import android.os.Bundle;
import android.view.View;


import gr.net.maroulis.library.EasySplashScreen;

public class SplashScreenActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EasySplashScreen config = new EasySplashScreen( activity: SplashScreenActivity.this)
                .withFullScreen()
                .withTargetActivity(LoginOrRegister.class)
                .withSplashTimeOut(1500)
                .withBackgroundResource(R.drawable.bg_ui)
                .withAfterLogoText("VOTING APP")
                .withLogo(R.mipmap.ic_launcher_round);
        config.getAfterLogoTextView().setTextColor(ContextCompat.getColor( context: this,R.color.imperialRed));
        View easySplashScreen = config.create();
        setContentView(easySplashScreen);
    }
}
```

```java
List<String> candidateName;
final String TAG = "DHYAN SE DEKHO IISEE";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_votingpage);
    //Spinner for candidates
    cast=(Spinner)findViewById(R.id.dropdown);
    candidateName = new ArrayList<String>();

    //Firebase to adapter
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference fDatabaseRoot = database.getReference();
    fDatabaseRoot.child("Candidate").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            for(DataSnapshot ds: snapshot.getChildren()){
                name = ds.child("name").getValue().toString();
                    candidateName.add(name);
            }
            ArrayAdapter<String> adapter=new ArrayAdapter<~>( context: votingpage.this,R.layout.spinner_item1,candidateName);
            adapter.setDropDownViewResource(R.layout.spinner_layout);
            adapter.notifyDataSetChanged();
            cast.setAdapter(adapter);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
```

5.2. Testing

5.2.1. Testcases

TEST CASE ID	TEST CASE OBJECTIVE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
MOD101	Test the Login Page	Correct Details (Positive Test Case)	Toast message saiyng "Login Successful"	Toast message saiyng "Login Successful"	PASS
MOD102	Test the login page	Incorrect Details (Negative Test Case)	Toast message saiyng "Login Failed"	Toast message saiyng "Login Failed"	PASS
MOD201	Testing Fingerprint Verification	No fingerprint (Negative Test Case)	Toast message saying "Verification failed"	Toast message saying "Verification failed"	PASS
MOD202	Testing Fingerprint Verification	Fingerprint given (Positive Test Case)	Toast message saying "Verification successful"	Toast message saying "Verification successful"	PASS
MOD301	Adding details to the database	Details given as input (Positive Test Case)	The details should be reflected in the database	The details are reflected in the database	PASS

Fig : Test Cases

6. Conclusion, Limitations and Scope for future Work

This app was made for the purpose of making the “fundamental duty” of voting, easy for the common man. This app successfully serves it purpose by giving the user a mobile platform to view the potential candidates, to vire their speeches and then cast a vote without the hassle of going to the poll booth and waiting in long queues to cast votes. This will increase the voter turnout by a great margin as the people who have moved out of their voting constituencies due to work will also be able to vote. Future work may include adding activities in the app which will play the candidate videos in the app itself rather than using the link in a browser.

7. References

1. https://www.researchgate.net/profile/Asoke_Nath/publication/324840628_Real-time_Communication_Application_Based_on_Android_Using_Google_Firebase/links/5ae721760f7e9b9793c82cbf/Real-time-Communication-Application-Based-on-Android-Using-Google-Firebase.pdf
2. https://www.researchgate.net/profile/Chunnu_Khawas/publication/325791990_Application_of_Firebase_in_Android_App_Development-A_Study/links/5bab55ed45851574f7e6801e/Application-of-Firebase-in-Android-App-Development-A-Study.pdf
3. https://www.cs.cornell.edu/people/egs/cornellonly/syslunch/fall03/voting.pdf
4. https://dspace.mit.edu/bitstream/handle/1721.1/51684/dennis-bounded.pdf?sequence=1&isAllowed=y

5. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.2530&rep=rep1&type=pdf
6. https://firebase.google.com/docs/android/setup
7. https://firebase.google.com/docs/auth/android/firebaseui
8. https://firebase.google.com/docs/database/android/start
9. https://firebase.google.com/docs/storage/android/start
10. https://firebase.google.com/docs/database/android/read-and-write