Report on

BIOMETRIC BASED ELECTRONIC VOTING MACHINE

in Course

Project II (EC4902D)

Of

B.Tech Electronics

Report By

Rushikesh Bute - 161060014
Nishad Patne - 161060028
Lalit Jadhav - 161060042
Saurabh Gurbhele- 161060063

Under the guidance of

Dr. D. P. Rathod

DEPARTMENT OF ELECTRICAL ENGINEERING

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE

MUMBAI – 400019

(2016-2020)

INDEX

1 ABSTRACT

The aim of the project is to develop a Biometric based Electronic Voting Machine that can be used to distant voting, better authentication. The key motivation lies in simplifying the complex electoral process of India. Biometric EVM also tries to find out new methodologies to address verification of voters specially thumb impression of voter in this project. The technology also provides a secured confirmation to voters about their registered vote and addresses an issues of security and tampering of traditional EVMs the technology encourages fast counting of votes provide quick result of elections.

The project uses a Raspberry Pi as a microprocessor of the technology. It uses fingerprint scanner for the verification of voters and single national database system of voters with SQL for database management.

Biometric Electronic Voting Machines connected with UIDAI would provide safe, secure and swift elections in India in the near future.

2 LIST OF FIGURES

3 INTRODUCTION

India is largest democracy in the world with 910 million voters and election as a complex and timing consuming exercise. The general election 2019 was carried out for over a month to elect new union government. The Vice President of India had once said, "Technology is a great democratic tool. It is only an enabler but also equalizer, which can help a billionaire as also the poorest of poor" [1].

The aim of the project is to develop a Biometric based Electronic Voting Machine. The system uses thumb impression for voter identification as a fingerprint of every human being has a unique pattern. The technology directs the voter as per his registered constituency. The EVM also notifies voter about his registered vote with a message of confirmation on a registered mobile number.

The applications of this project are enormous. The biometric EVM maintains a single common database of all voters of the nation. The Unique Identification Authority of India (UIDAI) [5] already has the biometric database of all citizens in the form of Aadhar. The database can be connected with electoral system to use in the elections. As per the study commissioned by Election Commission, 326 million voters are migrated from their home constituency [2] which makes it difficult for them to participate in elections. A 2011 study on political inclusion of seasonal migrant workers by Amrita Sharma found that 22 percent of seasonal migrant workers in India did not possess voter IDs or have their name in voting list [3]. The electoral process of democracy encourages the maximum participation of voters and this project simplifies the problem of distant voting. The biometric EVM can be used to vote for any constituency as per the requirement of the voters.

The important issue traditional electronic voting machine faces is issue of credibility. Opposition have always alleged that there are high chances of tampering EVMs [6] for the benefit of ruling party. This project addresses the problem by introducing new feature in the technology of confirmation of voting. The biometric EVM sends confirmation message to voter on registered mobile number which could be used to verify the authenticity of voting.

The project addresses some of the key challenges in electoral system of India and tries to simplify it for the citizens using simple technology.

4 PRIOR WORK AND LITERATURE REVIEW

Electronic Voting Machine (also known as EVM) is voting using electronic means to either aid or take care of the chores of casting and counting votes.
An EVM is designed with two units: the control unit and the balloting unit. These units are joined together by a cable. The control unit of the EVM is kept with the presiding officer or the polling officer. The balloting unit is kept within the voting compartment for electors to cast their votes. This is done to ensure that the polling officer verifies your identity. With the EVM, instead of issuing a ballot paper, the polling officer will press the Ballot Button which enables the voter to cast their vote. A list of candidates' names and/or symbols will be available on the machine with a blue button next to it. The voter can press the button next to the candidate's name they wish to vote for.
Recently biometric EVM has gained the attention of many researchers. In this section, the related works on electronic voting systems with biometric authentication are discussed briefly.

4.1 Biometrics

Biometrics is biological measurements or physical characteristics that can be used to identify individuals. Fingerprint mapping, facial recognition, and retina scans are all forms of biometric technology, but these are just the most recognized options. Because physical characteristics are relatively fixed and individualized even in the case of twins they are being used to replace or at least augment password systems for computers, phones, and restricted access rooms and buildings.

4.2 Protecting Biometric Identity

Unauthorized access becomes more difficult when systems require multiple means of authentication, such as life detection (like blinking) and matching encoded samples to users within encrypted domains. Some security systems also include additional features, such as age, gender, and height, in biometric data to thwart hackers.
India's Unique ID Authority of India Aadhaar program is a good example. Initiated in 2009, the multi-step authentication program incorporates iris scans, fingerprints from all 10 fingers, and facial recognition. This information is linked to a unique identification card that is issued to each of India's 1.2 billion residents. Soon, this card will be mandatory for anyone accessing social services in India.

4.3 Related works

In a study, a model of electronic voting machine was discussed where user verification was done using Near Field Communication (NFC) ID card and biometric technologies. In this process, multiple vote casting was restricted by marking this NFC card after the user had casted his vote once. Use of different biometric identification in e-voting and their security aspects were analyzed in another study conducted by Hof. He discussed some of the weaknesses of biometric systems such as spoofing, false accept and reject rate etc. and therefore, suggested implementing biometric in e-voting with precautions. In a study, an abstract model of voting system with fingerprint authentication and details matching process in fingerprint minutiae were introduced.
A study conducted by Sarkar provided a brief overview on existing e-voting systems and their framework and protocols. They discussed the recent developments of EVM in the context of Bangladesh and suggested some strategies to improve the security, accuracy of the existing design.

In another study, Sarker proposed a conceptual design of electronic voting machines with fingerprint authentication that helped to eradicate defrauding of the manual voting systems and prior versions of electronic voting. They used a four layer network system with three application servers and a client to send data from client to database.

A study conducted by Khasawneh proposed an idea of a multifaceted online e-voting system with combined biometric authentication like fingerprint, facial recognition, iris scanning etc. In this model, electronic ballot paper with multiple scope was introduced and computer simulations were run to test the robustness and accuracy.

In sum, each of the research work introduces different ways for the authentication of electronic voting systems. Though some studies show the implementation of Biometric EVM, each of them has pros and cons in their own use of context. Such as NFC or Adhar card needs to be used in some proposed system which introduces the issues of losing or stealing IDs. Some of the designed models did not ensure convenient user interface and integrated database, biometric authentication etc. like EVM that was tested in some countries- Bangladesh, India. Therefore, our contribution in this paper is to introduce a conceptual design and development of Biometric EVM which is unique, secured and convenient to solve the raised problems.

5 PROBLEM STATEMENT

5.1 Problem Definition

The malfunctions of the electoral system due to bogus voting and less participation of voters.

5.2 Context

There are several instances where bogus voting trends in the news during elections. The Indian National Congress suggested to Election Commission about presence of around 42 lakh bogus voters during Maharashtra Assembly Elections 2019 [7]. As per the study by Election Commission around 326 Million voters migrated from their constituency [2] which makes it difficult for them to participate in elections. The credibility of traditional EVMs is always questioned by the opposition due to complex voting procedures and insufficient verification of votes. There have been some successful attempts to improve with the introduction of VVPATs [4] but this measure hasn't made a significant impact. This project suggests some efficient models to resolve the issues.

5.3 Relevance

Low voter turnout has been shown to have negative associations with social cohesion and civic engagement. The democracy needs maximum voter participation as well as inclusion of all sections of society. The free and fair elections can contribute highly to elect honest & workaholic leaders for development of region. Electoral Process is start of the democracy and glitches in it can have serious impact on the human development addressing this problem we have developed Biometric based Electronic Voting Machine.

6 OBJECTIVES

The aim of this project is to develop a Biometric based Electronic Voting Machine and simplify the electoral procedures.

1. The Biometric based Electronic Voting Machine intends to:
2. Centralize single database of all voters in election.
3. Develop a fingerprint based unique voting system.
4. Enable the distance voting in elections
5. Provide SMS based vote verification for voters.
6. Calculate the election results swiftly and transparently.

7 PROPOSED METHOD

The method is research based on the news articles and various researches related to traditional Electronic Voting Machines. As a matured voter the researchers have experienced various flaws with the current electoral system and interviewed some of the student voters to identify the various problems undergone by them. The researchers also visited the Election Commission websites and reports to understand more about traditional EVMs.

The analysis involved listing down various set problems related to different stakeholders involved with electoral procedures. The point of views of administration, political parties, booth officers, immigrants and voters understood to develop a finest technological solutions with the introduction of Biometric based Electronic Voting Machine.

The proposed method creates a central database of voters which can be accessible from any Biometric based Electronic Voting Machine instead of constituency or ward wise database. The model involves solutions like unique identification based voting for better authentication of voters and avoid bogus voting. The method of distance voting is introduced in model to ensure maximum participation of voters. The SMS based vote verification system is developed to address the credibility issues questioned by political parties. The fast votes counting system is enabled to simplify the work of booth officers and avoid logistical problems faced by Election Commission in elections.

8 HARDWARE ASSEMBLING

8.1 Hardware Components

8.1.1 Raspberry Pi

Figure 8-1 Ransberry Pi 3.0 B

Raspberry Pi 3.0 is used as a microprocessor for the project. The memory chip of the processor stores the database of the voters and candidates. The processor performs tasks of control the GUI, fetching and updating database, counting the votes and sending notifications to voters. The USB hubs are used to connect the processor to fingerprint sensor (input device for biometric) and the HDMI port is used to connect with the LCD Display (output screen of voting machine).

8.1.2 Fingerprint Sensor

Figure 8-2 Fingerprint Sensor R305

Fingerprint Sensor R305 is used to capture digital image of fingerprint pattern of voter. The obtained digital image is then converted into hex code for documentation during voter verification. The sensor module connected to USB hub of Raspberry Pi using USB-TTL UART converter module. The fingerprint data in module can be configured in 1:1or 1:N mode for identifying the person, the finger print module is directly interface with 3v3 or 5v microcontroller.

8.1.3 USB-TTL UART Converter

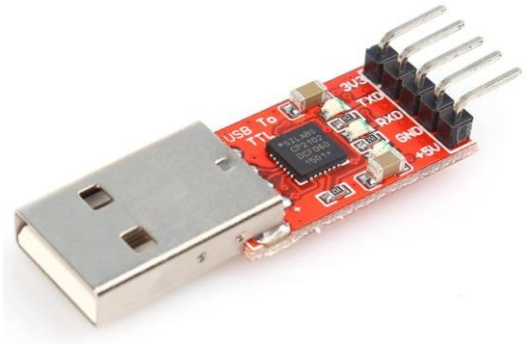


Figure 8-3 UART Converter

The USB to TTL UART converter module provided direct interface of fingerprint sensor with Raspberry Pi. It has standard USB type A and TTL 5 pin connector.

8.1.4 LCD Touch Display

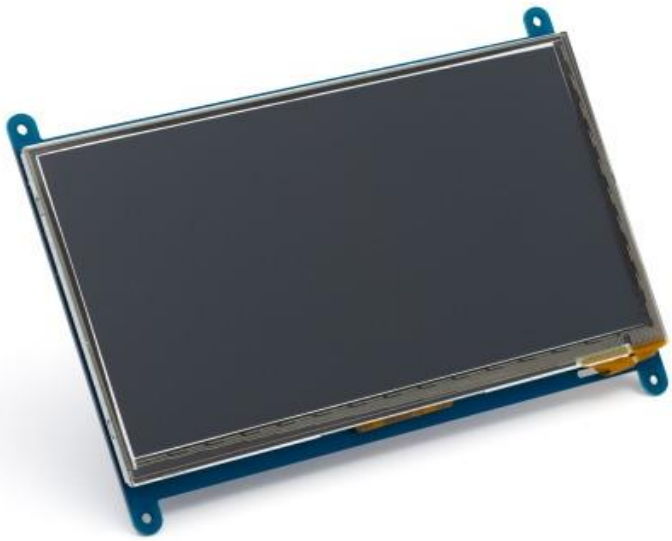


Figure 8-4 LCD Touch Display

The LCD Display of 7" Capacitive Touch Screen is used as an output device of Electronic Voting System. It supports Raspberry Pi and connected with the CPU using HDMI Cable for the transfer of graphical data and through USB as well to support power requirements. The built in HDMI interface enables the display to work as computer monitor just like any other HDMI screen. The built GUI for the EVM displays all the constituency details as well as result. It is the primary input for voters to vote for the candidate of their constituency. The results could also be displayed instantly as per necessity on the same LCD Display.

8.2 Hardware Installation

8.2.1 Raspberry Pi 3

To manage the memory and process of the Raspberry Pi we install Raspbian operating system [11]. It will act as a layer between all programs and hardware to use program standard Interface.

8.2.1.1 Here is a list of all the things used to get started with Raspberry Pi:

- A Raspberry Pi 3
- A Micro SD Card
- LCD Touch Display and HDMI Cable
- 5V 3A Power supply
- Keyboard and Mouse.
- Internet Connectivity (Wi-Fi or Ethernet).
- Micro SD Card Reader

8.2.1.2 Procedure

- Download the Raspbian OS ISO
  https://www.raspberrypi.org/downloads/raspbian/

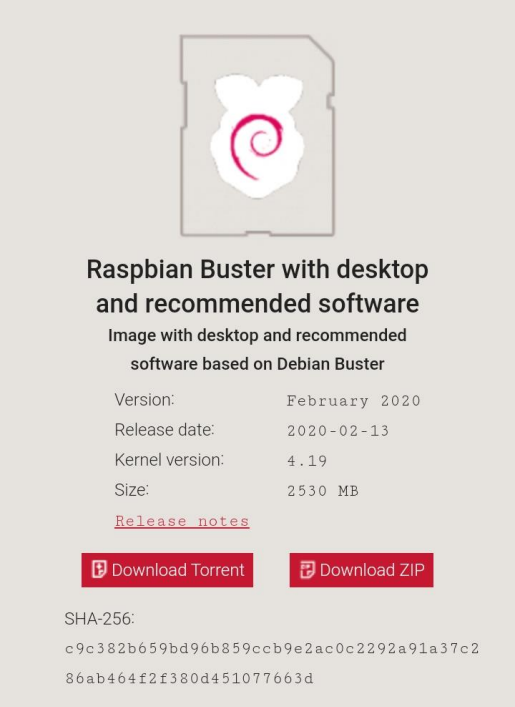


Figure 8-5 Raspbian Download

- Raspbian Stretch with desktop and recommended software :
  This version comes with the complete stack of Raspbian OS. Desktop environment (GUI) and also the bundled software tools and applications. This one is recommended for most people starting out with the Raspberry Pi.

- Prepare the Micro SD Card

  Etcher is used to flash the Micro SD Card. We opened up Etcher, insert the Micro SD card into your computer using a card reader, select the OS image just downloaded and click flash.

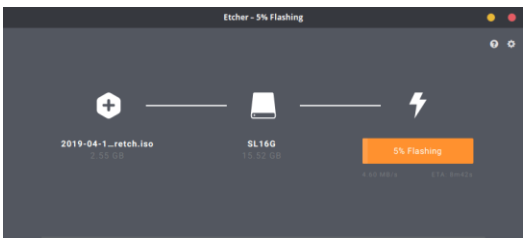


Figure 8-6 Preparation of Memory Card

- Powering up the Raspberry Pi
  Inserted SD Card to Raspberry Pi. Connected HDMI cable to display and wired up Keyboard and Mouse.

- Initial Configuration of Raspbian OS
  Raspberry Pi booted up.

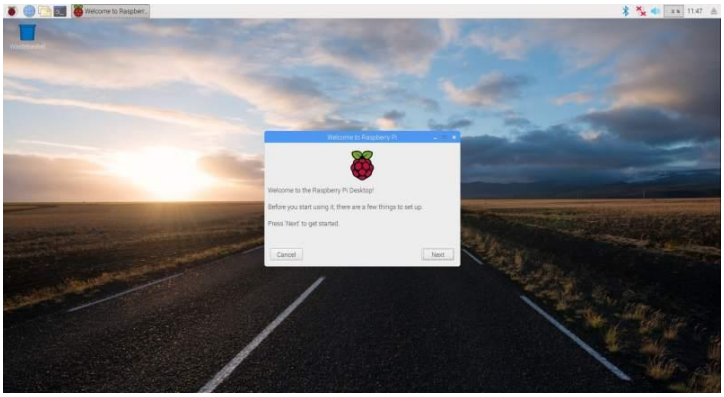


Figure 8-7 Initialise Raspbian

- Country and Password set.

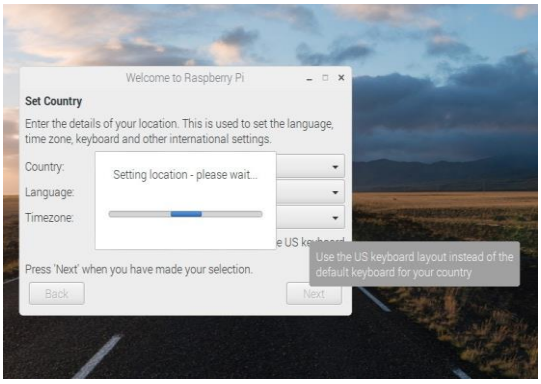


Figure 8-8 Set Country

Figure 8-9 Set Password

- Then connected to WIFI and updated the software if necessary.

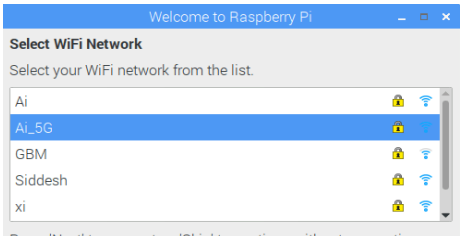


Figure 8-10 Selection of Network

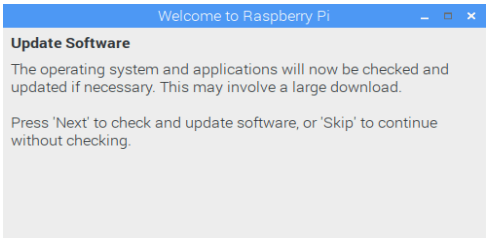


Figure 8-11 Sofware Updatet

8.2.2 Fingerprint Sensor Installation [12]:

8.2.2.1 Connection of the Raspberry Pi Fingerprint Sensor

The Fingerprint Sensor is connected to USB Converter as following:

- Red: 3.3V
- Blue: RXD
- Yellow: TXD
- Black: GND

To check if cabling is correct, following command is used at terminal after connection:

```
ls /dev/ttyUSB*
```

Installation of the Raspberry Pi Fingerprint Library,
For some commands of the installation, root privileges are required. So we type the following at terminal, which executes all the following commands as root:

```
sudo bash
```

The necessary package sources added at terminal:

```
wget -O - http://apt.pm-codeworks.de/pm-codeworks.de.gpg | apt-key add -
wget http://apt.pm-codeworks.de/pm-codeworks.list -P /etc./apt/sources.list.d/
```

Then updated the available packages and install the Python library:

```
apt-get update
apt-get install python3-fingerprint --yes
```

Test code & example scenario,
To see if sensor is detected we run sample files:

```
python3 /usr/share/doc/python-fingerprint/examples/example_index.py
```

The following result occurs:

```
Currently stored templates: 2
Waiting for finger...
Found template at position #1
The accuracy score is: 63
SHA-2 hash of template:
3aa1b01149abf0a7ad0d7803eaba65c22ba084009700c3c7f5f4ecc38f020851
```

8.3 Hardware Setup

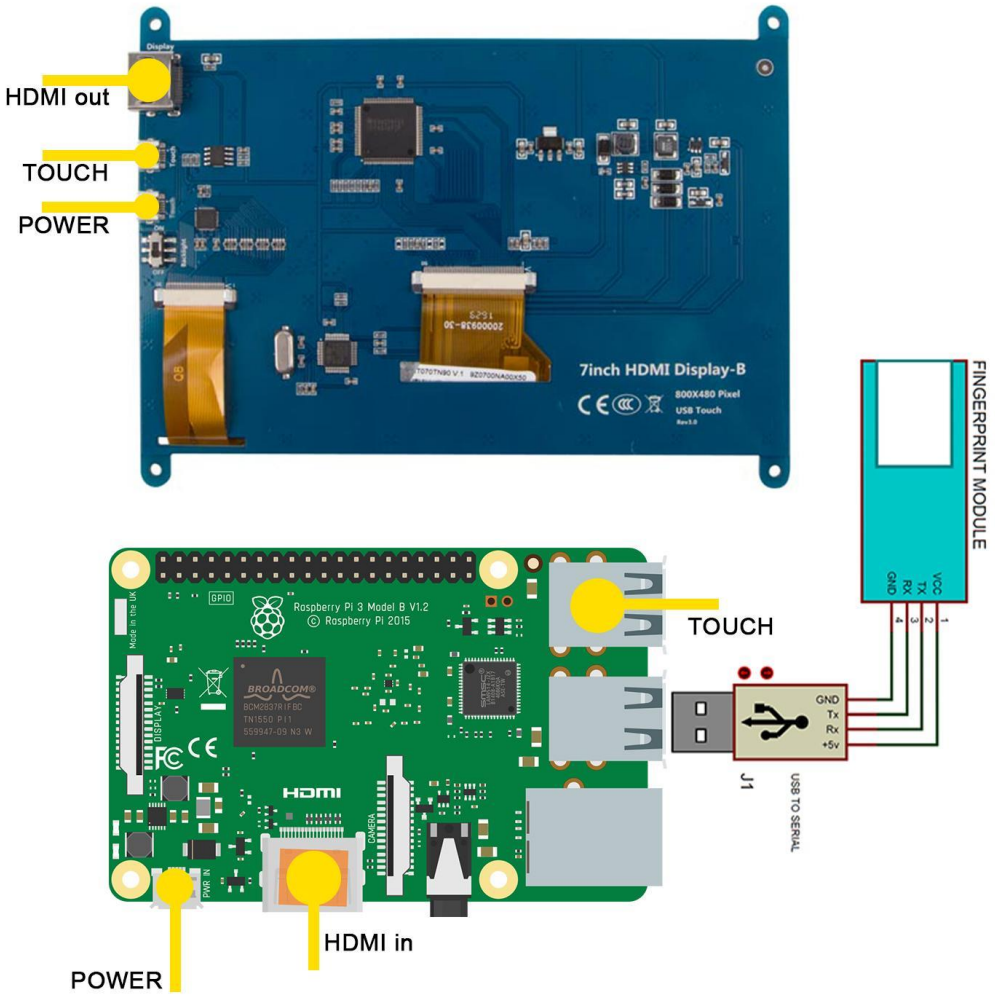


Figure 8-12 Hardware Setup

9 DATABASE MANAGEMENT

9.1 Software Used

9.1.1 SQLite

SQLite is an in-process library that implements a self-contained, server less, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.
In this biometric based EVM system SQLite acts as one of the most important elements. SQLite provides a platform for this system to manage the heavy database and to update the database from time to time. SQLite is used to compute the results faster and give more insights with the available data. The creation tables and the data entered in these tables is done on SQLite.

9.1.2 DB browser for SQLite

SQLite Database browser is a light GUI editor for SQLite databases, built on top of Qt. The main goal of the project is to allow non-technical users to create, modify and edit SQLite databases using a set of wizards and a spreadsheet-like interface.
DB browser for SQLite is used for a better visualization of the election database and for a simpler understanding of the data. This is also used to update the database with much ease.

9.1.3 Installation

9.1.3.1 SQLite:

SQLite comes pre-installed in Mac. For Windows we should follow these steps:

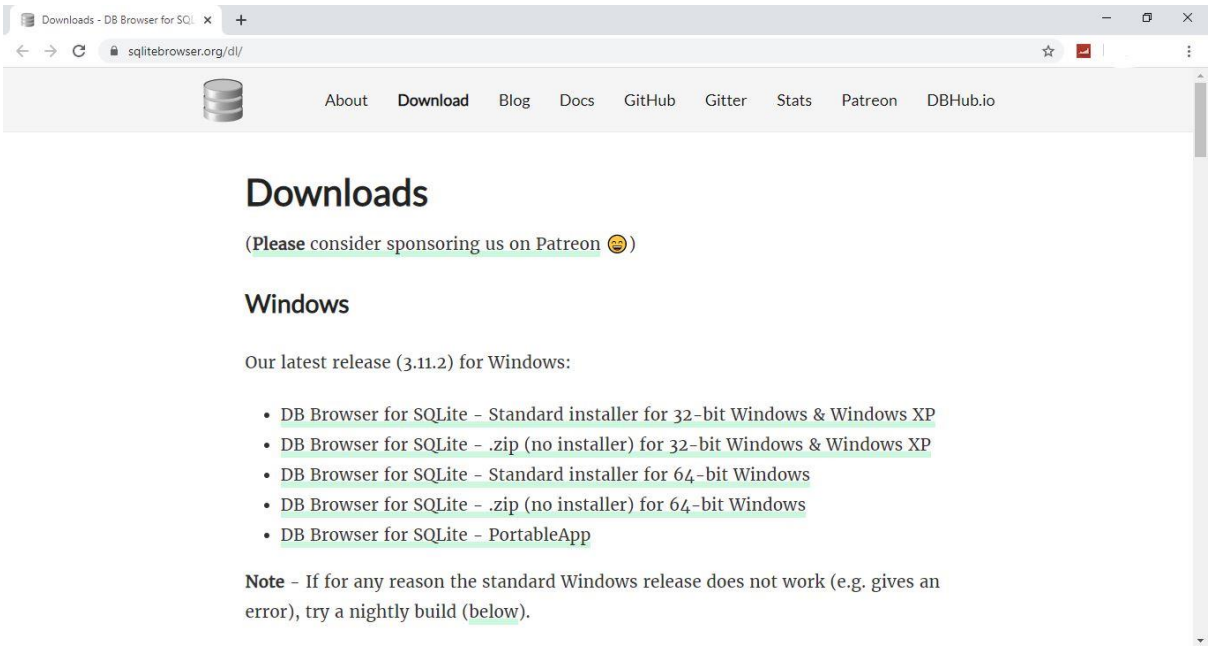
- Download the Zip file for SQLite website for your windows configuration.

Figure 9-1 Download link for SQLite

- Extract the files and open sqlite3.exe
- This program will open the command line and now you can process any SQL query.

9.1.3.2 DB browser for SQLite:

- Download the setup file from the official website.



Open the setup file and

Figure 9-2 DB Browser Installation

- Follow the instructions in the dialog box.
- Click install on the final dialog box.
- The installation process will start.
- After installation we can use the software.

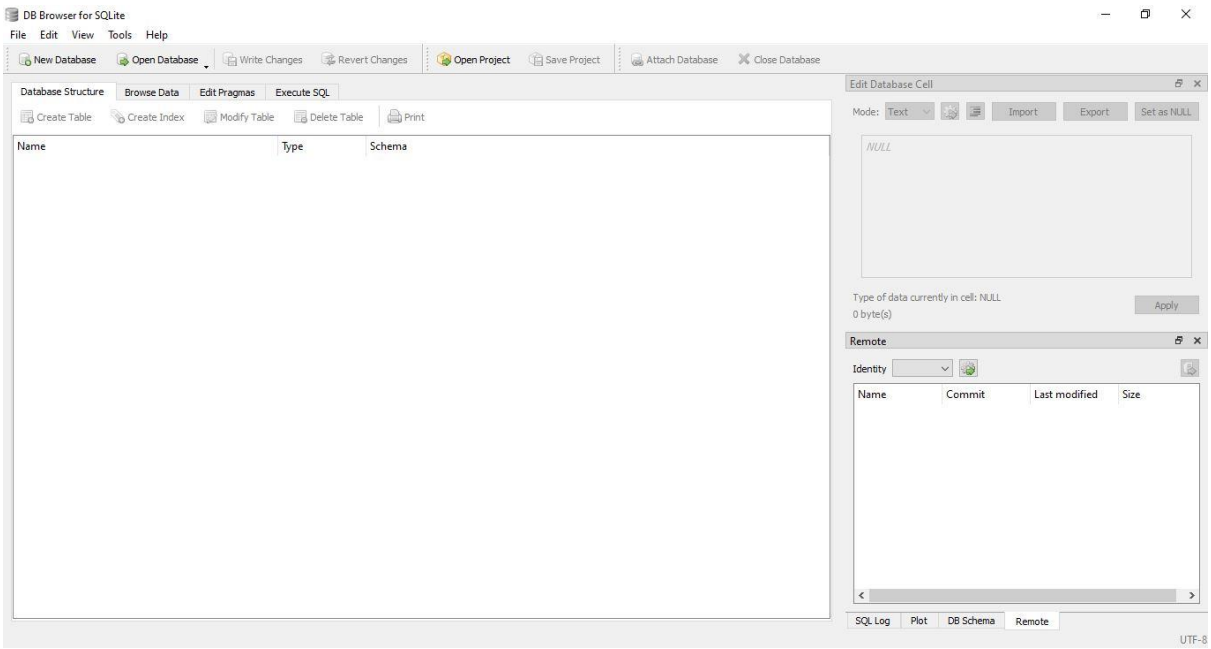


Figure 9-3 DB Browser Interface

9.1.4 Procedures:

To manage the data effectively we need 2 database tables

- Constituency table
- Voter table

The first constituency table will be multiple in number as different constituencies will have different tables. These will have details of the constituency. It will have the data of the participating candidates from that constituency in the elections.

Voter table will have data of all the voters which will be used for authentication, authorization and feedback.

We can visualize the data in DB Browser for SQLite software. In order to do so we have to create the above said tables.

9.1.4.1 Codes for creating the tables are as follows:

Constituency Table (Candidate):

```
CREATE TABLE "c_delhi" (
	"c_id"	NUMERIC NOT NULL UNIQUE,
	"name"	TEXT NOT NULL,
	"party"	TEXT NOT NULL,
	"sadd"	NUMERIC,
	"cadd"	NUMERIC,
	"count"	INTEGER,
	PRIMARY KEY("c_id")
);
```

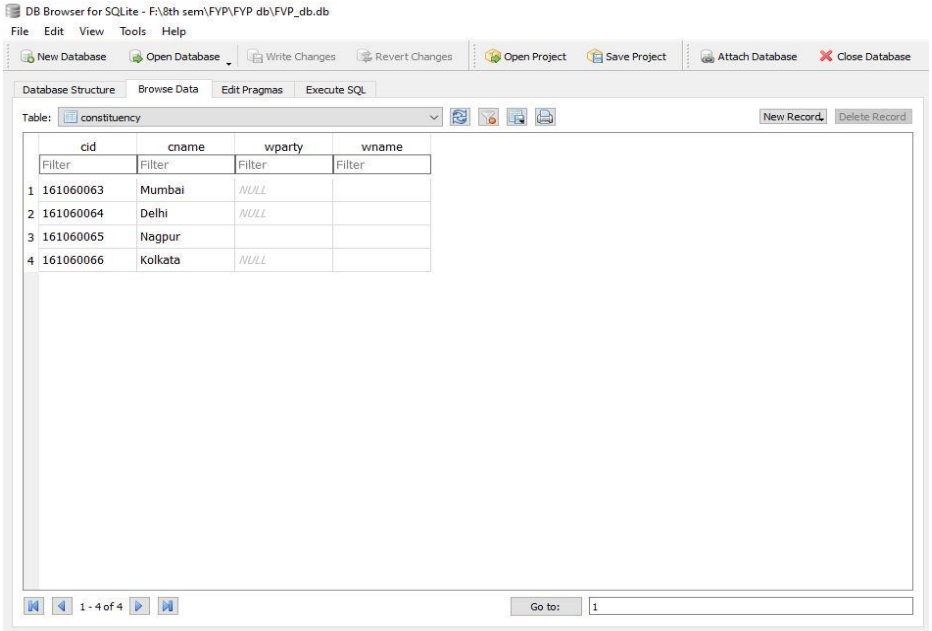


Figure 9-4 Constituency Table

Voter Table:

```
CREATE TABLE "voter" (
	"id"  INTEGER NOT NULL UNIQUE,
	"fhash"	NUMERIC UNIQUE,
	"name"	TEXT NOT NULL,
	"cst" TEXT NOT NULL,
	"phone"	NUMERIC NOT NULL,
	"vadd"	TEXT,
	"flag"	INTEGER NOT NULL,
	PRIMARY KEY("id")
);
```

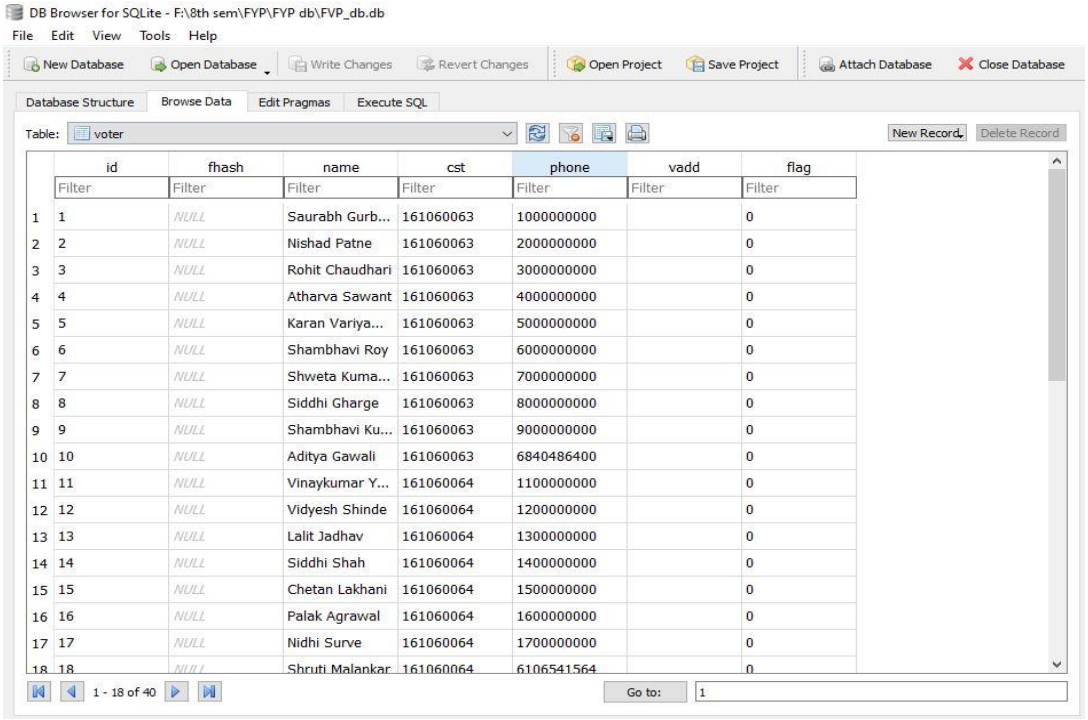


Figure 9-5 Voter Table

9.1.4.2 Counting

Counting of votes is an integral part of all the activities that are being carried out during elections. This has to be accurate, efficient and fast. By using the database management specified above counting process will be very easy, fast and at the same time reliable.

We can understand the process of vote registration from the figure below.

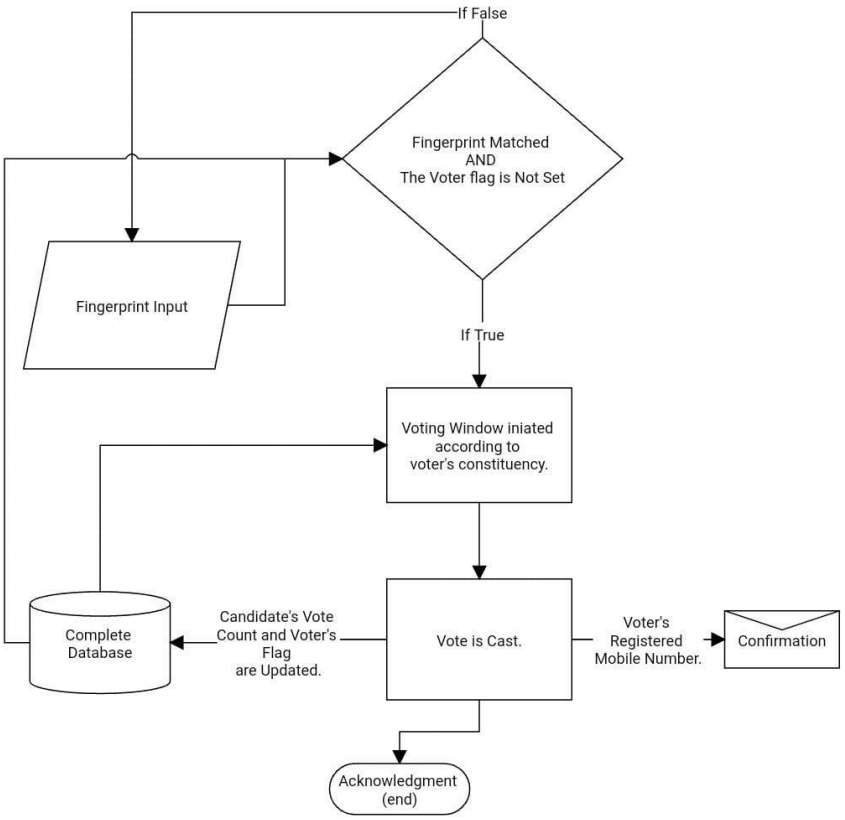


Figure 9-6 Counting Flowchart

Once the voter has casted the vote there will be an increment in the votes column of respective candidate. This process will go on till the voting process ends. When the voting process is finished we have to access the constituency table for results. This table will retrieve data from the constituency table for candidates of the two candidates who got the maximum and number of votes, party name etc. The candidate with more votes is the winner and the other one is the trailing candidate (lost candidate). The difference between the votes will appear in the margin column. All of this data will consist in the table for all constituencies. Thus the results can be announced accordingly.

10 GRAPHICAL USER INTERFACE

10.1 Software

The Kivy software is used for developing a graphic interface of application of Biometric based Electronic Voting Machine. Kivy is a free and open source Python library for developing mobile apps and other multitouch application software with a natural user interface (NUI). The framework contains all the elements for building an application like:

- Extensive input support for mouse, keyboard, TUIO, and OS-specific multitouch events,
- A graphic library using only opengl ES 2, and based on Vertex Buffer Object and shaders,
- A wide range of widgets that support multitouch,
- An intermediate language used to easily design custom widgets.

Figure 10-1 Kivy Logo

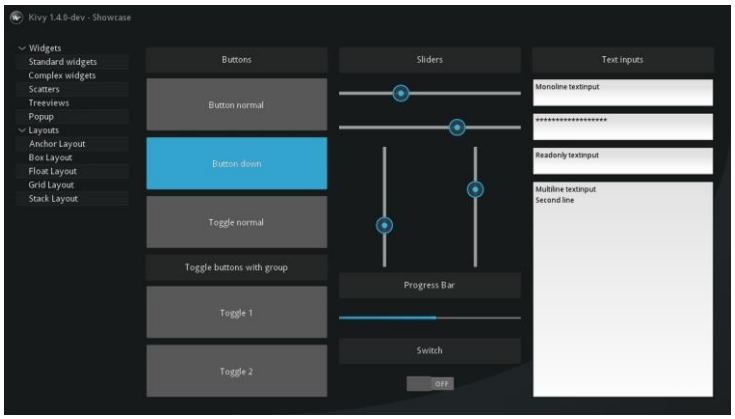


Figure 10-2 Kivy Example

10.1.1 Installation

The Kivy can be installed either manually or by downloading and booting KivyPie on the Raspberry Pi. We have performed Manual installation (On Raspbian Jessie/Stretch) for this project. The steps of installation are as follows [13]:

- Install the dependencies

```
sudo apt update
sudo apt install libsdl2-dev libsdl2-image-dev libsdl2-mixer-dev libsdl2-
ttf-dev \
pkg-config libgl1-mesa-dev libgles2-mesa-dev
python-setuptools libgstreamer1.0-dev git-core \
gstreamer1.0-plugins-{bad,base,good,ugly}
gstreamer1.0-{omx,alsa} python-dev libmtdev-dev \
xclip xsel libjpeg-dev
```

- Install pip dependencies:

```
python -m pip install --upgrade --user pip setuptools
python -m pip install --upgrade --user Cython==0.29.10 pillow
```

- Install Kivy to Python globally
  Install it like a normal python package with:

```
# to get the last release from pypi
python -m pip install --user kivy

# to install master
python -m pip install --user
https://github.com/kivy/kivy/archive/master.zip

# or clone locally then pip install
git clone https://github.com/kivy/kivy
cd kivy
python -m pip install --user
```

10.2 Voting Interface

The Electronic Voting Machine is concentrated around the appearance to voter. In a country like India, their is high technological illiteracy and to provide most simplified interface to voters is an important task. After all the EVM is not just a machine but also gadget so that poorest of poor people would be able to interact with democracy.

We have developed a user friendly interface which can help voter to understand the procedures to handle the EVM and vote should be explained simplifically. The different windows of voting interface are as follows:

10.2.1 Welcome window

The Welcome window is simple introductory interface which will be the first to interact with voter. It has a button named "Next to Vote" in the middle.

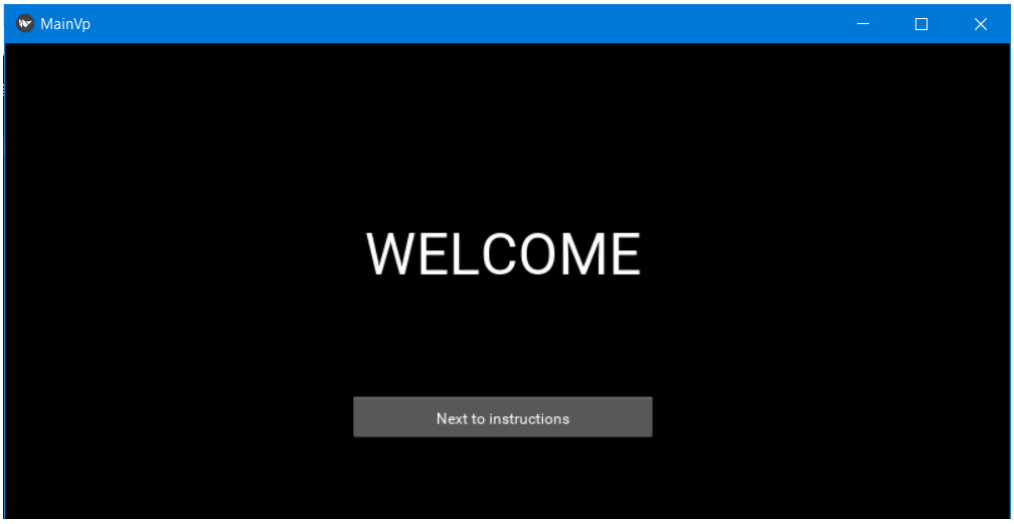


Figure 10-3 Welcome Window

10.2.2 Instruction window

The Instruction window has procedures to be followed by voters for the voting mentioned in it. The voting procedures mentioned as putting finger on scanner and waiting for the authentication, then pressing "Next to Vote" button again.

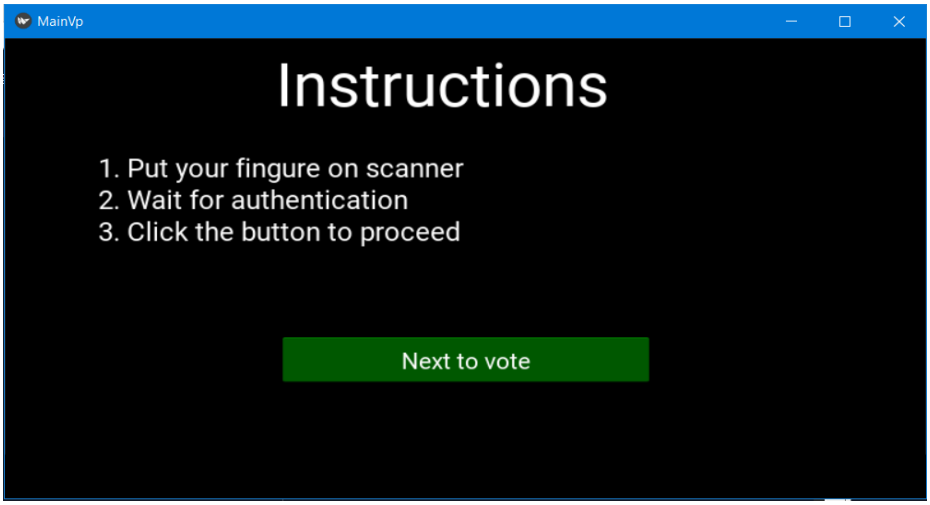


Figure 10-4 Instruction Window

10.2.3 Main voting window

The main voting window is divided in two parts as follows:

- Voter Details
  The voter details window has image, name, age, constituency and ID Number of the voter. Here the voter can verify all his details and proceed to candidate list to vote for candidate of his choice.
- Candidate List
  The candidate list is same part as traditional EVMs interface. All candidates of the constituency of voter are displayed here along with their Party and Party symbol. To choose the preferred candidate to vote, the voter can simply touch the button next to the candidate.

Figure 10-5 Main Voting Window

10.2.4 Confirmation window

The final confirmation window is simple text window with the message 'Your vote has been registered'. When the window appears, candidate will also get notification about his vote with the name of party he/she voted for. This verification by voter can be used instead of VVPATs of traditional EVMs.

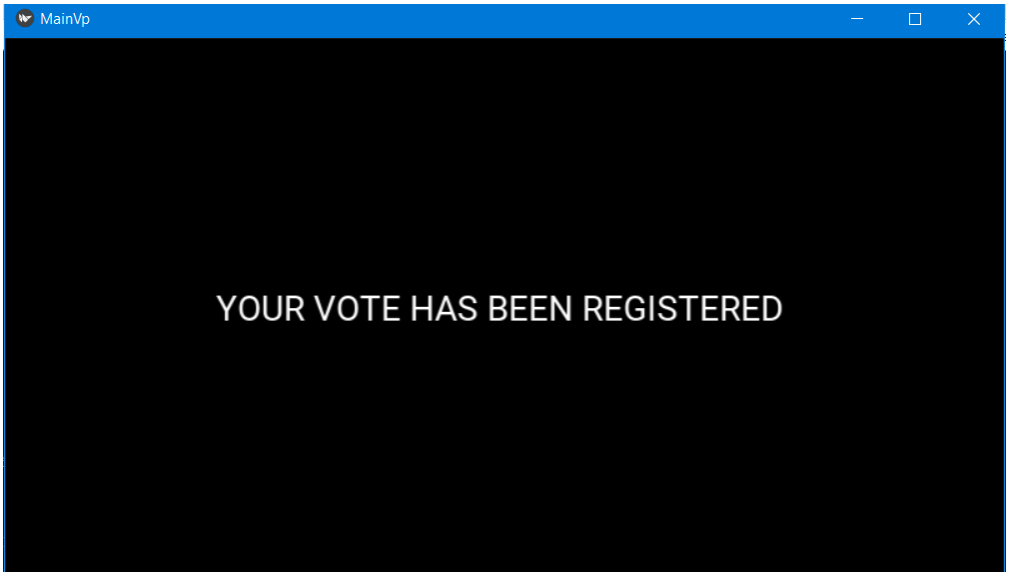


Figure 10-6 Confirmation Window

10.3 Counting Interface

The electoral process should be fast to prevent it from tampering and credibility questions. The traditional EVMs calculate the votes fast but only for that particular machine. Each constituency has thousands of booths and EVMs and to calculate the votes all have to manage a lot logistically. The gathering of all EVMs on counting day is time consuming process.

We have developed fast counting system of votes with simplified counting interface so that each EVM would be able to display the results as soon as possible after the voting. The different windows of counting interface are as follows:

10.3.1 Login window

Login window is simple entering window for election commission officials and booth officers. This window permits access to internal sections of EVMs like vote syncing and counting. It has two simple parameters to enter 'username' and 'password' with Login button.

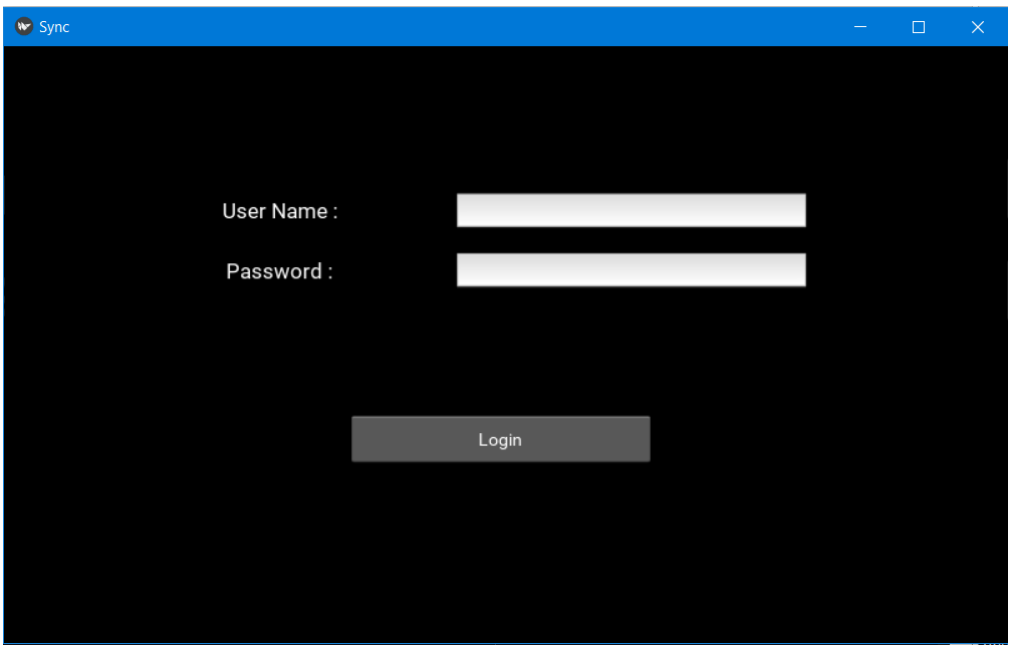


Figure 10-7 Login Window

10.3.2 Synchronization Window

Synchronization window is simple window which use to ask for permission to synchronize the vote data of that particular EVM with central counting cloud of that constituency. The button is 'Start Sync' is present there to initiate the same.

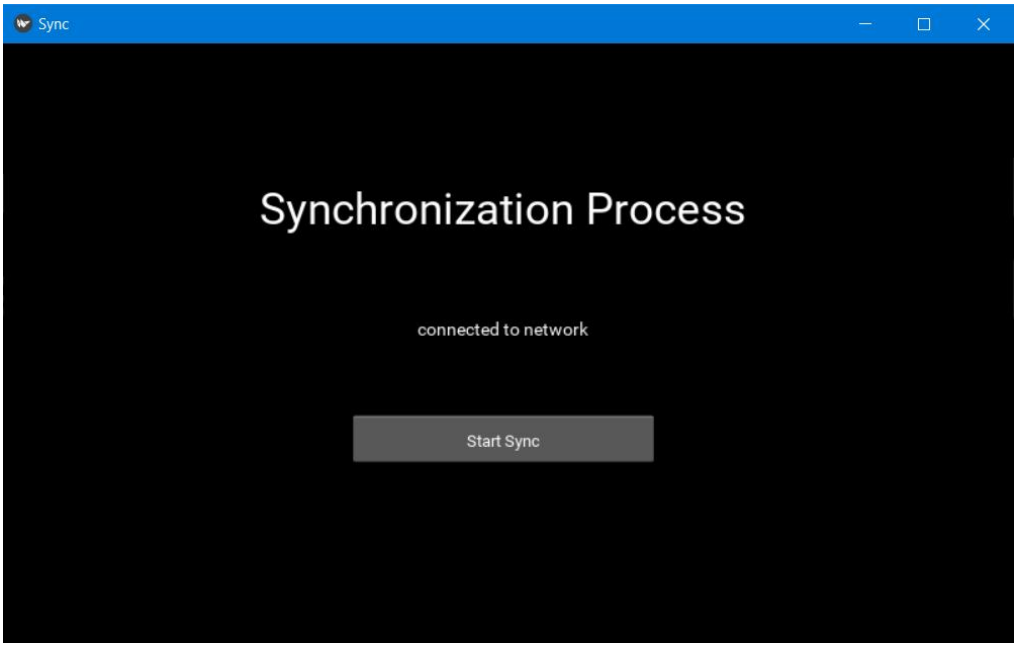


Figure 10-8 Synchronization Window

10.3.3 Dashboard window

The dashboard window is primary window to access results of each constituency. It will appear when syncing is complete. It has two main buttons 'Final Result' and 'Constituency Results' which direct to their respective windows. To compute final results the constituency results has to be calculate first. The individual can also directly sign out from this window.

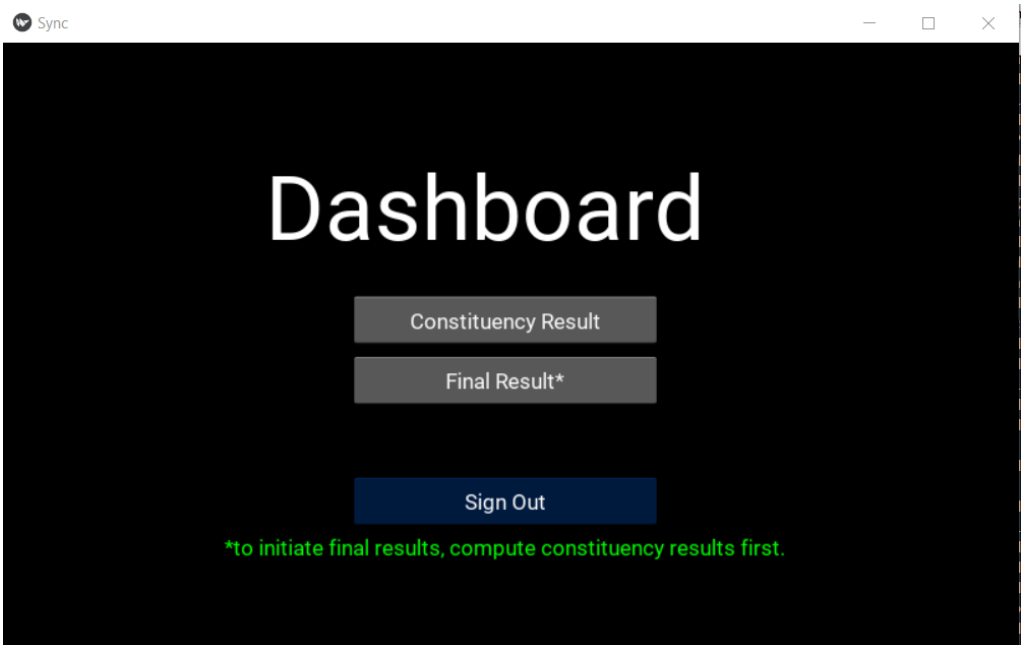


Figure 10-9 Dashboard

10.3.4 Constituency Result window

Constituency result window will display the winning candidates and parties of all constituencies. We can proceed to individual constituency counts from here by just clicking on a constituency name.

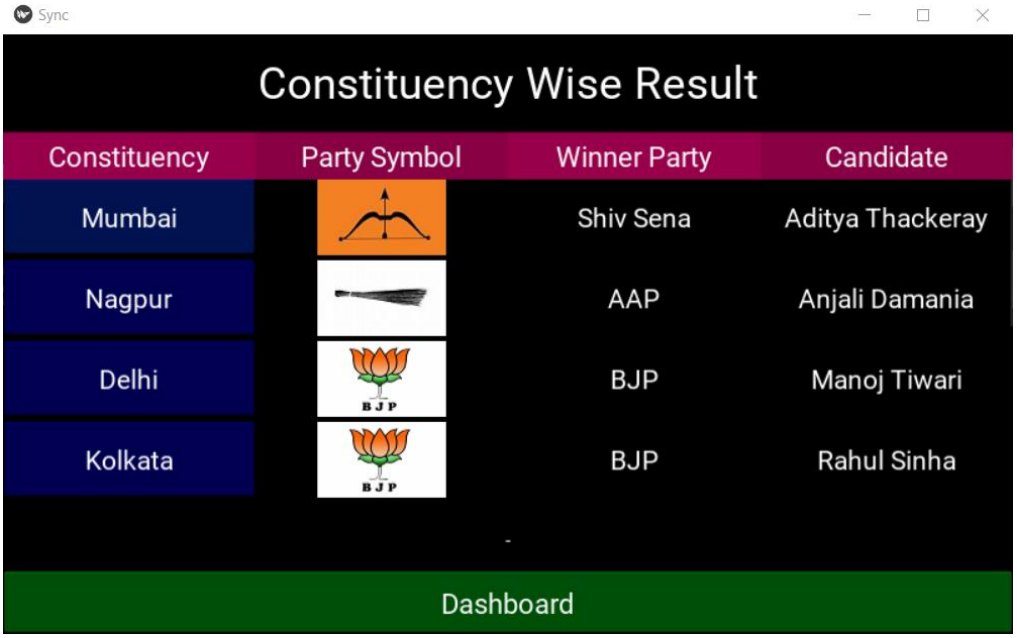


Figure 10-10 Constituency Window

10.3.5 Constituency wise results window

This window displays the votes counted for each candidate of particular constituency in a table. It also allows to return to dashboard so that one can return to see final result.

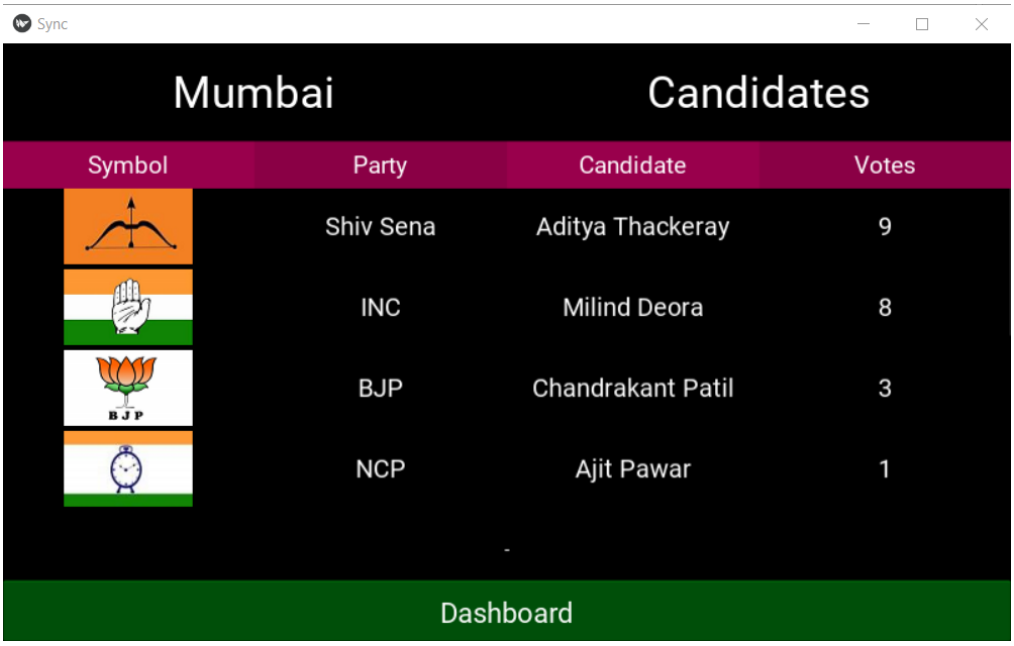


Figure 10-11 Mumbai Constituency Result

10.3.6 Final Result window

This is final result window of party wise seats. It displays all parties and their total seats won in whole election. This window directly gives analysis of the majority constituency winning party which can be invited to form government.

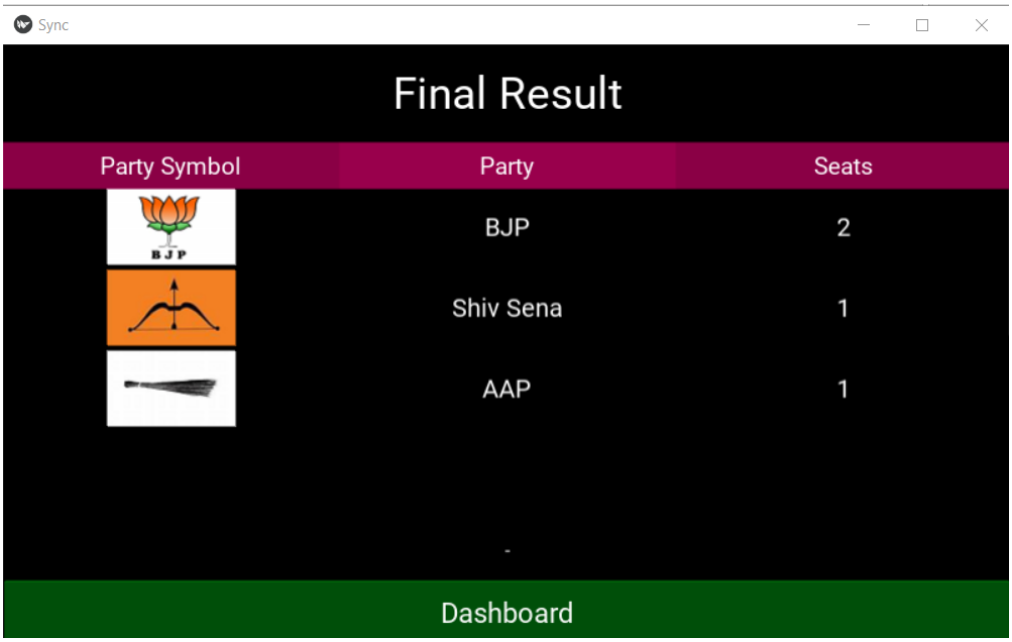


Figure 10-12 Final Result

11 MISCELLANEOUS

11.1 Voting Flowchart

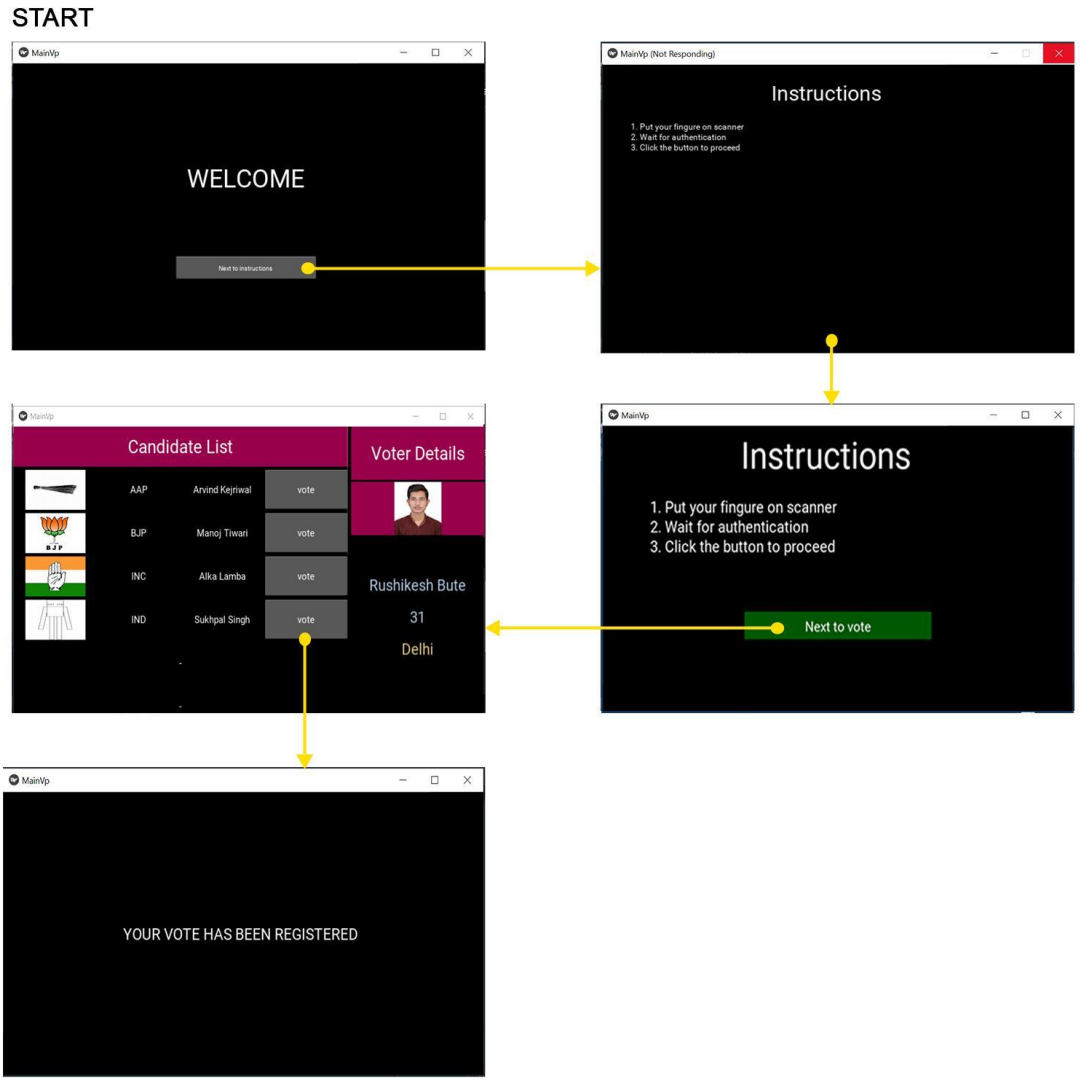


Figure 11-1 Voting Flow Chart

11.2 Counting Flowchart

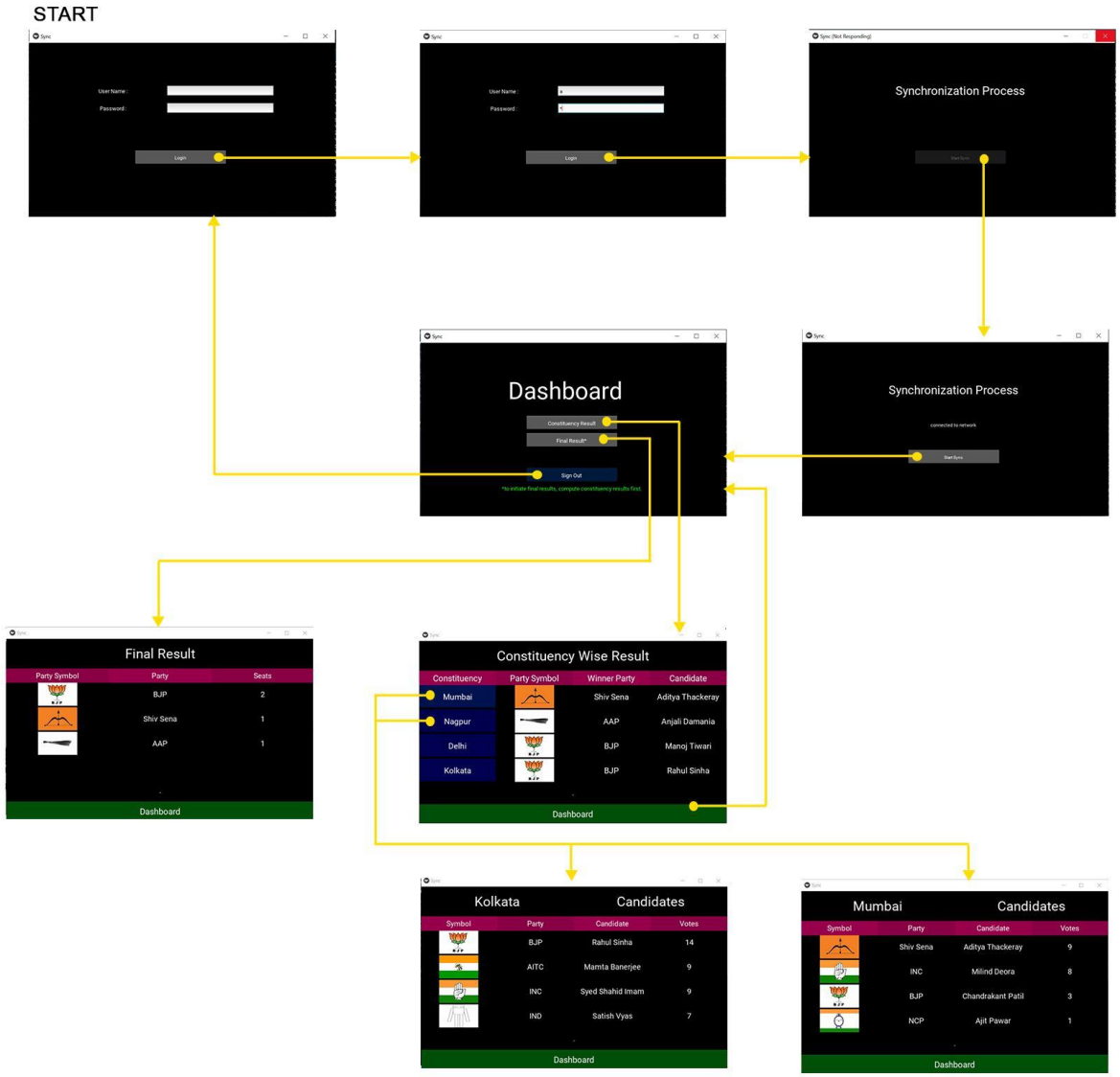


Figure 11-2 Counting Flow Chart

11.3 SMS Verification System

The Traditional EVMs use Voter-verified paper audit trail (VVPAT) to give confirmation to voters about their voting. The Biometric based Electronic Voting Machine uses a messaging service to inform voters about their vote and party which they voted for. In this project we have used third party system named FAST2SMS to send voter confirmation. The received message on voter's personal mobile phone is as:

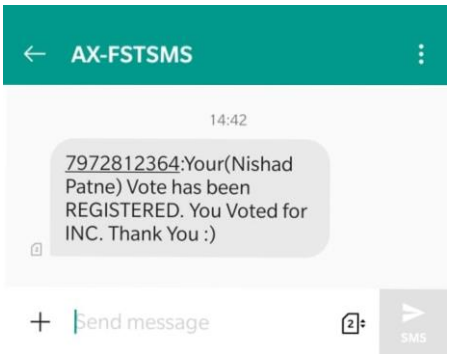


Figure 11-3 Example Message

11.3.1 Fast2SMS:

It is API SMS service which can be used for bulk messaging. Fast2SMS expects for the API Key to be included in all API requests to the server in a header for POST requests & in query parameters for GET requests.

Installation:

- Sign-up for a free account on the Fast2sms website.
- Use the bulk sms API for the SMS verification.
- API

```
import requests
class Sms():
    def sendtxt(self, number):
        url = "https://www.fast2sms.com/dev/bulk"
        payload = "sender_id=FSTSMS&message=Your Vote had been REGISTERED.
Thank You&language=english&route=p&numbers={}".format(number)
        headers = {
            'authorization':
"TK1kNbDAi06fwCHB7ZYjyPthzds8g3xvmcFeJSuLlGqa4onrIU4FIWbPrjU3XQYT6qGvwDp2Zk
Hx8f90",
            'Content-Type': "application/x-www-form-urlencoded",
            'Cache-Control': "no-cache",
            }
        response = requests.request("POST", url, data=payload,
headers=headers)
        print(response.text)
```

12 RESULT

The Biometric based Electronic Voting Machine was tested successfully at all objectives. The results of the system were positive and reliable. The fingerprint scanning system is providing fast authentication and entry to voting for their respective constituency. The verification message with name and party they have voted for is quickly delivered on the registered mobile number of voter. The machine keeps the count of all votes flawlessly without revealing the secrecy of voters. The counts sync to the server first before entering to calculate results of election. These results provide a silver line to use this system for voting than traditional EVMs.

13 CONCLUSION

As India is considered as the largest democracy in the world, Safe and accurate elections must take place to protect democracy in India. By using this Biometric based EVM we can ensure the most secure, safe and accurate elections. Biometric authentication makes sure that no fake votes are casted and prevents multiple voting. The centralized database used to cast distance voting and solve the participation problems of migrants in elections. The instant results lower the chances of tampering the EVMs for result and display the nationwide results on each Machine instantly when Election Day ends. This way Biometric based EVM helps to overcome the drawbacks of the current EVM system. As current EVM's integrity and accuracy is in doubt as a result of various elements in society and officials. Therefore we can say that implementation of Biometric based EVM can provide a transparent, fair, secure and accurate election process in India and thus hold strong roots of democracy in our country.

14 LIMITATIONS AND FUTURE SCOPE

1. The Biometric based Electronic Voting Machine tries to fulfil all the objectives throughout the project. It provides a way through to conduct safe and accurate elections. However the model has some limitations.
2. The verification of fingerprints may take some time extra than expected for the centralized database due to huge number of voters for election.
3. The Biometric authentication of voters can be challenging for physically disabled people.
4. Performance can be fluctuate to environmental conditions, badly maintained systems, age groups, etc.
5. The Biometric identifiers of voters cannot be changed, if compromised.
6. Distance voting can lead to problem of voters management in densely populated areas.
7. There are chances of breaching the secrecy of mobile phones of voters to learn about their voting patterns.
8. The reduction in time to vote and count can lead to administering as well as law and order issues.

These constraints can be overcome in future to make this system more trustworthy and reliable.
First thing can be done is, the system of authentication with fingerprint scanning can be extended to a multi-biometric system including retina scan, face recognition etc. Second, the UIDAI already has the biometric database of all Indian nationals in the form of AADHAR [5]. The biometric data can be used as a centralize voters database and can be modified from election to elections as per requirement. The already existing data would reduce the the efforts to allot voter id cards to each new voter and can directly recommend him to vote when he/she is qualified to vote by age. Third, the voting process can be shifted to Blockchain Technology for better security of the elections and lowering the chances of tampering.

15 REFERENCES

[1] Use technology to improve governance and timely delivery of services to citizens: Vice President, Press Information Bureau, Government of India
https://pib.gov.in/newsite/PrintRelease.aspx?relid=190477

[2] Let vote move with voter: Why it's not a reality yet, JANUARY 08, 2019, Times Of India
https://m.timesofindia.com/india/let-vote-move-with-voter-why-its-not-a-reality-yet/amp_articleshow/67429573.cms

[3] Political Inclusion of Seasonal Migrant Workers in India: Perceptions, Realities and Challenges, Aajeevika Bureau (Udaipur)
http://www.aajeevika.org/assets/pdfs/Political-Inclusion-of-Migrant-Workers-in-India.pdf

[4] Manual on Electronic Voting Machine and VVPAT, July 2018, Election Commission of India
https://eci.gov.in/files/file/8992-manual-on-electronic-voting-machine-and-vvpat-july-2018/

[5] Unique Identification Authority of India
https://uidai.gov.in/about-uidai/unique-identification-authority-of-india.html

[6] Security Analysis of India's Electronic Voting Machines, J. Alex Halderman, Hari K. Prasad, Rop Gonggrijp, Netindia, (P) Ltd., Hyderabad & The University of Michigan
https://indiaevm.org/evm_tr2010-jul29.pdf

[7] More than 42 lakh bogus voters in Maharashtra: Congress to Election Commission, September 19, 2019, Times Of India
https://www.indiatoday.in/elections/maharashtra-assembly-election/story/bogus-voters-maharashtra-congress-election-commission-1600623-2019-09-19
https://eci.gov.in/evm

[8] A Proposed Framework for Biometric Electronic Voting System by Md. Mahboob Karim1, Nabila Shahnaz Khan2, Ashratuz Zavin3, Shusmoy Kundu4, Asibul Islam, Brazab Nayak.

[9] https://usa.kaspersky.com/resource-center/definitions/biometrics

[10] How to Install Raspbian OS on Raspberry Pi
https://itsfoss.com/tutorial-how-to-install-raspberry-pi-os-raspbian-wheezy/

[11] How to use a Raspberry Pi Fingerprint Sensor for Authentication
https://tutorials-raspberrypi.com/how-to-use-raspberry-pi-fingerprint-sensor-authentication/

[12] Kivy Installation on Raspberry Pi
https://kivy.org/doc/stable/installation/installation-rpi.html

[13] FAST2SMS API
www.fast2sms.com

16 APPENDIX I (HARDWARE SPECIFICATIONS)

16.1 Raspberry Pi 3 Model B

Figure 16-1 Raspberry Pi 3.0 B

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

16.2 Fingerprint Sensor R305

Figure 16-2 Fingerprint Scanner R305

- Power DC : 3.6V-6.0V
- Interface : UART (TTL logical level)/ USB 1.1
- Working current : 100mA
- Peak Current : 150mA
- Matching Mode: 1:1 and 1:N
- Baud rate (9600*N)bps, N=1-12 (default N=6 57600bps)
- Character file size: 256 bytes
- Image acquiring time : <0.5s
- Template size : 512 bytes
- Storage capacity: 256
- Security level : 5 (1, 2, 3, 4, 5(highest))
- FAR : <0.001%
- FRR: <0.1%
- Average searching time: < 0.8s (1:880)
- Window dimension : 18mm*22mm

16.3 7" inch LCD Touch Display

Figure 16-3 LCD TFT Display

- 800×480 high resolution.
- Capacitive touch control.
- Supports Raspberry Pi, and driver is provided (works with custom Raspbian directly).
- Supports BB Black, comes with related images like : Angstrom.
- Supports Banana Pi / Banana Pro, comes with related images like : ubuntu, Raspbian.
- Not only for mini-PCs, it can work as a computer monitor just like any other general HDMI screen (touch function is unavailable in this case).
- HDMI interface for displaying, USB interface for touch control.
- Back light control to lower power consumption.

17 APPENDIX II (SOFTWARE SPECIFICATIONS)

17.1 Python 3.5+

Figure 17-1 Python Logo

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++. It provides large standard libraries that include the areas like string operations, Internet, web service tools, operating system interfaces and protocols. Most of the highly used programming tasks are already scripted into it that limits the length of the codes to be written in Python.

17.2 Raspbian OS

Figure 17-2 Raspbian Logo

Raspbian is a Debian-based computer operating system for Raspberry Pi. There are several versions of Raspbian including Raspbian Buster and Raspbian Stretch. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers. Raspbian comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspberry Pi.

17.3 Kivy

Figure 17-3 Kivy Logo

Kivy is a free and open source Python library for developing mobile apps and other multitouch application software with a natural user interface (NUI). It is distributed under the terms of the MIT License. The Kivy language (Kv) is a language dedicated to describing user interface and interactions. As with other user interface markup languages, it is possible to easily create a whole UI and attach interaction.

17.4 SQLite

Figure 17-4 SQLite Logo

SQLite is a relational database management system contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. The SQLite code base is supported by an international team of developers who work on SQLite full-time. The developers continue to expand the capabilities of SQLite and enhance its reliability and performance while maintaining backwards compatibility with the published interface spec, SQL syntax, and database file format. The source code is absolutely free to anybody who wants it, but professional support is also available. The SQLite project was started on 2000-05-09. The future is always hard to predict, but the intent of the developers is to support SQLite through the year 2050. Design decisions are made with that objective in mind.

17.5 DB Browser for SQL

Figure 17-5 DB Browser Logo

DB Browser for SQLite (DB4S) is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite. DB4S is for users and developers who want to create, search, and edit databases. DB4S uses a familiar spreadsheet-like interface, and complicated SQL commands do not have to be learned.

17.6 Fast2SMS

Figure 17-6 Fast2SMS logo

Fast2SMS simple platform helps to send promotional, marketing, OTP, multimedia & alerts SMS. Fast2SMS Bulk SMS API work with Java, PHP, C#, Python, Ruby, JavaScript. Fast2SMS Bulk SMS Service is used by various national companies, school, college, organization, developers, NGOs & government agencies for promotion, communication, transactional alerts, notification, reminders, OTP and feedback.

18 APPENDIX III (PYTHON CODES)

18.1 Voting process

18.1.1 Python code

```
import kivy
kivy.require('1.0.6')
from evmdb import Database
from evmdb import Voter
from evmdb import Candidate
from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.lang import Builder
from sms import Sms
from fig_auth import fingureprint

# Screen loaders
class StartWindow(Screen):
    pass

class InstWindow(Screen):


    def on_enter(self):
        pass

    def on_pre_enter(self):
        pass

    def onr(self):
        btn = self.ids['btn']
        btn.opacity = 0

    def onp(self):
        btn = self.ids['btn']
        rep = 1
        fp = fingureprint()
        while rep == 1:
            fp.search()
            fing = fp.fhash
            db.cmp_fp(fing)
            db.fhash = fing
            if db.flag == "set":
                btn.background_color = 0,1,0,1
                btn.opacity = 1
                rep = 0
            else:
                rep =1
```

```
class VoteWindow(Screen):


    def initialise(self):
        self.v = Voter(db.fhash)
        self.sms = Sms()
        self.can1 = Candidate(1,self.v.vcst)
        self.can2 = Candidate(2,self.v.vcst)
        self.can3 = Candidate(3,self.v.vcst)
        self.can4 = Candidate(4,self.v.vcst)

    def on_pre_enter(self):
        pass

    def details(self):
        self.initialise()
        self.cdetails()
        self.vdetails()

    def upcnt1(self):
        self.sms.sendtxt(self.v.vcno,self.v.vname,self.can1.cparty)
        self.v.update()
        self.can1.update()
        for x in range(0,1000):
            print(x)

    def upcnt2(self):
        self.sms.sendtxt(self.v.vcno,self.v.vname,self.can2.cparty)
        self.v.update()
        self.can2.update()
        for x in range(0,1000):
            print(x)

    def upcnt3(self):
        self.sms.sendtxt(self.v.vcno,self.v.vname,self.can3.cparty)
        self.v.update()
        self.can3.update()
        for x in range(0,1000):
            print(x)

    def upcnt4(self):
        self.sms.sendtxt(self.v.vcno,self.v.vname,self.can4.cparty)
        self.v.update()
        self.can4.update()
        for x in range(0,1000):
            print(x)

    def cdetails(self):

        c1party = self.ids['c1party']
        c1name = self.ids['c1name']
        c1img = self.ids['c1img']
```

```
        c1name.text = self.can1.cname
        c1party.text = self.can1.cparty
        c1img.source = self.can1.csadd

        c2party = self.ids['c2party']
        c2name = self.ids['c2name']
        c2img = self.ids['c2img']

        c2name.text = self.can2.cname
        c2party.text = self.can2.cparty
        c2img.source = self.can2.csadd

        c3party = self.ids['c3party']
        c3name = self.ids['c3name']
        c3img = self.ids['c3img']

        c3name.text = self.can3.cname
        c3party.text = self.can3.cparty
        c3img.source = self.can3.csadd

        c4party = self.ids['c4party']
        c4name = self.ids['c4name']
        c4img = self.ids['c4img']

        c4name.text = self.can4.cname
        c4party.text = self.can4.cparty
        c4img.source = self.can4.csadd

    def vdetails(self):
        vname = self.ids['vname']
        vuid = self.ids['vuid']
        vconst = self.ids['vconst']
        vpic = self.ids['vpic']
        vname.text = self.v.vname
        vuid.text = str(self.v.vid)
        vconst.text = self.v.vcst
        vpic.source = self.v.viadd

class AckWindow(Screen):

    def on_enter(self):
        pass

    def stag(self):

        for x in range(0,50000):
            print(x)

        self.manager.transition.direction = "down"
        sm.current = "start"

##############################################################################
```

```
class WindowManager(ScreenManager):
    pass

kv = Builder.load_file("mainvp.kv")
sm = WindowManager()
db=Database()

screens = [StartWindow(name="start"),
InstWindow(name="instructions"),VoteWindow(name="vote"),
AckWindow(name="acknowledgement")]
for screen in screens:
    sm.add_widget(screen)

sm.current = "start"

class MainVpApp(App):

    def build(self):
        return sm

if __name__ == '__main__':
    MainVpApp().run()
```

18.1.2 Kivy Code

```
<StartWindow>
    name: "start"
    FloatLayout:
        cols: 1

        Label:
            pos_hint:{"x":0.32,"top":0.7}
            size_hint: 0.35, 0.2
            font_size: (root.width**2 + root.height**2) / 11.5**4
            text: "WELCOME"


        Button:
            pos_hint:{"x":0.345,"y":0.25}
            size_hint: 0.3, 0.08
            font_size: (root.width**2 + root.height**2) / 16**4
            text: "Next to instructions"
            on_release:
                root.manager.transition.direction = "left"
                app.root.current = "instructions"


<InstWindow>
```

```
    name: "instructions"
    on_enter: root.onp()
    on_pre_enter: root.onr()
    btn : btn

    FloatLayout:
        Label:
            pos_hint:{"x":0.3,"top":1}
            size_hint: 0.35, 0.2
            font_size: (root.width**2 + root.height**2) / 11**4
            text: "Instructions"

        Label:
            pos_hint:{"x":0.2,"top":0.7}
            size_hint: 0.2, 0.1
            font_size: (root.width**2 + root.height**2) / 13.5**4
            text: " 1. Put your fingure on scanner \n 2. Wait for authentication
\n 3. Click the button to proceed "


        Button:
            id: btn
            pos_hint:{"x":0.3,"y":0.25}
            size_hint: 0.4, 0.1
            font_size: (root.width**2 + root.height**2) / 14**4
            background_color: 1,0,0,1
            text: "Next to vote"

            on_release:
                root.manager.transition.direction = "left"
                app.root.current = "vote"


<VoteWindow>
    name: "vote"
    #voter details

    vname:vname
    vudi: vuid
    vconst: vconst
    vpic: vpic

    #candidate details

    c1img: c1img
    c1name: c1name
    c1party: c1party
    c1btn: c1btn

    c2img: c2img
    c2name: c2name
    c2party: c2party
    c2btn: c2btn
```

```
c3img: c3img
c3name: c3name
c3party: c3party
c3btn: c3btn

c4img: c4img
c4name: c4name
c4party: c4party
c4btn: c4btn


on_pre_enter: root.details()


BoxLayout:
    spacing: 5

    ScrollView:
        GridLayout:
            orientation: "vertical"
            spacing: 5
            size_hint_y: None
            height: self.minimum_height
            row_default_height: 60
            cols:1

            BoxLayout:
                Label:
                    text: "Candidate List"
                    color: 1,1,1,1
                    font_size: (root.width**2 + root.height**2) / 13**4
                    canvas.before:
                        Color:
                            rgba: 0.6,0,0.3,1
                        Rectangle:
                            pos: self.pos
                            size: self.size

            BoxLayout:
                Image:
                    id: c1img

                Label:
                    id: c1party


                Label:
                    id: c1name


                Button:
                    id: c1btn
```

```
            text: "vote"
            on_release:
                root.upcnt1()
                root.manager.transition.direction = "left"
                app.root.current = "acknowledgement"

    BoxLayout:
        Image:
            id: c2img

        Label:
            id: c2party


        Label:
            id: c2name


        Button:
            id: c2btn
            text: "vote"
            on_release:
                root.upcnt2()
                root.manager.transition.direction = "left"
                app.root.current = "acknowledgement"

    BoxLayout:
        Image:
            id: c3img

        Label:
            id: c3party


        Label:
            id: c3name


        Button:
            id: c3btn
            text: "vote"
            on_release:
                root.upcnt3()
                root.manager.transition.direction = "left"
                app.root.current = "acknowledgement"

    BoxLayout:
        Image:
            id: c4img

        Label:
            id: c4party
```

```
            Label:
                id: c4name


            Button:
                id: c4btn
                text: "vote"
                on_release:
                    root.upcnt4()
                    root.manager.transition.direction = "left"
                    app.root.current = "acknowledgement"

        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"
        Label:
            text:"-"

BoxLayout:
    orientation: "vertical"
    size_hint: 0.4,1
    ScrollView:
        GridLayout:

            orientation: "vertical"
            spacing: 3
            size_hint_y: None
            height: self.minimum_height
            row_default_height: 80
            cols:1

            Label:
                font_size: (root.width**2 + root.height**2) / 13**4
                text: "Voter Details"
                color: 1,1,1,1
                canvas.before:
                    Color:
```

```
                            rgba: 0.6,0,0.3,1
                        Rectangle:
                            pos: self.pos
                            size: self.size

                Image:
                    id: vpic
                    canvas.before:
                        Color:
                            rgba: 0.6,0,0.3,1
                        Rectangle:
                            pos: self.pos
                            size: self.size

        ScrollView:

            GridLayout:
                orientation: "vertical"
                spacing: 3
                size_hint_y: None
                height: self.minimum_height
                row_default_height: 45
                cols:1

                Label:
                    id: vname
                    font_size: (root.width**2 + root.height**2) / 13.5**4
                    color: 0.7,0.8,0.9,1
                    text: ""

                Label:
                    id: vuid
                    font_size: (root.width**2 + root.height**2) / 13.5**4
                    text: ""
                    color: 0.7,0.8,0.9,1

                Label:
                    id: vconst
                    font_size: (root.width**2 + root.height**2) / 13.5**4
                    text: ""
                    color: 0.9,0.8,0.5,1

                Label:


                Label:



<AckWindow>
    name: "acknowledgement"
    on_enter: root.stag()
    FloatLayout:
```

```
Label:
    pos_hint:{"x":0.32,"top":0.6}
    size_hint: 0.35, 0.2
    font_size: (root.width**2 + root.height**2) / 13**4
    text: "YOUR VOTE HAS BEEN REGISTERED"
```

18.2 Synchronization Process

18.2.1 Python Code

```
import kivy
kivy.require('1.0.6')
from evmdb import Database
from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.lang import Builder


# Screen loaders
class LoginWindow(Screen):

    uName= "a"
    pName= "k"

    def validation(self):
        uname= self.ids['uname']
        pname= self.ids['pname']
        war= self.ids['war']
        war.opacity = 0

        if uname.text == self.uName and pname.text == self.pName:
             sm.current = "buffer"

        else:
            war.opacity = 1
            war.text ="invalid credentials"

            sm.current = "login"

    def clear(self):
        uname= self.ids['uname']
        pname= self.ids['pname']
        uname.text= ""
        pname.text=""

class BufferWindow(Screen):
    def switch(self):
         sm.current = "dash"

    def popy(self):
        pop= self.ids['pop']
        flag =input("press 1")
        flag = int(flag)
        if(flag == 1):
            pop.text="connected to network"
```

```
            btn= self.ids['btn']
            btn.disabled=False

class DashWindow(Screen):

    def switch(self,txt):
         sm.current = txt

class ConstiWindow(Screen):
    def switch(self):

         sm.current = "dash"

    def initwids(self):
        mum= self.ids['mum']
        mlogo = self.ids['mlogo']
        mwp= self.ids['mwp']
        mwcan = self.ids['mwcan']

        nag= self.ids['nag']
        nlogo = self.ids['nlogo']
        nwp= self.ids['nwp']
        nwcan = self.ids['nwcan']

        dli= self.ids['dli']
        dlogo = self.ids['dlogo']
        dwp= self.ids['dwp']
        dwcan = self.ids['dwcan']

        kol= self.ids['kol']
        klogo = self.ids['klogo']
        kwp= self.ids['kwp']
        kwcan = self.ids['kwcan']

        db.constit(mum.text,nag.text,dli.text,kol.text)

        mlogo.source= db.constis[0].wpsymb
        mwp.text = db.constis[0].wpname
        mwcan.text = db.constis[0].wpcand

        nlogo.source= db.constis[1].wpsymb
        nwp.text = db.constis[1].wpname
        nwcan.text = db.constis[1].wpcand

        dlogo.source= db.constis[2].wpsymb
        dwp.text = db.constis[2].wpname
        dwcan.text = db.constis[2].wpcand

        klogo.source= db.constis[3].wpsymb
        kwp.text = db.constis[3].wpname
        kwcan.text = db.constis[3].wpcand

    def on_pre_enter(self):
```

```
        pass

# switches
    def kolkata(self):
        db.currentconsti = 3
        sm.current = "candi"

    def mumbai(self):
        db.currentconsti = 0
        sm.current = "candi"

    def delhi(self):
        db.currentconsti = 2
        sm.current = "candi"

    def nagpur(self):
        db.currentconsti = 1
        sm.current = "candi"

class CandiWindow(Screen):

    def on_pre_enter(self):
        pass

    def initwids(self):

        c1name = self.ids['c1name']
        c1pic = self.ids['c1pic']
        c1party= self.ids['c1party']
        c1votes= self.ids['c1votes']

        c2name = self.ids['c2name']
        c2pic = self.ids['c2pic']
        c2party= self.ids['c2party']
        c2votes= self.ids['c2votes']

        c3name = self.ids['c3name']
        c3pic = self.ids['c3pic']
        c3party= self.ids['c3party']
        c3votes= self.ids['c3votes']

        c4name = self.ids['c4name']
        c4pic = self.ids['c4pic']
        c4party= self.ids['c4party']
        c4votes= self.ids['c4votes']

        cstname = self.ids['cstname']
        cstname.text = db.constis[db.currentconsti].name

        c1name.text = db.constis[db.currentconsti].candis[0].cname
        c1pic.source = db.constis[db.currentconsti].candis[0].csadd
        c1party.text = db.constis[db.currentconsti].candis[0].cparty
        c1votes.text = str(db.constis[db.currentconsti].candis[0].count)
```

```
        c2name.text = db.constis[db.currentconsti].candis[1].cname
        c2pic.source = db.constis[db.currentconsti].candis[1].csadd
        c2party.text = db.constis[db.currentconsti].candis[1].cparty
        c2votes.text = str(db.constis[db.currentconsti].candis[1].count)

        c3name.text = db.constis[db.currentconsti].candis[2].cname
        c3pic.source = db.constis[db.currentconsti].candis[2].csadd
        c3party.text = db.constis[db.currentconsti].candis[2].cparty
        c3votes.text = str(db.constis[db.currentconsti].candis[2].count)

        c4name.text = db.constis[db.currentconsti].candis[3].cname
        c4pic.source = db.constis[db.currentconsti].candis[3].csadd
        c4party.text = db.constis[db.currentconsti].candis[3].cparty
        c4votes.text = str(db.constis[db.currentconsti].candis[3].count)

    def switch(self):
         sm.current = "dash"

class FRsltWindow(Screen):

    def on_pre_enter(self):
        pass

    def initwids(self):
        p1name= self.ids['p1name']
        p1logo=self.ids['p1logo']
        p1seats=self.ids['p1seats']

        p2name= self.ids['p2name']
        p2logo=self.ids['p2logo']
        p2seats=self.ids['p2seats']

        p3name= self.ids['p3name']
        p3logo=self.ids['p3logo']
        p3seats=self.ids['p3seats']

        p4name= self.ids['p4name']
        #p4logo=self.ids['p4logo']
        p4seats=self.ids['p4seats']

        db.frslts()

        p1name.text = db.frslt[0].party
        p1logo.source = db.frslt[0].partysymb
        p1seats.text = str(db.frslt[0].seats)

        p2name.text = db.frslt[1].party
        p2logo.source = db.frslt[1].partysymb
        p2seats.text = str(db.frslt[1].seats)

        p3name.text = db.frslt[2].party
        p3logo.source = db.frslt[2].partysymb
```

```
        p3seats.text = str(db.frslt[2].seats)

        #p4name.text = db.frslt[3].party
        #p4logo.source = db.frslt[3].partysymb
        #p4seats.text = str(db.frslt[3].seats)

    def switch(self):
         sm.current = "dash"

##############################################################################
class WindowManager(ScreenManager):
    pass

kv = Builder.load_file("sync.kv")
sm = WindowManager()
screens = [LoginWindow(name="login"),
BufferWindow(name="buffer"),DashWindow(name="dash"),FRsltWindow(name="frslt"),Con
stiWindow(name="consti"),CandiWindow(name="candi")]
for screen in screens:
    sm.add_widget(screen)
sm.current = "login"
##############################################################################
db = Database()

class SyncApp(App):

    def build(self):
        return sm

if __name__ == '__main__':
    SyncApp().run()
```

18.2.2 Kivy Code

```
<LoginWindow>
    name: "login"
    uname: uname
    pname: pname
    war: war
    FloatLayout:

        Label:
            pos_hint:{"x":0.1, "top":0.8}
            size_hint:0.35, 0.15
            font_size: (root.width**2 + root.height**2) / 15**4
            text: "User Name :"

        TextInput:
```

```
            id: uname
            pos_hint:{"x": 0.45 , "top":0.755}
            size_hint:0.35, 0.06
            font_size: (root.width**2 + root.height**2) / 15.5**4
            multiline: False

        Label:
            pos_hint:{"x":0.1, "top":0.7}
            size_hint:0.35, 0.15
            font_size: (root.width**2 + root.height**2) / 15**4
            text: "Password :"

        TextInput:
            id: pname
            pos_hint:{"x": 0.45 , "top":0.655}
            size_hint:0.35, 0.06
            password: True
            font_size: (root.width**2 + root.height**2) / 15.5**4
            multiline: False

        Label:
            id: war
            opacity : 0
            pos_hint:{"x":0.33, "top":0.55}
            size_hint:0.35, 0.15
            font_size: (root.width**2 + root.height**2) / 15.5**4

        Button:
            pos_hint:{"x":0.345,"y":0.3}
            size_hint: 0.3, 0.08
            font_size: (root.width**2 + root.height**2) / 15.5**4
            text: "Login"
            on_release:
                root.validation()
                root.clear()


<BufferWindow>
    name: "buffer"
    on_enter: root.popy()

    btn : btn
    pop:pop

    FloatLayout:
        Label:
            pos_hint:{"x":0.32, "top":0.8}
            size_hint:0.35, 0.15
            font_size: (root.width**2 + root.height**2) / 12.5**4
            text: "Synchronization Process"

        Label:
            id:pop
```

```
            pos_hint:{"x":0.32, "top":0.6}
            size_hint:0.35, 0.15
            font_size: (root.width**2 + root.height**2) / 15.5**4

        Button:
            id: btn
            disabled: True
            pos_hint:{"x":0.345,"y":0.3}
            size_hint: 0.3, 0.08
            font_size: (root.width**2 + root.height**2) / 16**4
            text: "Start Sync"
            on_release:
                root.switch()


<DashWindow>
    name: "dash"
    FloatLayout:

        Label:
            pos_hint:{"x":0.30, "top":0.8}
            size_hint:0.35, 0.15
            font_size: (root.width**2 + root.height**2) / 10.5**4
            text: "Dashboard"

        Label:
            pos_hint:{"x":0.32, "y":0.09}
            size_hint:0.35, 0.15
            font_size: (root.width**2 + root.height**2) / 15**4
            color: 0,1,0,1
            text: "*to initiate final results, compute constituency results
first."

        Button:
            id: btn
            pos_hint:{"x":0.345,"y":0.2}
            size_hint: 0.3, 0.08
            background_color: 0,0.3,0.7,1
            font_size: (root.width**2 + root.height**2) / 15**4
            text: "Sign Out"
            on_release:
                root.switch( "login" )


        Button:
            id: btn
            pos_hint:{"x":0.345,"y":0.4}
            size_hint: 0.3, 0.08
            font_size: (root.width**2 + root.height**2) / 15**4
            text: "Final Result*"
            on_release:
                root.switch( "frslt" )
```

```
        Button:
            id: btn
            pos_hint:{"x":0.345,"y":0.5}
            size_hint: 0.3, 0.08
            font_size: (root.width**2 + root.height**2) / 15**4
            text: "Constituency Result"
            on_release:
                root.switch( "consti" )


<FRsltWindow>
    name: "frslt"
    on_pre_enter:root.initwids()
    #******result ids******

    p1name:p1name
    p1logo:p1logo
    p1seats: p1seats

    p2name:p2name
    p2logo:p2logo
    p2seats: p2seats

    p3name:p3name
    p3logo:p3logo
    p3seats: p3seats

    p4name:p4name
    #p4logo:p4logo
    p4seats: p4seats

    BoxLayout:
        orientation: "vertical"

        BoxLayout:
            size_hint: 1, 0.25
            Label:
                pos_hint:{"x":0.32,"top":0.6}
                size_hint: 0.35, 0.2
                font_size: (root.width**2 + root.height**2) / 12.5**4
                text: "Final Result"

        BoxLayout:
            size_hint: 1, 0.12

            Label:
                font_size: (root.width**2 + root.height**2) / 14.5**4
                text: "Party Symbol"
                color: 1,1,1,1
                canvas.before:
                    Color:
                        rgba: 0.6,0,0.3,0.9
                    Rectangle:
```

```
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14.5**4
        text: "Party"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,1
            Rectangle:
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14.5**4
        text: "Seats"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,0.9
            Rectangle:
                pos: self.pos
                size: self.size


ScrollView:
    GridLayout:
        orientation: "vertical"
        spacing: 4
        size_hint_y: None
        height: self.minimum_height
        row_default_height: 60
        cols:1

        BoxLayout:

            Image:
                id:p1logo

            Label:
                id:p1name
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:p1seats
                font_size: (root.width**2 + root.height**2) / 14.25**4

        BoxLayout:

            Image:
                id:p2logo
```

```
    Label:
        id:p2name
        font_size: (root.width**2 + root.height**2) / 14.25**4

    Label:
        id:p2seats
        font_size: (root.width**2 + root.height**2) / 14.25**4

BoxLayout:

    Image:
        id:p3logo

    Label:
        id:p3name
        font_size: (root.width**2 + root.height**2) / 14.25**4

    Label:
        id:p3seats
        font_size: (root.width**2 + root.height**2) / 14.25**4

BoxLayout:

    #Image:
        #id:p4logo

    Label:
        id:p4name
        font_size: (root.width**2 + root.height**2) / 14.25**4

    Label:
        id:p4seats
        font_size: (root.width**2 + root.height**2) / 14.25**4

Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
```

```
        BoxLayout:
            size_hint: 1, 0.16
            Button:
                pos_hint:{"x":0.9,"top":1}
                size_hint: 0.35, 1
                font_size: (root.width**2 + root.height**2) / 14**4
                text: "Dashboard"
                background_color: 0,0.9,0.1,1
                on_release:
                    root.switch()


<ConstiWindow>
    name: "consti"
    on_pre_enter:root.initwids()
    # *****consti ids******

    mum:mum
    mlogo:mlogo
    mwp:mwp
    mwcan:mwcan

    dli:dli
    dlogo:dlogo
    dwp:dwp
    dwcan:dwcan

    nag:nag
    nlogo:nlogo
    nwp:nwp
    nwcan:nwcan

    kol:kol
    klogo:klogo
    kwp:kwp
    kwcan:kwcan

    # *****end ids****

    BoxLayout:
        orientation: "vertical"

        BoxLayout:
            size_hint: 1, 0.25
            Label:
                pos_hint:{"x":0.32,"top":0.6}
                size_hint: 0.35, 0.2
                font_size: (root.width**2 + root.height**2) / 12.5**4
                text: "Constituency Wise Result"

        BoxLayout:
            size_hint: 1, 0.12
```

```
    Label:
        font_size: (root.width**2 + root.height**2) / 14**4
        text: "Constituency"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,1
            Rectangle:
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14**4
        text: "Party Symbol"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,0.9
            Rectangle:
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14**4
        text: "Winner Party"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,1
            Rectangle:
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14**4
        text: "Candidate"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,0.9
            Rectangle:
                pos: self.pos
                size: self.size

ScrollView:
    GridLayout:
        orientation: "vertical"
        spacing: 4
        size_hint_y: None
        height: self.minimum_height
        row_default_height: 60
        cols:1
```

```
BoxLayout:

    Button:
        id:mum
        font_size: (root.width**2 + root.height**2) / 14.2**4
        text: "Mumbai"
        background_color: 0,0.2,0.9,1
        on_release:
            root.mumbai()

    Image:
        id:mlogo

    Label:
        id:mwp
        font_size: (root.width**2 + root.height**2) / 14.2**4

    Label:
        id:mwcan
        font_size: (root.width**2 + root.height**2) / 14.2**4


BoxLayout:

    Button:
        id:nag
        font_size: (root.width**2 + root.height**2) / 14.2**4
        text: "Nagpur"
        background_color: 0,0,0.9,1
        on_release:
            root.nagpur()

    Image:
        id:nlogo


    Label:
        id:nwp
        font_size: (root.width**2 + root.height**2) / 14.2**4

    Label:
        id:nwcan
        font_size: (root.width**2 + root.height**2) / 14.2**4

BoxLayout:

    Button:
        id: dli
        font_size: (root.width**2 + root.height**2) / 14.2**4
        text: "Delhi"
        background_color: 0,0,0.9,1
        on_release:
            root.delhi()
```

```
    Image:
        id:dlogo


    Label:
        id:dwp
        font_size: (root.width**2 + root.height**2) / 14.2**4

    Label:
        id:dwcan
        font_size: (root.width**2 + root.height**2) / 14.2**4

BoxLayout:

    Button:
        id: kol
        font_size: (root.width**2 + root.height**2) / 14.2**4
        text: "Kolkata"
        background_color: 0,0,0.9,1
        on_release:
            root.kolkata()

    Image:
        id:klogo


    Label:
        id:kwp
        font_size: (root.width**2 + root.height**2) / 14.2**4

    Label:
        id:kwcan
        font_size: (root.width**2 + root.height**2) / 14.2**4

Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
Label:
    text: "-"
```

```
        BoxLayout:
            size_hint: 1, 0.16
            Button:
                pos_hint:{"x":0.9,"top":1}
                size_hint: 0.35, 1
                font_size: (root.width**2 + root.height**2) / 14**4
                text: "Dashboard"
                background_color: 0,0.9,0.1,1
                on_release:
                    root.switch()


<CandiWindow>
    name: "candi"
    on_pre_enter: root.initwids()
    cstname:cstname
    #****candi ids****

    c1name:c1name
    c1pic:c1pic
    c1party:c1party
    c1votes:c1votes

    c2name:c2name
    c2pic:c2pic
    c2party:c2party
    c2votes:c2votes

    c3name:c3name
    c3pic:c3pic
    c3party:c3party
    c3votes:c3votes

    c4name:c4name
    c4pic:c4pic
    c4party:c4party
    c4votes:c4votes

    # ****** end*******

    BoxLayout:
        orientation: "vertical"

        BoxLayout:
            size_hint: 1, 0.25
            Label:
                id: cstname

                pos_hint:{"x":0.32,"top":0.6}
                size_hint: 0.35, 0.2
                font_size: (root.width**2 + root.height**2) / 12.5**4
```

```
    Label:
        pos_hint:{"x":0.32,"top":0.6}
        size_hint: 0.35, 0.2
        font_size: (root.width**2 + root.height**2) / 12.5**4
        text: "Candidates"

BoxLayout:
    size_hint: 1, 0.12

    Label:
        font_size: (root.width**2 + root.height**2) / 14.5**4
        text: "Symbol"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,1
            Rectangle:
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14.5**4
        text: "Party"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,0.9
            Rectangle:
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14.5**4
        text: "Candidate"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,1
            Rectangle:
                pos: self.pos
                size: self.size

    Label:
        font_size: (root.width**2 + root.height**2) / 14.5**4
        text: "Votes"
        color: 1,1,1,1
        canvas.before:
            Color:
                rgba: 0.6,0,0.3,0.9
            Rectangle:
                pos: self.pos
                size: self.size
```

```
ScrollView:
    GridLayout:
        orientation: "vertical"
        spacing: 4
        size_hint_y: None
        height: self.minimum_height
        row_default_height: 60
        cols:1

        BoxLayout:

            Image:
                id:c1pic

            Label:
                id:c1party
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:c1name
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:c1votes
                font_size: (root.width**2 + root.height**2) / 14.25**4

        BoxLayout:

            Image:
                id:c2pic

            Label:
                id:c2party
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:c2name
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:c2votes
                font_size: (root.width**2 + root.height**2) / 14.25**4

        BoxLayout:

            Image:
                id:c3pic

            Label:
                id:c3party
                font_size: (root.width**2 + root.height**2) / 14.25**4
```

```
            Label:
                id:c3name
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:c3votes
                font_size: (root.width**2 + root.height**2) / 14.25**4

        BoxLayout:

            Image:
                id:c4pic

            Label:
                id:c4party
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:c4name
                font_size: (root.width**2 + root.height**2) / 14.25**4

            Label:
                id:c4votes
                font_size: (root.width**2 + root.height**2) / 14.25**4

        Label:
            text: "-"
        Label:
            text: "-"
        Label:
            text: "-"
        Label:
            text: "-"
        Label:
            text: "-"
        Label:
            text: "-"
        Label:
            text: "-"
        Label:
            text: "-"

BoxLayout:
    size_hint: 1, 0.16
    Button:
        pos_hint:{"x":0.9,"top":1}
        size_hint: 0.35, 1
        font_size: (root.width**2 + root.height**2) / 14**4
        text: "Dashboard"
        background_color: 0,0.9,0.1,1
        on_release:
            root.switch()
```

18.3 Auxiliary Code

18.3.1 Fingerprint Authentication Code

```
import time
import hashlib
from pyfingerprint.pyfingerprint import PyFingerprint

class fingureprint():

    def __init__(self):
       pass


    def enroll(self):

        ## Enrolls new finger
        ## Tries to initialize the sensor

        try:
            f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

            if ( f.verifyPassword() == False ):
                raise ValueError('The given fingerprint sensor password is
wrong!')

        except Exception as e:
            print('The fingerprint sensor could not be initialized!')
            print('Exception message: ' + str(e))
            exit(1)

        ## Gets some sensor information
        print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+
str(f.getStorageCapacity()))

        ## Tries to enroll new finger
        try:
            print('Waiting for finger...')

        ## Wait that finger is read
            while ( f.readImage() == False ):
                pass

        ## Converts read image to characteristics and stores
            f.convertImage(0x01)

        ## Checks if finger is already enrolled
            result = f.searchTemplate()
            positionNumber = result[0]

            if ( positionNumber >= 0 ):
```

```
                print('Template already exists at position #' +
str(positionNumber))
                exit(0)

            print('Remove finger...')
            time.sleep(2)

            print('Waiting for same finger again...')

            ## Wait that finger is read again
            while ( f.readImage() == False ):
                pass

    ## Converts read image to characteristics and stores it in charbuffer 2
            f.convertImage(0x02)

    ## Compares the charbuffers
            if ( f.compareCharacteristics() == 0 ):
                raise Exception('Fingers do not match')


    ## Creates a template
            f.createTemplate()

    ## Saves template at new position number
            positionNumber = f.storeTemplate()
            print('Finger enrolled successfully!')
            print('New template position #' + str(positionNumber))

            s=str(positionNumber)

            result = hashlib.sha256(s.encode())
            self.fhashen = result.hexdigest()

        except Exception as e:
            print('Operation failed!')
            print('Exception message: ' + str(e))
            exit(1)

    def search(self):

        try:
            f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

            if ( f.verifyPassword() == False ):
                raise ValueError('The given fingerprint sensor password is
wrong!')

        except Exception as e:
            print('The fingerprint sensor could not be initialized!')
            print('Exception message: ' + str(e))
            exit(1)
```

```
## Gets some sensor information
        print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+
str(f.getStorageCapacity()))

## Tries to search the finger and calculate hash
        try:
            print('Waiting for finger...')

    ## Wait that finger is read
            while ( f.readImage() == False ):
                pass

    ## Converts read image to characteristics and stores it in charbuffer 1
            f.convertImage(0x01)

    ## Searchs template
            result = f.searchTemplate()

            positionNumber = result[0]
            accuracyScore = result[1]

            if ( positionNumber == -1 ):
                print('No match found!')

            else:
                print('Found template at position #' + str(positionNumber))
                print('The accuracy score is: ' + str(accuracyScore))


            f.loadTemplate(positionNumber, 0x01)

    ## Downloads the characteristics of template loaded in charbuffer 1
            characterics = str(f.downloadCharacteristics(0x01)).encode('utf-8')

    ## Hashes characteristics of template
            self.fhash=hashlib.sha256(characterics).hexdigest()
        except Exception as e:
            print('Operation failed!')
            print('Exception message: ' + str(e))
```

18.3.2 Database Integration Code

```
import sqlite3

class Voter():


    def __init__(self, voterfpin):
        con = sqlite3.connect('evm.db')
        csr = con.cursor()
        csr.execute("SELECT * FROM voter WHERE fhash=?", (voterfpin,))
        voterdetails = csr.fetchall()

        for row in voterdetails:

            self.vname= row[2]
            self.vid = row[0]
            self.vcst = row[3]
            self.vcno = row[4]
            self.viadd = row[5]
            self.vfhash = row[1]
            self.vflag = row[6]

        con.close()

    def update(self):

        con = sqlite3.connect('evm.db')
        c = con.cursor()
        self.vflag = 1
        c.execute("UPDATE voter SET flag= 1 WHERE fhash=?",(self.vfhash,))
        con.commit()
        con.close()

class Candidate(object):

    def __init__(self,cid,tname):
        self.tname=tname
        con = sqlite3.connect('evm.db')
        csr = con.cursor()
        csr.execute("SELECT * FROM {} WHERE c_id={}".format(tname,cid) )
        candidetails = csr.fetchall()

        for row in candidetails:

            self.cname= row[1]
            self.csadd = row[3]
            self.ccadd = row[4]
            self.cparty = row[2]
            self.count = row[5]
            self.c_id = row[0]
```

```
        con.close()

    def update(self):
        con = sqlite3.connect('evm.db')
        c = con.cursor()
        c.execute("SELECT * FROM {}".format(self.tname))
        candidate = c.fetchall()
        cid = self.c_id
        self.count = candidate[cid-1][5]
        self.count= self.count+1
        c.execute("UPDATE {} SET count= {} WHERE
c_id={}".format(self.tname,self.count, cid))
        con.commit()
        con.close()


class Constituency():

    def __init__(self,name):
        self.name = name
        self.candis = []
        self.candis.append(Candidate(1,name))
        self.candis.append(Candidate(2,name))
        self.candis.append(Candidate(3,name))
        self.candis.append(Candidate(4,name))

        self.sortwin()

    def sortwin(self):
        self.candis.sort(key = lambda x: x.count, reverse = True)
        self.wpsymb = self.candis[0].csadd
        self.wpname = self.candis[0].cparty
        self.wpcand = self.candis[0].cname

class FinalResult(object):
    def __init__(self,party,seats):
        self.party = party
        self.seats = seats
        partysymb = "{}.png".format(party)
        partysymb = partysymb.replace(" ","")
        self.partysymb = partysymb.lower()
class Database():
    currentconsti = 0
    def __init__(self):
        self.fhash = " "
        self.flag=""
        self.constis = []
        self.frslt = []
    def constit(self, c1,c2,c3,c4):
        self.constis.append(Constituency(c1))
        self.constis.append(Constituency(c2))
        self.constis.append(Constituency(c3))
        self.constis.append(Constituency(c4))
```

```
        for i in range(len(self.constis)):
            self.constis[i].sortwin()
    def frslts(self):
            party = []
            for i in range(len(self.constis)):
                party.append(self.constis[i].wpname)
            party_dict = {i:party.count(i) for i in party}
            for x in party_dict:
               self.frslt.append(FinalResult(x,party_dict[x]))
            self.frslt.sort(key = lambda x: x.seats, reverse = True)
            for i in range(len(self.frslt)):
                print(self.frslt[i].partysymb)
    def cmp_fp(self, fpin):
        v=Voter(fpin)
        if v.vfhash == fpin:
            print("voter found")
            if v.vflag == 1:
                print("your vote has been registered")
                self.flag = ""
            else:
                self.flag = "set"
        else:
            print("voter not found")
```

18.3.3 SMS Service Code

```
import requests
class Sms():
    def sendtxt(self, number,name,party):

        url = "https://www.fast2sms.com/dev/bulk"
        payload = "sender_id=FSTSMS&message=Your({}) Vote has been REGISTERED.
You Voted for {}. Thank You
:)&language=english&route=p&numbers={}".format(name,party,number)
        headers = {
            'authorization':
"TK1kNbDAi06fwCHB7ZYjyPthzds8g3xvmcFeJSuLlGqa4onrIU4FIWbPrjU3XQYT6qGvwDp2ZkHx8f90
",
            'Content-Type': "application/x-www-form-urlencoded",
            'Cache-Control': "no-cache",
            }
        response = requests.request("POST", url, data=payload, headers=headers)
        print(response.text)
```