Chapter 1

INTRODUCTION

Voting is a method for a group such as a meeting or an electorate to make a decision or express an opinion, usually following discussions, debates or election campaigns. Democracies elect holders of high office by voting. Residents of a place represented by an elected official are called "constituents", and those constituents who cast a ballot for their chosen candidate are called "voters". A vote is a formal expression of an individual's choice in voting, for or against some motion, for or against some ballot question, for a certain candidate, a selection of candidates, or a political party.

1.1 Voting System in INDIA

India has an asymmetric federal government, with elected officials at the federal, state and local levels. At the national level, the head of government, Prime Minister, is elected by members of the Lok Sabha, the lower house of the parliament of India. The elections are conducted by the Election Commission of India. All members of the Lok Sabha, except two who can be nominated by the President of India, are directly elected through general elections which take place every five years, in normal circumstances, by universal adult suffrage and a first-past-the-post system. Members of the Rajya Sabha, the upper house of the Indian parliament, are elected by elected members of the legislative assemblies of the states and the Electoral college for the Union Territories of India.

The Election Commission of India is an autonomous, constitutionally established federal authority responsible for administering all the electoral processes in the Republic of India. Under the supervision of the commission, free and fair elections have been held in India at regular intervals as per the principles enshrined in the Constitution. The Election Commission has the power of superintendence, direction and control of all elections to the Parliament of India and the state legislatures and of elections in the office of the President of India and the Vice-President of India.

Electoral Process in India starts with the declaration of dates by the election commission. Publishing of electoral rolls is a key process that happens before the elections and is vital for the conduct of elections in India. The Indian Constitution sets the eligibility of an individual for voting as any person who is a citizen of India and above 18 years of age. It is the responsibility of the eligible voters to enroll their names. The model code of conduct comes in force from the day the dates are announced.

The polling is held normally from 7:00 am to 5:00 pm, whereas it might be changed under special circumstances. The Collector of each district is in charge of polling. Government employees are employed as poll officers at the polling stations. Electronic Voting Machines (EVMs) are being increasingly used instead of ballot boxes to prevent election fraud via booth capturing, which is heavily prevalent in certain parts of India. An indelible ink is applied usually on the left index finger of the voter as an indicator that the voter has cast his vote. This practice has been followed since the 1962 general elections to prevent a bad vote. Re-polling happens if the initial polling is unsuccessful due to reasons such as adverse weather, violence etc. The polled votes are counted to announce the winner.

1.2 Biometric Authentication

Biometrics refers to metrics related to human characteristics. Biometrics authentication is used in computer science as a form of identification and access control. It is also used to identify individuals in groups that are under surveillance.

Biometric identifiers are then distinctive, measurable characteristics used to label and describe individuals. Biometric identifiers are often categorized as physiological versus behavioral characteristics. Physiological characteristics are related to the shape of the body. Examples include, but are not limited to fingerprint, palm veins, face recognition, DNA, palm print, hand geometry, iris recognition, retina and odour/scent. Behavioral characteristics are related to the pattern of behavior of a person, including but not limited to typing rhythm, gait, and voice. Some researchers have coined the term behaviometrics to describe the latter class of biometrics.

1.2.1 Fingerprint Authentication

Fingerprint recognition or fingerprint authentication refers to the automated method of verifying a match between two human fingerprints. Fingerprints are one of many forms of biometrics used to identify individuals and verify their identity.

The analysis of fingerprints for matching purposes generally requires the comparison of several features of the print pattern. These include patterns, which are aggregate characteristics of ridges, and minutia points, which are unique features found within the patterns. It is also necessary to know the structure and properties of human skin in order to successfully employ some of the imaging technologies.

Patterns

The three basic patterns of fingerprint ridges are the arch, loop, and whorl:

- **Arch**: The ridges enter from one side of the finger, rise in the center forming an arc, and then exit the other side of the finger.
- **Loop**: The ridges enter from one side of a finger, form a curve, and then exit on that same side.
- **Whorl**: Ridges form circularly around a central point on the finger.

Scientists have found that family members often share the same general fingerprint patterns, leading to the belief that these patterns are inherited.

Fingerprint processing

Fingerprint processing has three primary functions: enrollment, searching and verification. Among these functions, enrollment which captures fingerprint image from the sensor plays an important role. A reason is that the way people put their fingerprints on a mirror to scan can affect to the result in the searching and verifying process. Regarding to verification function, there are several techniques to match fingerprints such as correlation-based matching, minutiae-based matching, ridge feature-based matching and minutiae-based algorithm. However, the most popular algorithm was minutiae based matching algorithm due to its efficiency and accuracy.

Minutiae features

The major minutia features of fingerprint ridges are ridge ending, bifurcation, and short ridge (or dot). The ridge ending is the point at which a ridge terminates. Fig 1.1 shows the ridge ending of a fingerprint.

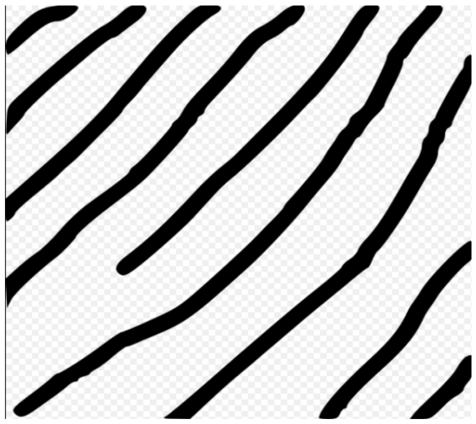


Fig 1.1 Ridge Ending

Bifurcations are points at which a single ridge splits into two ridges. Fig 1.2 shows the bifurcation point of a fingerprint.

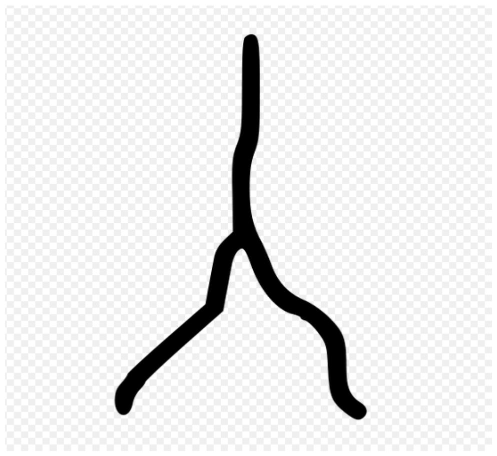


Fig 1.2 Ridge Bifurcation

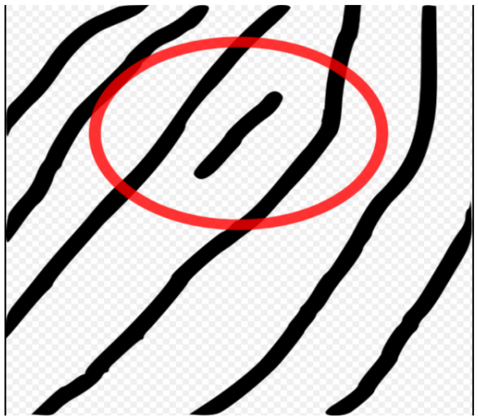


Fig 1.3 Short Ridge (Dot)

Short ridges are ridges which are significantly shorter than the average ridge length on the fingerprint. Fig 1.3 shows the short ridge or dot of a fingerprint.

Minutiae and patterns are very important in the analysis of fingerprints since no two fingers have been shown to be identical.

1.2.2 Face Recognition

A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiples methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database.

1.2.2.1 Face Recognition Using Principal Component Analysis (PCA)

In statistics, principal components analysis (PCA) is a technique that can be used to simplify a dataset. It is a linear transformation that chooses a new coordinate system for the data set such that the greatest variance by any projection of the data set comes to lie on the first axis (called the first principal component), the second greatest variance on the second axis, and so on. PCA can be used for reducing dimensionality in a dataset while retaining those characteristics of the dataset that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. The idea is that such low-order components often contain the "most important" aspects of the data.

The task of facial recognition is discriminating input signals (image data) into several classes (persons). The input signals are highly noisy (e.g. the noise is caused by differing lighting conditions, pose etc.), yet the input images are not completely random and in spite of their differences there are patterns which occur in any input signal. Such patterns, which can be observed in all signals could be - in the domain of facial recognition - the presence of some objects (eyes, nose, mouth) in any face as well as relative distances between these objects. These characteristic features are called eigenfaces in the facial recognition domain (or principal components generally). They can be extracted out of original image data by means of the mathematical tool called Principal Component Analysis (PCA).

By means of PCA one can transform each original image of the training set into a corresponding eigenface. original image. If one uses all the eigenfaces extracted from original images, one can reconstruct the original images from the eigenfaces exactly. But one can also use only a part of the eigenfaces. Then the reconstructed image is an approximation of the original image. However, losses due to omitting some of the eigenfaces can be minimized. This happens by choosing only the most important features (eigenfaces). Omission of eigenfaces is necessary due to scarcity of computational resources. Thus, the purpose of PCA is to reduce the large dimensionality of the face space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which are needed to describe the data economically. This is the case when there is a strong correlation between observed variables.

To generate a set of eigenfaces, a large set of digitized images of human faces, taken under the same lighting conditions, are normalized to line up the eyes and mouths. They are then all resample at the same pixel resolution (say $m \times n$), and then treated as mn-dimensional vectors whose components are the values of their pixels. The eigenvectors of the covariance matrix of the statistical distribution of face image vectors are then extracted. Since the eigenvectors belong to the same vector space as face images, they can be viewed as if they were $m \times n$ pixel face images: hence the name eigenfaces. Viewed in this way, the principal eigenface looks like a bland androgynous average human face. Some subsequent eigenfaces can be seen to correspond to generalized features such as left-right and top-bottom asymmetry, or the presence or lack of a beard. Other eigenfaces are hard to categorize and look rather strange. When properly weighted, eigenfaces can be summed together to create an approximate gray-scale rendering of a human face. Remarkably few eigenvector terms are needed to give a fair likeness of most people's faces, so eigenfaces provide a means of applying data compression to faces for identification purposes.

It is possible not only to extract the face from eigenfaces given a set of weights, but also to go the opposite way. This opposite way would be to extract the weights from eigenfaces and the face to be recognized. These weights tell nothing less, as the amount by which the face in question differs from "typical" faces represented by the eigenfaces.

Therefore, using this weight one can determine two important things:

- Determine if the image in question is a face at all. In the case the weights of the image differ too much from the weights of face images (i.e. images, from which we know for sure that they are faces) the image probably is not a face.
- Similar faces (images) possess similar features (eigenfaces) to similar degrees (weights). If one extracts weights from all the images available, the images could be grouped to clusters. That is, all images having similar weights are likely to be similar faces.

1.2.2.2 EIGENVALUES AND EIGENVECTORS

Large matrices can be costly, in terms of computational time, to use. Large matrices may have to be iterated hundreds or thousands of times for a calculation. Additionally, the behavior of matrices would be hard to explore without important mathematical tools. One mathematical tool, which has applications not only for Linear Algebra but for differential equations, calculus, and many other areas, is the concept of eigenvalues and eigenvectors. The words eigenvalue and eigenvector are derived from the German word "eigen" which means "proper" or "characteristic." An eigenvalue of a square matrix is a scalar that is usually represented by the Greek letter λ and an eigenvector is a non-zero vector denoted by the small letter x. For a given square matrix, A, all eigenvalues and eigenvectors satisfy the equation

$Ax= \lambda x$

In other words, an eigenvector of a matrix is a vector such that, if multiplied with the matrix, the result is always an integer multiple of that vector. This integer value is the corresponding eigenvalue of the eigenvector.

Eigenvectors possess following properties

- They can be determined only for square matrices
- There are n eigenvectors (and corresponding eigenvalues) in a n×n matrix.
- All eigenvectors are perpendicular, i.e. at right angle with each other.

Since each eigenvector is associated with an eigenvalue, we often refer to an x and λ that correspond to one another as an eigenpair. An eigenspace is a space consisting of all eigenvectors which have the same eigenvalue. These eigenvectors are derived from the covariance matrix of the probability distribution of the high-dimensional vector space of possible faces of human beings and hence eigenfaces are a set of eigenvectors as shown in fig 1.4.

STEPS FOR RECOGNITION USING PCA

The step by step instructions along with the formulas for the recognition of faces using Principal Component Analysis (PCA) are as follows:

STEP 1: Prepare the Data

The first step is to obtain a set **S** with **M** face images. Each image is transformed into a vector of size **N** and placed into the set.

$$S = \{\Gamma_1, \Gamma_2, \Gamma_3 \ldots\ldots\ldots\ldots \Gamma_n\}$$

STEP 2: Obtain the Mean

After obtaining the set, the mean image **Ψ** has to be obtained as,

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n$$

STEP 3: Subtract the Mean from Original Image

The difference between the input image and the mean image has to be calculated and the result is stored in **Φ**.

$$\Phi_i = \Gamma_i - \Psi$$

STEP 4: Calculate the Covariance Matrix

The covariance matrix **C** is calculated in the following manner

$$\mathbf{C} = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T$$

$$\mathbf{A} = \mathbf{A}\mathbf{A}^T$$

$$\mathbf{A} = \{\Phi_1, \Phi_2, \Phi_3 \ldots\ldots\ldots \Phi_n\}$$

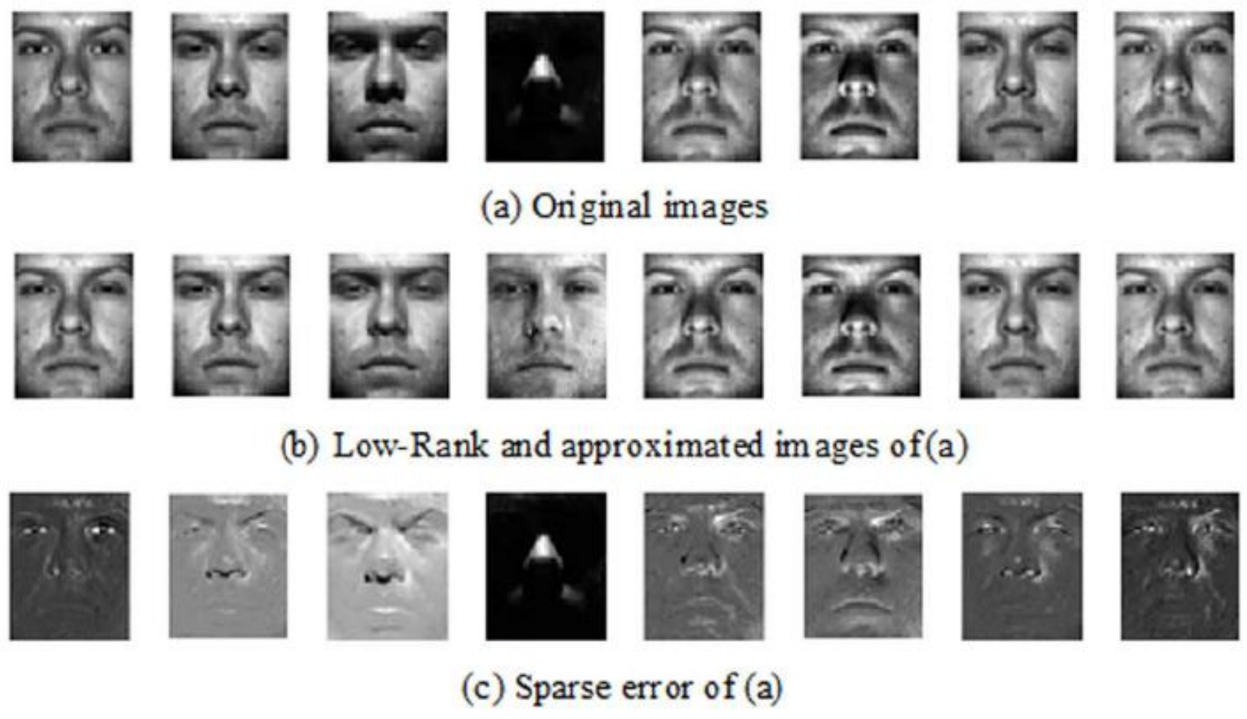


Fig 1.4 Eigen Faces

1.3 Aadhaar Card

Aadhaar is a 12 digit unique-identity number issued to all Indian residents based on their biometric and demographic data. The data is collected by the Unique Identification Authority of India (UIDAI), a statutory authority established on 12 July 2016 by the Government of India, under the Ministry of Electronics and Information Technology, under the provisions of the Aadhaar Act 2016.

Aadhaar is the world's largest biometric ID system, with over 1.133 billion enrolled members as of 31 March 2017. As of this date, over 99% of Indians aged 18 and above had been enrolled in Aadhaar. World Bank Chief Economist Paul Romer described Aadhaar as "the most sophisticated ID programme in world".

Aadhaar project has been linked to some public subsidy and unemployment benefit schemes like the domestic LPG scheme and MGNREGS. In these Direct Benefit Transfer schemes, the subsidy money is directly transferred to a bank account which is Aadhaar-linked.

On 29 July 2011, the Ministry of Petroleum and Natural Gas signed a memorandum of understanding with UIDAI. The Ministry had hoped the ID system would help them eliminate loss of the subsidized kerosene and LPG. In May 2012, the government announced that it will begin issuing Aadhaar-linked MGNREGS cards. On 26 November 2012, a pilot project was launched in 51 districts.

Aadhaar is not a proof of citizenship, and does not itself grant any rights to domicile in India. Fig 1.5 shows a digitally generated Aadhaar card. A number of features make the Aadhaar card a digital identity, and facilitate digital identity.

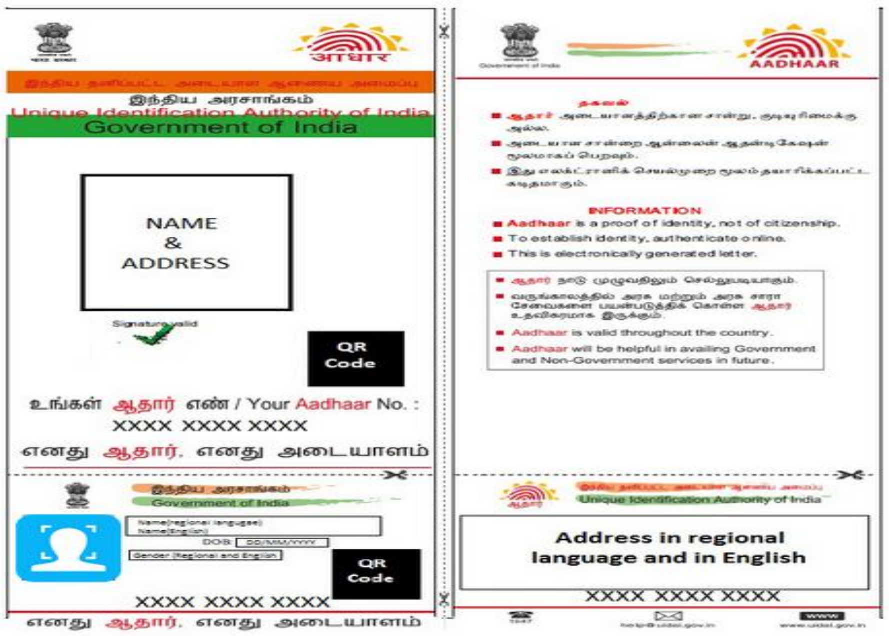

आधार
இந்திய அரசாங்கம்
Unique Identification Authority of India
Government of India
NAME & ADDRESS
Signature valid
QR Code
உங்கள் ஆதார் எண் / Your Aadhaar No. :
XXXX XXXX XXXX
எனது ஆதார், எனது அடையாளம்
Government of India
DOB : DD/MM/YYYY
QR Code
XXXX XXXX XXXX
எனது ஆதார், எனது அடையாளம்
AADHAAR
INFORMATION
Aadhaar is a proof of identity, not of citizenship.
To establish identity, authenticate online.
This is electronically generated letter.
Aadhaar is valid throughout the country.
Aadhaar will be helpful in availing Government and Non-Government services in future.
Unique Identification Authority of India
Address in regional language and in English
XXXX XXXX XXXX
help@uidai.gov.in
www.uidai.gov.in

Fig 1.5 Digitally Generated Aadhaar Card

- The document of the card itself is electronic in PDF format.
- A QR Code provides digital XML representation of some core details of the card.
- The number and some limited details can be validated online (with the notable exclusion of the name),
- Updating details can be done electronically using a mobile phone number and/or email as the second factor of authentication,
- The system collects a photo, all 10 finger scans, and eye scan, however there is no known common usage of this data to date to electronically validate a holder.

CHAPTER 2

LITERATURE SURVEY

2.1 Aadhaar Based Electoral System

An election voting system which is based on the fingerprint of voter which is saved as Aadhar card number in a central government database. In the Aadhar's centralized database, the government collects biometric and demographic data of citizens and provides a 12-digit unique identity number to individual. Fingerprint biometric provide secure authentication because fingerprint is unique to each individual. This system provides solutions to problems related to the Indian voting system. This system helps to increase voting percentage. In this voting process authentication can be done using fingerprint recognition to cast voter's votes, it ensures that vote casting cannot be altered by an unauthorized person. It requires Computer/ Touch screen computer, Fingerprint scanner, and electricity. Aadhaar's confidential biometric data may be hacked by the hacker. For the successful implementation of this system, it is very difficult, because it involves political issues, financial issues, and regional issues [1].

2.2 Rigging Free Electoral System Linked With AADHAR

This paper proposes a method for safe and secure aadhaar based biometric voting system to avoid misconceptions which are going to takes place in elections. The advantage of this project is, if an alcoholic person enters into polling booth, buzzer will alert authorized persons or constables who are in election duty. Because of Alcoholic sensor, it can provide peaceful environment at polling booth. If an unauthorized person enters into polling booth to cast his vote, buzzer will alert booth level officer. If already vote casted person enters into booth with his RFID tag for 2nd time voting, then also buzzer will alert booth level officer. Including RFID tags increases the cost of the system and are of additional work [2].

2.3 Anti-rigging Voting System Using Biometrics Based On Aadhar

The purpose of the project and implementation is to provide a secured and reliable environment to the customers is to electing the candidates by using the intelligent electronic voting machine by providing a unique identity to every user using the fingerprint identification technology. Here in this project and implementation we are going provide the at most security since it is taking the finger prints as the authentication for EVM. Intelligent EVM is an Embedded based project and implementation. It involves microcontroller and interfaces. Intelligent EVM has been specially designed to collect, record, store, count and display cent percent accurately. It has got two units control unit and ballot unit. It has “DISPLAY” section that will display the number of votes to respective candidate at the end of the poll.

The use of just the fingerprints makes the system vulnerable to spoofing attacks. As the fingerprints can be spoofed very easily using plenty of spoofing materials present in the market [3].

2.4 Biometrics using Electoral System with Embedded Security

The authenticating voters and polling data security aspects for e-voting systems was discussed. It ensures that vote casting cannot be altered by unauthorized person. The voter authentication in online e-voting process can be done by formal registration through administrators and by entering One time password (OTP). In Offline e-voting process authentication can be done using Iris recognition, finger vein sensing which enables the electronic ballot reset for allowing voters to cast their votes. Also the voted data and voters details can be sent to the nearby Database Administration unit in a timely manner using GSM System with cryptography technique.

The provision of online voting is not suitable as anyone can be forced at their places to fraud vote or there are lot of ways to hack the online process which can result in fraud voting masquerade. Future enhancements focused to design a system which can be easy to use and will provide security and privacy of votes on acceptable level by concentrating the authentication and processing section .In case of online e-voting some authentication parameters like facial recognition, In case of offline e-voting some authentication parameters like, Finger Vein and iris matching detection can be done [4].

2.5 Aadhaar based Electoral System providing Security

The main thesis of this project is to develop a secure Electronic voting machine using Finger print identification method, for finger print accessing we use AADHAR card database. At the time of voting in the elections, the e-voting process authentication can be done using finger vein sensing, which enables the electronic ballot reset for allowing voters to cast their votes. Also the voted data and voters details can be sent to the nearby Database Administration unit in a timely manner using Zigbee System with cryptography technique.

It is very difficult to design ideal e-voting system which can allow security and privacy on the high level with no compromise. Future enhancements focused to design a system which can be easy to use and will provide security and privacy of votes on acceptable level by concentrating the authentication and processing section [5].

2.6 Wireless Embedded System based Electoral System

This project is implemented with biometric system i.e. finger print scanning. This is used to ensure the security to avoid fake, repeated voting etc. It also enhances the accuracy and speed of the process. The system uses thumb impression for voter identification as we know that the thumb impression of every human being has a unique pattern. Thus it would have an edge over the present day voting systems. The purpose of such system is to ensure that the voting rights are accessed only by a legitimate user and no one else. In this, creation of a database consisting of the thumb impressions of all the eligible voters in a constituency is done as a pre-poll procedure. During elections, the thumb impression of a voter is entered as input to the system. This is then compared with the available records in the database. If the particular pattern matches with anyone in the available record, access to cast a vote is granted. But in case the pattern doesn't match with the records of the database or in case of repetition, access to cast a vote is denied or the vote gets rejected. The result is instantaneous and counting is done. The overall cost for conducting elections gets reduced and so does the maintenance cost of the systems [6].

2.7 IRIS Detection in Voting System

The individual's iris is been scanned and stored it in a voters database by giving appropriate AADHAR card no. If a person comes for voting then his or her iris is detected and this detected image is compared to image in voter's database. When the iris is detected we get the information about the voter in our PC, then that information is compared to the voter's ID. If both the details get matched then the person is allowed to vote. The current voting system is not secure, there are some individuals who give dummy votes or they are registered at more than one place. In this paper the Security of the voter is discussed and in general and the focus is on making the voting system more robust and reliable by eliminating dummy voters. By using Daughman’s algorithms will scan IRIS and check those details in our database for match.

There is an extra work of making a new database with iris with the Aadhaar card number which increases a lot of cost and work. The database needs to be updated every year or before election so that new eligible citizens may be enrolled and those who are dead are removed from the voter list [7].

2.8 Fingerprint Based e-Voting System using Aadhaar Database

This project proposes a secure online e-voting system that uses UIDAI or Aadhar database as its backend. The system ensures authentication of an individual by matching fingerprints and eligibility is checked by calculating the age of the voter thus making the existing voting cards redundant. The proposed system can handle voting at different levels such as Parliamentary, Municipality, State legislative assembly, etc simultaneously. The project will bring transparency in the voting process by assuring the voters that their votes will be in favor of the candidates of their choice. Besides electronic recording and counting of votes will be faster, more accurate and less labor intensive. The design of this system will make voting process more convenient and may therefore lead to improve the turnout.

E-voting systems have many advantages over conventional systems but it still has to solve many hurdles before becoming coming to fruition. India’s majority population is rural and illiterate. Also there is shortage of power and inadequate network between cities and villages which further adds to the problems. This system requires good bandwidth and high speed internet connection for operating, but it is still a distant reality in many cities in India. However conditions are improving with the onset of education in rural areas and with increasing urban population this project may soon become a reality [8].

2.9 Aadhaar based Electronic Voting Machine using Arduino

This paper describes an online electoral system for Indian election is proposed for the first time. The voting system is managed in a easier way as all the users should login by Aadhar card number and password and click on his/her favourable candidates to cast the vote. This features a larger security in the sense that voter high security password is confirmed before the vote is accepted in the main database of ECI. The extra feature of the model is that the voter will ensure if his/her vote has gone to correct candidate/party. The votes are going to be done automatically, therefore saving an enormous time and facultative ECI to announce the result at intervals a very short period.

This paper suggest that the EVM system has to be further studied and innovated to reach all level of community, so that the voter confidence will increase and election officials will make more involvement in purchasing the innovated EVM's for conduct smooth, secure, tamper-resistant Elections [9].

Sl. No.	**Title**	**Author**	**Year**	**Remarks**
1	Biometrics using electronic voting system with embedded security.	Alaguvel.R Gnanavel.G Jagadhambal.K 3	2013	- It ensures that vote casting cannot be altered by unauthorized person. - Iris and fingerprint verification.
2	Aadhar based electronic voting system and providing authentication	D.Krishna T.Hemalatha G.Dhana Mani Shankar K.Bala Krishna V.Bala Subhramanyam	2014	- Lesser cost, faster tabulation of results, improved accessibility, greater accuracy, and lower risk of human and mechanical errors.
3	Iris Detection In Voting System	Prof. Shital A Patil Mr.Praveen G Kote	2015	- Scanning individual's iris and storing it in a voters database by giving appropriate AADHAR card no. - The Security of the voter is discussed and in general and the focus is on making the voting system more robust and reliable by eliminating dummy voters.
4	Fingerprint and	B.Mary Havilah	2016	- Provides safety and security in voting

	RFID based electronic voting system linked with aadhaar for rigging free elections	Haque G.M.Owais Ahmed D.Sukruthi K.Venu Gopal Achary C.Mahendra Naidu		process. - The system provides security with RFID based Biometric Voting Method & provides safety from alcoholic persons and Maoists who comes to polling booth to blast polling booth.
5	Aadhar based election voting system	Ankita R. Kasliwal Jaya S. Gadekar Manjiri A. Lavadkar Pallavi K. Thorat Dr. Prapti Deshmukh	2014	Provides solutions to problems related to the Indian voting system. This system helps to increase voting percentage.
6	Antirigging Voting System Using Biometrics Based On Aadhar	M.Venkata Rao Venugopal Rao Ravula Pavani Pala	2015	Provide a secured and reliable environment to the customers is to electing the candidates by using the intelligent electronic voting machine by providing a unique identity to every user using the fingerprint identification technology.
7	Wireless Embedded System based Electoral System	Dhinesh Kumar.M Santhosh.A Aranganadhan.N.S Praveenkumar.D	2016	-Enhances the accuracy and speed of the process. - Ensure that the voting rights are accessed only by a legitimate user and no one else.

Table 2.1 Literature Survey

Chapter 3

SYSTEM ANALYSIS

3.1 Existing System

Electronic Voting Machines (EVM) are being used in Indian General and State Elections to implement electronic voting in part from 1999 elections and recently in 2017 state elections held in five states across India. EVMs have replaced paper ballots in local, state and general (parliamentary) elections in India. There were earlier claims regarding EVMs' tampering and security which have not been proved. After rulings of Delhi High Court, Supreme Court and demands from various political parties, Election Commission decided to introduce EVMs with Voter-verified paper audit trail (VVPAT) system. The Voter-verified paper audit trail (VVPAT) system was introduced in 8 of 543 parliamentary constituencies as a pilot project in Indian general election, 2014 and now implemented all over Karnataka.

The control unit is with the presiding officer or a polling officer and the balloting Unit is placed inside the voting compartment. The balloting unit presents the voter with blue buttons horizontally labeled with corresponding party symbol and candidate names. The Control Unit on the other hand provides the officer in-charge with a "Ballot" marked button to proceed to the next voter, instead of issuing a ballot paper to them. This activates the ballot unit for a single vote from the next voter in queue. The voter has to cast his vote by once pressing the blue button on the balloting unit against the candidate and symbol of his choice.

As soon as the last voter has voted, the Polling Officer in-charge of the Control Unit will press the 'Close' Button. Thereafter, the EVM will not accept any votes. Further, after the close of poll, the Balloting Unit is disconnected from the Control Unit and kept separately. Votes can be recorded only through the Balloting Unit. Again the Presiding officer, at the close of the poll, will hand over to each polling agent present an account of votes recorded. At the time of counting of votes, the total will be tallied with this account and if there is any discrepancy, this will be pointed out by the Counting Agents. During the counting of votes, the results are displayed by pressing the 'Result' button. There are two safeguards to prevent the 'Result' button from being pressed before the counting of votes officially begins.

- This button cannot be pressed till the 'Close' button is pressed by the Polling Officer in-charge at the end of the voting process in the polling booth.
- This button is hidden and sealed; this can be broken only at the counting center in the presence of designated office.

The cost per EVM was ₹5,500 at the time the machines were purchased in 1989–90. Even though the initial investment was heavy, it has since been expected to save costs of production and printing of ballot papers in lakhs, their transportation and storage, substantial reduction in the counting staff and the remuneration paid to them. For each national election, it is estimated that about 10,000 tons of ballot paper would be saved. The cost was estimated to be ₹10,500 per unit as per an additional order issued in 2014. EVMs are easier to transport compared to ballot boxes as EVMs are lighter, portable and come with polypropylene carrying cases. The vote counting is faster and in places where illiteracy is still a factor, illiterate people find EVMs easier than ballot paper system. Bogus voting is greatly reduced as the vote is recorded only once. The unit can store the result in its memory before it is erased manually. The battery is required only to activate the EVMs at the time of polling and counting and as soon as the polling is over, the battery can be switched off. The shelf life of Indian EVMs is estimated at 15 years.

3.1.1 Limitations

A candidate can know how many people from a polling station voted for him. This is a significant issue particularly if lop-sided votes for/against a candidate are cast in individual polling stations and the winning candidate might show favouritism or hold grudge on specific areas. The Election Commission of India has stated that the manufacturers of the EVMs have developed a Totaliser unit which can connect several balloting units and would display only the overall results from an Assembly or a Lok Sabha constituency instead of votes from individual polling stations.

The control units do not electronically transmit their results back to the Election Commission, even though a simple and unconditionally secure protocol for doing this exists. The Indian EVMs are purposely designed as stand-alone units to prevent any intrusion during electronic transmission of results. Instead, the EVMs are collected in counting booths and tallied on the assigned counting days in the presence of polling agents of the candidates.

3.1.2 Security Issues

An international conference on the Indian EVMs and its tamperability of the said machines was held under the chairmanship of Subramanian Swamy, President of the Janata Party and former Union Cabinet Minister for Law, Commerce and Justice at Chennai on 13 February 2010. The conclusion was that the Election Commission of India was shirking its responsibility on the transparency in the working of the EVMs. In April 2010, an independent security analysis was released by a research team led by Hari Prasad, RopGonggrijp, and J. Alex Halderman. The study included video demonstrations of two attacks that the researchers carried out on a real EVM as well as descriptions of several other potential vulnerabilities.

In order to mitigate these threats, the researchers suggest moving to a voting system that provides greater transparency, such as paper ballots, precinct count optical scan, or a voter verified paper audit trail, since, in any of these systems, sceptical voters could, in principle, observe the physical counting process to gain confidence that the outcome is fair. But Election Commission of India points out that for such tampering of the EVMs, one needs physical access to EVMs, and pretty high-tech skills are required. Given that EVMs are stored under strict security which can be monitored by candidates or their agents all the time, it's impossible to gain physical access to the machines. Plus, to impact the results of an election, hundreds to thousands of machines will be needed to tamper with, which is almost impossible given the hi-tech and time-consuming nature of the tampering process.

On 25 July 2011, responding to a PIL, Supreme Court of India asked EC to consider request to modify EVMs and respond within 3 months. The petitioner Rajendra Satyanarayan Gilda had alleged that EC has failed to take any decision despite his repeated representation. The petitioner suggested that the EVMs should be modified to give a slip printed with the symbol of the party in whose favour the voter cast his ballot.

On 17 January 2012, Delhi High Court in its ruling on Dr. Subramanian Swamy's Write Petition challenging the use of EVMs in the present form said that EVMs are not "tamper-proof". Further, it said that it is "difficult" to issue any directions to the EC in this regard. However, the court added that the EC should itself hold wider consultations with the executive, political parties and other stake holders on the matter.

Swamy appealed against Delhi High Court's refusal to order a VVPAT system in Supreme Court. On 27 September 2012, Election Commission's advocate Ashok Desai submitted to a Supreme Court bench of Justice P Sathasivam and Justice Ranjan Gogoi that field trial for VVPAT system is in progress and that a status report will be submitted by early January 2013. Desai said that on pressing of each vote, a paper receipt will be printed, which will be visible to the voters inside a glass but cannot be taken out of the machine. To this, Dr Swamy replied that the new system was acceptable to him. The Supreme Court posted the matter for further hearing to 22 January 2013.

Another similar writ petition filed by the AsomGanaParishad is still pending before the Gauhati High Court. On 8 October 2013,Supreme Court of India delivered its verdict on Subramanian Swamy PIL, that Election Commission of India will use VVPATs along with EVMs in a phased manner and the full completion should be achieved by 2019.

3.2 Problem Statement

The current scenario in the existing voting system has become so problematic as the election commission of India is unable to eliminate bogus vote possibilities due to inefficient identification system and lack of security in the EVM process. The above mentioned scenario leads to non-legitimate votes and fails to prove the current electoral process as secure, reliable and a fair way to conduct elections. The current electoral system also consumes a significant amount of country’s economical resources and human resources which needs to be minimized. Use of paper should also be minimized when considering from environmental point of view.

3.3 Proposed System

The purpose of the project and implementation is to provide a secured and reliable environment to the voters. Electing the candidates by using the intelligent means by providing a unique identity to every user using the fingerprint authentication and Facial authentication technology linked with the Aadhaar database. Here in this project and implementation we are going provide the at most security since it is taking the fingerprint and Face as the authentication for voting. It involves microcontroller and interfaces. It has been specially designed to authenticate, means to cast a vote and display cent percent accurately.

The person coming in to the polling booth has to scan his fingerprint and Face for authentication. These fingerprint and Face will be matched with the Aadhaar card database and the person will be authenticated accordingly. The person will not be allowed to vote again for the same elections if he/she has already voted.

The authenticated admin can check the results after the votes have been casted. This proposed method overcomes all the issues faced in existing systems as well as the system proposed based on RFID tags and online e-voting systems.

3.3.1 Advantages of proposed system

The proposed system has many advantages over the traditional way of voting, some of these advantages are:

- Consumes less time
- Improved accessibility
- Greater accuracy
- Lower risk of human and mechanical errors
- Removal of inedible ink
- Eradicates fraud voting
- Less manpower required
- Saves paper leading to eco-friendly environment

These are some advantages over the existing system. As most of the process will be automated, less man power will be needed which in turn reduces risk of human errors and it will also consume less time. Strict authentication using biometrics is done which eradicates fraud voting.

Chapter 4

SYSTEM DESIGN

The proposed system is divided into 3 modules for effective functions, they are as follows

- Fingerprint module
- Face Recognition module
- Voter module

The following figure shows the ER diagram of the system. There are 3 entities taken namely, voter, candidate and admin. There are 3 relationships shown between the entities.

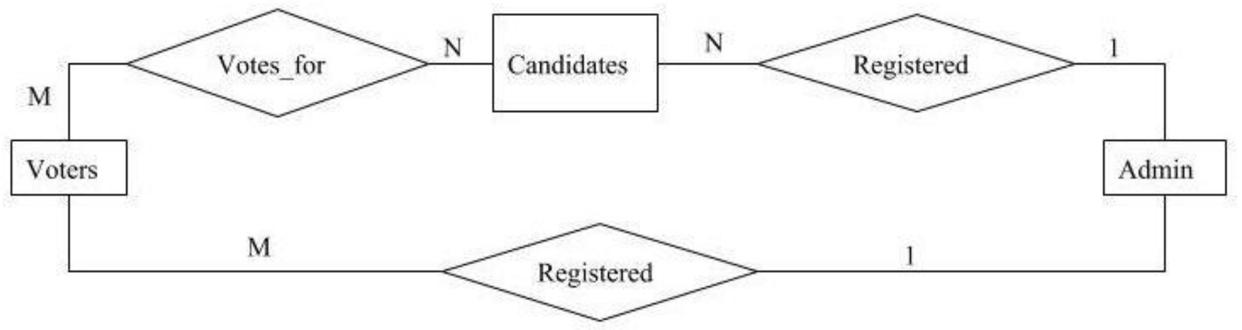


Fig 4.1 ER Diagram of proposed system

The voter has attributes like face and fingerprint and the candidate has the name as its attribute. The relationships of the entities are shown in the diagram.

The voter coming in to the polling booth has to scan his fingerprint and real-time facial image for authentication. These fingerprint and facial will be matched with the already available database and the voter will be authenticated accordingly ensuring that the voter is valid and the voter is going to cast a legitimate vote.

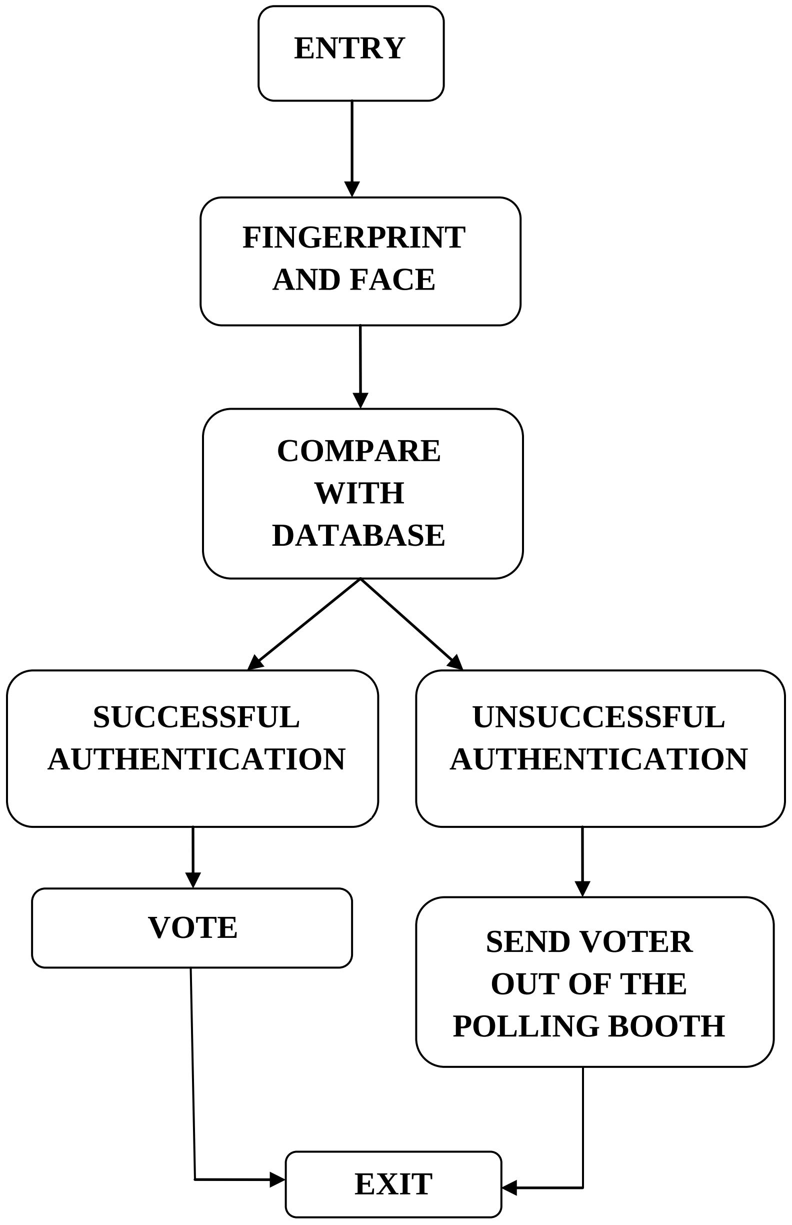

ENTRY
FINGERPRINT AND FACE
COMPARE WITH DATABASE
SUCCESSFUL AUTHENTICATION
UNSUCCESSFUL AUTHENTICATION
VOTE
SEND VOTER OUT OF THE POLLING BOOTH
EXIT

Fig 4.2 Flow diagram of proposed system

If the data of the person is present in the database then the voter’s identification details are matched with fingerprint and facial images stored in the database, upon successful recognition and authentication, he/she will be allowed to vote. In other conditions like voting second time for the same election or trying to vote in behalf of another person will not be allowed by the system. The proposed design ensures “one person, one vote” by implementing a 2-level recognition and authentication system which are biometric in nature. If authentication is unsuccessful in any recognition step then the voter is asked to leave.

4.1 Fingerprint Authentication Design

One of the biometric traits used is fingerprint so the voter scans his fingerprint for authentication. The person to be authenticated indicates his/her identity and places his/her finger on the fingerprint sensor. If the person is authenticated successfully then he can vote.

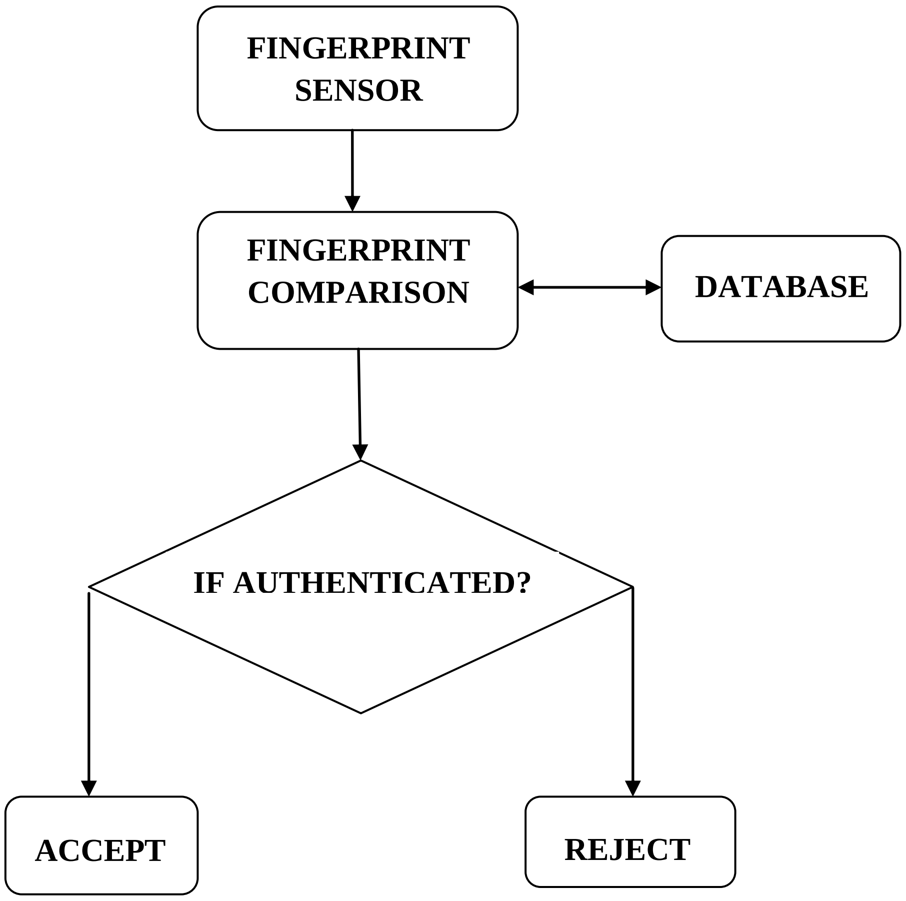


Fig 4.3 Dataflow diagram for fingerprint authentication

For the captured fingerprint image, the biometric template is extracted during the feature extraction stage. After the feature extraction it is fed to a matching algorithm which matches it against the person’s biometric template stored in the database. This fingerprint authentication allows voter to vote or not. If the authentication is successful the voter is allowed to vote for his desired candidate.

4.2 Face Recognition Design

The Face images to be used are stored in the database. The test image will be taken for matching. Then PCA technique is used on the images. This technique extracts the main variations in the feature vector and allows an accurate reconstruction of the data to be produced from the extracted feature values and reduces the amount of computation needed.

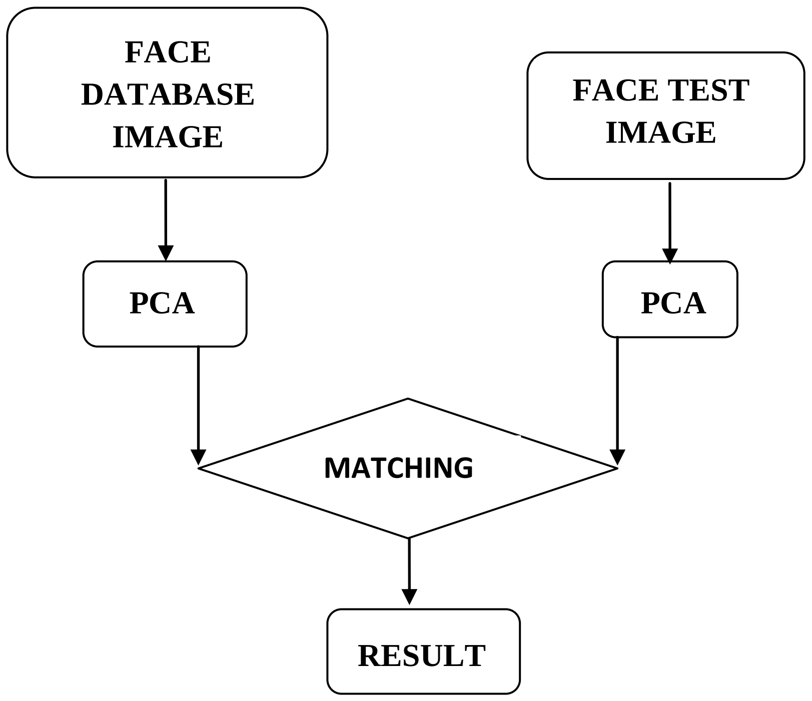


Fig 4.4 Dataflow diagram for Face Recognition

4.3 Voter Module

The voter module is the integrated system consisting of various subsystems which includes fingerprint recognition module, face recognition module and front-end voting interface. This module deals with counting of votes, storing the count with the Aadhaar database in the background and providing legitimate voting interface which has different status to monitor the count of each and every voter. The status is namely

- Yet to cast vote (Voter has yet to undergo the recognition process)

- Vote (The voter has successfully undergone the recognition process and can cast a vote now)
- Vote casted successfully (Voter has casted a vote and the voter can’t cast a bogus vote)

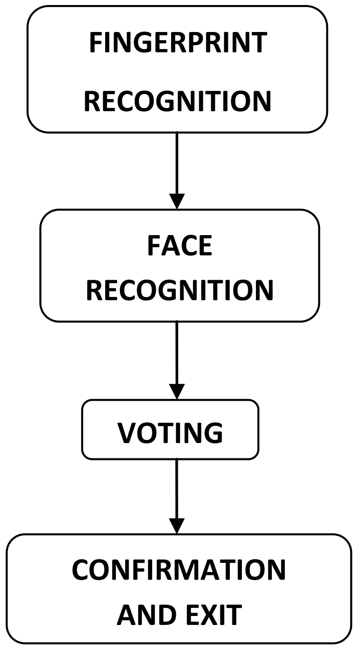


Fig 4.5 Dataflow diagram for Voter module

Chapter 5

IMPLEMENTATION

The implementation of the system is divided in three modules based on their functionalities. They are as follows

- Fingerprint module
- Face Recognition module
- Voter module

5.1 Fingerprint Module

The following code matches the fingerprint with the registered fingerprint and calls the Face Recognition programs in the MATLAB and displays the appropriate result on the screen on different situations.

```
#include<EEPROM.h>
#include<LiquidCrystal.h>
LiquidCrystallcd(12,11,5,4,3,2);
 #include <Adafruit_Fingerprint.h>
uint8_t id;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial);
 #define enroll 14
#define del 15
#define up 16
#define down 17
#define match 18
#define indVote 13
#define sw1 6
#define sw2 7
#define sw3 8
#define resultsw 9
#define indFinger 10
#define buzzer 19
#define records 25
int vote1,vote2,vote3;

int flag;
```

```
void setup()
{
delay(1000);
pinMode(enroll, INPUT_PULLUP);
pinMode(up, INPUT_PULLUP);
pinMode(down, INPUT_PULLUP);
pinMode(del, INPUT_PULLUP);
pinMode(match, INPUT_PULLUP);
pinMode(sw1, INPUT_PULLUP);
pinMode(sw2, INPUT_PULLUP);
pinMode(sw3, INPUT_PULLUP);
pinMode(resultsw, INPUT_PULLUP);
pinMode(buzzer, OUTPUT);
pinMode(indVote, OUTPUT);
pinMode(indFinger, OUTPUT);
    // digitalWrite(buzzer, HIGH);

lcd.begin(16,2);
  if(digitalRead(resultsw) ==0)
  {
for(inti=0;i<records;i++)
EEPROM.write(i+10,0xff);
EEPROM.write(0,0);
EEPROM.write(1,0);
EEPROM.write(2,0);
lcd.clear();
lcd.print("System Reset");
      // flag=1;
delay(1000);
  }

lcd.clear();
lcd.print("Voting Machine");
lcd.setCursor(0,1);
lcd.print("by Finger Print");
delay(2000);

if(EEPROM.read(0) == 0xff)
EEPROM.write(0,0);

if(EEPROM.read(1) == 0xff)
EEPROM.write(1,0);

if(EEPROM.read(1) == 0xff)
EEPROM.write(1,0);
```

```
//finger.begin(57600);
          Serial.begin(9600);
          lcd.clear();
          lcd.print("Finding Module");
          lcd.setCursor(0,1);
          delay(1000);
             if (finger.verifyPassword())
             {
               //Serial.println("Found fingerprint sensor!");
          lcd.clear();
          lcd.print("Found Module ");
          delay(1000);
             }
             else
             {
             //Serial.println("Did not find fingerprint sensor :(");
          lcd.clear();
          lcd.print("module not Found");
          lcd.setCursor(0,1);
          lcd.print("Check Connections");
             while (1);
             }

          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("BJP");
          lcd.setCursor(4,0);
          lcd.print("CONG");
          lcd.setCursor(8,0);
          lcd.print("JDS");
          lcd.setCursor(12,0);
          lcd.print("RESULT");

          lcd.setCursor(0,1);
            vote1=EEPROM.read(0);
          lcd.print(vote1);
          lcd.setCursor(6,1);
            vote2=EEPROM.read(1);
          lcd.print(vote2);
          lcd.setCursor(12,1);
            vote3=EEPROM.read(2);
          lcd.print(vote3);
          delay(2000);
          }

          void loop()
```

```
{
lcd.setCursor(0,0);
lcd.print("Press Match Key ");
lcd.setCursor(0,1);
lcd.print("to start system");
//digitalWrite(buzzer, HIGH);
digitalWrite(indVote, LOW);
digitalWrite(indFinger, LOW);
 if(digitalRead(match)==0)
 {
digitalWrite(buzzer, HIGH);
delay(500);
digitalWrite(buzzer, LOW);
digitalWrite(indFinger, HIGH);
for(inti=0;i<3;i++)
  {
lcd.clear();
lcd.print("Place Finger");
delay(2000);
int result=getFingerprintIDez();
    if(result>=0)
    {
      flag=0;
for(inti=0;i<records;i++)
        {
if(result == EEPROM.read(i+10))
          {
lcd.clear();
lcd.print("Authorised Voter");
lcd.setCursor(0,1);
lcd.print("Please Wait....");
delay(1000);
Vote();
EEPROM.write(i+10, 0xff);
              flag=1;
              return;
          }
        }

if(flag == 0)
        {
lcd.clear();
lcd.print("Already Voted");
        //lcd.setCursor(0,1);
        //lcd.print("")
digitalWrite(buzzer, HIGH);
delay(5000);
digitalWrite(buzzer, LOW);
       return;
       }
     }
   }
lcd.clear();
 }
checkKeys();
delay(1000);
}

void checkKeys()
{
   if(digitalRead(enroll) == 0)
   {
lcd.clear();
lcd.print("Please Wait");
delay(1000);
    while(digitalRead(enroll) == 0);
Enroll();
   }

   else if(digitalRead(del) == 0)
   {
lcd.clear();
lcd.print("Please Wait");
delay(1000);
delet();
   }
}

void Enroll()
{
int count=0;
lcd.clear();
lcd.print("Enter Finger ID:");

while(1)
   {
lcd.setCursor(0,1);
lcd.print(count);
     if(digitalRead(up) == 0)
     {
       count++;
       if(count>25)
       count=0;
delay(500);
     }

     else if(digitalRead(down) == 0)
     {
       count--;
       if(count<0)
       count=25;
delay(500);
     }
     else if(digitalRead(del) == 0)
     {
          id=count;
getFingerprintEnroll();
for(inti=0;i<records;i++)
          {
            if(EEPROM.read(i+10) == 0xff)
            {
EEPROM.write(i+10, id);
              break;
            }
          }
          return;
     }

       else if(digitalRead(enroll) == 0)
     {
          return;
     }
 }
}

void delet()
{
int count=0;
lcd.clear();
lcd.print("Enter Finger ID");

while(1)
   {
lcd.setCursor(0,1);
lcd.print(count);
     if(digitalRead(up) == 0)
     {
       count++;
       if(count>25)
       count=0;
delay(500);
     }

     else if(digitalRead(down) == 0)
     {
       count--;
       if(count<0)
       count=25;
delay(500);
     }
     else if(digitalRead(del) == 0)
     {
          id=count;
deleteFingerprint(id);
for(inti=0;i<records;i++)
          {
            if(EEPROM.read(i+10) == id)
            {
EEPROM.write(i+10, 0xff);
              break;
            }
          }
          return;
     }
       else if(digitalRead(enroll) == 0)
     {
          return;
     }
 }
}

uint8_t getFingerprintEnroll()
{
int p = -1;
lcd.clear();
lcd.print("finger ID:");
lcd.print(id);
lcd.setCursor(0,1);
lcd.print("Place Finger");
delay(2000);
  while (p != FINGERPRINT_OK)
  {
    p = finger.getImage();
    switch (p)
    {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
lcd.clear();
lcd.print("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      //Serial.println("No Finger");
lcd.clear();
lcd.print("No Finger");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      //Serial.println("Communication error");
lcd.clear();
lcd.print("Comm Error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      //Serial.println("Imaging error");
lcd.clear();
lcd.print("Imaging Error");
      break;
    default:
      //Serial.println("Unknown error");
lcd.clear();
lcd.print("Unknown Error");
      break;
    }
  }

  // OK success!

  p = finger.image2Tz(1);
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image converted");
lcd.clear();
lcd.print("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      //Serial.println("Image too messy");
lcd.clear();
lcd.print("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      //Serial.println("Communication error");
lcd.clear();
lcd.print("Comm Error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      //Serial.println("Could not find fingerprint features");
lcd.clear();
lcd.print("Feature Not Found");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      //Serial.println("Could not find fingerprint features");
lcd.clear();
lcd.print("Feature Not Found");
      return p;
    default:
      //Serial.println("Unknown error");
lcd.clear();
lcd.print("Unknown Error");
      return p;
   }

  //Serial.println("Remove finger");
lcd.clear();
lcd.print("Remove Finger");
delay(2000);
  p = 0;
  while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
  }
  //Serial.print("ID "); //Serial.println(id);
  p = -1;
  //Serial.println("Place same finger again");
lcd.clear();
lcd.print("Place Finger");
lcd.setCursor(0,1);
lcd.print("   Again");
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      //Serial.print(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      //Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      //Serial.println("Imaging error");
      break;
    default:
      //Serial.println("Unknown error");
      return;
    }
  }

  // OK success!

  p = finger.image2Tz(2);
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      //Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      //Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      //Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      //Serial.println("Could not find fingerprint features");
      return p;
    default:
      //Serial.println("Unknown error");
      return p;
  }

  // OK converted!
  //Serial.print("Creating model for #");  //Serial.println(id);

  p = finger.createModel();
  if (p == FINGERPRINT_OK) {
    //Serial.println("Prints matched!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    //Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_ENROLLMISMATCH) {
    //Serial.println("Fingerprints did not match");
    return p;
  } else {
  //Serial.println("Unknown error");
  return p;
 }

 //Serial.print("ID "); //Serial.println(id);
 p = finger.storeModel(id);
 if (p == FINGERPRINT_OK) {
  //Serial.println("Stored!");
lcd.clear();
lcd.print("Stored!");
delay(2000);
 } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  //Serial.println("Communication error");
  return p;
 } else if (p == FINGERPRINT_BADLOCATION) {
  //Serial.println("Could not store in that location");
  return p;
 } else if (p == FINGERPRINT_FLASHERR) {
  //Serial.println("Error writing to flash");
  return p;
 }
 else {
  //Serial.println("Unknown error");
  return p;
 }
}

intgetFingerprintIDez()
{
 uint8_t p = finger.getImage();

 if (p != FINGERPRINT_OK)
 return -1;

 p = finger.image2Tz();
 if (p != FINGERPRINT_OK)
 return -1;

 p = finger.fingerFastSearch();
 if (p != FINGERPRINT_OK)
 {
lcd.clear();
lcd.print("Finger Not Found");
lcd.setCursor(0,1);
lcd.print("Try Later");
delay(2000);
```

```
  return -1;
  }
  // found a match!
//  Serial.print("Found ID #");
 // Serial.print(finger.fingerID);
  switch(finger.fingerID)
  {
      case 1:lcd.clear();
lcd.print("NAME:SSSS");
lcd.setCursor(0,1);
lcd.print("AGE:20");
delay(2000);
             return finger.fingerID;
             break;

       case 2:lcd.clear();
lcd.print("YYYYY");
lcd.setCursor(0,1);
lcd.print("AGE:21");
delay(2000);
             return finger.fingerID;
             break;

      case 3:lcd.clear();
lcd.print("AAAAAA");
lcd.setCursor(0,1);
lcd.print("AGE:22");
delay(2000);
             return finger.fingerID;
             break;

      case 4:lcd.clear();
lcd.print("BBBBBB");
lcd.setCursor(0,1);
lcd.print("AGE:23");
delay(2000);
             return finger.fingerID;
             break;

       case 5:lcd.clear();
lcd.print("CCCCCC");
lcd.setCursor(0,1);
lcd.print("AGE:24");
delay(2000);
             return finger.fingerID;
             break;
```

```
    case 6:lcd.clear();
lcd.print("DDDDDD");
lcd.setCursor(0,1);
lcd.print("AGE:25");
delay(2000);
        return finger.fingerID;
        break;

    case 7:lcd.clear();
lcd.print("EEEEE");
lcd.setCursor(0,1);
lcd.print("AGE:26");
delay(2000);
        return finger.fingerID;
        break;

  }
  return finger.fingerID;
}
uint8_t deleteFingerprint(uint8_t id)
{
  uint8_t p = -1;
lcd.clear();
lcd.print("Please wait");
  p = finger.deleteModel(id);
  if (p == FINGERPRINT_OK)
  {
    //Serial.println("Deleted!");
lcd.clear();
lcd.print("Finger Deleted");
lcd.setCursor(0,1);
lcd.print("Successfully");
delay(1000);
  }

  else
  {
    //Serial.print("Something Wrong");
lcd.clear();
lcd.print("Something Wrong");
lcd.setCursor(0,1);
lcd.print("Try Again Later");
delay(2000);
    return p;
  }
```

```
}

void Vote()
{
lcd.clear();
lcd.print("Please Place");
lcd.setCursor(0,1);
lcd.print("Your Vote");
digitalWrite(indVote, HIGH);
digitalWrite(indFinger, LOW);
digitalWrite(buzzer, HIGH);
delay(500);
digitalWrite(buzzer, LOW);
delay(1000);
while(1)
  {
        if(digitalRead(sw1)==0)
        {
          vote1++;
voteSubmit(1);
EEPROM.write(0, vote1);
          while(digitalRead(sw1)==0);
          return;
        }
        if(digitalRead(sw2)==0)
        {
          vote2++;
voteSubmit(2);
EEPROM.write(1, vote2);
          while(digitalRead(sw2)==0);
          return;
        }
        if(digitalRead(sw3)==0)
        {
          vote3++;
voteSubmit(3);
EEPROM.write(2, vote3);
          while(digitalRead(sw3)==0);
          return;
        }

         if(digitalRead(resultsw)==0)
         {
lcd.clear();
lcd.setCursor(0,0);
lcd.print("BJP:");
lcd.setCursor(6,0);
lcd.print("CONG:");
lcd.setCursor(12,0);
lcd.print("JDS:");
for(inti=0;i<3;i++)
         {
lcd.setCursor(i*6,1);
lcd.print(EEPROM.read(i));
         }
delay(2000);
int vote=vote1+vote2+vote3;
       if(vote)
       {
if((vote1 > vote2 && vote1 > vote3))
        {
lcd.clear();
lcd.print("BJP Wins");
delay(2000);
lcd.clear();
        }
        else if(vote2 > vote1 && vote2 > vote3)
        {
lcd.clear();
lcd.print("CONG Wins");
delay(2000);
lcd.clear();
        }
        else if((vote3 > vote1 && vote3 > vote2))
        {
lcd.clear();
lcd.print("JDS Wins");
delay(2000);
lcd.clear();
        }

       else
       {
lcd.clear();
lcd.print("   Tie Up Or   ");
lcd.setCursor(0,1);
lcd.print("   No Result   ");
delay(1000);
lcd.clear();
       }

      }
```

```
           else
           {
lcd.clear();
lcd.print("No Voting....");
delay(1000);
lcd.clear();
           }
           vote1=0;vote2=0;vote3=0;vote=0;
lcd.clear();
           return;
          }
  }
digitalWrite(indVote, LOW);
}

void voteSubmit(intcn)
{
lcd.clear();
if(cn == 1)
lcd.print("BJP:");
  else if(cn == 2)
lcd.print("CONG:");
  else if(cn == 3)
lcd.print("JDS:");
lcd.setCursor(0,1);
lcd.print("Vote Submitted");
digitalWrite(buzzer , HIGH);
delay(1000);
digitalWrite(buzzer, LOW);
digitalWrite(indVote, LOW);
 // flag=1;
  return;
}
```

5.2 Face Recognition Module

The following code runs on MATLAB as a Backend service when matched person finger imprints matches with the fingerprint imprints stored in the database.

The Face Recognition Module will return a Character associated with the particular person profile stored in the database to the fingerprint module.

function varargout = realtimefacerecognition(varargin)

```
% REALTIMEFACERECOGNITION MATLAB code for realtimefacerecognition.fig
%      REALTIMEFACERECOGNITION, by itself, creates a new
REALTIMEFACERECOGNITION or raises the existing
%      singleton*.
%
%      H = REALTIMEFACERECOGNITION returns the handle to a new
REALTIMEFACERECOGNITION or the handle to
%      the existing singleton*.
%
%      REALTIMEFACERECOGNITION('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in REALTIMEFACERECOGNITION.M with the given
input arguments.
%
%      REALTIMEFACERECOGNITION('Property','Value',...) creates a new
REALTIMEFACERECOGNITION or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before realtimefacerecognition_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to realtimefacerecognition_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help realtimefacerecognition

% Last Modified by GUIDE v2.5 19-Feb-2018 12:05:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @realtimefacerecognition_OpeningFcn, ...
                   'gui_OutputFcn',  @realtimefacerecognition_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin&&ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before realtimefacerecognition is made visible.
function realtimefacerecognition_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to realtimefacerecognition (see VARARGIN)

% Choose default command line output for realtimefacerecognition
handles.output = hObject;

axes(handles.axes1);
axis off;
axes(handles.axes2);
axis off;
axes(handles.axes3);
axis off;
axes(handles.axes4);
axis off;
axes(handles.axes5);
axis off;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes realtimefacerecognition wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = realtimefacerecognition_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
vidobj  =videoinput('winvideo', 1, 'YUY2_320x240');

vidRes = get(vidobj , 'VideoResolution');
nBands = get(vidobj , 'NumberOfBands');

% Display the video data in your GUI.
% axes(handles.axes1);
axes(handles.axes4);
hImage = image( zeros(vidRes(2), vidRes(1), nBands) );

preview(vidobj, hImage);

 handles.vidobj = vidobj;
%
% % set(handles.pushbutton_CaptureFrame, 'Enable', 'on');
% % set(handles.pushbutton_StartVideo, 'Enable', 'off');
% % Update handles structure
 guidata(hObject, handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.webcam_image = getsnapshot(handles.vidobj);

% url='http://192.168.1.31:8080/photo.jpg';
%
% handles.webcam_image=imread(url);
% stoppreview(handles.vidobj);

axes(handles.axes2);
handles.webcam_image = ycbcr2rgb(handles.webcam_image);
imshow(handles.webcam_image);
guidata(hObject, handles);
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global imgCrop
handles.imgCrop = [];
[handles] = locateface(hObject, handles);
```

```
% handles.imgCrop = crop_image(handles.webcam_image, 1);

imgCrop = handles.imgCrop;
if ~isempty(imgCrop)
    axes(handles.axes3);
    imshow(uint8(imgCrop));
    imgCrop=uint8(imgCrop);
    %save cropImgimgCrop;

%     set(handles.pushbutton_Save, 'Enable', 'on');
%     set(handles.pushbutton_Enrolement, 'Enable', 'on');
end
guidata(hObject, handles);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global imgCrop
%TrainDatabasePath = uigetdir('D:\2016\Practie 2016\ERachith RV Face\delivered', 'Select training database path' );
TrainDatabasePath = '.\TrainDatabase';
prompt = {'Enter image name (a number between 1 to 10):'};
dg_title = 'Input of PCA-Based Face Recognition System';
num_lines= 1;
def = {'1'};

TrainImage  =inputdlg(prompt,dg_title,num_lines,def);
TrainImage = strcat(TrainDatabasePath,'\',char(TrainImage),'.jpg');
imwrite(imgCrop,TrainImage)
guidata(hObject, handles);
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
clear all
close all

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
TrainDatabasePath ='.\TrainDatabase';

%
% prompt = {'Enter test image name (a number between 1 to 10):'};
% dg_title = 'Input of PCA-Based Face Recognition System';
% num_lines= 1;
% def = {'1'};
%
% TestImage  =inputdlg(prompt,dg_title,num_lines,def);
% TestImage = strcat(TestDatabasePath,'\',char(TestImage),'.jpg');
% im = imread(TestImage);
%%

vidobj  =videoinput('winvideo', 1, 'YUY2_320x240');

vidRes = get(vidobj , 'VideoResolution');
nBands = get(vidobj , 'NumberOfBands');
% axes(handles.axes1);
axes(handles.axes4);
hImage = image( zeros(vidRes(2), vidRes(1), nBands) );

% Display the video data in your GUI.

preview(vidobj, hImage);

 handles.vidobj = vidobj;
 pause(3);
handles.webcam_image = getsnapshot(handles.vidobj);

% url='http://192.168.1.31:8080/photo.jpg';
% handles.webcam_image=imread(url);

% stoppreview(handles.vidobj);

axes(handles.axes2);
handles.webcam_image = ycbcr2rgb(handles.webcam_image);
imshow(handles.webcam_image);

handles.imgCrop = [];
[handles k] = locateface(hObject, handles);
% handles.imgCrop = crop_image(handles.webcam_image, 1);

%imgCrop = handles.imgCrop;

% if ~isempty(imgCrop)
%     axes(handles.axes3);
%     imshow(imgCrop);
%   end
    %save cropImgimgCrop;

%     set(handles.pushbutton_Save, 'Enable', 'on');
%     set(handles.pushbutton_Enrolement, 'Enable', 'on');

for i=1:k
%%

imgCrop = handles.imgCrop(:,:,i);
axes(handles.axes3);
    imshow(uint8(imgCrop));
imwrite(uint8(imgCrop),'test.jpg')
TestImage = 'test.jpg';
T = CreateDatabase(TrainDatabasePath);

[m, A, Eigenfaces] = EigenfaceCore(T);
[OutputName index] = Recognition(TestImage, m, A, Eigenfaces);

SelectedImage = strcat(TrainDatabasePath,'\',OutputName);
SelectedImage = imread(SelectedImage);

% imshow(im)
% title('Test Image');

stop(vidobj);
% SelectedImage1 = strcat(TrainDatabasePath,'\',OutputName);
% SelectedImage1 = imread(SelectedImage1);

% str = strcat('Matched image is PCA :  ',OutputName);
% disp(str)

guidata(hObject, handles);
%%
imgCrop=uint8(imgCrop);
eyeloc=locateeye(imgCrop);
noseloc=locatenose(imgCrop);
eye1=[eyeloc(1)+10 eyeloc(4)/2];
eye2=[eyeloc(1)+eyeloc(3)-10 eyeloc(4)/2];
nose1=[noseloc(1)+noseloc(3)/2 noseloc(2)+noseloc(4)/2];
eye_diff=eu_dist(eye1,eye2);
nose_eye1_diff=eu_dist(eye1,nose1)
nose_eye2_diff=eu_dist(eye2,nose1)
s=serial('COM3');
```

```
if index<6 && nose_eye1_diff<130&& nose_eye1_diff>110 && nose_eye2_diff<130 &&
nose_eye2_diff>115
   disp('Person 1 matched')
   axes(handles.axes1);
imshow(SelectedImage);
title('Equivalent Image using PCA');

elseif index>5 && index <11 && nose_eye1_diff<120&& nose_eye1_diff>110 &&
nose_eye2_diff<122 && nose_eye2_diff>110
   disp('Person 2 matched')
   axes(handles.axes1);
imshow(SelectedImage);
title('Equivalent Image using PCA');
   fopen(s);
   fwrite(s,'#N');
   fclose(s);
elseif index>10 && index <16 && nose_eye1_diff<120&& nose_eye1_diff>110 &&
nose_eye2_diff<120 && nose_eye2_diff>110
   disp('Person 3 matched')
   axes(handles.axes1);
imshow(SelectedImage);
title('Equivalent Image using PCA');

elseif index>15 && index <21 && nose_eye1_diff<130&& nose_eye1_diff>120 &&
nose_eye2_diff<135 && nose_eye2_diff>125
   disp('Person 4 matched')
   axes(handles.axes1);
imshow(SelectedImage);
title('Equivalent Image using PCA');

elseif index>20 && index <23 && nose_eye1_diff<130&& nose_eye1_diff>116 &&
nose_eye2_diff<130 && nose_eye2_diff>110
   disp('Person 5 matched')
   axes(handles.axes1);
imshow(SelectedImage);
title('Equivalent Image using PCA');

elseif index>22 && index <26 && nose_eye1_diff<130&& nose_eye1_diff>118 &&
nose_eye2_diff<135 && nose_eye2_diff>118
   disp('Person 6 matched')
   axes(handles.axes1);
imshow(SelectedImage);
title('Equivalent Image using PCA');


else
```

```
   disp('Not matched')
   d='NOT MATCHED';
   set(handles.text2,'string',d);

end
end
```

5.3 Voter Module

The voter module works on MVC architecture. MVC architecture provides interface and security for the authentication and validation process.

5.3.1 Spring Web MVC framework

The Spring Web model-view-controller (MVC) framework is designed around a Dispatcher Servlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale and theme resolution as well as support for uploading files. The default handler is based on the @Controller and @RequestMapping annotations, offering a wide range of flexible handling methods. With the introduction of Spring 3.0, the @Controller mechanism also allows to create RESTful Web sites and applications, through the @PathVariable annotation and other features.

5.3.2 Dispatcher Servlet

Spring's web MVC framework is, like many other web MVC frameworks, request-driven, designed around a central Servlet that dispatches requests to controllers and offers other functionality that facilitates the development of web applications. Spring's Dispatcher Servlet however, does more than just that. It is completely integrated with the Spring IoC container and as such allows you to use every other feature that Spring has.

The request processing workflow of the Spring Web MVC Dispatcher Servlet is illustrated in the following diagram. The Dispatcher Servlet is an expression of the "Front Controller" design pattern (this is a pattern that Spring Web MVC shares with many other leading web frameworks).

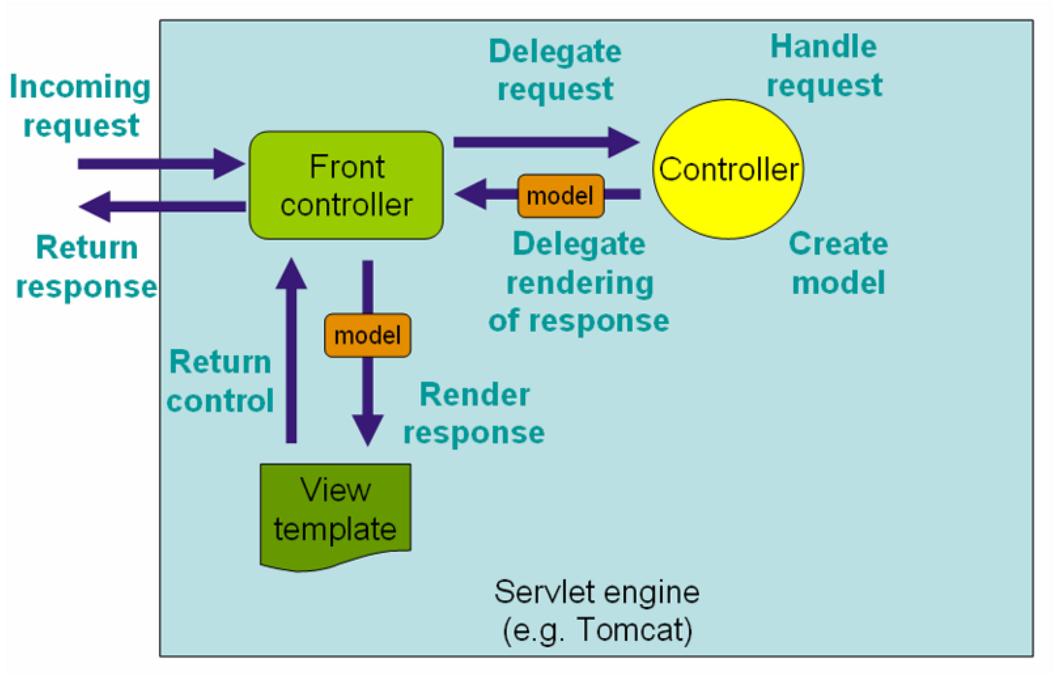


Fig 5.1 Request processing workflow in Spring Web MVC (high level)

The Dispatcher Servlet is an actual Servlet (it inherits from then HttpServlet base class), and as such is declared in the web.xml of the web application. In this project, dom.xml has been used to establish dependencies.

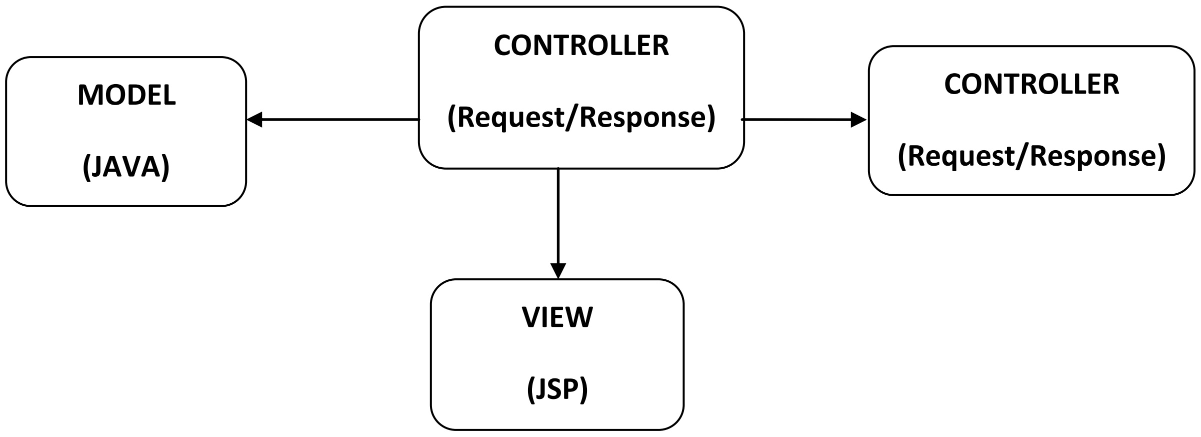


Fig 5.2 Programming Implementation of MVC

Whenever a service (cast vote) is requested by the voter, the controller initiates the model that is programmed in java, it checks with the status of the current voter and updates the model, the model in turn initiates the view and updates it. The view is programmed in JSP for this project. This project is a MAVEN project, dom.xml provides the dependencies.

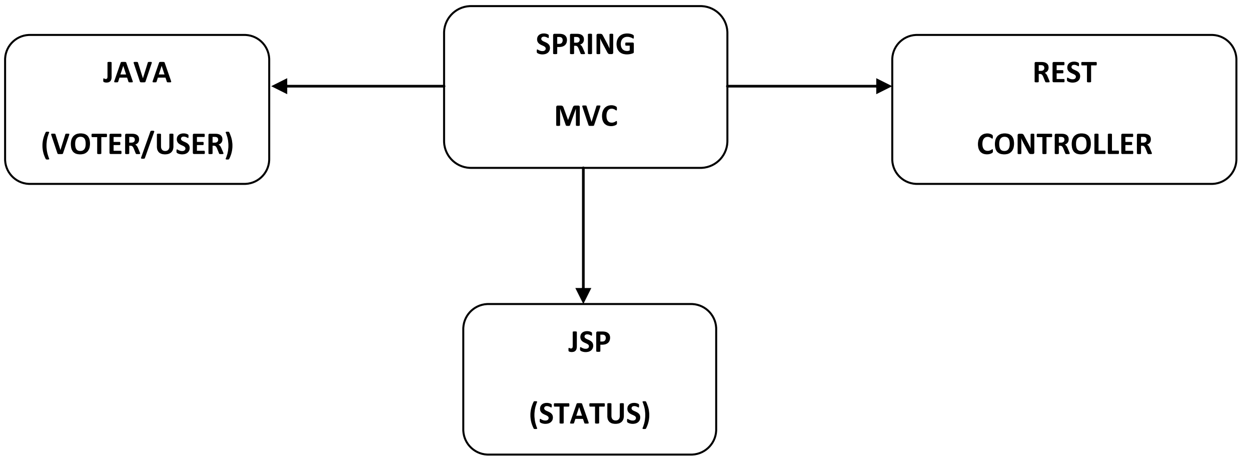


Fig 5.3 Implementation Structure of MVC

A character is sent from fingerprint recognition module to face recognition module on successful authentication. Then, a character is sent from face recognition module to Rest controller which in turn matches it with the id number in MySql database. This overall process involves ensuring “one person, one vote” and updating vote count in the database.

5.3.3 Source Code Implementation

As stated, our voter module is based upon MVC architecture, hence there are different files used for writing different component. All of which is mentioned below.

5.3.3.1 Java Model and User View

This is the model view where all the user information is stored and displayed, information includes name, address, Date of Birth and a status key which keeps the track whether the user have voted or is still left to caste the vote.

```
package com.handinhand.model;
public class VoterUser {
        int VoterId;
        String Name;
        String Address;
        int status;
        String dob;
```

```
	public VoterUser() {
	super();
	}
	public VoterUser(int voterId, String name, String address, int status) {
	super();
	VoterId = voterId;
	Name = name;
	Address = address;
	this.status = status;
	}
	public int getVoterId() {
	return VoterId;
	}
	public void setVoterId(int voterId) {
	VoterId = voterId;
	}
	public String getName() {
	return Name;
	}
	public void setName(String name) {
	Name = name;
	}
	public String getAddress() {
	return Address;
	}
	public void setAddress(String address) {
	Address = address;
	}
	public int getStatus() {
	return status;
	}
	public void setStatus(int status) {
	this.status = status;
	}
	public String getDob() {
	return dob;
	}
	public void setDob(String dob) {
	this.dob = dob;
	}
	@Override
	public String toString() {
	return "VoterUser [VoterId=" + VoterId + ", Name=" + Name
				+ ", Address=" + Address + ", status=" + status + "]";
	}
}
```

5.3.3.2 Dispatcher Servlet Configuration module

This module is responsible for loading the content listener of the Spring Framework, configure the location of the context and for creating and setting up the Servlet Framework.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/root-context.xml</param-value>
  </context-param>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/dispatcher-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

5.3.3.3 The Dispatcher Servlet Module

1. DispatcherServlet Context: defines this servlet's request-processing infrastructure.
2. Enables the Spring MVC @Controller programming model.
3. Handles HTTP GET requests for resources by efficiently serving up static resources in the ${webappRoot}/resources directory.
4. Resolves views selected for rendering by @Controllers to JSP resources in the /WEB-INF/views directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
	xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:context="http://www.springframework.org/schema/context"
	xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
	http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
	http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
	<!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
	<!-- Enables the Spring MVC @Controller programming model -->
	<annotation-driven />
	<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources
in the ${webappRoot}/resources directory -->
	<resources mapping="/resources/**" location="/resources/" />
	<resources mapping="/images/**" location="/images/" />
	<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-
INF/views directory -->
       <beans:bean  class="org.springframework.web.servlet.view.InternalResourceViewResolver">
	<beans:property name="prefix" value="/WEB-INF/views/" />
	<beans:property name="suffix" value=".jsp" />
	</beans:bean>
	<context:component-scan base-package="com.handinhand.controller" />
</beans:beans>
```

5.3.3.4 Status View

This is a JSP view which is responsible for the change in the Status of the Individual voter whether he/she has casted vote successfully or he/she is yet to caste vote. This is monitored with the help of the status key which keeps the track of the above-mentioned status. It also displays the Result button triggered after the end of the voting or after all voters have casted their votes Successfully.

```
<%@page import="com.handinhand.model.User"%>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="com.handinhand.controller.MySQLAccess"%>
<%@ page import="com.handinhand.model.*"%>
<%@ page import="java.util.*"%>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link                                                    rel="stylesheet"
```

```
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body style="background-color: #9dc8e2" align="center">
<h3>Finger Print Authentication and Online Voting Application</h3>

        <br>
        <br>
<%
        //UserBean user = new UserBean();
        MySQLAccess mySQLAccess = new MySQLAccess();
        List<VoterUser> trsList = mySQLAccess.getAllVotingUser();
%>
<table align="center" class="table table-bordered" style="width: 1000px">
        <tr>
        <th></th>
        <th>Voter Id</th>
        <th>Date of Birth</th>
        <th>Voter Name</th>
        <th>Voter Address</th>
        <th>Status</th>
        </tr>
        <tr>
        <%  int count=0;
                for (VoterUser user1 : trsList) {
        %>
        <% if(user1.getVoterId() == 1){
        %>
        <td><img src="./images/omprakash.jpg" style="width:100px;height:100px;"/>
        </td>
        <%
        } else if(user1.getVoterId() == 2){
        %>
        <td><img src="./images/aditya.jpg" style="width:100px;height:100px;" />
         </td>
        <%
        } else if(user1.getVoterId() == 3){
        %>
        <td><img src="./images/apurva.jpg"  style="width:100px;height:100px;"/>
        </td>
        <%
        } else if(user1.getVoterId() == 4){
        %>
        <td><img src="./images/kundan.jpg"  style="width:100px;height:100px;"/>
         </td>
        <%
        }
        %>
        <td><%=user1.getVoterId()%></td>
        <td><%=user1.getDob() %></td>
        <td><%=user1.getName()%></td>
        <td><%=user1.getAddress()%></td>
        <% if(user1.getStatus() == 1){
        %>
        <td><a                    href="/Voting/casteVote/?VoterId=<%=user1.getVoterId()%>"
style="color:red;">Vote</a></td>
        <%
        } else if(user1.getStatus() == 2){
                count++;
        %>
        <td>Vote Casted Successfully </td>
        <%
        } else if(user1.getStatus() == 3){
        %>
        <td>Yet to Cast Vote </td>
        <%
        }
        %>
        </tr>
        <%
        }
        //}
        %>
        <tr>
        <% if(count == 4){ %>
        <td ></td>
        <td ></td>
        <td ></td>
        <td ></td>
        <td > <a href="/Voting/getResults" style="color:red;">Show Results</a></td>
        <% } %>
        </tr>
</table>
</body>
</html>
```

5.3.3.5 JSP Module

This module is responsible for the view in which user/voter can see and caste the vote to his or her desired party and candidate.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body style="background-color: #9dc8e2" align="center">
<h3>Finger Print Authentication and Online Voting Application Cast Vote Details</h3>

        <br>
        <br>
<form:form method="post" commandName="vote" action="/Voting/Vote1">
 <div class="table-responsive">
        <table align="center" class="table table-bordered" style="width: 300px">
                <tr>
                        <td>Voter Id :</td>
                        <td><form:input type="text" path="voterId" /></td>
                </tr>
                <tr>
                        <td>Party Name </td>
                        <td>
                                <form:select path="partName">
                                        <form:option value="BJP" label="BJP"/>
                                        <form:option value="CONGRESS" label="CONGRESS"/>
                                        <form:option value="JDS" label="JDS"/>
                                        <form:option value="AAP" label="AAP"/>
                                        <form:option value="NOTA" label="NOTA" />
                                </form:select>
                        </td>
                </tr>
                <tr>
                        <td colspan="2"><input type="submit" value="Vote" /></td>
                </tr>
        </table>
        </div>
        </form:form>
```

```
</body>
</html>
```

5.3.3.6 The Result Module

After the end of the voting, this module i shows the various count of the votes like total votes casted, Vote casted to the individual candidate and the Winner of the voting based on the count .

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body style="background-color: #9dc8e2" align="center">
<h3>Finger Print Authentication and Online Voting Application Cast Vote Details</h3>

	<br>
	<br>
<div class="table-responsive">
	<table align="center" class="table table-bordered" style="width: 300px">
		<tr>
			<td>Total Vote Count :</td>
			<td><input type="text" value="4" editable="true"/></td>
		</tr>
		<tr>
			<td>Total BJP Party Count </td>
			<td><input type="text" value="${BJP}" editable="true"/></td>
		</tr>
		<tr>
			<td>Total CONGRESS Party Count  </td>
			<td><input type="text" value="${CONGRESS}" editable="true"/></td>
		</tr>
		<tr>
			<td>Total JDS Party Count  </td>
			<td><input type="text" value="${JDS}" editable="true"/></td>
		</tr>
		<tr>
```

```
                <td>Total AAP Party Count  </td>
                <td><input type="text" value="${AAP}" editable="true"/></td>
            </tr>
            <tr>
                <td>Total NOTA Count  </td>
                <td><input type="text" value="${NOTA}" editable="true"/></td>
            </tr>
            <tr>
                <td>Winner Of the Polling </td>
                <td><input type="text" value="${winner}" editable="true"/></td>
            </tr>

        </table>
        </div>
        <form id="dischargeForm" method="post" action="resetVotings" >
        <div class="table-responsive">
        <table align="center" class="table table-bordered" width="300px;">
            <tr>
                <td></td>
                <td><input class="btn btn-primary btn-sm" type="submit" value="Reset
Voting System" /></td>
            </tr>
        </table>
        </div>
        </form>
</body>
</html>
```

Chapter 6

TESTING

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also, to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words, software testing is a verification and validation process.

6.1 Types of Tests

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing

System testing is the testing to ensure that by putting the software in different environments it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

Beta Testing

Beta testing is the testing which is done by end users, a team outside development, or publicly releasing full pre-version of the product which is known as beta version. The aim of beta testing is to cover unexpected errors. It falls under the class of black box testing.

6.2 Unit Testing

Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.

Ideally, each test case is independent from the others. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

A unit test should usually not go outside of its own class boundary, and especially should not cross such process/network boundaries because this can introduce unacceptable performance problems to the unit test-suite. Crossing such unit boundaries turns unit tests into integration tests, and when test cases fail, makes it less clear which component is causing the failure. Instead, the software developer should create an abstract interface around the database queries, and then implement that interface with their own mock object. By abstracting this necessary attachment from the code, the independent unit can be more thoroughly tested than may have been previously achieved. This results in a higher-quality unit that is also more maintainable.

6.3 Acceptance Testing

The acceptance test suite may need to be performed multiple times, as all of the test cases may not be executed within a single test iteration. The acceptance test suite is run using predefined acceptance test procedures to direct the testers which data to use, the step-by-step processes to follow and the expected result following execution. The actual results are retained for comparison with the expected results. If the actual results match the expected results for each test case, the test case is said to pass. If the quantity of non-passing test cases does not breach the project's predetermined threshold, the test suite is said to pass. If it does, the system may either be rejected or accepted on conditions previously agreed between the sponsor and the manufacturer.

The anticipated result of a successful test execution:

- test cases are executed, using predetermined data
- actual results are recorded
- actual and expected results are compared, and
- test results are determined.

The objective is to provide confidence that the developed product meets both the functional and non-functional requirements. The purpose of conducting acceptance testing is that once completed, and provided the acceptance criteria are met, it is expected the sponsors will sign-off on the product development/enhancement as satisfying the defined requirements. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

6.4 Integration Testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. This is the testing that we have used for testing our system because it lets you tests interfaces and look for bugs that unit testing didn't catch. This testing is used because a set of modules that work fine individually rarely work together correctly for the first time.

6.5 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware systems. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

Test Strategy and Approach

- Testing will be performed manually and functional tests will be written in detail.

Test Objectives

- All the hardware should work properly.
- Database must be updated periodically.
- The messages and responses must not be delayed.

Features To Be Tested

- Verify the person and enroll in database.
- No duplicate entries should be allowed.
- Correct message should be displayed for authentication.
- Votes should be incremented after each vote for the respective candidates.

Software Testing

Test Case ID	**Scenario Description**	**Type of Testing**	**Expected Results**	**Actual Result**
TC1	Authenticating a user who is enrolled in Aadhaar database	Functionality Testing	Authenticated	Authenticated
TC2	Authenticating a user who is not enrolled in Aadhaar database	Functionality Testing	Unsuccessful Not eligible for voting	Unsuccessful Not eligible for voting
TC3	Person voting for the 2nd time	Functionality Testing	Not allowed	Not allowed
TC4	Casting a vote	Functionality Testing	Increment number of votes	Increment number of votes
TC5	Matching fingerprint but Face Recognition doesn't match	Integration Testing	Unsuccessful authentication	Unsuccessful authentication
TC6	Fingerprint doesn't match	Unit Testing	Unsuccessful authentication	Unsuccessful authentication
TC7	Reset for next elections	System Testing	Resets successfully	Resets successfully

Table 6.1 Test Cases

Chapter 7

SNAPSHOTS

The figures below show the data-flow sequence in real-time. When a voter enters a polling booth, he/she is asked to get the fingerprint recognized by placing the finger as shown in fig 7.1.

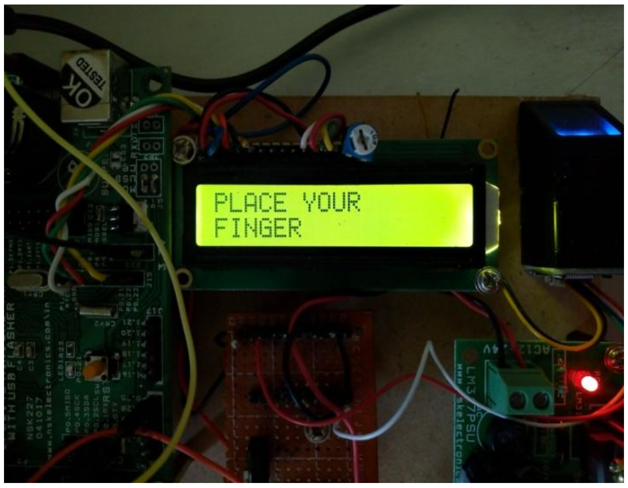


Fig 7.1 Asking the Voter to Place their Finger

Upon placing the finger, the fingerprint image is processed and compared to the fingerprint stored in the database. The ARM microcontroller has been used for this process and a wifi module serves as a tool to connect to the database available in the front-end capturing server. The database has been implemented in the ARM microcontroller which has the fingerprint images stored in its memory. After a voter places his/her finger on the fingerprint scanner the fingerprint module initiates image capturing of the finger and then finding the image in the stored database. If the scanned image matches with the image of fingerprint in the database the voter is found valid, the valid id is displayed as shown in the figure 7.2.

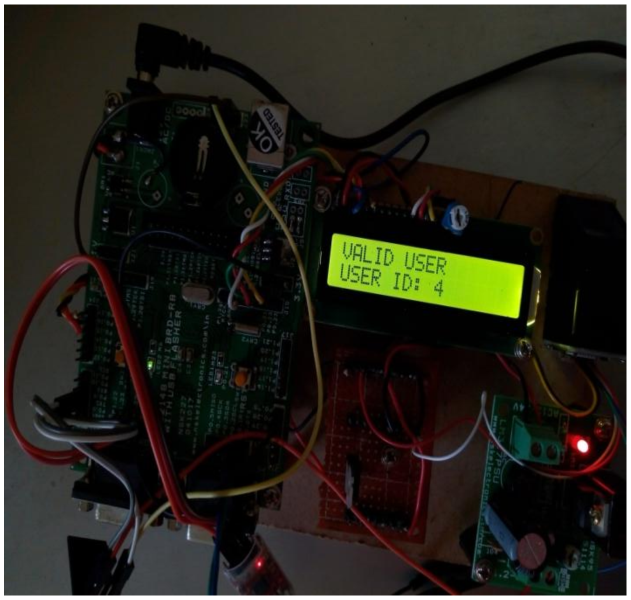


Fig 7.2 Displays Valid User ID after Successful Fingerprint Authentication

Upon successful verification of the voter’s fingerprint, a character is passed over to the face recognition system in MATLAB and the fingerprint module waits for the response as shown in fig 7.3.

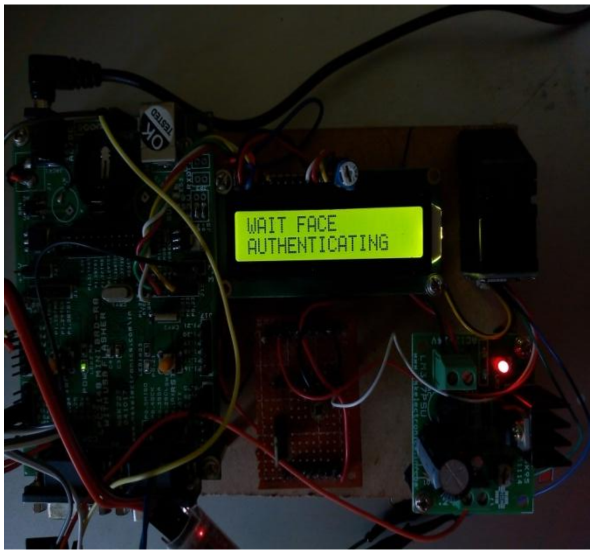


Fig 7.3 Requesting facial Authentication response

The facial module has been implemented using MATLAB and the test database is pre-stored as shown in fig 7.4.

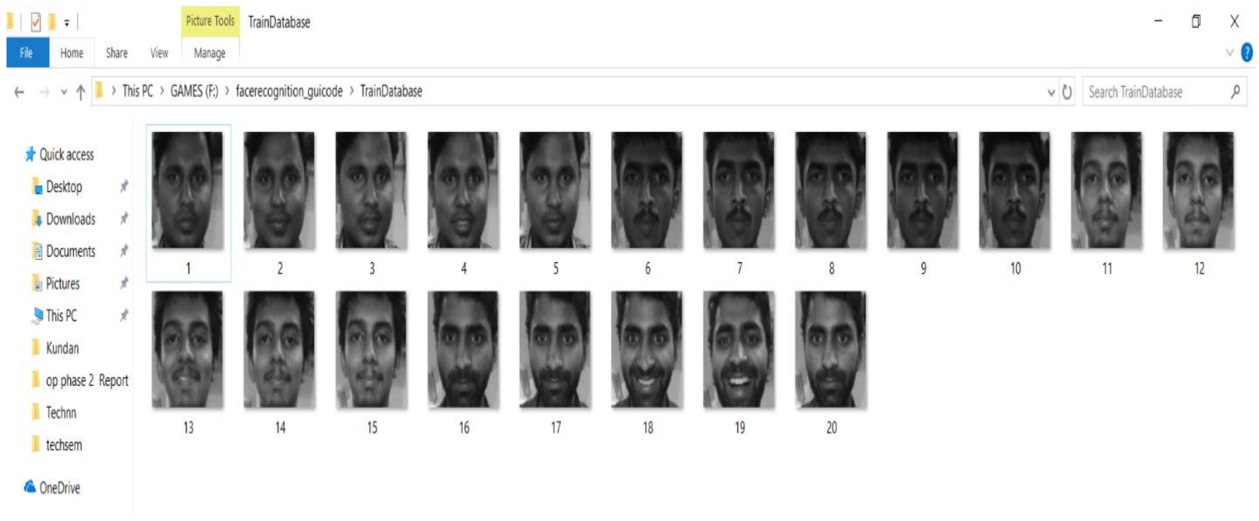
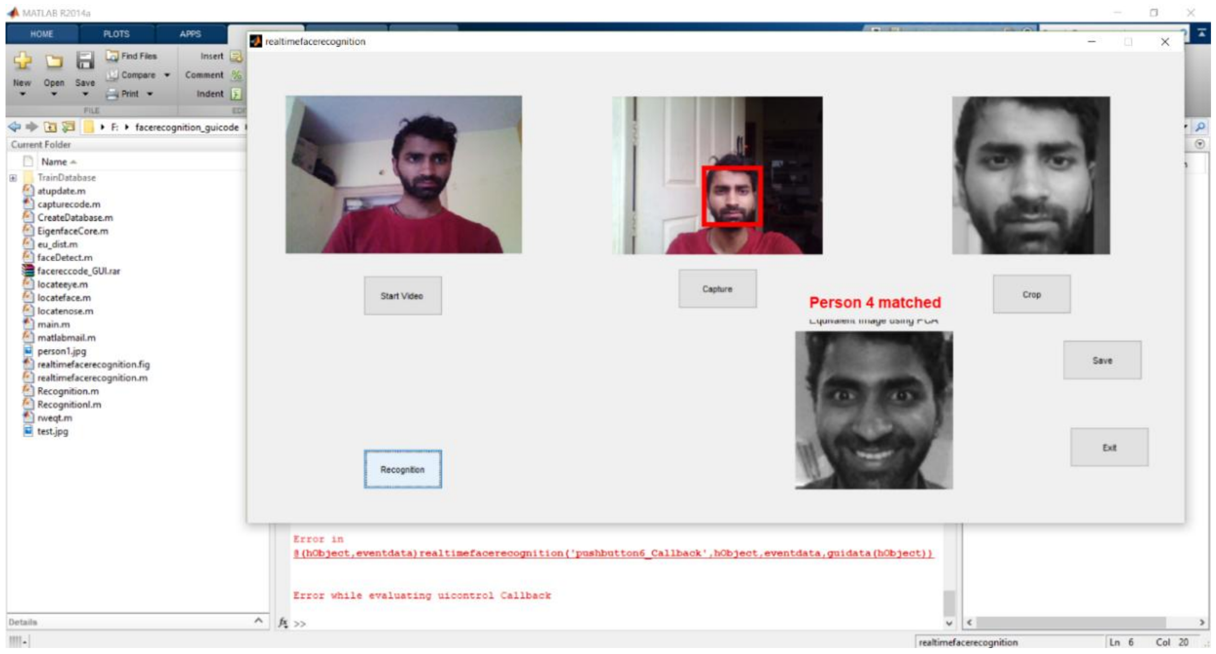


Fig 7.4 Stored Images in the Database

After fingerprint recognition a GUI console appears to initiate facial recognition, the real-time image of voter’s face is captured and matched with the images in the test database. Upon successful matching, the voter with the valid id is displayed as shown in fig 7.5.

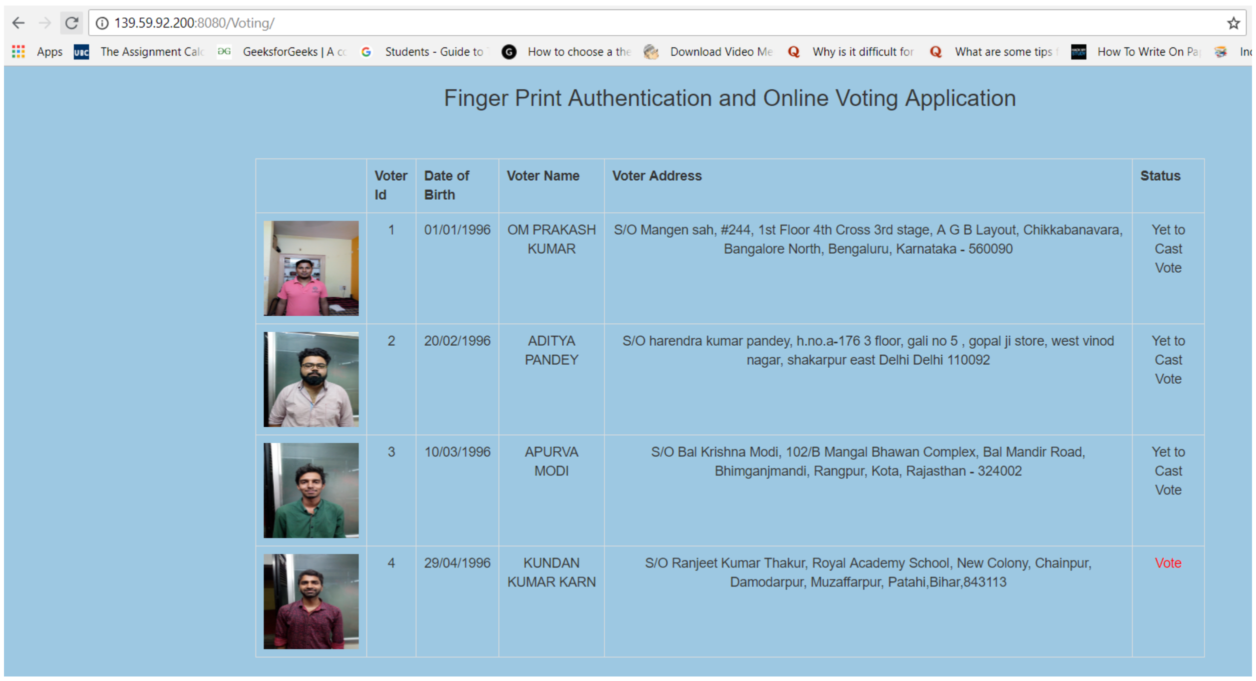


```
Error in
@(hObject,eventdata)realtimefacerecognition('pushbutton6_Callback',hObject,eventdata,guidata(hObject))


Error while evaluating uicontrol Callback
```

Fig 7.5 Successful Face Recognition of a Voter

After the face recognition process is successful, a character is passed from MATLAB to the frontend capturing server. An interactive voting portal opens up on the screen and the voter with associated voter id is now allowed to cast a vote as shown in fig 7.6.

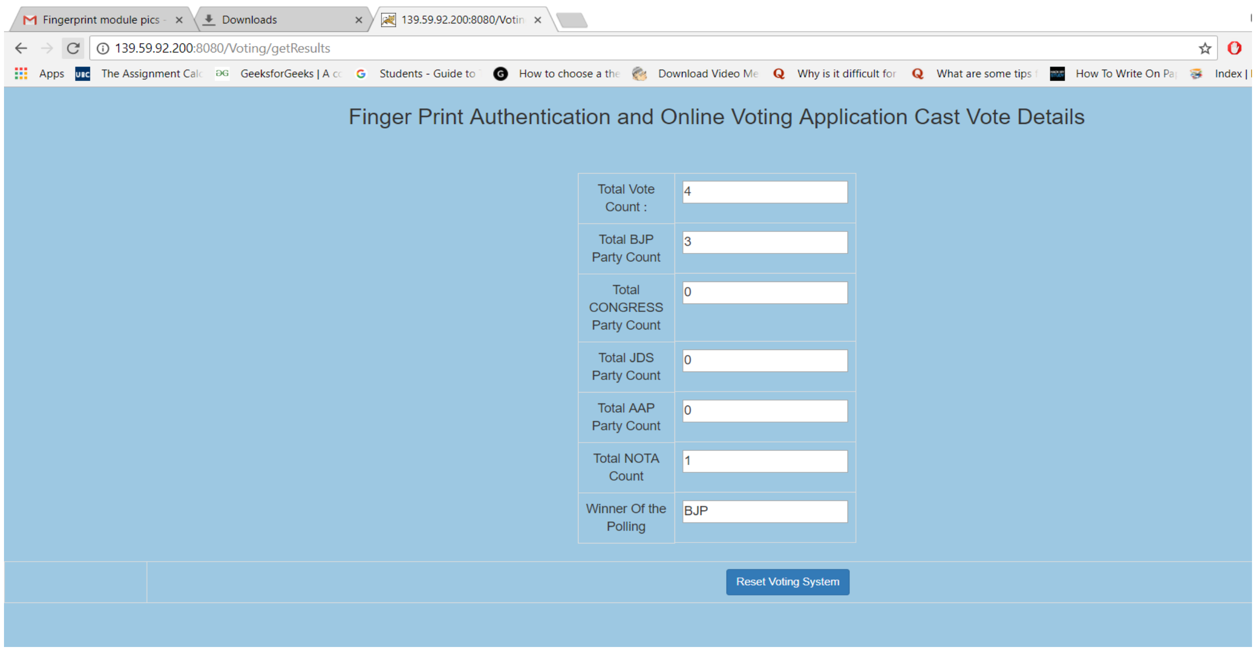


Finger Print Authentication and Online Voting Application

	Voter Id	Date of Birth	Voter Name	Voter Address	Status
	1	01/01/1996	OM PRAKASH KUMAR	S/O Mangen sah, #244, 1st Floor 4th Cross 3rd stage, A G B Layout, Chikkabanavara, Bangalore North, Bengaluru, Karnataka - 560090	Yet to Cast Vote
	2	20/02/1996	ADITYA PANDEY	S/O harendra kumar pandey, h.no.a-176 3 floor, gali no 5 , gopal ji store, west vinod nagar, shakarpur east Delhi Delhi 110092	Yet to Cast Vote
	3	10/03/1996	APURVA MODI	S/O Bal Krishna Modi, 102/B Mangal Bhawan Complex, Bal Mandir Road, Bhimganjmandi, Rangpur, Kota, Rajasthan - 324002	Yet to Cast Vote
	4	29/04/1996	KUNDAN KUMAR KARN	S/O Ranjeet Kumar Thakur, Royal Academy School, New Colony, Chainpur, Damodarpur, Muzaffarpur, Patahi,Bihar,843113	Vote

Fig 7.6 Voting Portal opens up for the Voter id 4 with status to caste his/her vote

After the election gets over, the winner is selected based on the number of vote count which is accessible only to authorized officers of election commission. The result count is calculated using the result sub module in the voting module. The result is shown on an interacting “result” portal which shows the total number of count, each party’s individual count and the winner of election based on the maximum no. of votes received. The voting portal also includes NOTA (None of the above) along with the names of other parties in that constituency. The voting count is maintained in the front end database connected to the front end capturing server. The result module also provides an option to reset the entire voting process. The result is displayed on the result portal as shown in fig 7.7.



Finger Print Authentication and Online Voting Application Cast Vote Details

Total Vote Count :	4
Total BJP Party Count	3
Total CONGRESS Party Count	0
Total JDS Party Count	0
Total AAP Party Count	0
Total NOTA Count	1
Winner Of the Polling	BJP

Fig 7.7 Displaying Results

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENT

This system provides best solutions to the problems related to the voting system. It focuses on providing end-to-end transparency in the voting system to protect and save democracy. In our voting process authentication is done by using fingerprint and iris authentication to cast votes which ensures that vote casting cannot be altered by an unauthorized person. The proposed systems have many advantages over the traditional way of voting.

The importance of voting is maintained and in general the focus is on making the voting system more robust and reliable by eliminating fraud voting. This voting system helps everybody to cast their votes without any problem. By using this newly developed system we can overcome many problems of existing system. This system is more efficient than the existing one.

Thus the advent of this Aadhar enabled authenticated electoral system would enable hosting of fair elections in India. This will preclude the illegal practices. The citizens can be sure that they can choose their leaders securely and fairly, thus exercising their right in the democracy.

The future scope for enhancements in this project are listed below

- It can be enhanced to support regional languages so that it will be easy for people in rural areas.
- To embed an intelligent speaking system for visually impaired people so that they can hear the instructions.
- To introduce another type of authentication for differently abled people.
- To facilitate remote voting for military personnel and on duty officials.

REFERENCES

[1] Ankita R. Kasliwal, Jaya S. Gadekar, Manjiri A. Lavadkar, Pallavi K. Thorat, Dr.Prapti Deshmukh, “Aadhar Based Election Voting System”, March 2014.

[2]B.Mary Havilah Haque, G.M.Owais Ahmed, D.Sukruthi, K.Venu Gopal Achary, C.Mahendra Naidu , “Fingerprint and RFID Based Electronic Voting System Linked With AADHAAR For Rigging Free Elections”, March 2016.

[3]M.Venkata Rao, Venugopal Rao Ravula, Pavani Pala , “Development Of Antirigging Voting System Using Biometrics Based On Adhar Card Numbering”, February 2015.

[4]Alaguvel.R, Gnanavel.G, Jagadhambal.K, “Biometrics using Electronic Voting System with Embedded Security”, IJARCET, March 2013.

[5]D.Krishna, T.Hemalatha, G.Dhana Mani Shankar, K.Bala Krishna, V.BalaSubhramanyam, “Aadhar based electronic voting system and providing authentication”, IJESAT, March 2014

[6]DhineshKumar.M, Santhosh.A, Aranganadhan.N.S, Praveenkumar.D, “Embedded System for Voting Machine System using Wireless Technology”, February 2016.

[7]Shital A PAtil, Praveen G Kote, “IRIS Detection in Voting System”, August 2015.

[8]Rohan Patel, Vaibhav Ghorpade, Vinay Jain ,MansiKambli, “Fingerprint Based e-Voting System using Aadhar Database”, March 2015

[9]R. Murali Prasad, Polaiah Bojja, Madhu Nakirekanti, “Aadhar based Electronic Voting Machine using Arduino”, July 2016.