# **Two Factor Authentication Voting System**

# **ABSTRACT**

Implementing a secure electronic voting system is crucial to ensure the integrity and reliability of elections. One effective approach involves integrating multi-factor authentication (MFA) mechanisms, such as biometrics and cryptographically secured smart cards, to enhance voter authentication. For instance, combining fingerprint biometrics with smart cards can significantly reduce the risk of voter impersonation and unauthorized access.

Additionally, employing MFA methods that require voters to present multiple verification factors—something they know, something they have, or something they are—further strengthens the security framework.

These measures collectively address common challenges in traditional voting systems, such as vote-rigging and vote falsification, thereby ensuring that the final count accurately reflects the true will of the voters.

# **INTRODUCTION**

Electronic voting (e-voting) systems have transformed the electoral landscape by offering efficiency and accessibility. However, they also present challenges related to security and voter authentication. Traditional voting methods, while familiar, are susceptible to issues such as vote-rigging, voter impersonation, and vote falsification. The integration of digital technologies in voting processes necessitates robust security measures to maintain electoral integrity.

Two-factor authentication (2FA) has emerged as a pivotal solution to enhance the security of e-voting systems. By requiring voters to present two distinct forms of verification—typically something they know (like a password or PIN) and something they have (such as a mobile device or biometric identifier)—2FA significantly reduces the risk of unauthorized access. For instance, implementing a mobile voting system that utilizes blockchain technology and 2FA can securely store cast votes and authenticate voters before they cast their votes, providing an easily accessible, secure, and transparent m-voting system.

The application of 2FA in e-voting is not merely a theoretical concept but a practical approach adopted in various systems. For example, Vero Voting employs 2FA by distributing unique user IDs and PINs to voters via email or SMS. Upon logging in, voters are prompted to enter a code sent to their registered mobile number or email address, adding an extra layer of security to the voting process.

Moreover, the U.S. Election Assistance Commission's Voluntary Voting System Guidelines (VVSG 2.0) emphasize the importance of multi-factor authentication for critical operations within voting systems. These guidelines advocate for the use of 2FA to ensure that only authorized individuals can access sensitive components of the voting infrastructure, thereby safeguarding the electoral process from potential threats.

Incorporating 2FA into e-voting systems addresses several security concerns inherent in digital voting platforms. By enhancing voter authentication mechanisms, 2FA ensures that only eligible voters can cast ballots, thereby upholding the principles of democratic participation and electoral integrity.

# **LITERATURE SURVEY**

**Rivest, R. L. (2001). Electronic voting. Proceedings of Financial Cryptography, Springer.**  
Rivest explores the fundamental challenges and security concerns of electronic voting (e-voting) systems. He highlights the risks associated with software-based voting systems, including malware attacks, vote tampering, and voter privacy concerns. The study emphasizes the need for verifiable voting mechanisms to ensure election integrity. Rivest discusses the limitations of paper-based voting and argues that electronic voting should incorporate cryptographic techniques for end-to-end verification. He also proposes that voter authentication mechanisms must be robust to prevent unauthorized access. The paper further suggests that two-factor authentication (2FA) could enhance voter identity verification, reducing risks of fraudulent voting. Additionally, Rivest emphasizes that e-voting must be both accessible and secure, advocating for transparent systems that allow voters to verify their choices. This work laid the groundwork for subsequent research in e-voting security, influencing the design of cryptographic voting protocols and the development of secure voting technologies.

**Mercuri, R. (2002). A better ballot box? IEEE Spectrum, 39(10), 46-50.**  
Mercuri critically examines the vulnerabilities in traditional and electronic voting systems, questioning their reliability and security. She argues that while electronic voting systems offer efficiency and convenience, they also introduce risks such as software bugs, hacking, and lack of transparency in vote counting. The paper proposes the concept of a "voter-verified paper audit trail" (VVPAT) to enhance trust in e-voting. Mercuri discusses real-world cases where electronic voting machines have failed, emphasizing the importance of physical records for election verification. The study suggests that an ideal voting system should be a hybrid model, integrating digital efficiency with paper-based security. Mercuri also highlights the importance of usability and accessibility in voting systems, ensuring that all voters, including those with disabilities, can participate securely. Her work played a crucial role in shaping policy discussions on electronic voting security, influencing later implementations of VVPAT in electoral processes worldwide.

**Rubin, A. D. (2002). Security considerations for remote electronic voting. Communications of the ACM, 45(12), 39-44.**  
Rubin addresses the security risks associated with remote electronic voting, particularly focusing on internet-based systems. He explores threats such as denial-of-service (DoS) attacks, phishing, malware, and man-in-the-middle (MITM) attacks, which can compromise voter authentication and ballot integrity. The study underscores the difficulty of ensuring voter anonymity while maintaining election transparency. Rubin proposes using strong cryptographic authentication techniques, including two-factor authentication (2FA), to mitigate identity fraud. He also discusses potential solutions such as digital signatures, secure socket layer (SSL) encryption, and blockchain-based voting for ensuring tamper-proof records. Furthermore, the paper highlights the need for rigorous third-party audits and verifiable election processes. Rubin’s research serves as a foundational work for developing secure remote voting models, influencing later studies on cryptographic voting protocols and secure voter authentication. His insights remain relevant as governments and organizations explore internet voting solutions in modern elections.

**Kohno, T., Stubblefield, A., Rubin, A. D., & Wallach, D. S. (2004). Analysis of an electronic voting system. IEEE Symposium on Security and Privacy, 27-40.**  
This study provides a detailed security analysis of a widely used electronic voting system, exposing its vulnerabilities and potential attack vectors. Kohno et al. conducted a comprehensive review of the Diebold AccuVote-TS system, demonstrating that the machine lacked robust security mechanisms, making it susceptible to vote tampering. The paper highlights issues such as weak encryption, insecure storage of votes, and lack of proper voter authentication. The researchers also show how malicious software could be installed to manipulate election results without detection. As a response, they propose improvements, including stronger authentication measures like two-factor authentication (2FA), voter-verifiable audit trails, and cryptographic verification methods. The study was instrumental in raising awareness about e-voting security risks, prompting legislative and regulatory discussions on electronic voting system integrity. Their findings led to increased scrutiny of e-voting machines and encouraged the development of more secure and transparent electoral systems worldwide.

**Neumann, P. G. (2004). The risks of electronic voting. Communications of the ACM, 50(11), 136.**  
Neumann provides an in-depth discussion of the risks associated with electronic voting systems, emphasizing their susceptibility to fraud, software errors, and insider attacks. He argues that while e-voting can improve accessibility and efficiency, it also introduces new attack surfaces that threaten electoral integrity. The paper categorizes threats into software vulnerabilities, hardware manipulation, network security breaches, and inadequate authentication measures. Neumann stresses the need for verifiable election systems where voters can confirm that their votes are correctly recorded and counted. He suggests incorporating end-to-end encryption, biometric authentication, and two-factor authentication (2FA) to prevent unauthorized voting. Additionally, the study calls for independent security audits and open-source software development to increase transparency. His research significantly influenced policy discussions on election security, advocating for stronger regulatory frameworks to ensure secure and trustworthy e-voting systems. Neumann's insights continue to shape modern electronic voting security protocols.

**Cranor, L. F. (2008). A framework for reasoning about the human in the loop. USENIX Security Symposium.**  
Cranor’s study explores the role of human factors in electronic voting security, emphasizing how usability issues impact election integrity. She argues that while strong cryptographic security measures are essential, they must be designed with user experience in mind to ensure widespread adoption. The paper highlights how poorly designed authentication mechanisms can lead to voter errors, disenfranchisement, or security bypasses. Cranor proposes a human-centered framework for designing secure voting systems, incorporating principles of usability testing, accessibility, and cognitive load analysis. She suggests that voter authentication methods, including two-factor authentication (2FA), should balance security with ease of use to prevent resistance or system misuse. Additionally, the study discusses how education and awareness campaigns can help voters recognize phishing attempts and fraudulent authentication requests. Cranor’s work is instrumental in bridging the gap between security engineering and user experience, leading to the development of more user-friendly yet secure electronic voting systems.

**Estehghari, S., & Desmedt, Y. (2010). Exploiting the client vulnerabilities in internet e-voting systems. IEEE International Conference on e-Voting Security.**  
Estehghari and Desmedt analyze security flaws in internet-based electronic voting systems, focusing on client-side vulnerabilities. They demonstrate how malware, keyloggers, and compromised devices can be exploited to alter votes or steal voter credentials. The study examines the risks associated with browser-based voting interfaces, where attackers can inject malicious scripts to manipulate voter interactions. The authors propose countermeasures such as hardware-based security tokens, biometric authentication, and two-factor authentication (2FA) to enhance voter authentication. They also discuss using blockchain technology to create a decentralized and tamper-proof voting record. The research highlights the importance of educating voters on cybersecurity risks and enforcing strict security protocols to mitigate external threats. Their findings underscore the need for continuous security audits and improvements in online voting systems, influencing later research on secure digital election mechanisms. This study remains a valuable resource for policymakers and developers designing secure online voting platforms.

# **EXISTING SYSTEM**

The traditional voting system primarily relies on paper-based ballots and manual vote counting. In this method, voters physically visit designated polling stations, mark their choices on paper ballots, and submit them into a ballot box. Once the voting process is completed, election officials manually count the votes, which is time-consuming and prone to human errors. This method, while considered reliable, faces several challenges, including logistical complexities, high costs, and risks of ballot tampering or misplacement. Additionally, manual vote counting is susceptible to fraud, such as multiple voting, miscounting, or manipulation by election officials. In some cases, electronic voting machines (EVMs) have been introduced, but these systems lack transparency and verifiability, leading to security concerns. Furthermore, voter authentication in traditional methods relies on physical identity verification, which can be bypassed through impersonation or forged identification documents. These limitations highlight the need for a more secure, efficient, and transparent voting system.

# **DISADVANTAGES**

1. **Time-Consuming Process** – Traditional voting methods, including paper ballots and manual counting, require significant time for vote collection, counting, and result declaration, leading to delays.
2. **Security and Fraud Risks** – Paper-based voting is vulnerable to tampering, ballot stuffing, miscounting, and impersonation, which can compromise election integrity.
3. **High Operational Costs** – Conducting elections using paper ballots involves expenses related to printing, transportation, storage, and security, making it costly and resource-intensive.
4. **Limited Accessibility** – Voters need to physically visit polling stations, which may be challenging for individuals with disabilities, those living in remote areas, or expatriates.
5. **Lack of Transparency** – In electronic voting machines (EVMs), the absence of a verifiable audit trail raises concerns about result manipulation and reduces public trust in the election process.

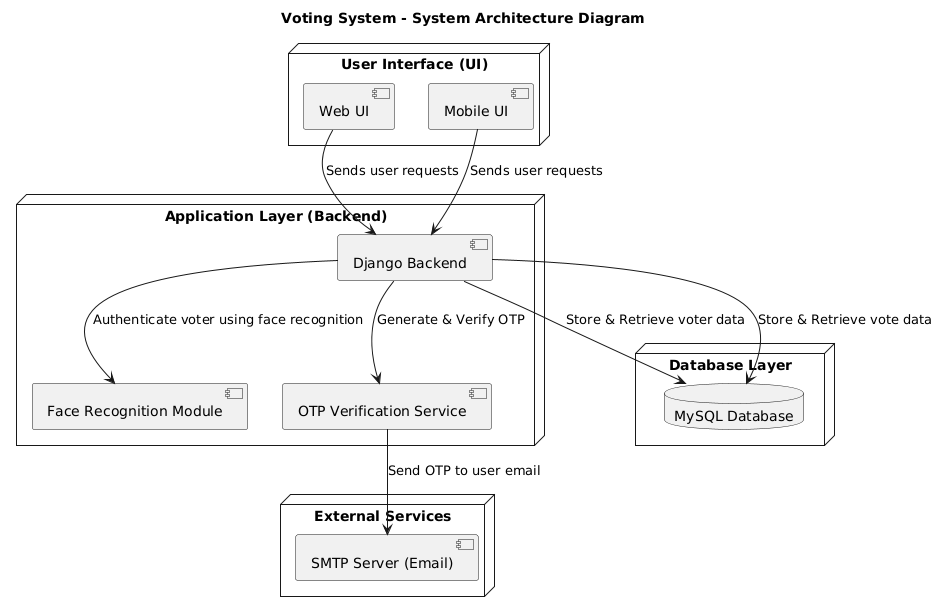
# **PROPOSED SYSTEM**

The proposed system aims to enhance the efficiency, security, and transparency of the voting process by leveraging advanced cryptographic techniques and secure authentication mechanisms. This system ensures a tamper-proof environment by integrating blockchain technology, which provides a decentralized and immutable ledger for vote recording. Additionally, biometric authentication and multi-factor verification methods are incorporated to eliminate voter fraud and ensure that only authorized individuals can cast their votes. The proposed system also focuses on accessibility by enabling remote electronic voting through a secure web-based or mobile application, allowing voters to participate from any location while maintaining data integrity and confidentiality. Furthermore, real-time vote tallying and verification mechanisms improve the accuracy and speed of result computation, reducing the reliance on manual counting. By addressing the limitations of traditional and existing electronic voting methods, the proposed system enhances voter trust, minimizes security risks, and promotes a seamless, user-friendly election experience.

# **ADVANTAGES**

1. **Enhanced Security** – Unlike traditional voting systems that are prone to tampering and fraud, the proposed system integrates blockchain technology and encryption methods to ensure the integrity and confidentiality of votes. This prevents unauthorized access and manipulation of election results.
2. **Elimination of Voter Fraud** – The existing system lacks robust authentication, allowing multiple voting or identity fraud. The proposed system incorporates biometric authentication and multi-factor verification to ensure that only legitimate voters can cast their votes.
3. **Improved Accessibility** – Traditional voting methods require physical presence at polling stations, making it difficult for remote voters to participate. The proposed system enables online voting, allowing voters to cast their votes securely from anywhere, increasing voter turnout.
4. **Faster and Accurate Vote Counting** – Manual vote counting in the existing system is time-consuming and prone to errors. The proposed system provides real-time vote tallying, ensuring quick and precise election results with minimal human intervention.
5. **Tamper-Proof and Transparent Elections** – Traditional voting systems face risks of ballot manipulation and result discrepancies. By using a decentralized ledger, the proposed system ensures transparency, making election data verifiable and immutable, thereby increasing trust in the electoral process.

# **SYSTEM ARCHITECTURE**



# **SYSTEM REQUIREMENTS**

**➢ H/W System Configuration:-**

**➢ Processor - Pentium –IV**

**➢ RAM - 4 GB (min)**

**➢ Hard Disk - 20 GB**

**➢ Key Board - Standard Windows Keyboard**

**➢ Mouse - Two or Three Button Mouse**

**➢ Monitor - SVGA**

**SOFTWARE REQUIREMENTS:**

1. **Operating system : Windows 7 Ultimate.**
2. **Coding Language : Python.**
3. **Front-End : Python.**
4. **Back-End : Django-ORM**
5. **Designing : Html, css, javascript.**
6. **Data Base : MySQL / (WAMP Server).**

# **SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# **SYSTEM DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram was capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

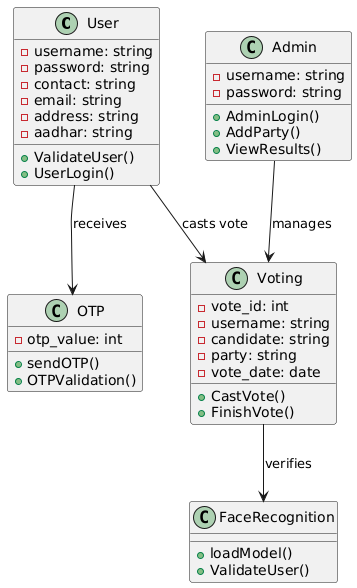


Figure-5.1: Class Diagram

**Sequence Diagram**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

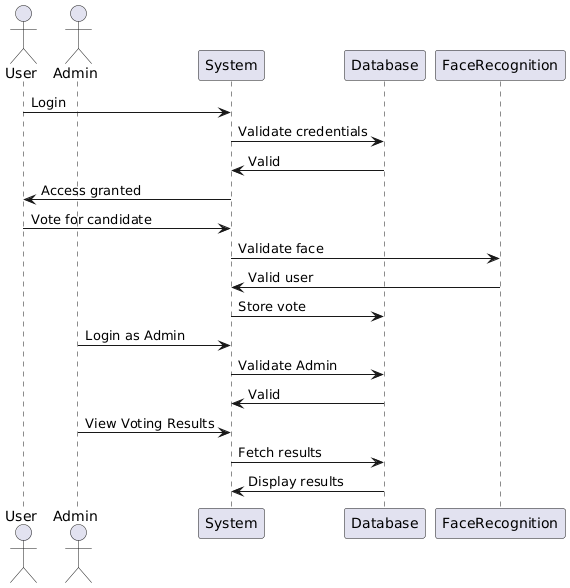


Figure-5.2: Sequence Diagram

**Activity diagram**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

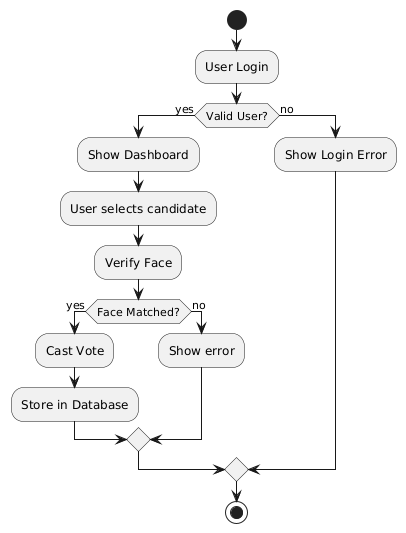


Figure-5.3: Activity Diagram

**Data flow diagram**

A data flow diagram (DFD) is a graphical representation of how data moves within an information system. It is a modeling technique used in system analysis and design to illustrate the flow of data between various processes, data stores, data sources, and data destinations within a system or between systems. Data flow diagrams are often used to depict the structure and behavior of a system, emphasizing the flow of data and the transformations it undergoes as it moves through the system.

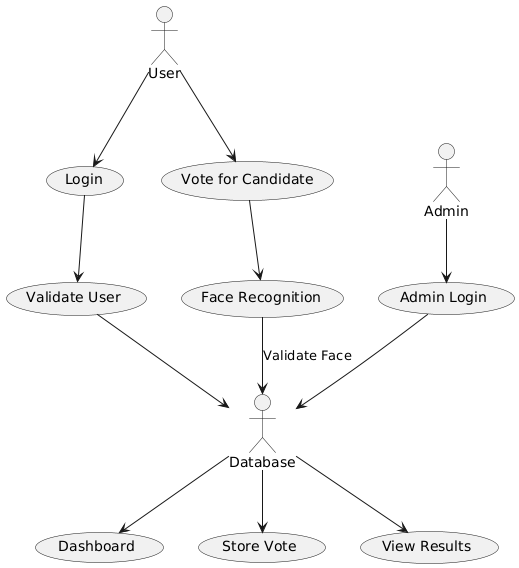


Figure-5.4: Dataflow Diagram

**Component diagram:** Component diagram describes the organization and wiring of the physical components in a system.

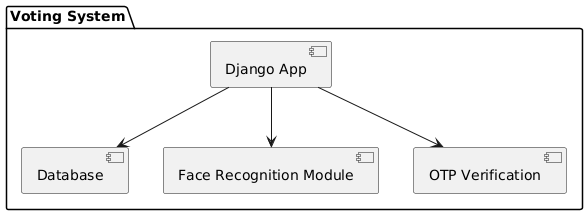
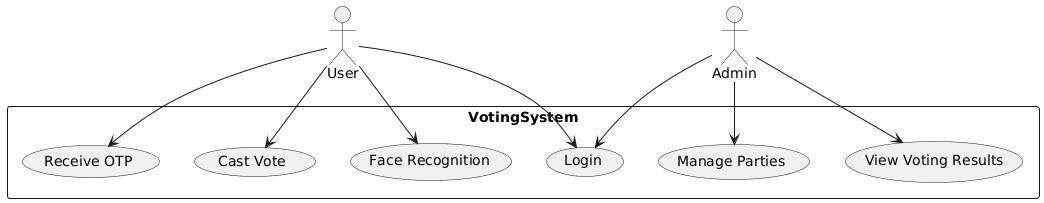


Figure-5.5: Component Diagram

**Use Case diagram:** A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Deployment Diagram:**

A deployment diagram in UML illustrates the physical arrangement of hardware and software components in the system. It visualizes how different software artifacts, such as data processing scripts and model training components, are deployed across hardware nodes and interact with each other, providing insight into the system’s infrastructure and deployment strategy.

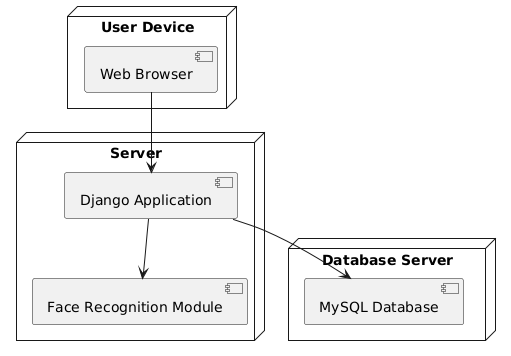
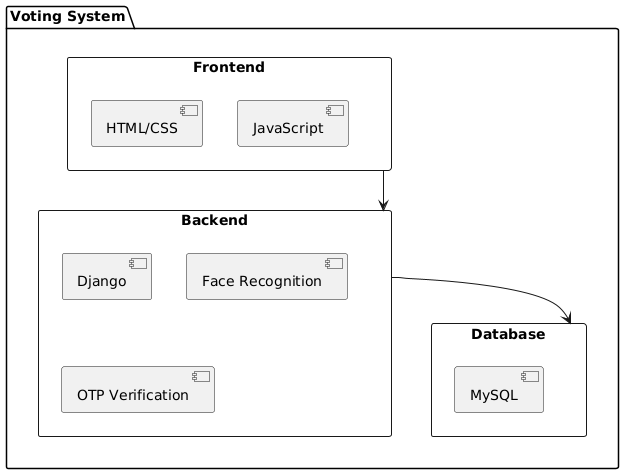


Figure-5.4: DeploymentDiagram

**Architectural Block Diagram**

An architectural block diagram offers a high-level view of a system’s structure, showcasing the main components and their interactions. It represents how major modules, such as data sources, processing units, and evaluation components, are organized and how they communicate with each other to accomplish the system’s objectives. This diagram helps in understanding the overall design and flow of the system.



# 

# **SOFTWARE ENVIRONMENT**

**What is Python :-**

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* + [Machine Learning](https://www.geeksforgeeks.org/machine-learning/)
  + GUI Applications (like Kivy, Tkinter, PyQt etc. )
  + Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  + Image processing (like Opencv, Pillow)
  + Web scraping (like Scrapy, BeautifulSoup, Selenium)
  + Test frameworks
  + Multimedia

**Advantages of Python :-**

Let’s see how Python dominates over other languages.

**1. Extensive Libraries**

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you [download Python](https://data-flair.training/blogs/install-python-windows/) for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features**.**

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

*Any doubts till now in the advantages of Python? Mention in the comment section.*

**Advantages of Python Over Other Languages**

**1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and [machine learning](https://data-flair.training/blogs/machine-learning-tutorials-home/), automate things,do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

**1. Speed Limitations**

We have seen that Python code is executed line by line. But since [Python](https://www.python.org/) is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically-typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**History of Python : -**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**Modules Used in Project**

**TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.‍

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**What is Machine Learning : -**

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

**Categories Of Machine Leaning :-**

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

*Supervised learning* involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

*Unsupervised learning* involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

**Need for Machine Learning**

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

**Challenges in Machines Learning :-**

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

**Applications of Machines Learning :-**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

**How to Start Learning Machine Learning?**

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to [Indeed](http://blog.indeed.com/2019/03/14/best-jobs-2019/), Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

**How to start learning ML?**

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

**Step 1 – Understand the Prerequisites**

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

**(a) Learn Linear Algebra and Multivariate Calculus**

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is veryimportant as you will have to implement many ML algorithms from scratch.

**(b) Learn Statistics**

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!  
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

**(c) Learn Python**

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is [Python](https://www.geeksforgeeks.org/python-programming-language/)! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as [Keras](https://keras.io/" \t "_blank), [TensorFlow](https://www.tensorflow.org/), [Scikit-learn](https://scikit-learn.org/stable/), etc.

So if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as [Fork Python](https://practice.geeksforgeeks.org/courses/fork-python) available Free on GeeksforGeeks.

**Step 2 – Learn Various ML Concepts**

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

**(a) Terminologies of Machine Learning**

* Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
* Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
* Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
* Training – The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
* Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

**(b) Types of Machine Learning**

* Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
* Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
* Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
* Reinforcement Learning – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

**Advantages of Machine learning :-**

**1. Easily identifies trends and patterns -**

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

**2. No human intervention needed (automation)**

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

**3. Continuous Improvement**

As [ML algorithms](https://data-flair.training/blogs/machine-learning-algorithms/) gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

**4. Handling multi-dimensional and multi-variety data**

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

**5. Wide Applications**

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Disadvantages of Machine Learning :-**

**1. Data Acquisition**

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated**.**

**2. Time and Resources**

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

**3. Interpretation of Results**

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

**4. High error-susceptibility**

[Machine Learning](https://en.wikipedia.org/wiki/Machine_learning) is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

# **SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# **IMPLIMENTATION**

**MODULES**

In this project as per your requirements we have designed following modules

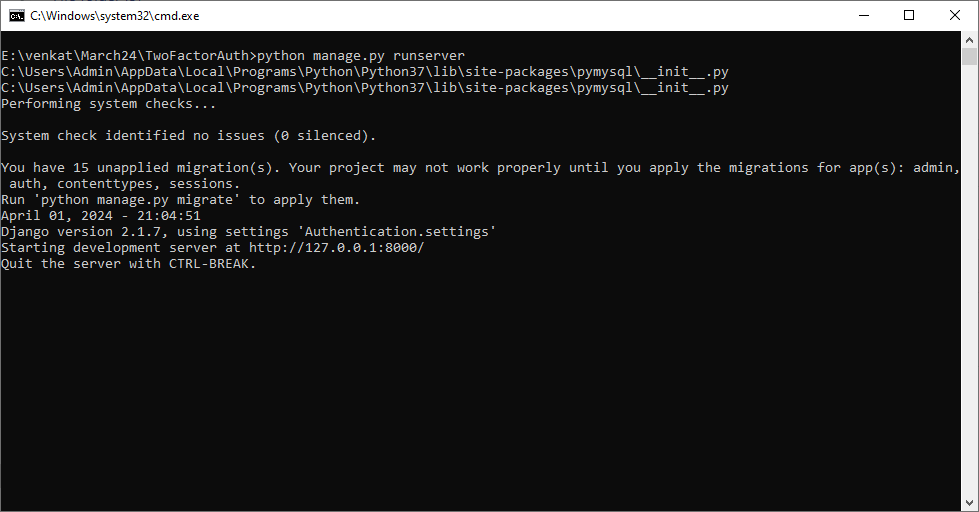
1. Admin module: admin can login to system using username and password as ‘admin and admin’. After login admin will add new party details along with symbols. Admin will add voter details by capturing information like AADHAR number, thumb image, email, phone number and then capture users photo. Admin can view all part details and can view vote count for each party
2. User Module: user can login to system by using username, AADHAR number and thumb image. If all this details matched then OTP will be forward to user registered mail. After OTP validation user will be forward for face capturing and recognition. Once face recognized then user will be forward to cast vote by displaying various parties. User can cast his vote to desired party.

.

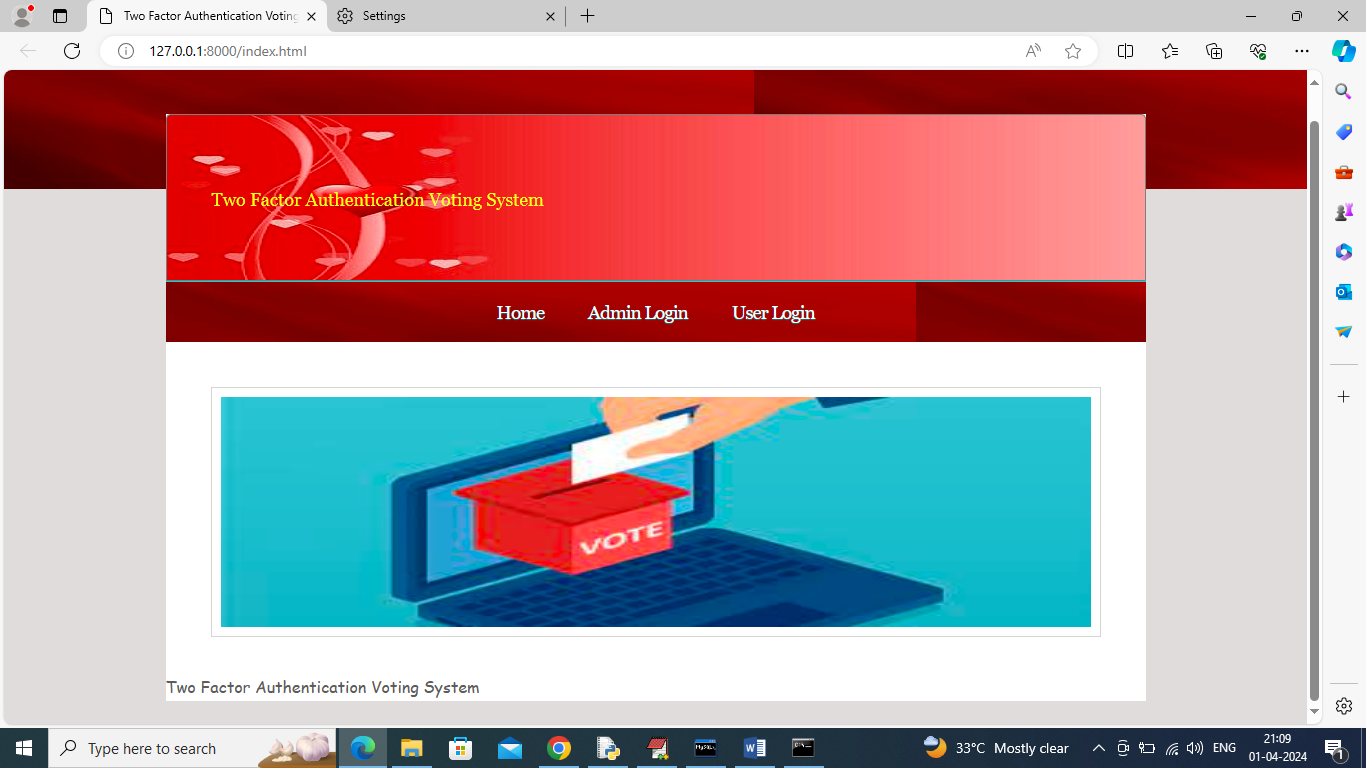
**SCREEN SHOTS**

To run project install PYTHON 3.7 and then install all packages for requirements file and then install MYSQL database and then copy content from ‘database.txt’ file and paste in MYSQL console to create database.

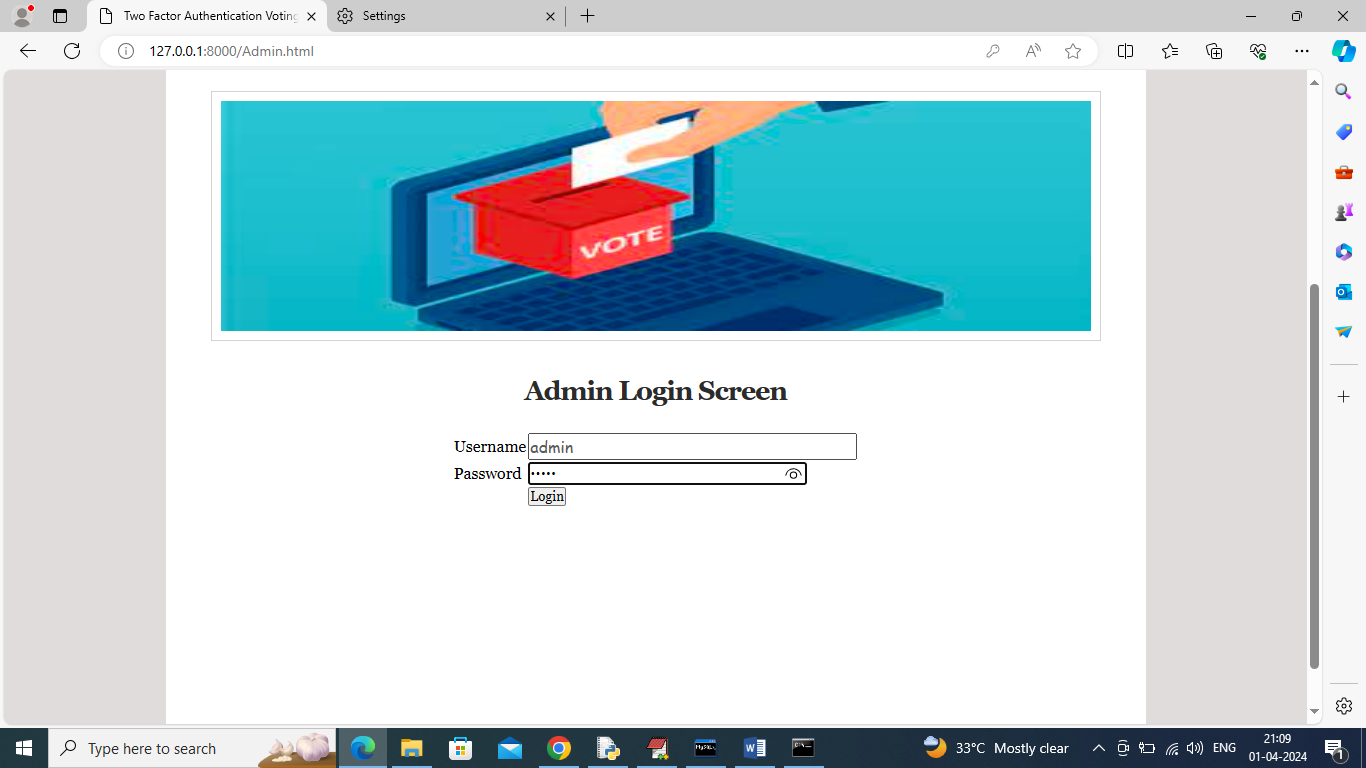
Now double click on run.bat file to start python server and get below page



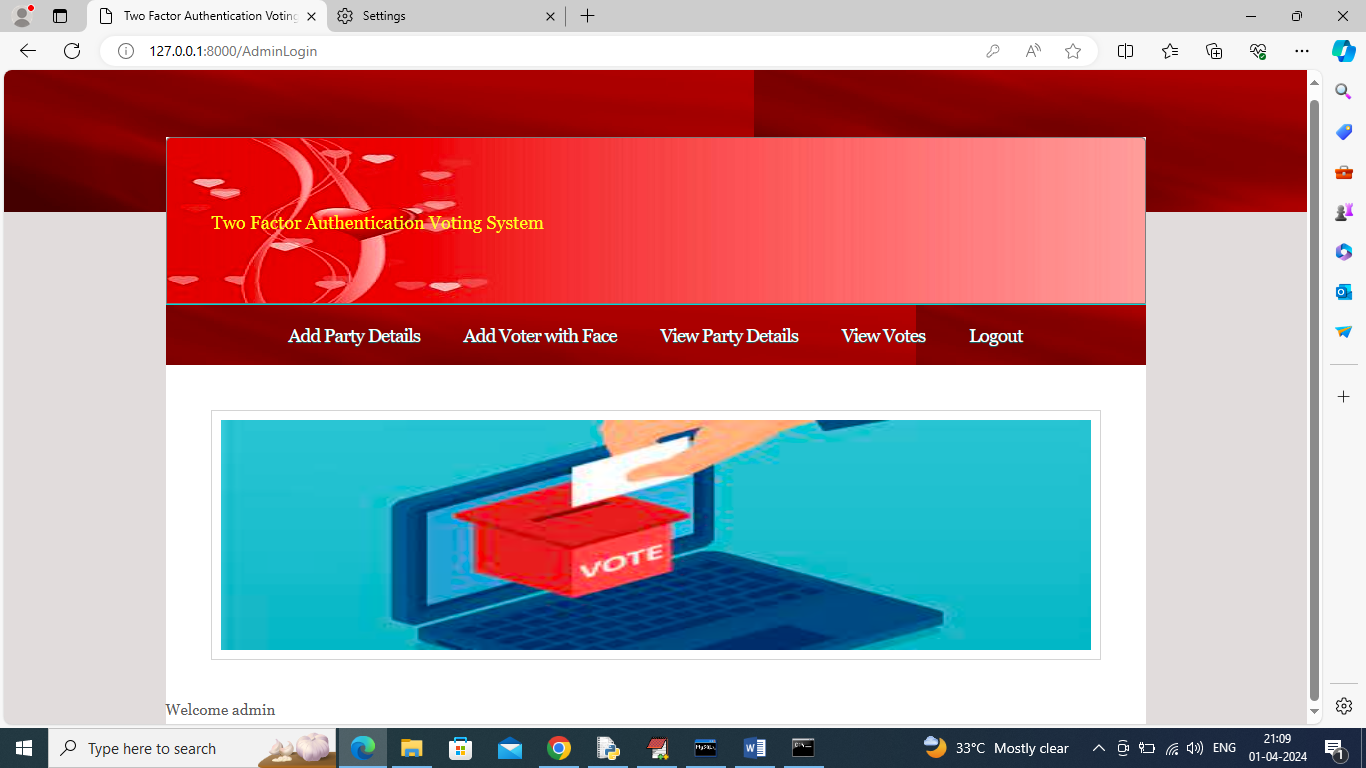
In above screen python server started and now open browser and enter URL as <http://127.0.0.1:8000/index.html> and press enter key to get below page



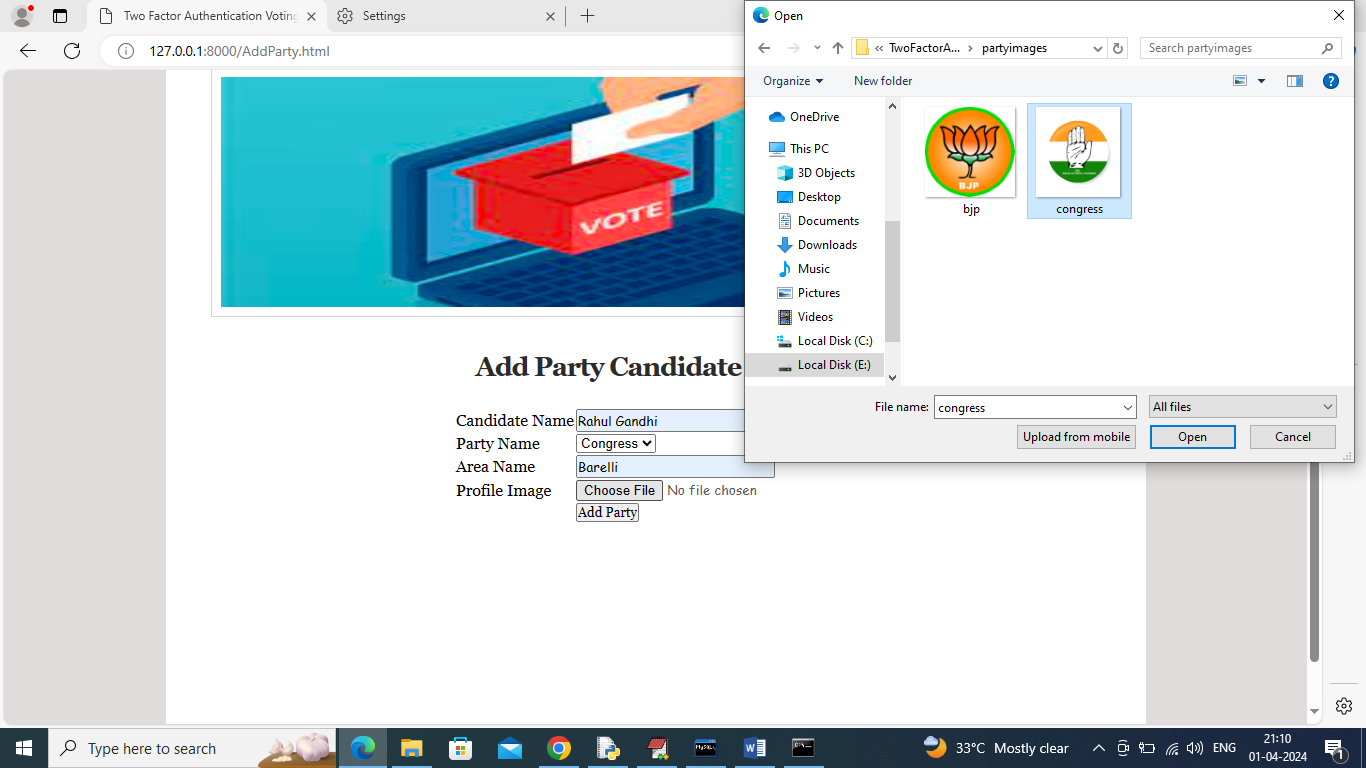
In above screen click on ‘Admin Login’ link to get below page



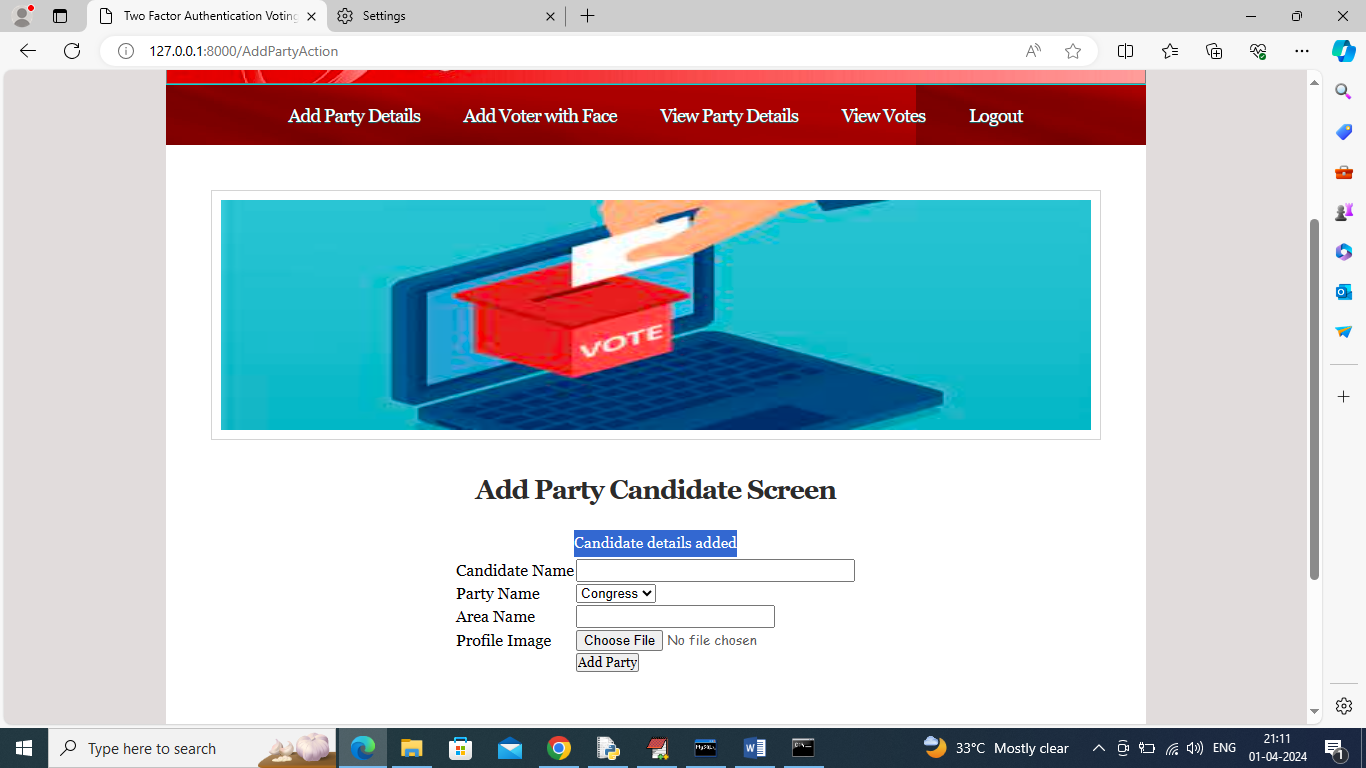
In above screen admin is login and after login will get below page



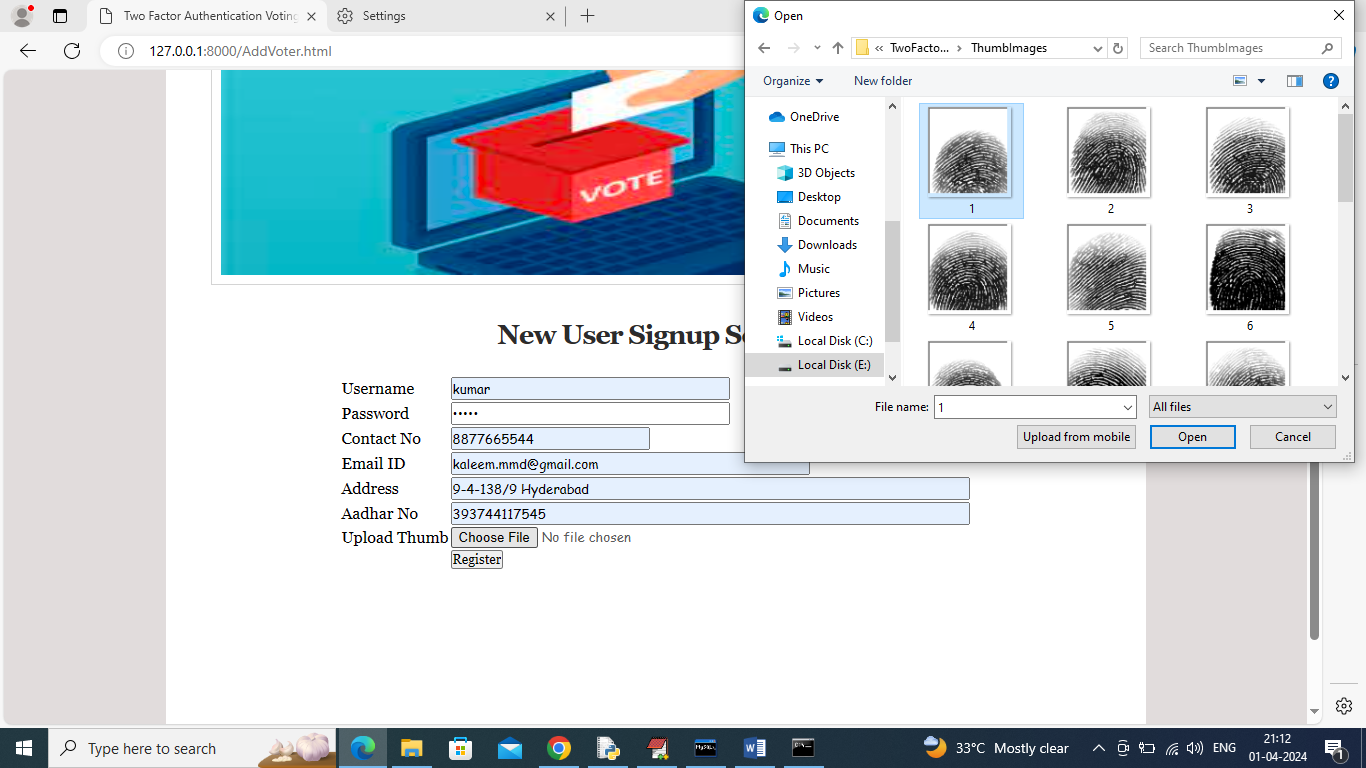
In above screen admin will click on ‘Add Party Details’ link to get below page



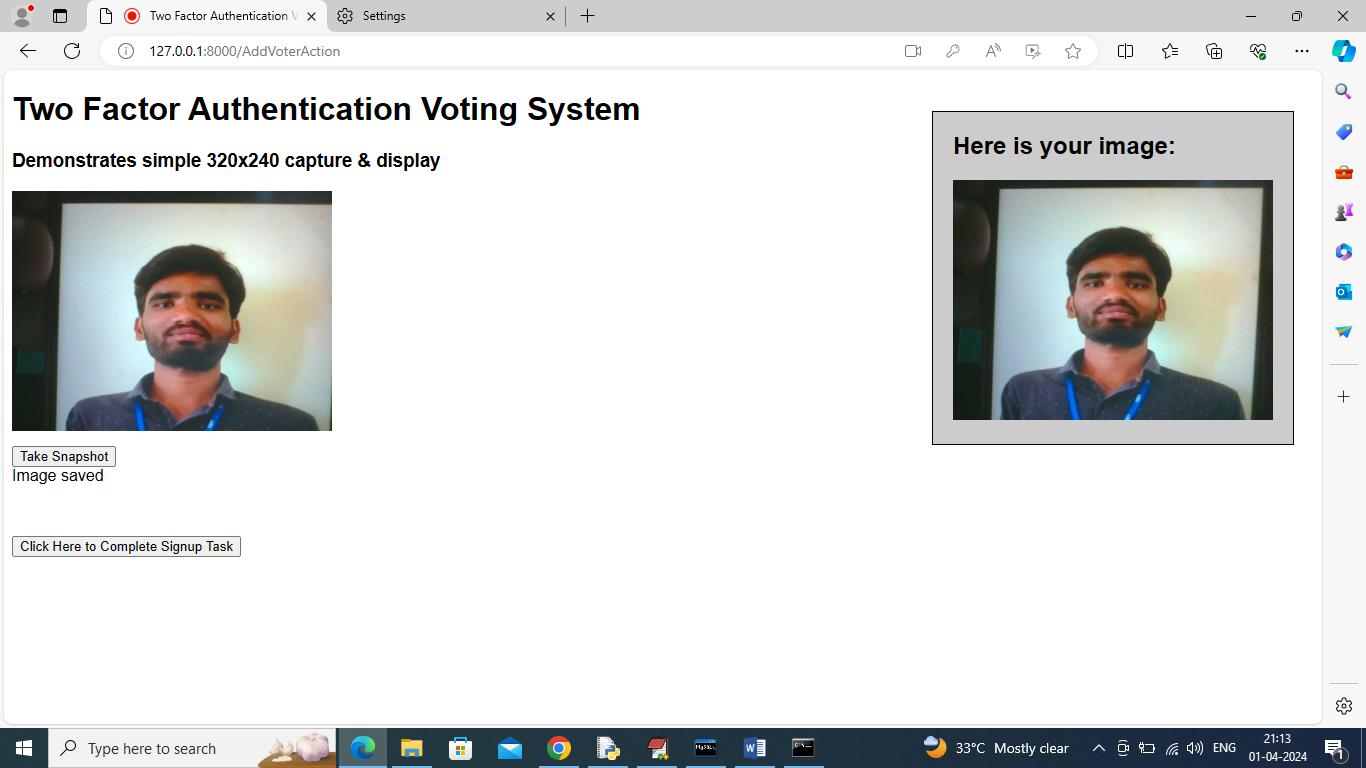
In above screen adding party candidate details along with party symbols and similarly you can add for any number of parties and now click on ‘Add Party’ button to get below page



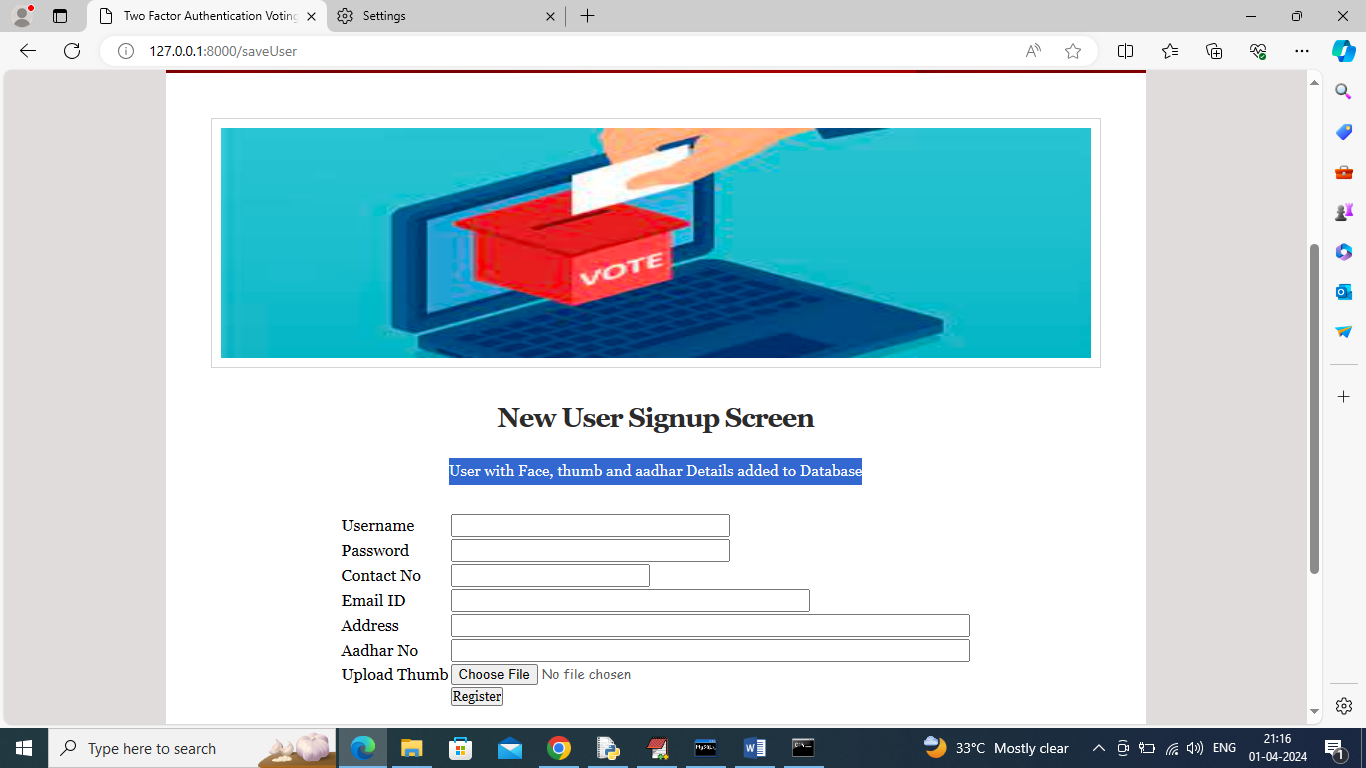
In above screen candidate details added and now click on ‘Add Voter with Face’ link to get below page



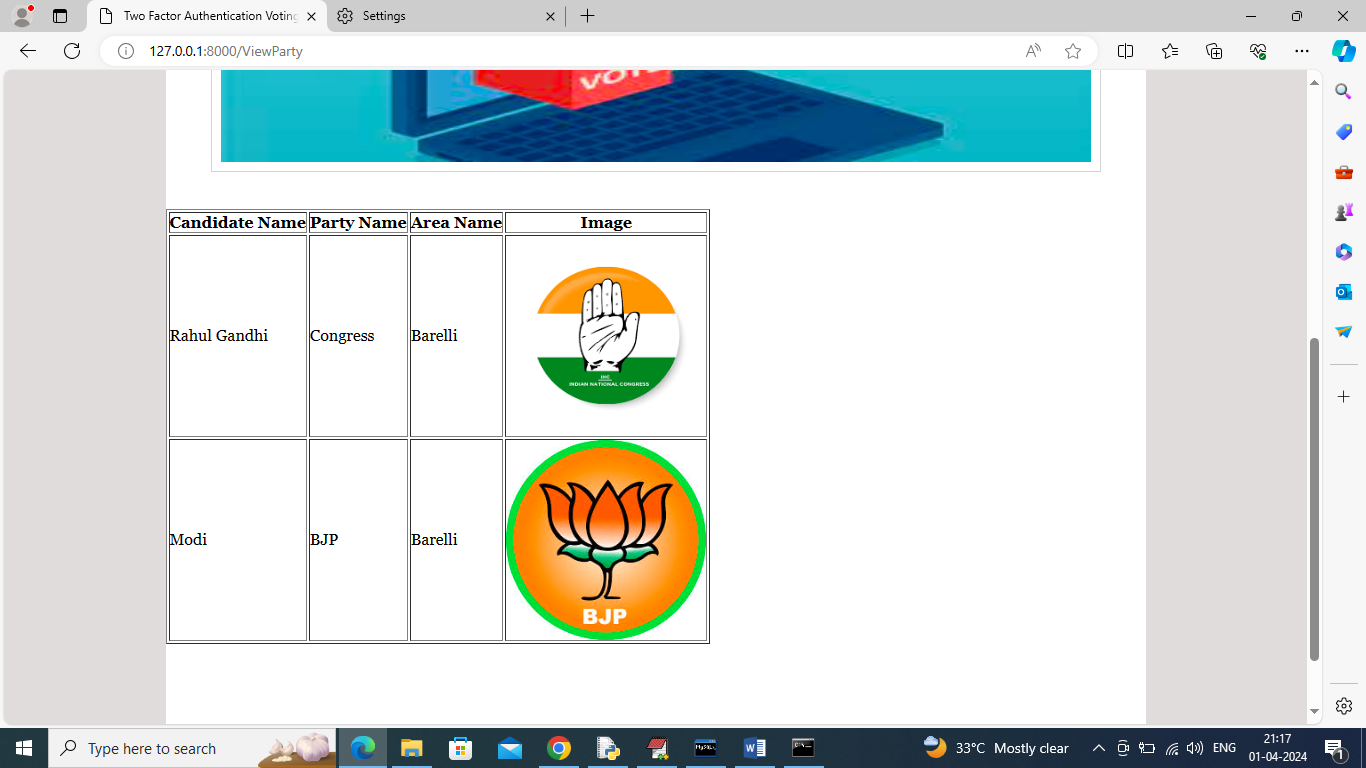
In above screen adding voter details along with username, AADHAR number, thumb image and give valid email to receive OTP and now click on ‘Register’ button to get below page



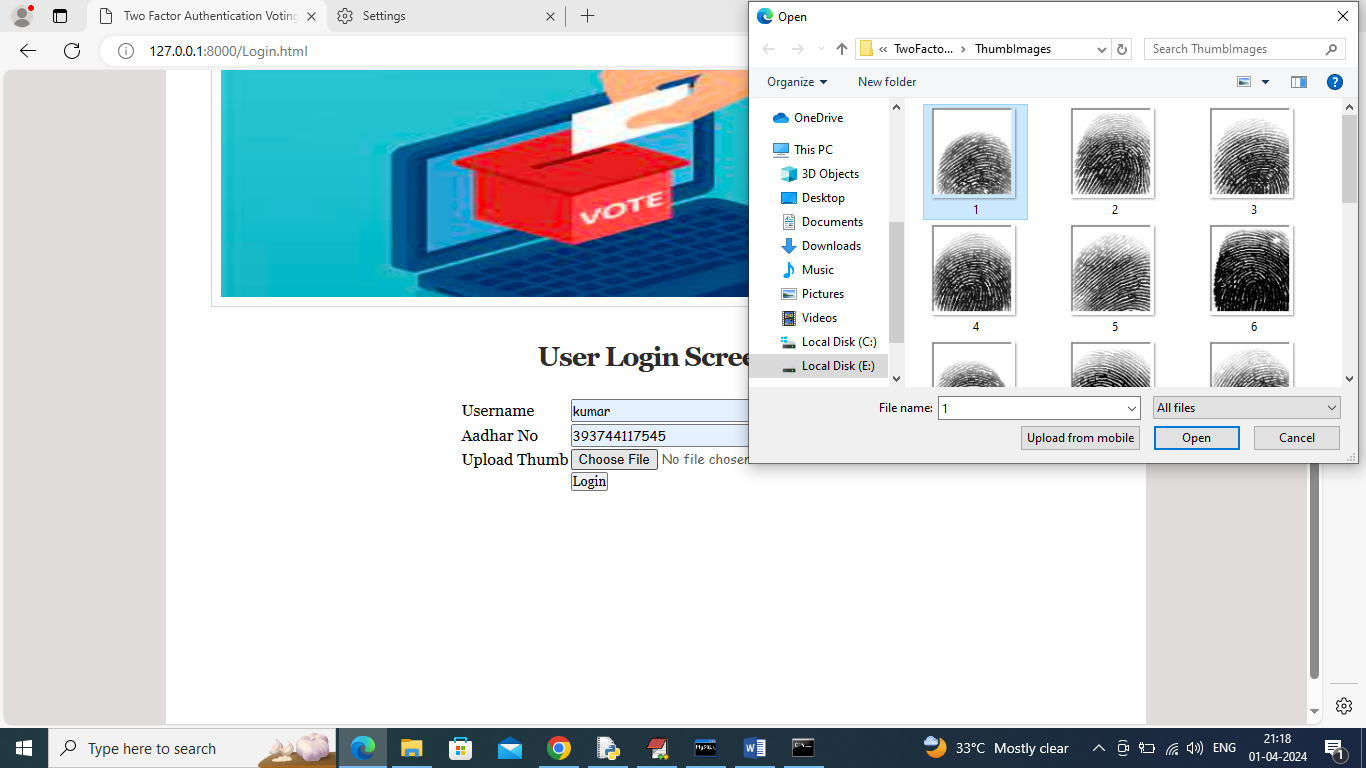
In above screen in left side webcam you can view face and then click on ‘Take Snapshot’ button to capture and preview face in right side panel and then click on ‘Click Here to Complete Sign up Task’ button to complete voter adding details and get below page



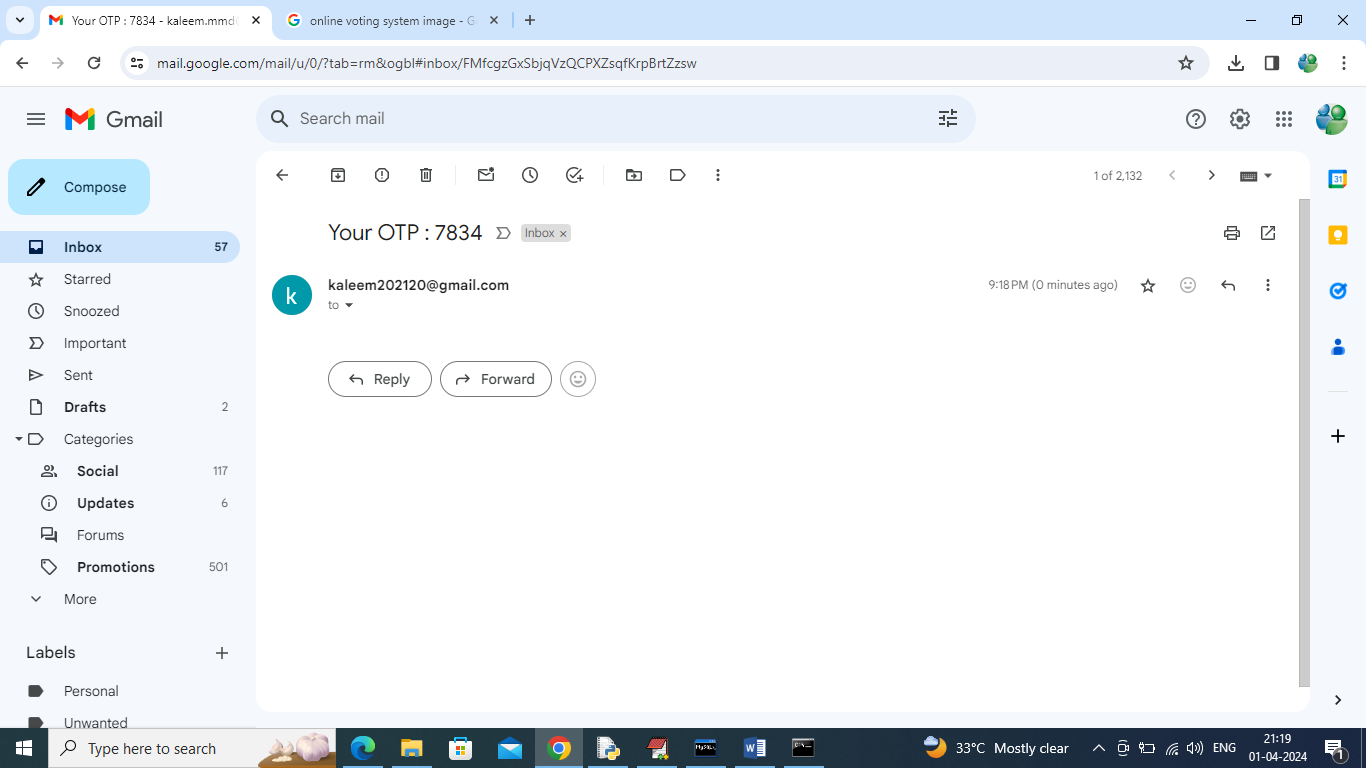
In above screen in blue colour text can see user with face, thumb and AADHAR details added to database. Similarly you can add any number of voters and now click on ‘View Party Details’ to view all existing party details like below page



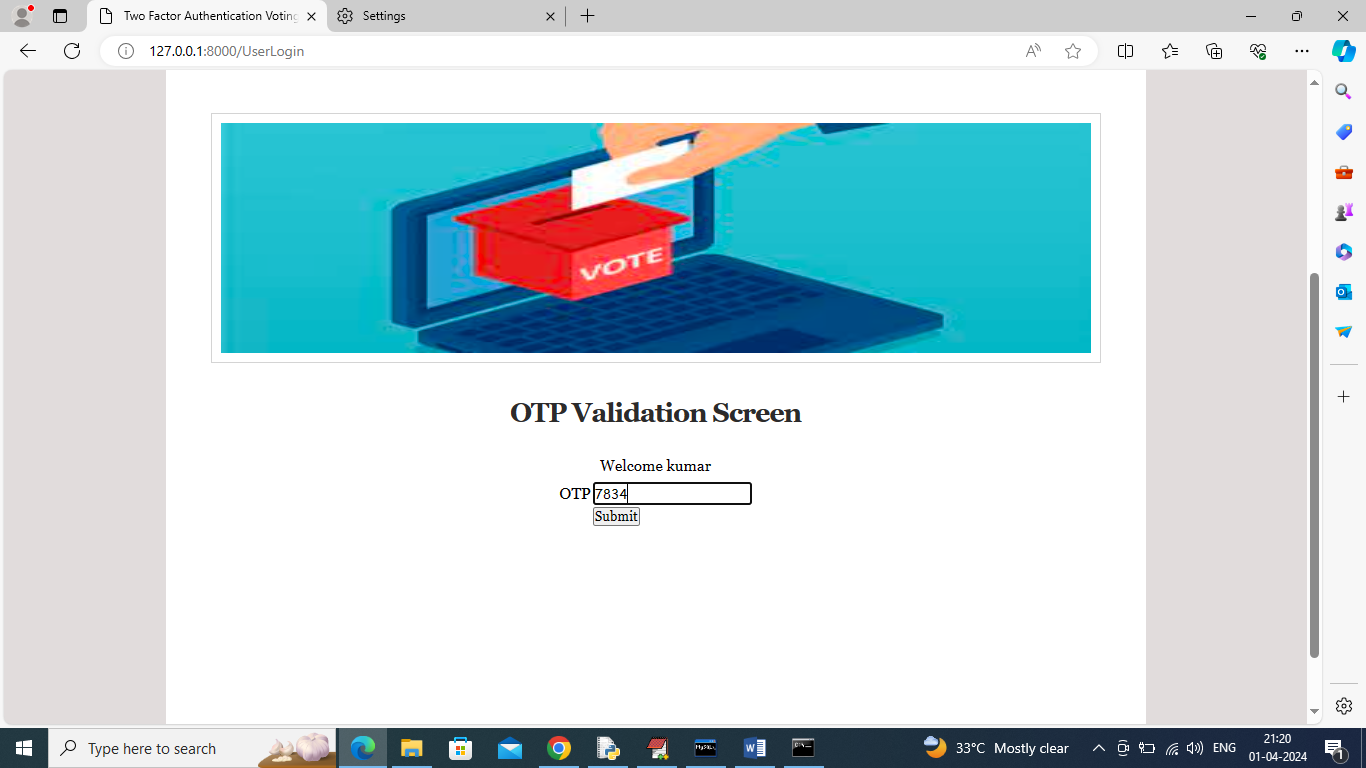
In above screen admin can view all added party candidate details and now logout and login as user to cast vote



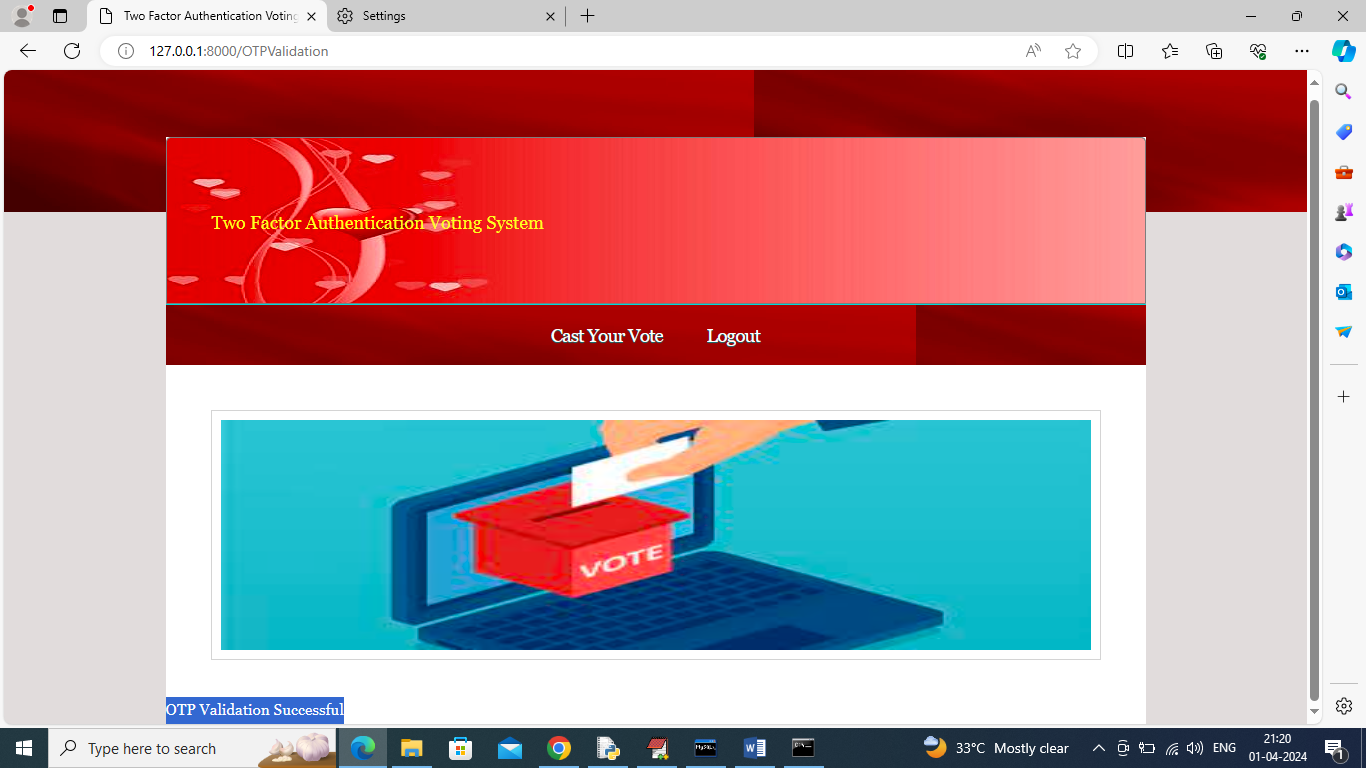
In above screen user is login with name, AADHAR and thumb image and then click on ‘Login’ button to get OTP in email



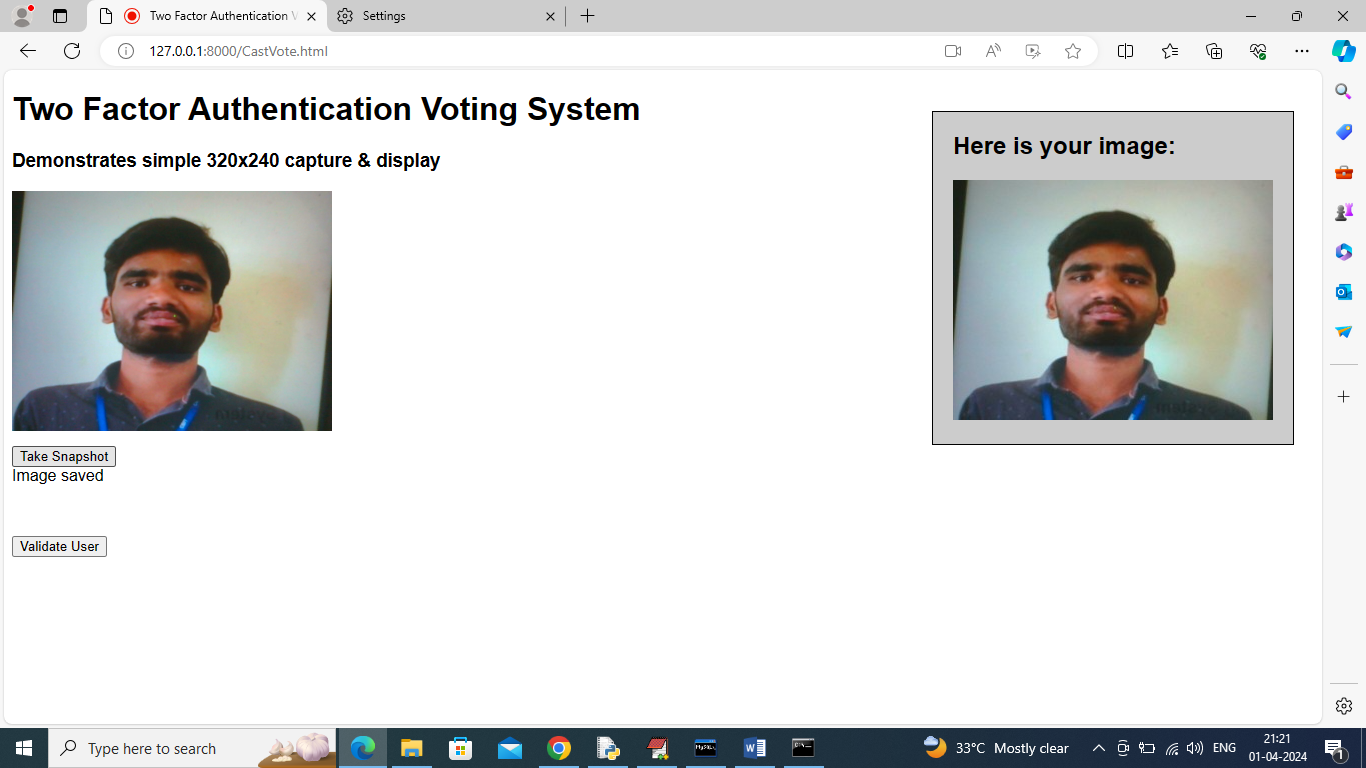
In above screen OTP received in email and then enter this OTP in application like below page



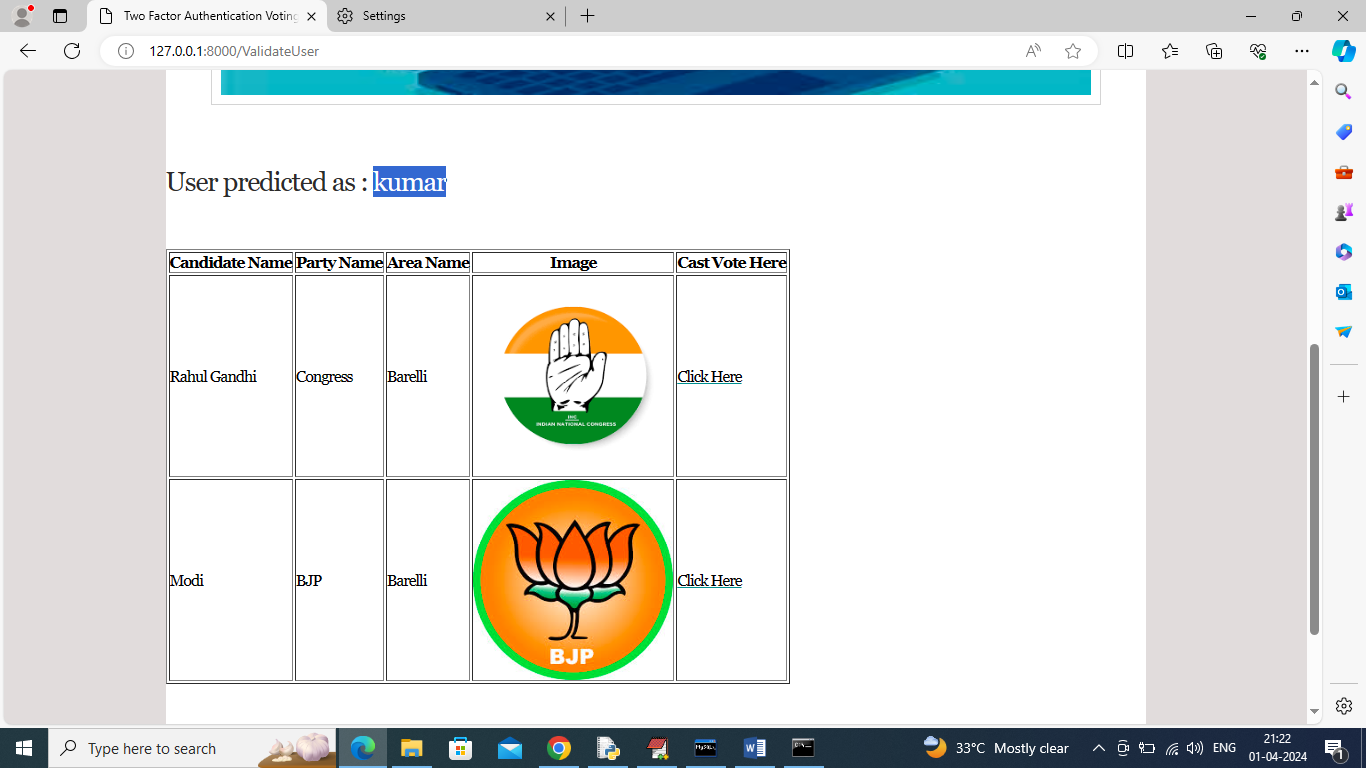
In above screen after entering OTP press button to get below page



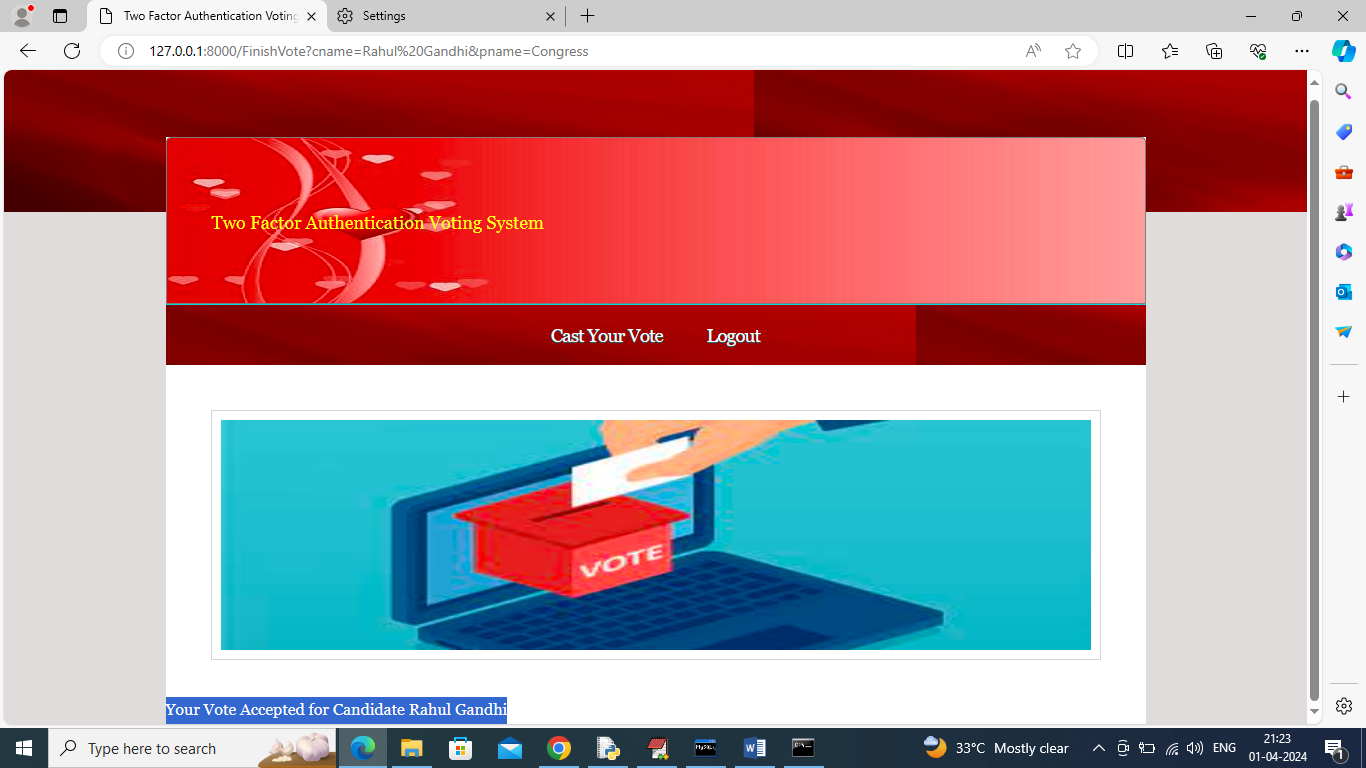
In above screen in blue text can see OTP validation successful and now click on ‘Cast Your Vote’ link to forward for face recognition and get below webcam



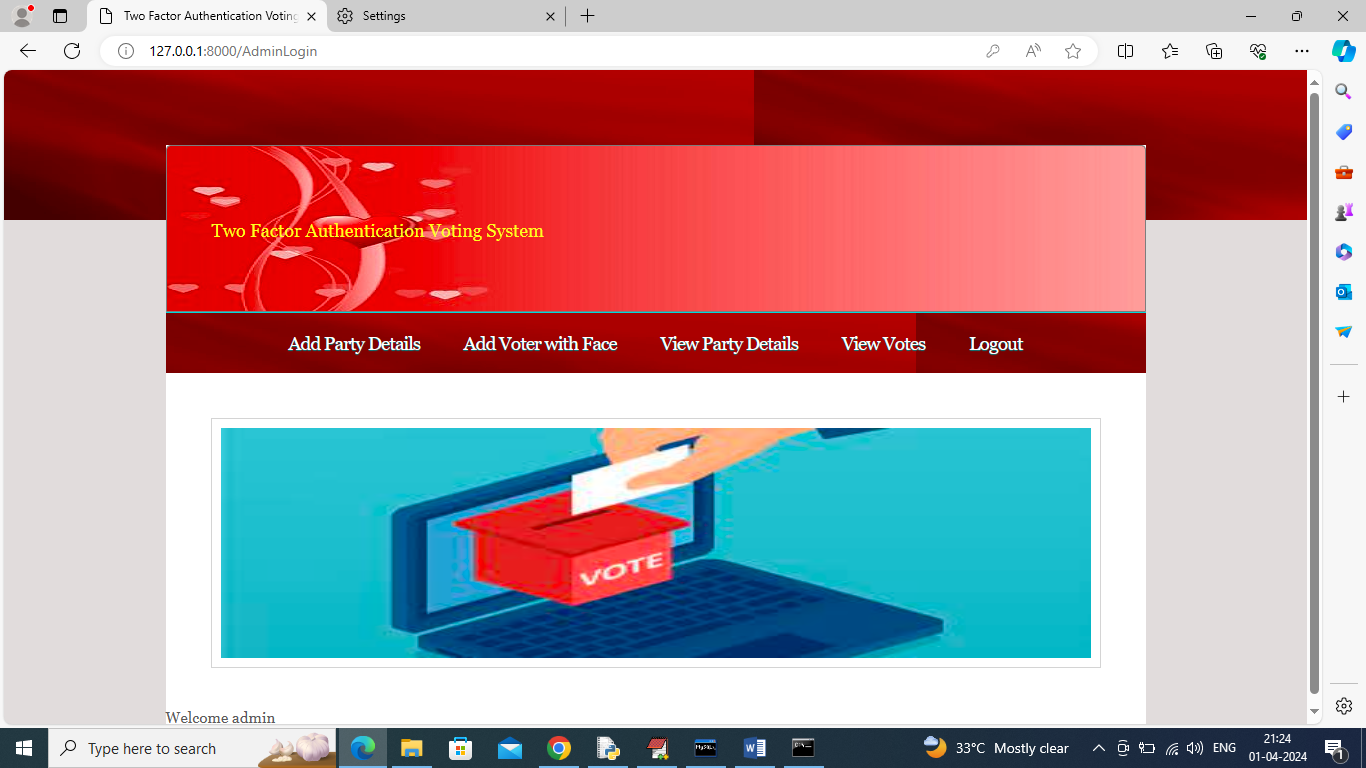
In above screen view face in webcam and then click on ‘Take Snapshot’ and then click on ‘Validate User’ button to recognized face and once recognition successful then will get below page



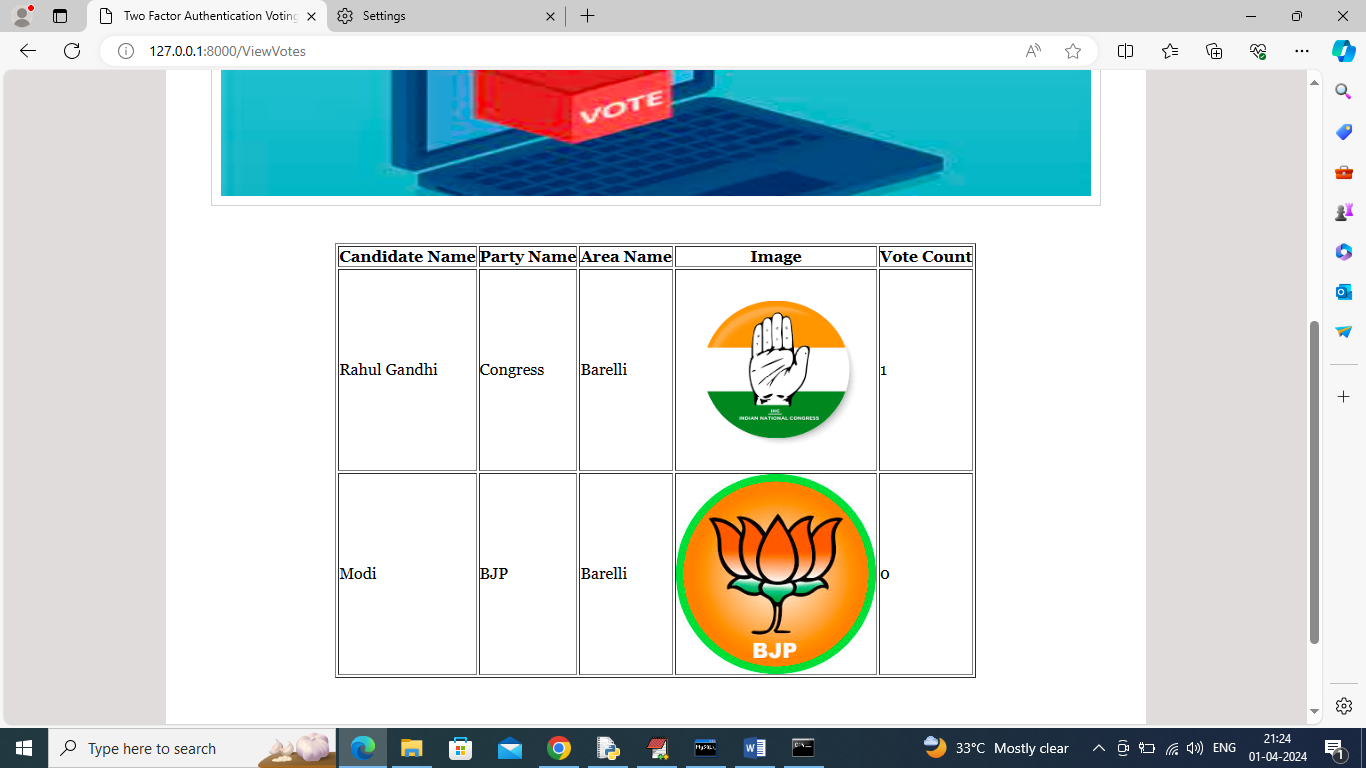
In above screen blue text can see user recognized as ‘kumar’ and now user can click on ‘Click Here’ link beside each candidate to cast his vote and then will get below page



In above screen in blue text can see ‘Vote accepted for selected candidate’ and now logout and login as admin to view vote count



In above screen admin can click on ‘View Votes’ link to get below page



In above screen in last column admin can see which candidate got how many votes.

Similarly by following above screens you can cast vote to parties.

# **CONCLUSION**

The proposed electronic voting system addresses the limitations of traditional voting methods by integrating advanced security mechanisms, enhancing accessibility, and ensuring transparency. By leveraging technologies such as blockchain, biometric authentication, and encryption, the system significantly reduces the risks of voter fraud, tampering, and result manipulation. Additionally, real-time vote counting and remote accessibility improve efficiency and voter participation. The implementation of this system can revolutionize the electoral process, making it more reliable, secure, and trustworthy. As electronic voting continues to evolve, further research and development can enhance its adoption, ensuring a seamless and democratic voting experience for all.

# **FUTURE SCOPE OF THE PROJECT**

The future of electronic voting systems holds significant potential for advancements in security, accessibility, and efficiency. With the integration of emerging technologies such as blockchain, artificial intelligence, and homomorphic encryption, electronic voting can become even more secure and tamper-proof. Blockchain ensures transparency and immutability, reducing concerns regarding vote manipulation, while AI can enhance fraud detection by identifying anomalies in voting patterns. Additionally, biometric authentication methods such as facial recognition and fingerprint scanning can further improve voter verification, minimizing identity fraud.

Another area of improvement is enhancing accessibility for remote voters, including overseas citizens and individuals with disabilities. Future electronic voting systems could leverage mobile applications and cloud-based platforms to allow secure voting from any location while maintaining the integrity of the process. Furthermore, the implementation of decentralized and verifiable audit mechanisms will ensure public trust in election outcomes. By continuously refining security protocols and expanding accessibility, electronic voting can become the standard for democratic elections worldwide.

# **REFERENCES**

1. Rivest, R. L. (2001). Electronic voting. *Proceedings of Financial Cryptography*, Springer.
2. Mercuri, R. (2002). A better ballot box? *IEEE Spectrum, 39*(10), 46-50.
3. Rubin, A. D. (2002). Security considerations for remote electronic voting. *Communications of the ACM, 45*(12), 39-44.
4. Kohno, T., Stubblefield, A., Rubin, A. D., & Wallach, D. S. (2004). Analysis of an electronic voting system. *IEEE Symposium on Security and Privacy*, 27-40.
5. Neumann, P. G. (2004). The risks of electronic voting. *Communications of the ACM, 50*(11), 136.
6. Cranor, L. F. (2008). A framework for reasoning about the human in the loop. *USENIX Security Symposium*.
7. Estehghari, S., & Desmedt, Y. (2010). Exploiting the client vulnerabilities in internet e-voting systems. *IEEE International Conference on e-Voting Security*.
8. Ryan, P. Y. (2011). Veriﬁable election schemes. *Security & Privacy, IEEE, 9*(1), 48-54.
9. Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer Networks, 54*(15), 2787-2805.
10. López, J., & Oppliger, R. (2013). Authentication and electronic voting. *IEEE Security & Privacy, 11*(4), 74-76.
11. Neff, C. A. (2004). Practical high-certainty intent verification for encrypted votes. *USENIX Security Symposium*.
12. Adida, B. (2008). Helios: Web-based open-audit voting. *USENIX Security Symposium*.
13. Benaloh, J., Jones, D., Lazarus, E., Lindeman, M., & Stark, P. (2011). *Star-Vote: A Secure, Transparent, Auditable, and Reliable Voting System*.
14. Gritzalis, D. (2002). Secure electronic voting. *Advances in Information Security, 7*, Springer.
15. Chaum, D., Rivest, R. L., & Sherman, A. T. (2020). *Secure electronic voting: The technology of democracy*. Springer.
16. Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2017). Digital identity guidelines. *NIST Special Publication 800-63*.
17. Keller, S., & Goodman, J. (2018). The security risks of internet voting. *Journal of Cybersecurity, 4*(1), tyy002.
18. Volk, C., & Preneel, B. (2019). Blockchain for electronic voting: A systematic literature review. *IEEE Transactions on Blockchain, 2*(3), 324-336.
19. Gligor, V. D. (2003). A guide to understanding authentication. *National Computer Security Center*.
20. U.S. Election Assistance Commission. (2021). Voluntary Voting System Guidelines (VVSG 2.0). *U.S. Election Assistance Commission Report*.