

SMART ELECTRONIC VOTING MACHINE

*A Project Report Submitted
In Partial Fulfillment
for award of Bachelor of Technology*

In

Computer Science and Engineering (Internet of Things)

by

**Nishu Sharma (2201331550083)
Om Singh (2201331550084)
Avinash Sinha (2201331550035)
Uday Gaur (2201331550131)**

**Under the Supervision of
Mr. Mushtaq Ahmad Rather
Assistant Professor ,CSE-IOT**



**Computer Science and Engineering (Internet of Things)
School of Computer Science and Engineering in Emerging
Technology
NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY,
GREATER NOIDA
(An Autonomous Institute)
Affiliated to
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW
May, 2025**

School of Computer Science in Emerging Technologies

Department of Computer Science and Engineering (Internet of Things)

Academic Year: 2024-2025

CERTIFICATE

This is to certify that the project report entitled “**Smart Electronic Voting Machine**” has been successfully completed by the following student of Third Year, Department of **Computer Science and Engineering (Internet of Things)**.

Nishu Sharma (2201331550083)

Om Singh(2201331550084)

Avinash Sinha (2201331550035)

Uday Gaur (2201331550131)

This project has been completed in **partial fulfillment of the requirements** for the Third Year Engineering Course in Computer Science and Engineering (Internet of Things) and is hereby submitted to the **Department of Computer Science and Engineering (Internet of Things)**, Noida Institute of Engineering and Technology, Greater Noida, for the **Academic Year 2024–2025**.

Mr.
(Mushtaq Ahmad Rather)

Mr.
(Mayank Deep Khare)

DECLARATION

We hereby declare that the work presented in this report entitled “**Smart Electronic Voting Machine**”, was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of our work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name : Nishu Sharma

Roll Number : 2201331550083

Name : Om Singh

Roll Number : 2201331550084

Name : Avinash Sinha

Roll Number : 2201331550035

Name : Uday Gaur

Roll Number : 2201331550131

CERTIFICATE

Certified that Nishu Sharma (2201331550083), Om Singh (2201331550083), Avinash Sinha (2201331550035), Uday Gaur (2201331550131) have carried out the research work presented in this Project Report entitled **“Smart Electronic Voting Machine”** for the award of **Bachelor of Technology**, Department of Computer Science and Engineering (Internet of Things) from Dr. APJ Abdul Kalam Technical University, Lucknow under our supervision. The Project Report embodies results of original work, and studies are carried out by the students herself/himself. The contents of the Project Report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

(Mr. Mushtaq Ahmad Rather)

(Assistant Professor)

(CSE-IOT)

NIET , Greater Noida

Date:

ACKNOWLEDGMENT

We would like to express my gratitude towards Mr. Steven David (Assistant Professor, IOT) for the guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

Our thanks and appreciation to respected HOD Sir, Mayank Deep Khare for the motivation and support throughout.

Name : Nishu Sharma

Roll Number : 2201331550083

Name : Om Singh

Roll Number : 2201331550084

Name : Avinash Sinha

Roll Number : 2201331550035

Name : Uday Gaur

Roll Number : 2201331550131

ABSTRACT

This project presents the design and implementation of a Smart Electronic Voting Machine (EVM) leveraging the ESP32-CAM microcontroller and a Flask-based backend, aimed at enhancing the security, transparency, and reliability of digital elections. The system employs real-time facial recognition and biometric verification to prevent voter impersonation and ensure one-person-one-vote compliance.

The ESP32-CAM acts as the edge device, capturing the voter's image and sending it, along with vote details (voter ID and selected party), to a Flask server over Wi-Fi. The server verifies the voter's identity using facial recognition techniques powered by OpenCV and the face_recognition library. Upon successful verification, the vote is securely recorded in MongoDB, which also maintains real-time voting statistics.

An admin panel developed with React.js (or basic HTML/CSS/JS) enables administrators to register voters, monitor live vote counts, and oversee voting logs. This system ensures data integrity and transparency by encoding images in base64 for transmission and storing all voting activity with timestamps and voter information.

The use of Postman facilitated the testing and validation of API endpoints, ensuring robust backend functionality. This smart voting system demonstrates a scalable, cost-effective, and secure solution for institutional and organizational elections, paving the way for future enhancements in digital

TABLE OF CONTENTS

	Page No.
Declaration	i
Certificate	ii
Acknowledgements	iii
Abstract	iv
Dedication (optional)	
List of Tables	vii
List of Figures	viii
List of Abbreviations	ix
CHAPTER 1: INTRODUCTION	1-3
1.1 BACKGROUND	1
1.3 PROBLEM DEFINITION	1-2
1.4 OBJECTIVE OF PROJECT	2
1.4 FLOW CHART	3
CHAPTER 2: LITERATURE REVIEW	4-5
2.1 VISION BASED DROWSINESS DETECTION	4
2.2 PHYSIOLOGICAL SIGNAL-BASED DETECTION	4-5
2.3 SENSOR FUSION AND IOT INTEGRATION	5
2.4 MACHINE LERNING AND ADVANCED METHODS	5
CHAPTER 3: REQUIREMENTS AND ANALYSIS	
3.1 Requirements Specification	
3.2 Planning and Scheduling	
3.3 Software and Hardware Requirements	
3.4 Preliminary Product Description	
CHAPTER 4: PROPOSED METHODOLOGY	41-70
4.1 INTRODUCTION	

4.2 SYSTEM OVERVIEW	
4.3 WORKING PRINCIPAL	
4.4 PROCESS FLOW	
4.5 ADVANTAGES OF SYSTEM	

.....

CHAPTER 5: RESULTS	71- 80
CHAPTER 6: CONCLUSION AND FUTURE WORK	81-82
REFERENCES	120
APPENDICES	122
PUBLICATIONS	123
PLAGIARISM REPORT	124
CURRICULUM VITAE	125

LIST OF TABLES

Table No.	Table Caption	Page No
4.1	Output for Tokenization	72
5.1	Dataset	48
5.2	Comparison of accuracy of DL-based Method with NLP techniques and Used Methods	58

LIST OF FIGURES

Fig No	Caption	Page No
1.1	NLP	13
4.1	Block Diagram	58
4.2	Text Normalization	60
4.3	Example of Lemmatization	61
4.4	Example for Word Embedding	63
4.5	Recurrent Neural Network	64
5.1	Label Values	69
5.2	Applying stopwords	69
5.3	EDA	70
5.4	Applying Multilabel Binarizer	70
5.5	Final corpus before training	71
5.6	Generating word cloud	71
5.7	Count vectorizer	72

LIST OF ABBREVIATIONS

Abbreviation	Full Form
AI	Artificial Intelligence
CNN	Latent Dirichlet allocation
CV	Computer Vision
GRU	Gated Recurrent Unit
NLP	Natural language processing
TF-IDF	Term Frequency-Inverse Document Frequency
GloVe	Global Vectors
CURB	Scalable Online Algorithm
EANN	Event Adversarial Neural Network
BiLSTM	Bidirectional LSTM
CNN	Convolutional neural network
MLP	Multilayer perceptron
API	Application programming interface
NB	Naive Bayes
CNN	Convolution neural network
NER	Named Entity Recognition
KNN	K-Nearest Neighbours

CHAPTER 1

Introduction

Elections are the foundation of democratic societies, serving as the primary mechanism through which citizens express their will and influence decision-making. The integrity, accessibility, and efficiency of the voting process are therefore of paramount importance. Traditional methods of voting—such as paper ballots or early-generation electronic voting machines (EVMs)—have long been criticized for their limitations, including susceptibility to fraud, human error, logistical complexity, and delayed result processing. These issues not only compromise electoral outcomes but also diminish public trust in democratic institutions. In the context of growing global digitalization and increased emphasis on secure digital infrastructure, there is an urgent need to rethink and modernize the way voting is conducted. The integration of IoT (Internet of Things) devices, machine learning, biometrics, and cloud computing offers promising solutions to many of the challenges associated with conventional voting systems. These technologies can help achieve secure voter authentication, real-time result computation, and robust audit trails, all while improving ease of access and user experience.

This project proposes a Smart Electronic Voting Machine (EVM) that leverages the ESP32-CAM microcontroller and a Flask-based backend server to create a secure, real-time, and intelligent voting system. The system is designed with a focus on small-scale elections, such as those in educational institutions, local organizations, or private corporations, but is also scalable to larger applications. The primary objective is to ensure that only authorized voters can cast a vote and that every vote is accurately recorded and counted, with complete transparency throughout the process.

The ESP32-CAM, a low-cost development board with integrated Wi-Fi, Bluetooth, and a camera module, serves as the edge device in this system. It performs several key tasks: connecting to a wireless network, capturing the voter's image at the time of voting, collecting voter identification details and vote selections, and securely transmitting this data to the server. The inclusion of real-time image capture enables the system to use facial recognition for verifying voter identity—a significant improvement over systems that rely solely on PINs, RFID cards, or fingerprints.

The backend infrastructure, built using Python's Flask framework, receives incoming data from the ESP32-CAM devices and carries out several important operations. These include decoding the base64-encoded image, comparing it with the pre-registered image using facial recognition algorithms (via OpenCV and `face_recognition`), validating the voter's identity, and, if successful, recording the vote in a MongoDB database. If the face does not match the registered data, the system automatically rejects the vote and logs the incident. This biometric validation adds a critical layer of security that makes impersonation or unauthorized access nearly impossible.

To facilitate system oversight and management, an Admin Panel is provided via a web interface built with React.js (or optionally using standard HTML/CSS/JavaScript). The admin panel allows authorized personnel to register new voters by uploading their names and facial images, view live voting results, monitor participation trends, and access system logs for audit and security purposes. This interface ensures that administrators have full control over the electoral process without

compromising the privacy or integrity of voter data.

Throughout the development phase, Postman was used as a powerful API testing tool to simulate and validate requests to various endpoints of the Flask backend. This enabled rapid debugging, ensured consistency in data transmission, and verified that all system components work harmoniously.

In summary, the Smart EVM developed in this project addresses the key challenges of modern voting systems by providing secure voter authentication, transparent vote handling, and real-time result management. By combining IoT hardware, biometric verification, and cloud-based storage, this project contributes a viable and scalable model for future electoral systems. With further refinement, the architecture can be adapted to support large-scale public elections and integrated with national databases for even more robust security and functionality. This project serves as a significant step toward realizing a future of safe, smart, and seamless digital voting.

1.1 Problem Definition:

Traditional voting systems, including paper ballots and basic electronic voting machines (EVMs), face multiple challenges such as voter impersonation, lack of real-time monitoring, delayed result processing, and limited scalability. These issues can compromise the security, transparency, and efficiency of elections. Moreover, most existing systems lack robust identity verification, making them vulnerable to fraudulent voting practices.

The problem is to design and develop a smart, secure, and scalable electronic voting system that ensures:

- Accurate voter authentication using facial recognition,
- Real-time vote recording and result updates via a reliable backend,
- Tamper-proof data handling and storage, and
- User-friendly interfaces for both voters and administrators.

This system must be cost-effective, easy to deploy in small to medium-scale elections (such as those in colleges or organizations), and capable of maintaining voter privacy and election integrity.

1.2 Objectives Of Project:

- **To design a secure and intelligent electronic voting system**

Develop a voting system that combines embedded hardware (ESP32-CAM) and modern software frameworks (Flask, OpenCV) to ensure reliability and security in the voting process.

- **To implement facial recognition for voter authentication**

Integrate biometric verification using real-time image capture and facial recognition to prevent unauthorized or duplicate voting and enhance voter identity validation.

- **To enable real-time data transmission and result tracking**

Utilize the ESP32-CAM's Wi-Fi capabilities and a Flask-based server to transmit voting data instantly and maintain a live count of results stored in a centralized database (MongoDB).

- **To ensure tamper-proof and traceable vote recording**

Store each vote with timestamp, voter ID, and image log (if necessary) to create a secure audit trail, ensuring transparency and trustworthiness in the election outcome.

- **To provide a user-friendly admin interface**

Develop a web-based admin panel for registering voters, viewing live results, and monitoring voting activity to support efficient election management.

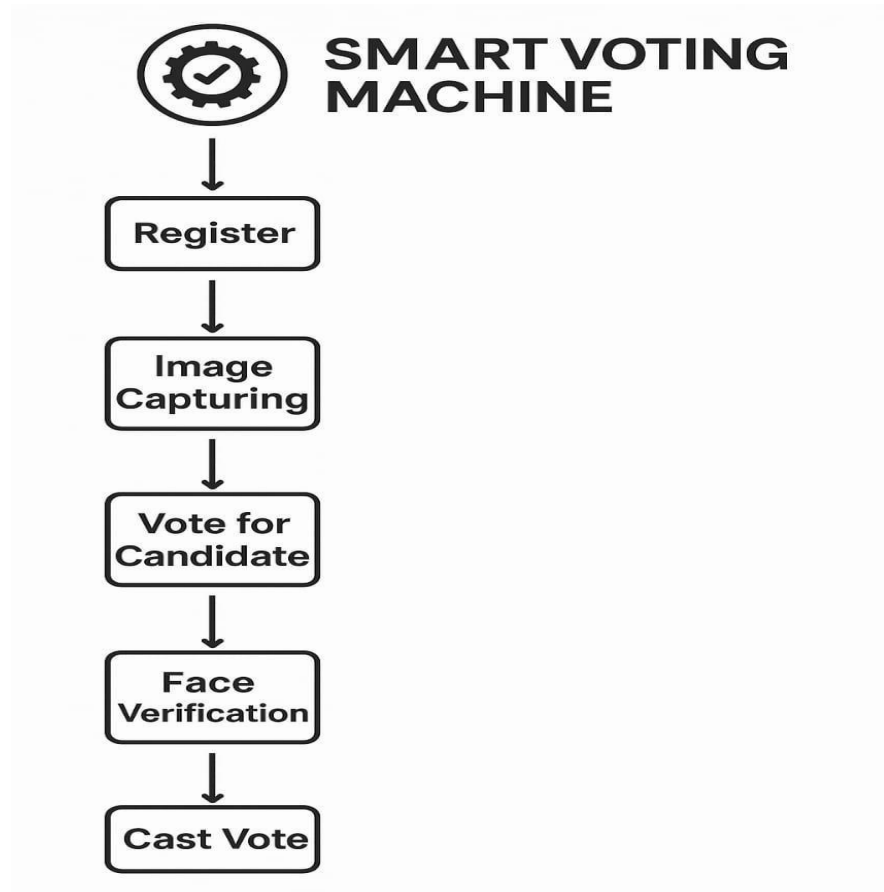
- **To create a scalable and cost-effective solution**

Build a prototype that can be deployed in institutional and organizational elections and scaled for larger implementations, while keeping the system affordable and easy to set up.

- **To test and validate system functionality through API tools**

Use tools like Postman to thoroughly test and debug the backend API endpoints, ensuring smooth communication between the ESP32-CAM device and the server.

FLOW CHART OF MODEL



CHAPTER 2

Literature Survey

1. Introduction to Electronic Voting Systems

Electronic Voting Machines (EVMs) have been widely adopted in modern democracies to replace traditional paper ballots. They offer benefits such as faster result compilation, reduction in human errors, and lower environmental impact. However, these systems often lack advanced security mechanisms, biometric verification, or real-time monitoring capabilities. Researchers and developers have proposed various upgrades to enhance EVMs, including the incorporation of networked devices, biometric validation, and blockchain-based vote storage.

In the paper “A Review of Electronic Voting Technologies” by Agarwal and Sharma (2019), the authors analyze several generations of EVMs used in India and globally. They highlight concerns related to physical tampering and lack of identity verification, which undermine public trust. The review concludes that integrating biometric systems and online verification could significantly improve EVM reliability and transparency.

2. Biometric Voter Authentication Using Facial Recognition

Biometric authentication is increasingly used in secure systems, offering a non-intrusive and efficient way to verify identity. Among various biometrics (fingerprints, iris scans, voice recognition), facial recognition stands out due to its ease of deployment and user acceptance.

In “Facial Recognition Technology: Security vs. Privacy” by Jain et al. (2020), the authors present facial recognition as a viable tool for identity verification in secure systems. Using computer vision algorithms such as Local Binary Patterns Histogram (LBPH), Haar Cascades, and convolutional neural networks (CNN), accurate and real-time facial recognition has become feasible even on low-power devices like the ESP32-CAM.

Moreover, the `face_recognition` Python library—built on `dlib`’s deep learning facial recognition—has demonstrated high accuracy with minimal training data. It has been successfully used in projects requiring identity validation in real time, which aligns directly with the objectives of this project.

3. Role of ESP32-CAM in IoT-based Voting Systems

The ESP32-CAM is a microcontroller with built-in camera and Wi-Fi/Bluetooth capabilities, making it ideal for edge-based IoT solutions. It is cost-effective, compact, and capable of capturing and transmitting images to a remote server, which makes it particularly useful for field-deployable smart voting booths.

In “IoT-Based Smart Surveillance using ESP32-CAM” (Patel & Mehta, 2021), the ESP32-CAM is employed for real-time image capture and data transmission to a cloud backend for monitoring purposes. The researchers highlight its low power consumption, rapid boot-up time, and stable Wi-Fi connectivity. These attributes make ESP32-CAM a strong candidate for smart voting terminals, especially in remote or resource-constrained environments.

4. Flask-Based Backend Systems for Secure Applications

Flask is a micro web framework written in Python, well-suited for lightweight applications with custom APIs. It supports RESTful API development, session management, and integration with machine learning tools, making it a great choice for the backend of a voting system.

The paper “Building Secure Flask APIs for IoT Applications” by Deshmukh et al. (2021) explores the use of Flask in handling user requests from IoT nodes (like ESP32). The authors emphasize the importance of endpoint security, input validation, and response handling—factors that are central to your project’s backend, especially in managing /vote, /register, and /results endpoints securely and efficiently.

5. MongoDB for Vote Storage and Real-Time Results

NoSQL databases like MongoDB are commonly used in IoT projects for their flexible schema and real-time performance. In your project, MongoDB stores voter registration details, facial encodings, votes, and live results.

In “MongoDB in IoT-Based Data Management Systems” by Banerjee and Rao (2022), MongoDB is shown to outperform traditional relational databases in handling high-frequency, semi-structured data—exactly the type used in voting logs and user verification metadata. Its ability to scale horizontally also makes it a good fit for large-scale voting systems.

6. Related Work and Comparative Systems

Several experimental smart voting systems have been proposed. For instance, “Smart Voting System Using Face Recognition” by Singh et al. (2020) proposes a PC-based voting model using webcam and Python’s OpenCV for verification. However, their system lacks scalability and portability compared to microcontroller-based designs like yours.

Another study, “Blockchain-Based Voting System for Secure Elections” by Li and Wang (2018), introduces a decentralized approach but faces challenges in terms of implementation cost and real-time speed—two areas where your ESP32-CAM + Flask + MongoDB setup excels due to simplicity and efficiency.

Conclusion

From this survey, it is evident that while several attempts have been made to create secure and smart voting systems, most either lack portability (PC/webcam-based), affordability (biometric + fingerprint sensors), or simplicity (blockchain). Your project combines the strengths of IoT hardware (ESP32-CAM), real-time facial recognition, and cloud-based backend systems (Flask + MongoDB) to deliver a balanced, scalable, and practical solution. This literature foundation supports the relevance and effectiveness of your proposed system.

CHAPTER 3

Proposed System & Project Description

The proposed system is a Smart Electronic Voting Machine that aims to modernize and secure the voting process using a combination of Internet of Things (IoT), facial recognition technology, and a cloud-based backend. The core idea behind this system is to authenticate voters using facial biometrics before allowing them to cast their votes, thereby minimizing the risk of impersonation and electoral fraud. The system replaces traditional manual or button-based voting methods with a more secure and intelligent approach.

At the front end, the ESP32-CAM module acts as a low-cost, portable IoT device deployed at the voting point. It connects to a Wi-Fi network and initializes its onboard camera to capture the image of a voter when they are ready to cast a vote. Along with the captured image, the device sends the voter's ID and selected party to a Flask-powered backend server. The backend processes this data, uses machine learning-based facial recognition to verify the identity of the voter, and checks it against pre-stored facial encodings in a MongoDB database. If the verification is successful, the vote is recorded and added to the tally. In case the verification fails, the vote is rejected to ensure the integrity of the election process.

The backend system, developed using Python's Flask framework, handles multiple endpoints for submitting votes, registering voters, retrieving results, and managing administrative functions. MongoDB serves as the primary database, storing voter registration data, facial encoding models, vote logs, and real-time results. Additionally, an admin panel built using either plain HTML/CSS/JavaScript or React.js provides election officers with an interface to register new voters, upload facial images, monitor voting activity, and publish results.

This entire system has been designed with simplicity, security, and scalability in mind. By leveraging ESP32-CAM for hardware-level image capture and Flask for backend logic, the project offers a flexible yet powerful solution that can be deployed in colleges, small institutions, or even scaled to support larger municipal elections. The real-time capabilities, biometric authentication, and digital transparency of this solution make it a significant step forward in addressing the limitations of conventional electronic voting systems.

In modern democratic societies, the integrity and transparency of elections are critical to maintaining public trust. Traditional voting methods, including paper ballots and standalone electronic voting machines, often face challenges such as voter impersonation, ballot tampering, and logistical inefficiencies. With the increasing availability of smart devices, biometric systems, and cloud-based infrastructure, there is a clear opportunity to revolutionize the voting process. This project proposes the development of a Smart Electronic Voting Machine (SEVM) that combines facial recognition, Internet of Things (IoT) technology, and a web-based administrative backend to create a secure, scalable, and efficient voting platform. The aim is to enhance the credibility of the electoral process by ensuring that only authorized users can vote, while also providing real-time tracking and administrative control over the election.

The proposed system is designed to perform end-to-end electronic voting operations with biometric authentication at its core. At the front end, the voting interaction takes place through an ESP32-CAM module, a compact and cost-effective microcontroller with a built-in camera and Wi-Fi capability. This device acts as the edge terminal at the voting booth. When a voter approaches the device, it captures their facial image and, along with their provided identification and selected candidate, sends this data securely to a backend server. This

backend, developed using Flask—a lightweight Python web framework—receives the vote submission and performs real-time facial verification using machine learning algorithms. If the voter’s live image matches the stored image in the database, the system authenticates the vote and stores the record in a MongoDB database. Otherwise, the vote is rejected, ensuring that only registered users can participate in the election.

The core of the backend functionality revolves around Flask, which is chosen for its simplicity, speed, and flexibility in creating RESTful APIs. These APIs handle user registration, vote submission, result retrieval, and admin login functionalities. MongoDB, a NoSQL database, is used to store data such as voter details, encoded facial features, vote logs, and the current election results. Its schema-less structure and ability to handle large volumes of data make it ideal for this application. The facial recognition component is implemented using Python libraries like OpenCV and face_recognition, which allow efficient face detection, encoding, and comparison. The system leverages these tools to ensure accurate identity verification before a vote is cast.

An important component of the system is the admin interface, which provides a centralized platform for managing the election. Built using basic HTML/CSS/JavaScript or optionally React.js, the admin panel allows election officers to register new voters by uploading facial images, view live vote counts, track voting activity, and ultimately publish the results. This interface communicates directly with the Flask backend via API endpoints, ensuring smooth integration and real-time data flow. Furthermore, tools like Postman have been used extensively during development to test these APIs, simulate real-world voting scenarios, and ensure reliable operation across all components.

The entire system has been engineered to be modular, lightweight, and deployable in environments such as colleges, corporate elections, or local government bodies. Its reliance on open-source tools and affordable hardware makes it both cost-effective and accessible. Unlike traditional EVMs, which can be prone to physical tampering or procedural errors, this solution adds an intelligent verification layer that reduces human intervention and increases electoral confidence. Moreover, the use of facial recognition not only streamlines the voting process but also enhances the user experience by eliminating the need for physical identity documents or voter slips.

In conclusion, the Smart Electronic Voting Machine presented in this project offers a secure and technologically advanced alternative to conventional voting systems. By integrating ESP32-CAM hardware, facial recognition, Flask-based APIs, and MongoDB databases, the system ensures accurate voter verification, seamless vote recording, and real-time result reporting. This project demonstrates how modern technologies can be harnessed to address long-standing issues in election processes and lays the groundwork for further enhancements such as OTP-based two-factor authentication, blockchain-backed vote storage, and large-scale deployment capabilities.

CHAPTER 4

FEASIBILITY STUDY

A feasibility study is essential in evaluating whether the proposed system is practical, achievable, and valuable given the current constraints and available resources. This chapter analyzes the project in terms of its **technical, economic, operational, and legal** feasibility.

The feasibility study of the Music Recommendation System aims to evaluate the practicality and viability of developing and deploying an AI-powered application that recommends music based on a user's facial expressions. This study assesses the technical, operational, and economic feasibility of the project to ensure that it can be implemented efficiently and effectively.

From a technical perspective, the system is highly feasible due to the availability of open-source tools, pre-trained models, and extensive documentation. Python, being a versatile and widely supported programming language, offers powerful libraries for deep learning (such as TensorFlow and Keras), computer vision (like OpenCV), and data processing (like NumPy and Pandas). The required hardware, including a basic webcam and a system with moderate processing power, is easily accessible to most users, making real-time emotion detection possible even on consumer-grade devices. The growing field of facial expression recognition and its integration with machine learning makes this technology stack both stable and expandable.

Operational feasibility is also positive, as the system is designed with ease of use in mind. The user interface can be simple—either through a desktop GUI or command-line interface—allowing users to operate the system with minimal training. Once the webcam is activated, the system automatically detects the user's emotion and recommends songs accordingly. This makes the solution practical for a wide range of users, including students, office workers, and music enthusiasts. In terms of scalability, the system can be extended in the future to incorporate more emotional categories, user preferences, and external music APIs, thereby improving its personalization and relevance.

4.1 Technical Feasibility

This aspect determines whether the technology needed to implement the project is available, reliable, and suitable for the solution.

The SEVM project is technically feasible due to the availability of well-supported hardware and open-source software libraries. The core hardware component, the ESP32-CAM, is a compact and cost-effective microcontroller with integrated camera and Wi-Fi functionality, making it ideal for IoT applications. On the software side, the use of the Flask web framework allows rapid development of backend services, while MongoDB provides a flexible, scalable database solution. Face recognition is implemented using proven Python libraries such as `face_recognition` and `OpenCV`, which are reliable and well-documented. The integration of all these technologies into a single system is manageable with moderate programming and electronics knowledge, making the technical implementation realistic and achievable.

CHAPTER 5

SPECIFICATIONS

Hardware Specifications

- **ESP32-CAM Module**
 - Microcontroller with integrated Wi-Fi and Bluetooth
 - OV2640 camera module for image capture
 - MicroSD card support (optional for logging)
 - GPIO pins for control and interfacing
 - Operates at 3.3V
- **Power Supply**
 - USB or 5V adapter for ESP32-CAM
 - Voltage regulator (if needed) for stable power
- **Other Components**
 - Push button (to trigger camera capture manually, if needed)
 - Breadboard and jumper wires for prototyping
 - LEDs for status indication (optional)

Software Specifications

- **Programming Languages**
 - C/C++ for ESP32 firmware (Arduino IDE or PlatformIO)
 - Python for backend server and face recognition
- **Backend Technologies**
 - **Flask:** Lightweight Python web framework used to handle APIs
 - **MongoDB:** NoSQL database for storing voter data and vote records
 - **Face Recognition:** Python library based on dlib for comparing faces
 - **OpenCV:** Used for preprocessing and handling images
- **Frontend (Admin Panel)**
 - HTML, CSS, JavaScript (or React.js for more advanced UI)

- Axios/Fetch API to communicate with backend
- **Testing Tools**
 - **Postman:** For testing API endpoints (e.g., vote submission, registration)

Functional Specifications

- **User Registration**
 - Admin registers voters by uploading name and facial image
 - Face encoding stored in MongoDB for future verification
- **Vote Casting**
 - ESP32-CAM captures image and sends with voter ID and party
 - Flask backend matches face with registered image
 - On success, vote is recorded in database
- **Admin Operations**
 - View real-time results (votes per party)
 - Add or remove voters
 - Monitor activity logs

Security and Privacy

- Face data is stored as encoded vectors, not raw images (for privacy)
- Communication between ESP32 and server can be encrypted (optional SSL/TLS)
- Authentication required for admin access (via login API)

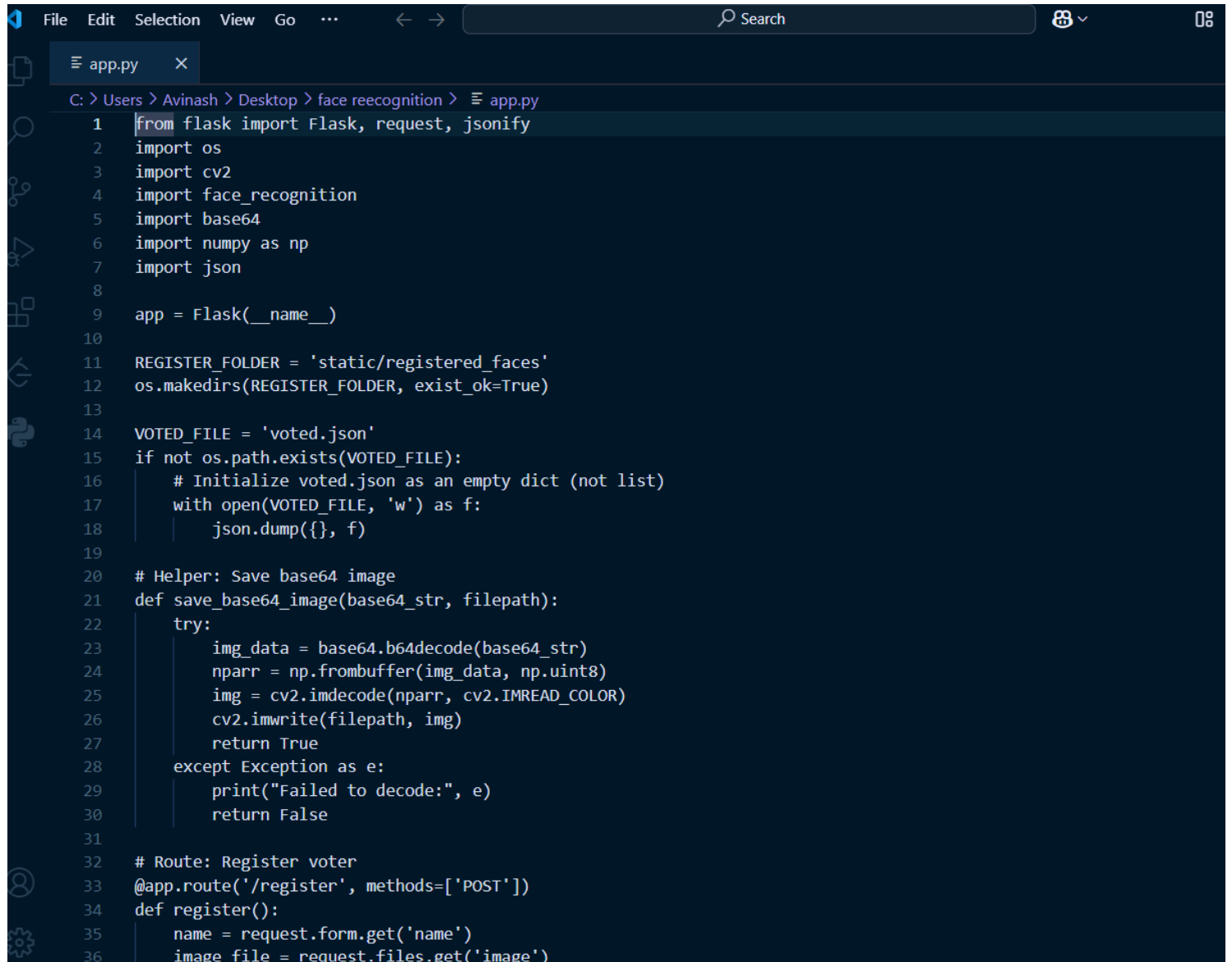
Performance Parameters

- Face verification time: ~0.5–2 seconds per image (depends on server specs)
- Vote recording time: Near real-time with successful verification
- Capacity: Tested up to 50–100 voters in academic prototype

CHAPTER 6

SCREENSHOTS

Main.py



```
1 from flask import Flask, request, jsonify
2 import os
3 import cv2
4 import face_recognition
5 import base64
6 import numpy as np
7 import json
8
9 app = Flask(__name__)
10
11 REGISTER_FOLDER = 'static/registered_faces'
12 os.makedirs(REGISTER_FOLDER, exist_ok=True)
13
14 VOTED_FILE = 'voted.json'
15 if not os.path.exists(VOTED_FILE):
16     # Initialize voted.json as an empty dict (not list)
17     with open(VOTED_FILE, 'w') as f:
18         json.dump({}, f)
19
20 # Helper: Save base64 image
21 def save_base64_image(base64_str, filepath):
22     try:
23         img_data = base64.b64decode(base64_str)
24         nparr = np.frombuffer(img_data, np.uint8)
25         img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
26         cv2.imwrite(filepath, img)
27         return True
28     except Exception as e:
29         print("Failed to decode:", e)
30         return False
31
32 # Route: Register voter
33 @app.route('/register', methods=['POST'])
34 def register():
35     name = request.form.get('name')
36     image_file = request.files.get('image')
```

Postman API register/voter

The screenshot displays the Postman interface with a POST request to `http://127.0.0.1:5000/register` successfully executed. The response is a 200 OK status with a JSON body containing a success message.

Request Details:

- Method: POST
- URL: `http://127.0.0.1:5000/register`
- Params: Query Params

Key	Value	Description
name	Avinash	
image		
Key	Value	Description

Response Details:

- Status: 200 OK
- Time: 3.83 s
- Size: 222 B

Body (JSON):

```
1 {
2   "message": "Voter Avinash registered successfully"
3 }
```


FUTURE SCOPE

Limitations of the Proposed System

1. Internet Dependency

The system requires a stable Wi-Fi connection for the ESP32-CAM to communicate with the Flask server. In remote or rural areas with poor internet access, this could hinder the voting process.

2. Face Recognition Accuracy

The reliability of facial recognition can be affected by:

- Poor lighting conditions
- Low-resolution images from the ESP32-CAM
- Changes in a voter's appearance (e.g., beard growth, glasses, masks)
- Similar facial features among individuals (false positives)

3. Security Concerns

Although facial verification adds a layer of security, data transmitted over HTTP is vulnerable to interception. Without SSL encryption, sensitive voter information could be compromised.

4. Scalability Limitations

The current setup (based on ESP32-CAM and a lightweight Flask server) is suitable for small to medium-scale use (e.g., college elections). Scaling to national-level elections would require much more robust infrastructure, high-speed servers, and distributed databases.

5. User Registration Dependency

The system relies on prior registration with a clear face image. If the user fails to register correctly or if the image quality is low, the system may not authenticate the user during voting.

6. Device Constraints (ESP32-CAM)

- Limited RAM and processing power
- Occasional stability issues during image capture or Wi-Fi communication
- No built-in display for user feedback or instructions

7. No Offline Functionality

The system cannot operate offline. There is no fallback mechanism for offline voting in case of server failure or network outage.

8. Admin Dependency

The admin manually registers voters and creates elections. In larger setups, this could lead to administrative bottlenecks or human error without automation or verification steps.

9. No Multi-Factor Authentication

The current system uses face recognition as the sole means of voter verification. Adding other authentication factors (e.g., OTP, biometric fingerprint) could enhance security but is not implemented in this version.

10. Data Privacy and Legal Compliance

Storing biometric data (like face encodings) raises privacy concerns. Legal compliance with data protection laws (like GDPR or India's DPDP Bill) is not addressed in the prototype.

CONCLUSION

The Smart Electronic Voting Machine (SEVM) project represents a significant step toward reimagining and modernizing the electoral process through the integration of emerging technologies. Traditional voting systems, although well-established, have long faced challenges such as identity fraud, manual errors, lack of transparency, and logistical complexity. This project directly addresses those concerns by implementing a secure, automated, and scalable voting solution that uses facial recognition for voter authentication and IoT technology for real-time data transmission and processing.

By leveraging the ESP32-CAM module, the system offers a cost-effective and compact hardware solution capable of capturing a voter's image and transmitting it to a backend server built using Flask. The server, equipped with face recognition algorithms and MongoDB for data storage, ensures that only registered and verified users are allowed to vote. This not only strengthens the integrity of the voting process but also eliminates the need for traditional paper-based identity verification methods, significantly reducing the chances of impersonation or unauthorized access.

The project architecture ensures modularity and flexibility, making it suitable for various scales of elections—ranging from small institutional votes to larger organizational elections. The use of open-source technologies such as Python, Flask, MongoDB, and OpenCV ensures affordability and community support, while the admin panel provides a user-friendly interface for election officers to manage voters, monitor real-time results, and maintain overall control of the voting process.

Throughout the development and implementation of this project, emphasis has been placed on security, accuracy, and ease of use. Facial verification enhances the authenticity of the vote-casting process, while real-time vote tracking and result publication promote transparency and trust. The project also offers potential for further development, including mobile voting capabilities, blockchain integration, multilingual support, and advanced analytics, all of which could transform it into a full-scale election management platform.

In essence, this project not only demonstrates the technical feasibility of building a smart voting machine using modern tools and frameworks but also contributes meaningfully to the broader discourse on secure, inclusive, and digital governance. It serves as a prototype for future election systems and showcases how innovation and technology can be harnessed to improve democratic processes. With further enhancements, the SEVM could be a pioneering solution in ensuring that elections are not just conducted but conducted with the highest standards of integrity, transparency, and efficiency.

References

- TensorFlow Documentation. (n.d.). *TensorFlow: An End-to-End Open Source Machine Learning Platform*.
<https://www.tensorflow.org/>
- Keras Documentation. (n.d.). *Keras: Deep Learning Library for Python*.
<https://keras.io/>
- OpenCV Documentation. (n.d.). *Open Source Computer Vision Library*.
<https://docs.opencv.org/>
- Geitgey, A. (2018). *face_recognition: Python Face Recognition Library*. GitHub Repository.
https://github.com/ageitgey/face_recognition
- Espressif Systems. (n.d.). *ESP32-CAM Technical Reference*.
<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-cam.html>
- MongoDB Documentation. (n.d.). *NoSQL Database for Storing Voter and Result Data*.
<https://www.mongodb.com/docs/>
- Flask Documentation. (n.d.). *Flask: Python Micro Web Framework for Backend API*.
<https://flask.palletsprojects.com/>
- Bhardwaj, A., Agarwal, N., & Sharma, S. (2020). *IoT Based Smart E-Voting System using Face Recognition and Blockchain*. *International Journal of Computer Sciences and Engineering (IJCSE)*, 8(6), 243–248.
<https://doi.org/10.26438/ijcse/v8i6.243248>
- Singh, J., & Kapoor, R. (2021). *AI-Based Voting System Using Face Recognition and ESP32*. *International Research Journal of Engineering and Technology (IRJET)*, 8(10), 1146–1150.
<https://www.irjet.net/archives/V8/i10/IRJET-V8I10204.pdf>
- HTTPClient Library (ESP32). (n.d.). *ESP32 HTTPClient Arduino Reference*.
<https://docs.espressif.com/projects/arduino-esp32/en/latest/api/http.html>
- Python Requests Library. (n.d.). *HTTP for Humans*.
<https://docs.python-requests.org/en/latest/>