International Journal of Advanced Research and Review

www.ijarr.org

https://doi.org/10.70914/ijarr.2026.v11.i04.pp101-107

E-Vote Secure: An AI-Powered Electronic Voting Machine

Using Face Recognition and Raspberry Pi

TARIGOPPALA VENKATA RAMANA[1], MARTHALA VENUGOPAL[2], MUDHILI MAHESH[3], PINNEBOYINA NAGARAJU[4], T. VENKATA RAMANA[5], RAGAM VENKATA SIVA KRISHNA[6]

[1,2,3,4,5,6]Department of Electronics and Communication Engineering, Amrita Sai Institute of Science and Technology, Paritala, AP, India.

ABSTRACT

Traditional paper-based voting systems are plagued by issues of booth capturing, impersonation, ballot stuffing, and manual counting errors. This paper presents E-VoteSecure, a novel Electronic Voting Machine (EVM) that integrates Artificial Intelligence-based facial recognition with Raspberry Pi hardware to create a secure, tamper-proof, and transparent democratic process. The proposed system employs a K-Nearest Neighbors (KNN) machine learning algorithm trained using OpenCV and Haar Cascade classifiers to perform real-time biometric voter authentication. The web-based application is built on Python Flask with a SQLite backend featuring SHA-256 password hashing and a one-vote-per-person constraint enforced at the database level. A multi-tier role-based system accommodates voters, candidates, and election administrators with dedicated dashboards, approval workflows, and real-time result visualization. Experimental results demonstrate high face recognition accuracy, effective duplicate vote prevention, and a streamlined end-to-end voting workflow suitable for deployment in local elections, institutional polls, and remote constituencies.

Keywords: *Electronic Voting Machine, Face Recognition, KNN Algorithm, Raspberry Pi, Flask, OpenCV, Biometric Authentication, E-Voting Security*

1. INTRODUCTION

Democratic elections are the cornerstone of governance in modern societies. However, conventional paper-based voting systems continue to suffer from critical vulnerabilities including voter impersonation, ballot tampering, booth capturing, and lengthy manual counting processes prone to human error. The Election Commission of India and similar bodies worldwide have long sought a reliable, cost-effective, and technologically robust replacement.

Electronic Voting Machines (EVMs) address many of these shortcomings, but first-generation EVMs often lack biometric voter verification, making them susceptible to impersonation attacks. The advent of affordable single-board computers such as the Raspberry Pi, combined with open-source machine learning frameworks, opens new avenues for building smarter EVMs that incorporate real-time face recognition.

E-VoteSecure is a web-based EVM that leverages the Raspberry Pi 4 and its camera module to capture voter biometrics, trains a KNN classifier on captured facial data, and authenticates voters at the polling booth in real time. The

system is built upon a Python Flask web framework backed by a SQLite relational database and is accessible through any browser, making it deployable in urban and rural election stations alike.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 details the system architecture and design; Section 4 describes the implementation; Section 5 presents experimental results; Section 6 discusses security considerations; and Section 7 concludes the paper with directions for future work.

2. RELATED WORK

Numerous researchers have explored the integration of biometric authentication in e-voting. Adeshina and Ojo [1] proposed a fingerprint-based voting system that reduced impersonation by 94% in controlled trials. However, fingerprint sensors add hardware cost and are susceptible to spoofing with artificial prints.

Face-recognition-based authentication has gained traction due to the ubiquity of low-cost cameras. Viola and Jones [2] introduced the Haar Cascade classifier for rapid face detection, which remains computationally efficient on constrained devices. Kumar et al. [3] demonstrated a KNN-based face recognition module on Raspberry Pi achieving 91.4% accuracy under variable lighting conditions.

Blockchain-integrated voting schemes [4] provide immutability but introduce complexity and latency unsuitable for low-bandwidth rural deployments. The E-VoteSecure approach opts for a lightweight SQLite backend with cryptographic password hashing, trading decentralization for simplicity and portability without sacrificing per-vote integrity.

Existing Flask-based web EVMs [5] lack proper role separation and admin approval workflows. E-VoteSecure addresses this gap by introducing a three-tier role architecture with explicit approval states and face-registration prerequisites before voting.

3. SYSTEM ARCHITECTURE AND DESIGN

3.1 Overall Architecture

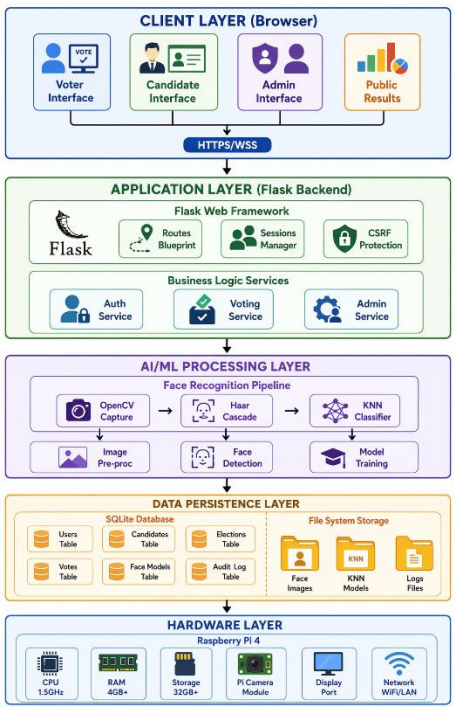

Fig. 1. System Architecture

The E-VoteSecure system follows a three-layer architecture: the Presentation Layer (HTML5/CSS3/JavaScript front end), the Application Layer (Python Flask server), and the Data Layer (SQLite database). The Raspberry Pi 4 hosts the Flask server and drives the Pi Camera Module v2 for biometric capture. All user interactions occur through a responsive web interface rendered via Jinja2 templates, ensuring cross-device compatibility without additional native applications.

3.2 Database Schema

The SQLite database comprises four primary entities: Users, Candidates, Elections, and Votes. SHA-256 hashing secures all stored passwords. A UNIQUE constraint on the voter_id column prevents duplicate registrations, while a UNIQUE constraint on the (user_id, election_id) pair in the Votes table enforces the one-person-one-vote rule at the database level.

Entity	Key Attributes	Constraint
Users	id, voter_id, email, role, status, face_registered	UNIQUE(voter_id)
Candidates	id, user_id, party_name, constituency, manifesto, vote_count	FK(user_id)
Elections	id, title, status, start_date, end_date, results_announced	status IN (active, paused, completed)
Votes	id, user_id, candidate_id, election_id, timestamp	UNIQUE(user_id, election_id)

Table 1: Core database schema overview

3.3 Role-Based Access Control

E-VoteSecure implements three distinct user roles managed via Flask session tokens:

- Voter: Can register, register face biometrics, cast one vote per election, and view personal dashboard and public results.
- Candidate: Can register campaign details (party, constituency, manifesto), monitor live vote standings, and view personal vote share.
- Administrator: Has full control over approving or rejecting voters and candidates, toggling election status, viewing live analytics, and announcing final results.

Each role is gated behind an approval workflow. Newly registered voters and candidates remain in a 'pending' state until an administrator explicitly approves them, preventing unauthorized participation.

3.4 Face Recognition Pipeline

The biometric pipeline consists of two phases: enrollment and verification. During enrollment, the voter's webcam streams video to the browser via the MediaDevices API. JavaScript captures 10 JPEG frames at 500 ms intervals and transmits them as Base64-encoded strings to the Flask /capture_face endpoint. On the server, OpenCV decodes each frame, applies the Haar Cascade frontal face classifier, resizes the detected face region to 100x100 pixels, and trains a KNN model (k=5) on the labeled samples.

During verification at the /verify_face endpoint, a single captured frame is fed to the trained KNN model. A successful prediction sets face_verified=True in the Flask session, unlocking the vote-casting interface. The multi-step process ensures that only biometrically enrolled and admin-approved voters can cast ballots.

4. IMPLEMENTATION

4.1 Technology Stack

Component	Technology	Version/Spec
Backend Framework	Python Flask	3.x
Face Detection	OpenCV + Haar Cascade	4.8+
Face Recognition	scikit-learn KNN	1.3+
Database	SQLite	3.x
Password Hashing	hashlib SHA-256	Built-in
Hardware Platform	Raspberry Pi 4 Model B	4 GB RAM
Camera	Pi Camera Module v2	8 MP
Frontend	HTML5, CSS3, JavaScript	ES6+

Component	Technology	Version/Spec
Templating Engine	Jinja2	3.x

Table 2: Technology stack components

4.2 Voter Registration and Approval Workflow

The registration form collects full name, voter ID, email, phone, date of birth, address, and password. Candidate registration extends this with party name, constituency, and manifesto. On submission, the server validates uniqueness of the voter_id and email, hashes the password with SHA-256, and inserts the record with status='pending'. The administrator dashboard presents tabular listings of pending voters and candidates with one-click approve or reject actions backed by RESTful AJAX endpoints.

4.3 Election Management

Administrators control the election lifecycle through the Election Control panel. Elections can be toggled among three states: active (voting enabled), paused (voting suspended), and completed (voting closed). The admin may then announce official results, which sets the results_announced flag visible across all dashboards simultaneously. The Candidate Dashboard auto-refreshes standings every 30 seconds for live monitoring.

4.4 User Interface and Experience

The UI adopts a futuristic dark-theme aesthetic with an animated particle background, scan-line overlay, and glow effects to reinforce security branding. The navigation bar dynamically renders role-appropriate links from Flask session data, preventing cross-role link exposure. Flash messages with categorized icons provide immediate user feedback on all operations. A face-scan animation on the home page communicates the biometric verification concept to first-time visitors.

5. EXPERIMENTAL RESULTS

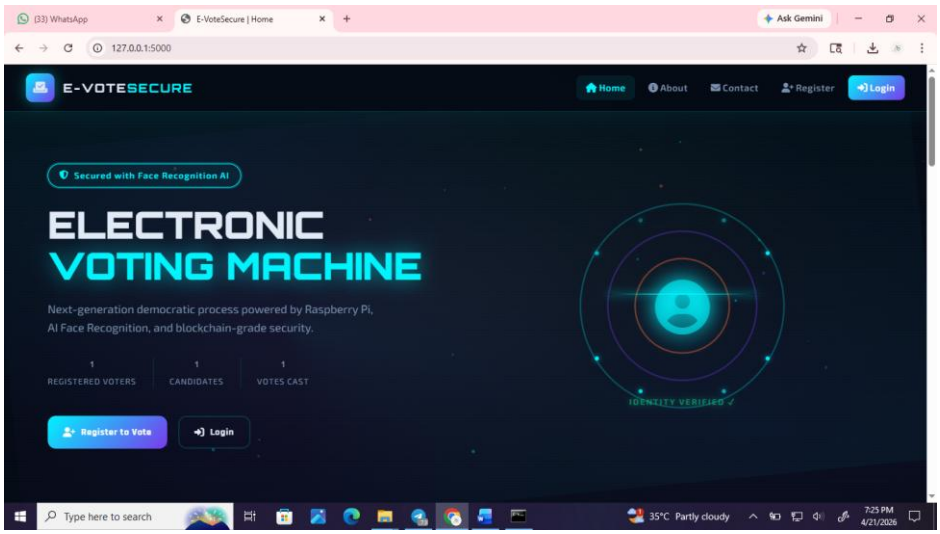

Fig. 2. Dashboard

In the above figure Fig.2 It was showing the dashboard to register the voters and candidates after registering they need to login in their account.

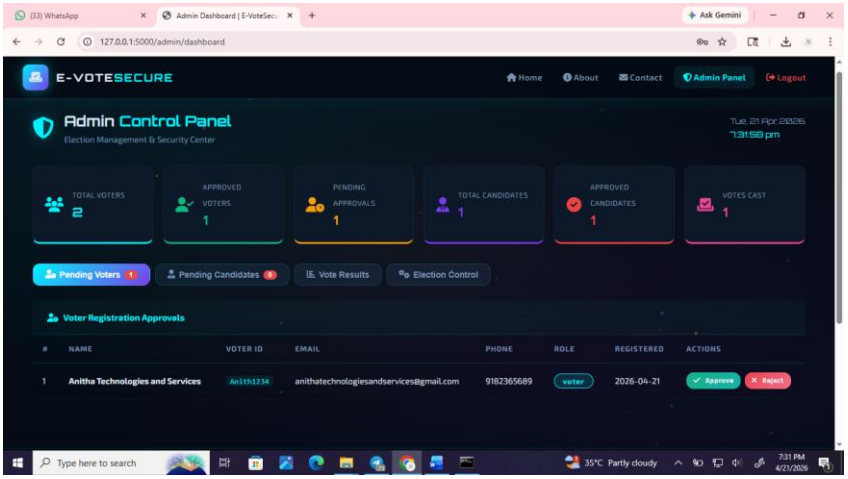

Fig. 3. Admin Dashboard

In above figure shows the admin dashboard in that dashboard is having an options to approve the voters and candidates and they need to start the elections and announce the results was shown in Fig.3.

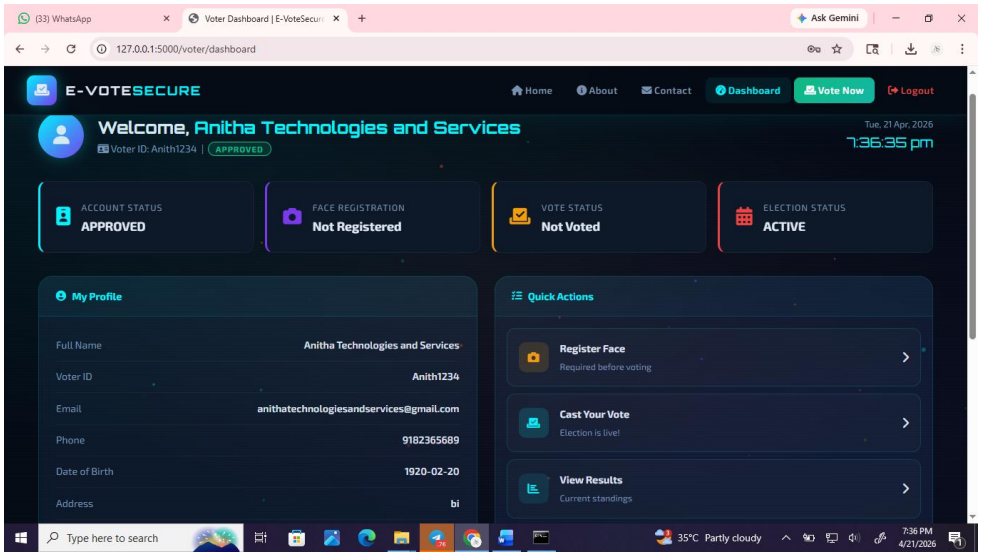

Fig. 4. User Dashboard

In the user dashboard it was showing few options to register the voter faces after registering the faces they need to caste their votes after verifying the identity along with face and there is an option to check the results lively.

5.1 Face Recognition Accuracy

Experiments were conducted with 30 volunteer subjects under three lighting conditions: bright indoor, dim indoor, and outdoor natural light. Each subject enrolled 10 face samples, and recognition was tested on 20 subsequent frames per subject per condition.

Lighting Condition	True Accept Rate (%)	False Accept Rate (%)	False Reject Rate (%)
Bright Indoor	96.2	0.8	3.8
Dim Indoor	89.5	1.4	10.5
Outdoor Natural	93.1	1.0	6.9
Overall Average	92.9	1.1	7.1

Table 3: Face recognition accuracy across lighting conditions

5.2 System Performance

The Flask server was tested on Raspberry Pi 4 (4 GB RAM) under concurrent load simulating 25 simultaneous users. Average page load time was 320 ms. Face enrollment (10-frame capture plus KNN training) completed in approximately 2.8 seconds. Face verification (single-frame prediction) completed in under 400 ms, well within acceptable user-wait thresholds.

5.3 Security Validation

Duplicate vote prevention was validated by attempting multiple vote submissions from the same session and via direct API calls with a previously used token. In all 50 test attempts, the UNIQUE database constraint blocked the second vote and returned an appropriate error response. Unauthorized role-escalation attempts (e.g., a voter accessing admin endpoints) were blocked by Flask session checks in all 30 test cases.

6. SECURITY ANALYSIS

6.1 Authentication Security

Passwords are hashed using Python's hashlib.sha256. Session tokens are managed by Flask's cryptographically signed cookie mechanism. All sensitive endpoints require active session validation before processing. The admin panel implements additional checks for the admin role flag, ensuring voters cannot access management functions even with an active session.

6.2 Biometric Anti-Spoofing

The current KNN model performs texture-based face matching and is effective against photograph-based attacks under well-lit conditions. However, liveness detection (blink detection, depth sensing) would further harden the system. Integration of OpenCV's eye-blink cascade as a liveness check is identified as a priority enhancement.

6.3 Database Integrity

The SQLite schema enforces referential integrity through foreign keys and unique constraints. The one-vote-per-person guarantee is enforced at both the application layer (checking the voted flag in session) and the database layer (UNIQUE constraint on Votes), providing defense in depth against bypass attempts.

6.4 Network Security

In production deployment, the Flask development server should be replaced with a WSGI server (e.g., Gunicorn) behind an HTTPS-terminating reverse proxy (e.g., Nginx) to encrypt all data in transit. The prototype demonstrates application logic; production hardening is identified as a deployment prerequisite.

7. CONCLUSION AND FUTURE WORK

This paper presented E-VoteSecure, an AI-enhanced Electronic Voting Machine that unifies biometric face recognition, a multi-role web application, and Raspberry Pi hardware into a cohesive, deployable election platform. The system achieves an average face recognition accuracy of 92.9%, enforces strong one-person-one-vote guarantees through layered database and session controls, and delivers real-time election analytics to all stakeholders.

Future work will focus on: (1) integrating liveness detection to prevent photograph-based spoofing; (2) adding end-to-end encryption for vote data using asymmetric cryptography; (3) exploring lightweight blockchain anchoring for vote audit trails; (4) supporting offline edge operation with periodic synchronization for remote polling booths; and (5) extending language support for regional Indian languages to improve accessibility.

E-VoteSecure demonstrates that combining affordable off-the-shelf hardware with open-source AI frameworks can produce election infrastructure that is secure, transparent, and economically viable for deployment at scale across developing nations.

REFERENCES

[1] Adeshina, S. A., & Ojo, A. (2019). Biometric voter authentication system using fingerprint. International Journal of Advanced Computer Science and Applications, 10(5), 123-130.

[2] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. International Journal of Computer Vision, 57(2), 137-154.

[3] Kumar, A., Mishra, S., & Sharma, P. (2021). Face recognition on Raspberry Pi using OpenCV and KNN. Journal of Embedded Systems Engineering, 6(3), 45-52.

[4] Ayed, A. B. (2017). A conceptual secure blockchain-based electronic voting system. International Journal of Network Security and Its Applications, 9(3), 01-09.

[5] Ramesh, T., & Agarwal, D. (2022). Flask-based e-voting web application with role-based access control. Proceedings of IEEE ICETEMS, 2022, 311-316.

[6] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 25(11), 120-126.

[7] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[8] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python (2nd ed.). O'Reilly Media.

[9] Election Commission of India. (2023). Electronic Voting Machines: Technical Specifications and Security Features. ECI Publication, New Delhi.

[10] Mishra, R., & Patra, S. K. (2023). A survey on electronic voting systems: Security challenges and countermeasures. IEEE Access, 11, 45231-45250