GOVERNMENT OF WEST BENGAL

# CENTRAL CALCUTTA POLYTECHNIC

21, CONVENT ROAD, KOLKATA-14, WB

**DEPARTMENT OF COMPUTER SCIENCE & TECHNOLOGY**

# CERTIFICATE

This is to certify that this report of Third Year project, entitled "**New Generation Voting System**" is a record of bona-fide work, carried out by **Saikat Roy, Debdeep Sen Gupta, Dipayan Sardar, Ankita Pan, Sneha Sadhu, Samridhi Mandal.**

In my opinion, the report in its present form is in partial fulfillment of all the requirements, as specified by the Central Calcutta Polytechnic and as per regulations of the West Bengal State Council of Technical & Vocational Education and Skill Development. In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for final year project in Computer Science and Technology in the year 2024-2025.

Guide / Supervisor

_____

Mr./Ms./Dr. LMN

HOD of CST department

_____

Mr./Ms./Dr. LMN

Department of Computer Science and Technology

# Acknowledgement

I would like to express

- my sincere gratitude to all these individuals for mentoring and supporting me & my team members in completing this project.

- Our guide **Mr. Samaresh Hazra,** for proving us with invaluable insights and direction.

- Our respective Principal Mr. Tapas Sen, for fostering an environment of learning and creativity within our college.

- To my parents, their constant encouragement, patience and understanding have been the pillars of my success.

- I am also thankful to my seniors, friends, team members who contributed ideas and perspectives that enriched the project.

- Thank you everyone for shaping this project and enhancing my learning experiences.

# PROJECT SYNOPSIS

This biometric voting system is highly secure and authentic. It ensures voters' data protection, the Polling Officer's authentication and also provides a cross-checking option during voting for security.

It comes with a biometric-enabled EVM, ensuring that a valid voter gives their own vote.

It also ensures that no one can give double votes

It comes with a highly protected encryption system that ensures data security.

# CONTENTS

# INTRODUCTION

The biometric voting system is a highly secure and authentic method of conducting elections. It is specifically designed to enhance the integrity of the voting process and minimize the risk of electoral fraud. One of its key features is the protection of voter data, ensuring that personal and biometric information is stored and handled securely. This system makes it nearly impossible for unauthorized individuals to access or tamper with voter information. It also includes a mechanism to authenticate the identity of Polling Officers, which prevents misuse of authority during the election process. Only verified officers are allowed to manage and oversee the polling activities. A built-in cross-checking option during the voting process adds an extra layer of security. This feature verifies the identity of the voter in real-time to ensure the right person is voting. By confirming the identity at multiple stages, the system reduces the possibility of impersonation. The use of biometric-enabled Electronic Voting Machines (EVMs) ensures that only valid, registered voters are allowed to cast their votes. These machines can recognize unique biological traits such as fingerprints or iris scans. This prevents duplicate voting and ensures that each person votes only once. It also eliminates proxy voting and identity theft during elections. The system brings more transparency and trust to the voting process. It reduces manual errors and enhances the efficiency of polling operations. Voters can be processed quickly and accurately. Election officials find it easier to manage and verify voters. The biometric system also supports secure data tracking for post-election audits. This advanced technology contributes to fair and credible elections. It is an important step toward modernizing the democratic process.

# POLLING OFFICER REGISTRATION

## 📘 Officer Registration Module Documentation :

**Module:** Officer Registration
**Application:** Online Biometric Voting System(OBVS)
**Version:** 1.0
**Author:** [Group A]
**Last Updated:** [09.06.2024]

## 📜 Overview:

The Officer Registration module is responsible for securely registering election officers into the Online Voting System (OVS). These officers are granted access to administrative functionalities such as voter verification, system monitoring, and election result validation. The registration process involves collecting officer details, validating them via OTP (One Time Password), encrypting sensitive information, and storing the data in a secure database.

## ✅ Key features:

- Secure Officer Signup Form
- Email-based OTP Verification
- Profile Image Upload
- Password Encryption
- Data Storage in MongoDB

## 📐 Technologies used:

- **Frontend Framework:** Flet (Python)

- **Backend:** Python
- **Database:** MongoDB
- **Security:** bcrypt encryption for password
- **OTP Delivery:** SMTP (email)
- **Image Handling:** PIL or base64 + GridFS

## 📝 Data Fields Collected:

| Field Name | Type | Description |
|---|---|---|
| Full Name | String | Officer's full legal name |
| Email Address | String | Used for login and OTP verification |
| Password | String | Encrypted using bcrypt before storage |
| Phone Number | String | Optional, may be used for contact |
| Profile Image | File (Image) | Stored in MongoDB GridFS |
| OTP | Integer | 6-digit code sent to email for verification |

## 🔐 Registration workflow:

### 1. Form Submission

- Officer enters name, email, password, and uploads photo.
- System validates input (empty fields, email format, etc.)

### 2. OTP Generation

- 6-digit OTP is generated using  random.randint().
- Sent to the officer's email using Python's SMTP protocol.

### 3. OTP Verification

- Officer enters the OTP received via email.
- System compares with generated OTP stored temporarily in memory/cache.

**4. Password Encryption**

- The password is hashed using bcrypt.hashpw() for security.

**5. Data Storage**

- Textual data saved in MongoDB collection officers.
- Profile image saved in GridFS with reference in officer document.

## MongoDB Schema (Example):

```
{
 "name": "Samridhi Mandal",
 "email": "samridhimandal255@gmail.com",
 "password": "$2b$12$xxxxxxxxxxxxxxxxxxx",
 "image_file_id": ObjectId("image-file-objectid"),
}
```

## Validation Rules

- Email must be unique.
- Password must be a character with a mix of letters and numbers.
- OTP must match and expire after 5 minutes.
- Only image formats allowed: PNG, JPG, JPEG.

## Error Handling:

| Scenario | Message Displayed |
|---|---|
| Invalid Email Format | "Please enter a valid email." |
| OTP Mismatch | "Incorrect OTP. Try again." |
| Email Already Exists | "Officer already registered." |
| Image Upload Failed | "Could not upload profile image." |

| | |
|---|---|
| MongoDB Insert Error | "Registration failed. Try again." |

## 🧰 Future Enhancement :

- Add phone OTP verification
- Admin approval step after officer registration
- Email domain whitelisting for government officials
- Dashboard to manage registered officers

## 📷 Page Layout:

# VOTER REGISTRATION

📄 **Documentation: Voter Registration Page – Online Voting System (Flet App):**

**Module:** Voter Registration
**Application:** Online Biometric Voting System (OBVS)
**Version:** 1.0
**Author:** [Group A]
**Last Updated:** [09.06.2024]

🧩 **Overview:**

This is a graphical Voter Registration Form developed using the Flet framework (Python-based front-end for Flutter). It allows users to input and submit voter information, capture a photo, verify email, and scan fingerprints.

✅ **Key Features:**

- **User-Friendly Interface:** Simple and organised form layout for quick voter registration.
- **Photo Capture Integration:** Allows users to capture their photo directly from the form (camera-enabled).
- **Gender & Location Selection:** Dropdowns for selecting gender, state, district, and constituency to ensure accurate regional data entry.
- **Date of Birth Picker:** Built-in date selection for ease of entering birth dates.
- **Email Verification:** Button to validate the voter's email address (format or via OTP/API logic).

- **Fingerprint Authentication:** Connects to a biometric scanner through selectable COM ports for fingerprint capture.
- **Error Handling:** Real-time feedback (e.g., missing photo error) ensures proper completion before submission.
- **Form Submission:** Collects and sends validated data to the backend for secure voter registration.

### 💻 User Interface Components:

| Components | Widget Type | Purpose |
|---|---|---|
| Voter's Name | TextField | Input for voter's full name |
| Capture Photo Button | ElevatedButton | Initiates camera to capture photo |
| Photo Display | Image | Display captured photo; Requires src or sec_base64 |
| Gender | Dropdown | For selecting gender |
| Date of Birth | DatePicker/Button | Opens calender to select DOB |
| State | Dropdown | Select state (e.g West Bengal) |
| District | Dropdown | Select district (e.g Purulia) |
| Constituency | Dropdown | Select voter constituency |
| Vill/City | TextField | Enter village or city |
| PIN Code | TextField | Enter 6-digit postal PIN code |
| Voter's Email | TextField | Input voter's email address |
| Verify Email Button | ElevatedButton | Trigger email format validation or verification process |
| COM Port Selection | Dropdown | Select hardware COM part for fingerprint scanner |
| Add Fingerprint Button | ElevatedButton | Initiate fingerprint capture via connected device |
| Submit Button | ElevatedButton | Submits the form data after validation. |

### ⚠ Known Issues / Validation Errors:

- **Missing Image Source Error:** Ensure that captured image is stored as Image(src=...) or Image(src_base64=...).
- **Email Field:** Currently contains a space after "NASKAR", which can cause validation failure. Must be trimmed and validated with regex.

### 📐 Technologies used:

- **Frontend framework**: Flet(Python)
- **Backend:** Python
- **Database:** MongoDB
- **Security:** Cryptography encryption for passwords
- **OTP Delivery:** SMTP (email)
- **Image Handling:** PIL or base64 + GridFS

### 🔄 Expected Backend Integration:

- **Email Verification:** Likely uses SMTP/OTP or regex + API.
- **Fingerprint:** Connects to a biometric device via selected COM port.
- **Data Submission**: Final data sent to a backend API or database (MongoDB).

### 🧪 Future Improvement:

✅ Add real-time input validation.

✅ Implement loader/spinner on "Verify Email".

✅ Auto-fill address from PIN (optional).

### 📁 Project Structure :

voter_registration/

├── Voter_register_APP.py

├── state_district_library.py

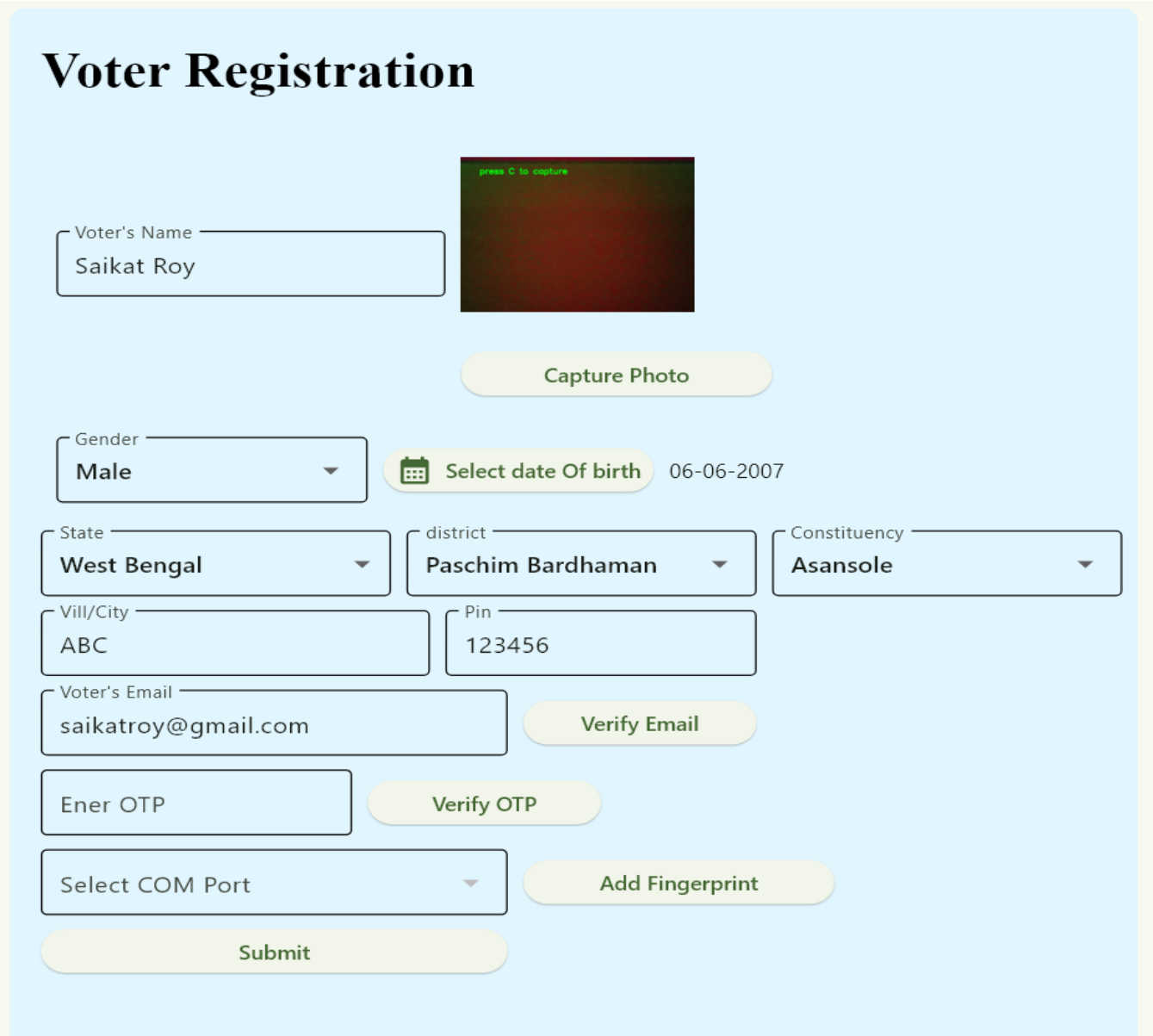
├── constitution_list.py

└── captured_photo.jpg

## MongoDB Schema (Example) :


```
{
 "name": "test 1",
 "date_of_birth": "25-04-2025",
 "gender": "others",
 "state": "bihar",
 "district": "begusarai",
 "constituency": "buxar",
 "area": "vill 1",
 "pin": "00000000",
 "photo": {
  "$oid": "680b5d707726c9bfe57f13ed"
 },
 "email": {
  "$binary": {
   "base64":
"Z0FBQUFBQm9DMTF3OURUdEs2cE1WNV9OS2NBOVFmaHg4Z2U3TkVqYXoxQW16azQzQjcxNn
N1U0p3Zlg0OVRDREdiUENkLXc3RFZFajg0WmF6dnFGQlVOazZIeG5lNWRrOXo2bi01NXMwVDAxT
npMUko2RTdmNGM9",
    "subType": "00"
  }
 },
 "voter_id": "B998B478",
 "identity_key": "begusaraitest_125-04-2025vill_1",
 "fingerprint_data": [768]
}
```

## Page Layout:


Voter Registration
press C to capture
Voter's Name
Saikat Roy
Capture Photo
Gender
Male
Select date Of birth
06-06-2007
State
West Bengal
district
Paschim Bardhaman
Constituency
Asansole
Vill/City
ABC
Pin
123456
Voter's Email
saikatroy@gmail.com
Verify Email
Ener OTP
Verify OTP
Select COM Port
Add Fingerprint
Submit

# POLLING OFFICER LOGIN

📄 **Documentation: Polling Officer Login Page – Online Voting System (Flet App):**

**Module:** Officer Login
**Application:** Online Biometric Voting System(OBVS)
**Version:** 1.0
**Author:** [Group A]
**Last Updated:** [09.06.2024]

## ☑ Objective:

To allow polling officers to securely log into the online voting system, enabling them to perform election-related duties such as verifying voter identities, starting/stopping voting sessions, and monitoring the voting process.

## 🧩 Key Features:

1. Secure Login Interface
2. Email and Password Authentication
3. Optional OTP Verification for Extra Security
4. Input Validation & Error Handling
5. Session Management

## 💻 User Interface Components:

| UI Element | Description |
|---|---|
| **Enter Polling ID** | Unique ID assigned to the polling officer |
| **Enter Email** | Email ID registered in the system |
| **Enter Password** | Secure password (masked) |
| **Select Center** | Dropdown menu to select assigned polling center |
| **Verify Identity Button** | Button to submit credentials and login |

**Styling:**

- The login card has a rounded border with a blue outline.
- The **"Verify Identity"** button is styled in green to highlight its importance.

### 🔐 Authentication Process:

1. **<u>Polling Officer enters:</u>**
    a. Polling ID
    b. Email
    c. Password
    d. Selects assigned center
2. **<u>Validation on Frontend:</u>**
    a. All fields are required.
    b. Email format must be valid.
    c. Password must meet security criteria (length, characters, etc.).
3. **<u>Backend Verification:</u>**
    a. Match Polling ID + Email + Center from MongoDB.
    b. Check hashed password match using bcrypt or similar.
    c. If verified, officer is logged in and redirected to the dashboard.
4. **<u>Login Response:</u>**
    a. ✅ Success: Redirect to Polling Dashboard
    b. ❌ Failure: Show appropriate error messages

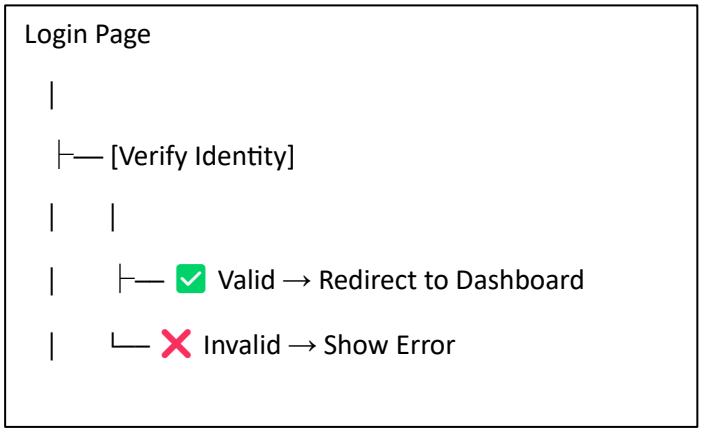
### 🗃️ Database Schema (MongoDB):


```
{"polling_id": "A9715F1F",
 "email": dipsarda27@gmail.com
 "password": "encrypted_hash_here",
 "center": "CCP",
 "last_login": "2025-06-05T10:41:00Z"}
```

## ⚠ Error Handling:

| Condition | Error Message |
|---|---|
| Empty Fields | "All fields are required." |
| Wrong Email or Password | "Invalid credentials." |
| Center Mismatch | "Center does not match record." |
| Backend Failure | "Something went wrong. Try again." |

## 🔁 Navigation Flow:



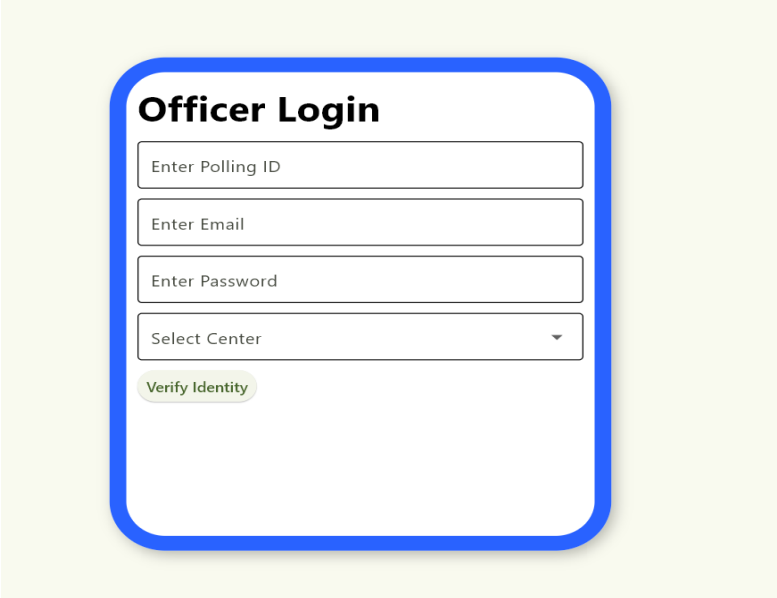

## 🔒 Security Considerations:

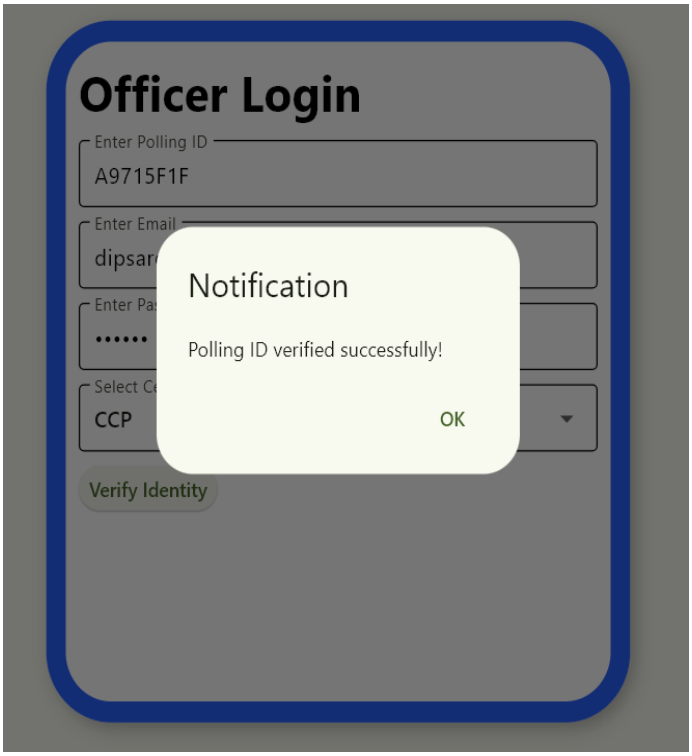
- Password is stored in hashed form (never plain text)
- Center selection avoids unauthorized cross-center access
- Backend uses secure login API over HTTPS

## 📌 Future Enhancements:

- Password recovery feature
- Biometric login (if on Android devices)
- Role-based dashboard access
- Login audit trail

## 📷 Page Layout:


Officer Login
Enter Polling ID
Enter Email
Enter Password
Select Center
Verify Identity


Officer Login
Enter Polling ID
A9715F1F
Enter Email
Notification
Polling ID verified successfully!
OK
CCP
Verify Identity

# MAIN WEBSITE

📄 **Documentation: Main Website – Online Voting System**

**Module:** Informational Website
**Application:** Online Biometric Voting System (OBVS)
**Version:** 1.0
**Author:** [Group A]
**Last Updated:** 09.06.2024

🧩 **Overview:**

The **Main Website** of the **Online Biometric Voting System (OBVS)** serves as a static yet dynamic-like **informational portal** for voters and stakeholders. It is developed using **Node.js with Express.js** for the backend, and **HTML, CSS, and JavaScript** for the frontend, with modular architecture that allows reusable components such as a dynamic header and footer.

The website focuses on improving public awareness about the election process, voting rights, and candidate/party information. It provides essential resources like voter details, election stories, downloadable voter cards (linked module), and real-time announcements. While this module **does not support actual biometric voting or login**, it supports the OBVS infrastructure by being an accessible, responsive, and modern communication platform.

This version is served via an **Express server**, and the public directory includes static assets and frontend pages, making it suitable for both local and live deployment.

✅ **Key Features**

- **Express.js Backend:** Serves static files and handles client routing cleanly.
- **Reusable Components:** Header and footer loaded dynamically using JavaScript for consistency across pages.
- **Informational Homepage:** Includes banner, slogan and key action buttons.

- **Navigation Bar:** Sticky top nav with Home, About Us, Management, FAQ, and Contact.
- **Content Sections:**
  It has a very useful layout. Main actions, such as Voter Details, Political Parties, Election Stories, Election Results, are easy to see for a user.
- **Announcements Panel:**
  Displays the latest public updates including registration guidelines, updated schedules, and voting machine instructions.
- **Myth vs Reality + Press Releases:**
  Quick access section to combat misinformation and provide official communications.
- **Contact Form with JS Handling:**
  Accepts user concerns and sends emails via backend service.
- **Download Voter Card (Linked):**
  A feature allowing users to securely download their voter ID card in PDF format (linked to a different module).
- **Responsive Design:**
  Built using Bootstrap and media queries for mobile and desktop views. Website automatically adjusts to different screen sizes, ensuring usability on both desktop and mobile browsers.

### ⚠ Known Limitations

- Has a very less database interaction.
- Voting, registration, or authentication modules are **not included** in this system.
- No session/user authentication is implemented.

### Technologies Used

- **Frontend:** HTML5, CSS3, JavaScript (Vanilla), Bootstrap 4.6
- **Backend:** Node.js (v18+), Express.js (for static file serving)

- **Dynamic Layout Injection:** JavaScript fetch() used to inject header/footer HTML into each page
- **Server Platform:** Compatible with Node.js runtime (local or cloud server like Heroku, Render, etc.)
- **Hosting Environment:** Local development via localhost:3000 (Express server)

## 🧪 Future Improvements

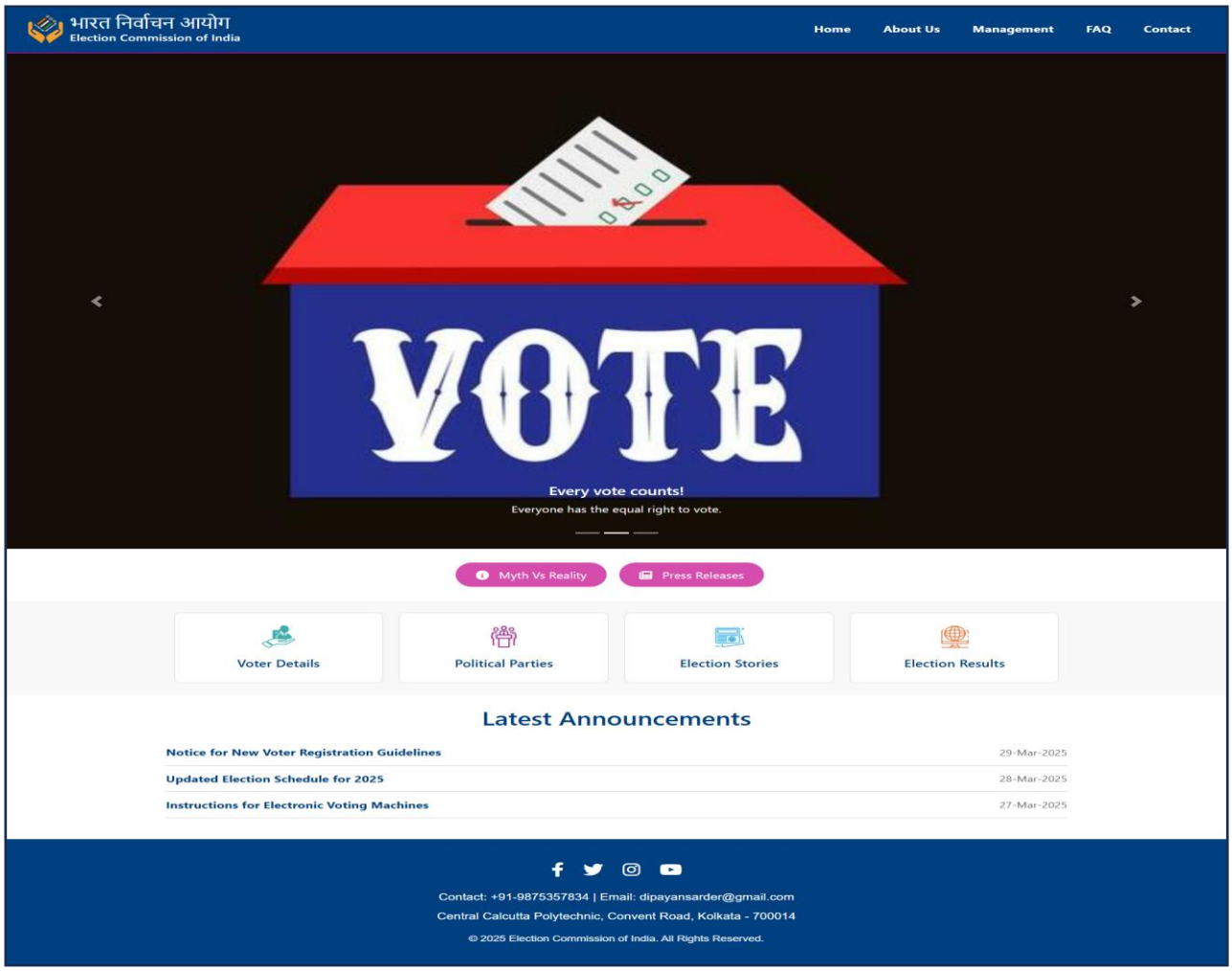
- Convert announcements into JSON data and load via AJAX to mimic dynamic behaviour
- Add content management system (headless CMS or admin panel) for easier updates
- Include search functionality across pages
- Add chatbot or live chat widget for real-time support
- Add accessibility improvements: high contrast mode, text reader support
- Countdown timer for election events
- Newsletter subscription system
- Implement a secure login panel (future admin/voter role)

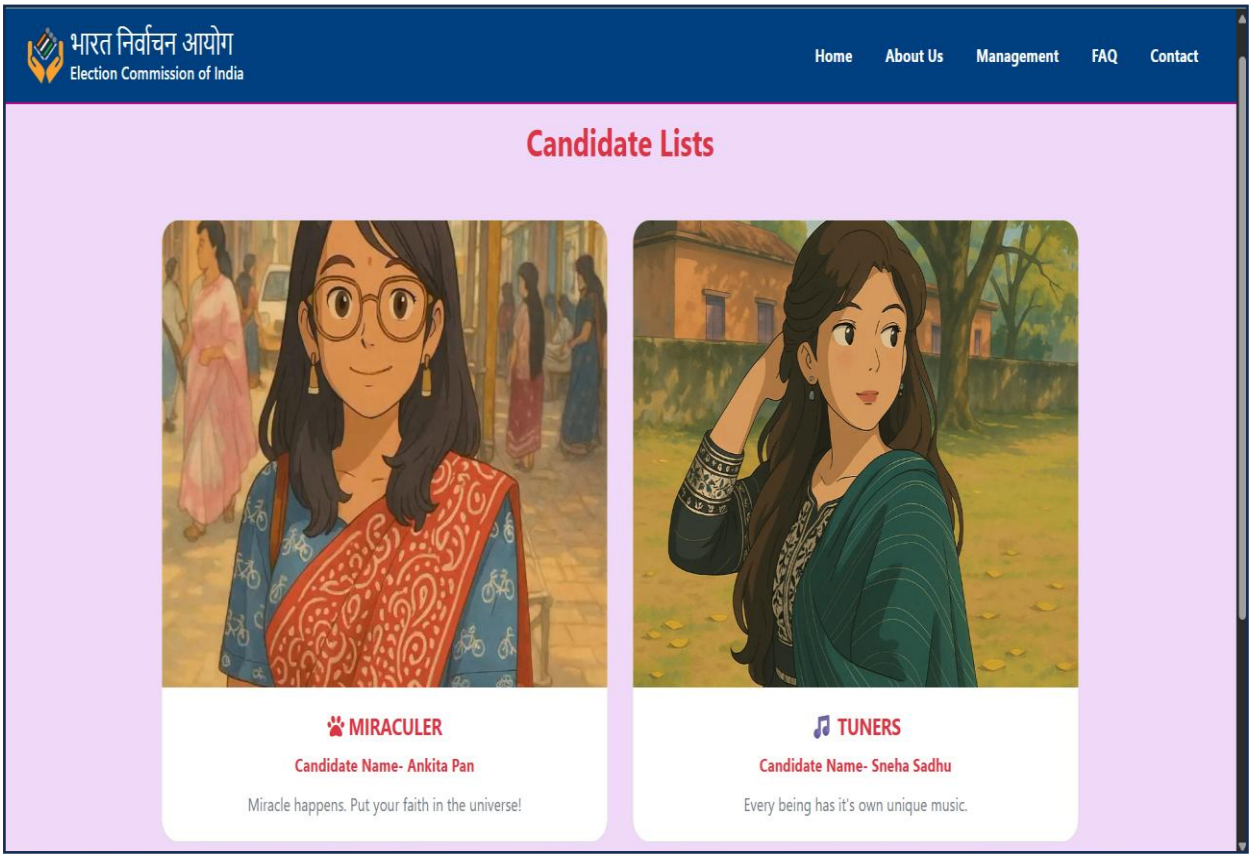
## 📁 Project Structure:

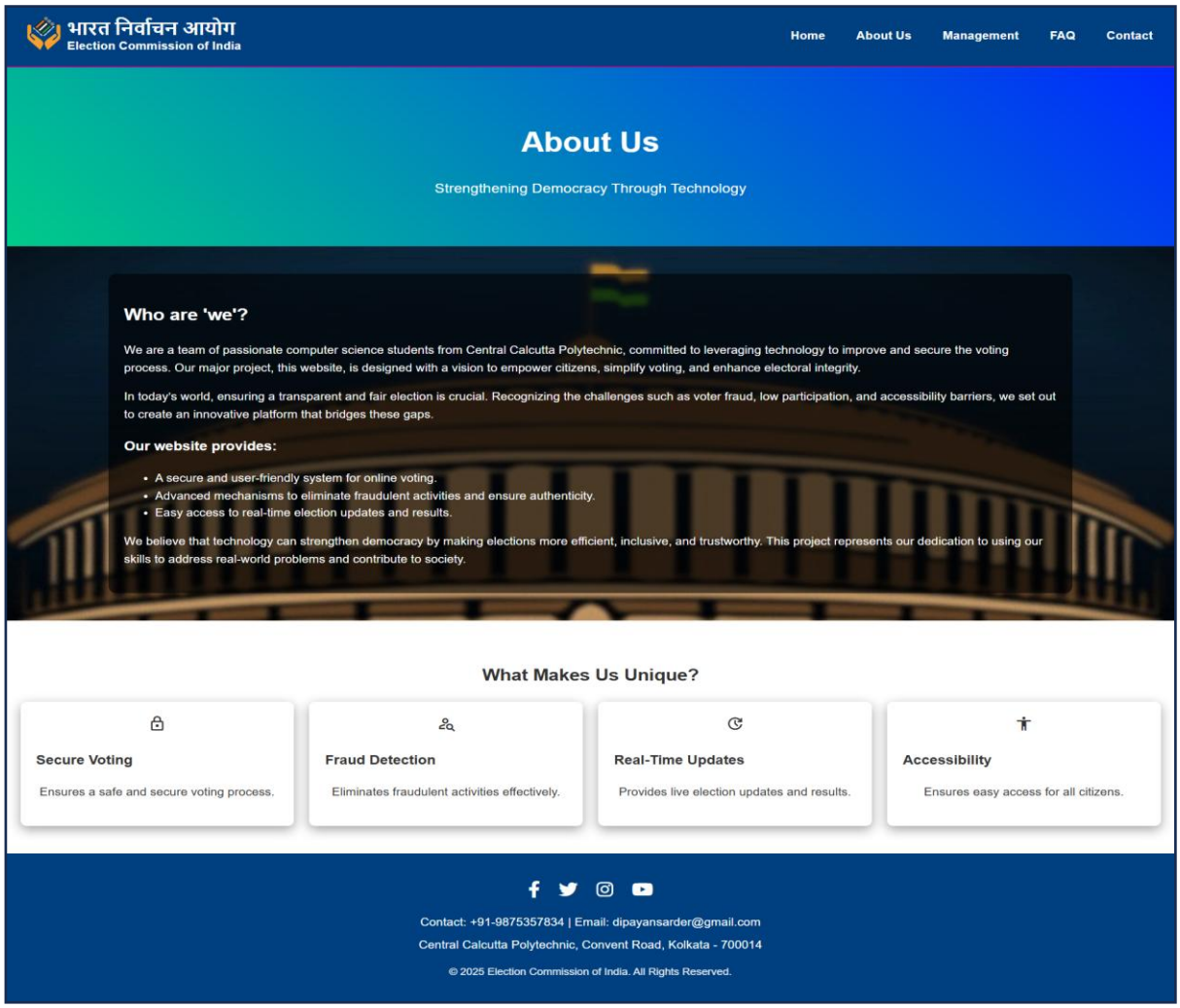
```
ECI-NODE-APP/
    ├── node_modules/          # Node dependencies
    ├── public/                # Public directory served by Express
    │   ├── assets/            # Static images and media
    │   │   ├── logo & other images
    │   ├── static/            # JavaScript and CSS files
    │   │   ├── script.js
    │   │   └── style.css
```

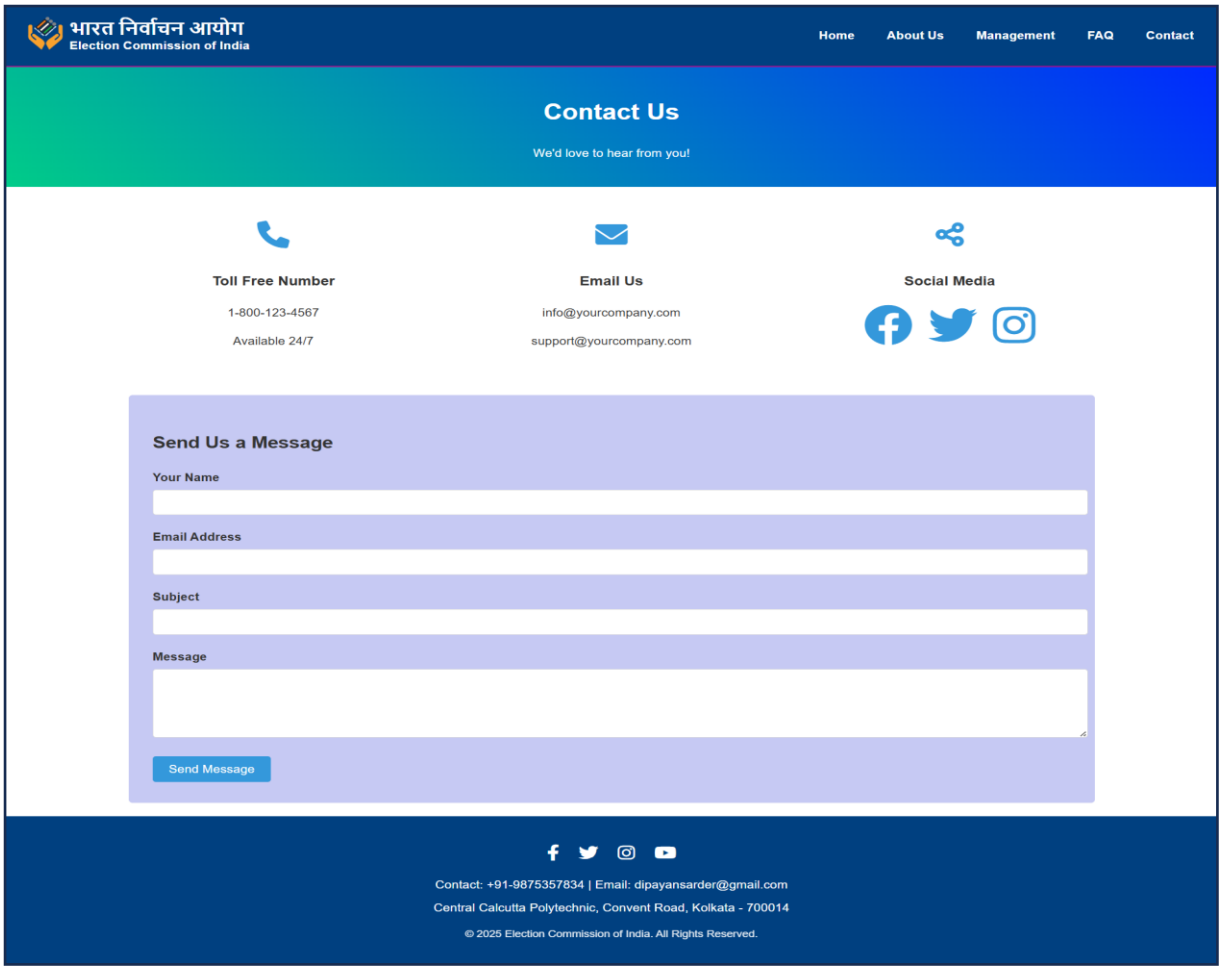
```
│   ├── aboutus.html
│   ├── contactus.html
│   ├── footer.html
│   ├── header.html
│   ├── index.html
│   ├── mythVSreality.html
│   └── politicalparty.html
├── package.json            # Node project metadata and dependencies
├── package-lock.json       # Exact dependency versions
└── server.js               # Node.js Express server script
```

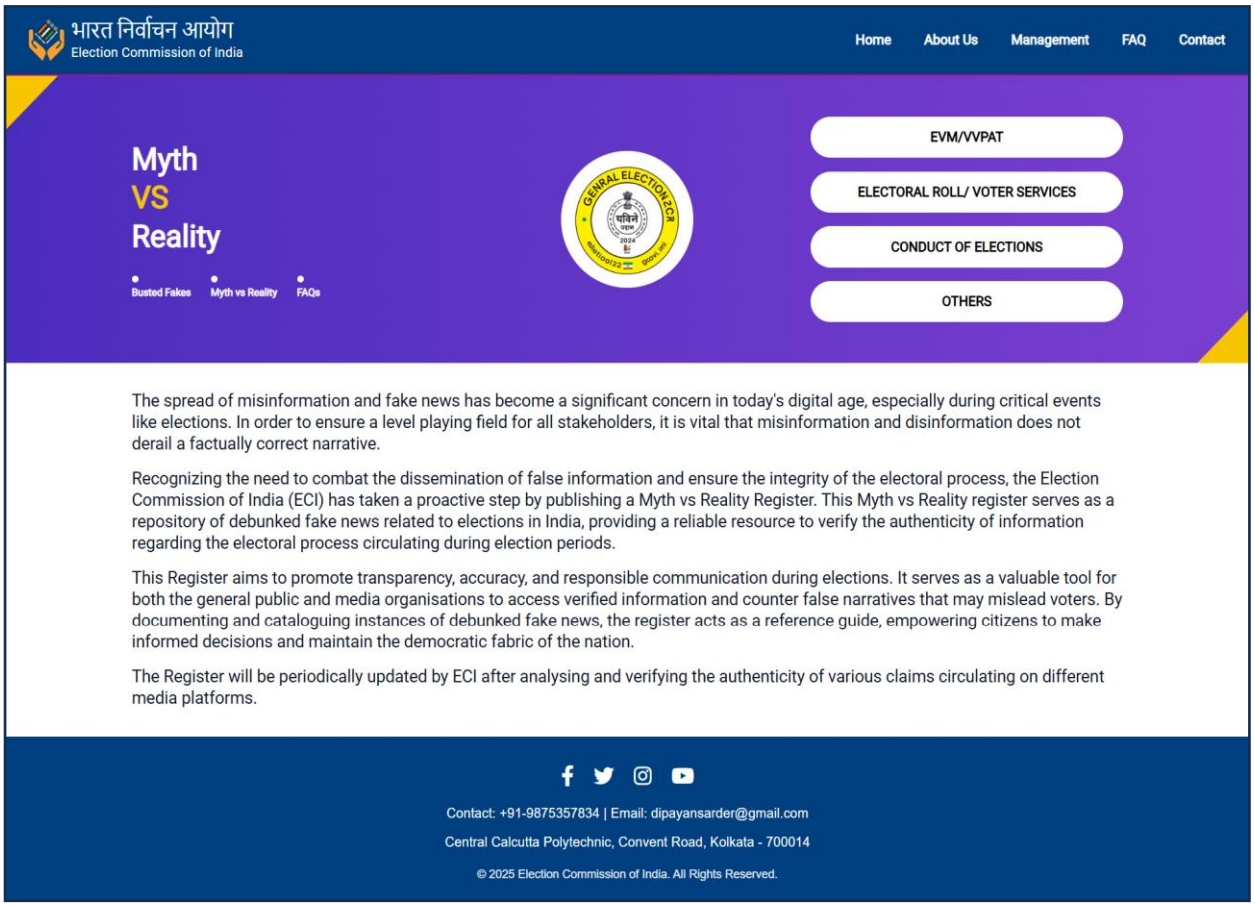
## 📷 Page Layout:




भारत निर्वाचन आयोग
Election Commission of India
Home
About Us
Management
FAQ
Contact
Candidate Lists
MIRACULER
Candidate Name- Ankita Pan
Miracle happens. Put your faith in the universe!
TUNERS
Candidate Name- Sneha Sadhu
Every being has it's own unique music.


भारत निर्वाचन आयोग
Election Commission of India
Home
About Us
Management
FAQ
Contact
About Us
Strengthening Democracy Through Technology
Who are 'we'?
We are a team of passionate computer science students from Central Calcutta Polytechnic, committed to leveraging technology to improve and secure the voting process. Our major project, this website, is designed with a vision to empower citizens, simplify voting, and enhance electoral integrity.
In today's world, ensuring a transparent and fair election is crucial. Recognizing the challenges such as voter fraud, low participation, and accessibility barriers, we set out to create an innovative platform that bridges these gaps.
Our website provides:
A secure and user-friendly system for online voting.
Advanced mechanisms to eliminate fraudulent activities and ensure authenticity.
Easy access to real-time election updates and results.
We believe that technology can strengthen democracy by making elections more efficient, inclusive, and trustworthy. This project represents our dedication to using our skills to address real-world problems and contribute to society.
What Makes Us Unique?
Secure Voting
Ensures a safe and secure voting process.
Fraud Detection
Eliminates fraudulent activities effectively.
Real-Time Updates
Provides live election updates and results.
Accessibility
Ensures easy access for all citizens.
Contact: +91-9875357834 | Email: dipayansarder@gmail.com
Central Calcutta Polytechnic, Convent Road, Kolkata - 700014
© 2025 Election Commission of India. All Rights Reserved.


भारत निर्वाचन आयोग
Election Commission of India
Home
About Us
Management
FAQ
Contact
Contact Us
We'd love to hear from you!
Toll Free Number
1-800-123-4567
Available 24/7
Email Us
info@yourcompany.com
support@yourcompany.com
Social Media
Send Us a Message
Your Name
Email Address
Subject
Message
Send Message
Contact: +91-9875357834 | Email: dipayansarder@gmail.com
Central Calcutta Polytechnic, Convent Road, Kolkata - 700014
© 2025 Election Commission of India. All Rights Reserved.


भारत निर्वाचन आयोग
Election Commission of India
Home
About Us
Management
FAQ
Contact
Myth
VS
Reality
Busted Fakes
Myth vs Reality
FAQs
EVM/VVPAT
ELECTORAL ROLL/ VOTER SERVICES
CONDUCT OF ELECTIONS
OTHERS
The spread of misinformation and fake news has become a significant concern in today's digital age, especially during critical events like elections. In order to ensure a level playing field for all stakeholders, it is vital that misinformation and disinformation does not derail a factually correct narrative.
Recognizing the need to combat the dissemination of false information and ensure the integrity of the electoral process, the Election Commission of India (ECI) has taken a proactive step by publishing a Myth vs Reality Register. This Myth vs Reality register serves as a repository of debunked fake news related to elections in India, providing a reliable resource to verify the authenticity of information regarding the electoral process circulating during election periods.
This Register aims to promote transparency, accuracy, and responsible communication during elections. It serves as a valuable tool for both the general public and media organisations to access verified information and counter false narratives that may mislead voters. By documenting and cataloguing instances of debunked fake news, the register acts as a reference guide, empowering citizens to make informed decisions and maintain the democratic fabric of the nation.
The Register will be periodically updated by ECI after analysing and verifying the authenticity of various claims circulating on different media platforms.
Contact: +91-9875357834 | Email: dipayansarder@gmail.com
Central Calcutta Polytechnic, Convent Road, Kolkata - 700014
© 2025 Election Commission of India. All Rights Reserved.

# E-VOTER CARD DOWNLOAD

📄 **Documentation: Voter Card Download System**
**Module:** Voter ID Generation & Download
**Application:** Online Biometric Voting System (OBVS)
**Version:** 1.0
**Author:** [Group A]
**Last Updated:** [09.06.2024]

🧩 **Overview:**
The Voter Card Download System is an integrated digital solution designed to allow users to retrieve voter information and generate voter ID cards in downloadable PDF format. It utilizes a hybrid technology stack including PHP for the frontend interface, Node.js/Express for backend APIs, MongoDB for data storage, and GridFS for image storage. PDFKit or TCPDF libraries are used for dynamic PDF generation. This module ensures voter data is fetched securely and presented professionally with embedded photos.

✅ **Key Features:**
- Simple and secure HTML form for user input (voter ID, DOB, gender)
- Data validation and processing through REST API using Node.js
- Voter image retrieval from GridFS (stored in binary chunks)
- PDF voter card generation with embedded personal data and photo
- Alternative support for PDF generation using TCPDF in PHP
- CORS-enabled communication between PHP frontend and Node.js backend
- Body-parser usage for handling JSON POST requests
- Error handling for missing images or invalid input
- Well-structured voter card format with text and photo integration
- Supports photo rendering via HTTP endpoint /photo/:id
- Clean backend response with structured JSON object

⚠ **Known Issues / Limitations:**

- No authentication currently implemented (public access only)
- No OTP verification or advanced user verification at this stage
- No admin interface for managing or uploading voter entries
- Photos must already be uploaded into GridFS via admin backend (not part of current module)
- Does not use HTTPS by default (for local/testing use only)

 **Technologies Used:**

- **Frontend:** PHP (voter_card_download.php), HTML
- **Backend:** Node.js + Express
- **Database:** MongoDB
- **File Storage:** GridFS (MongoDB's file storage system)
- **PDF Generation:** PDFKit (Node.js) or TCPDF (PHP alternative)
- **Middleware:** body-parser, cors
- **Hosting Options:** Localhost (XAMPP), Node.js server, or cloud-based (e.g., Heroku, Render)

 **System Architecture:**

**Frontend:** voter_card_download.php (form UI in HTML & PHP)
**Backend:** server.js (Node.js + Express server)
**Data Flow Overview:**

1. User submits form with Voter ID, Date of Birth, and Gender
2. PHP sends POST request to Node.js API /get-voter-details
3. Node.js validates request and returns voter data in JSON
4. PHP uses photo ObjectId from response to request image via /photo/:id
5. PHP embeds the photo and voter data into a formatted PDF using PDFKit or TCPDF

🗂️ **Database Design:**

**MongoDB Collection:** voter_data_collection

Fields:

- name: String
- voter_id: String
- date_of_birth: String
- gender: String
- constituency: String
- photo: GridFS ObjectId

**GridFS Collections Used:**

- fs.files
- fs.chunks

*GridFS stores binary voter photos in 255kB-sized chunks*

🧪 **API Endpoints:**

**Endpoint:** /get-voter-details

**Method:** POST

**Request Payload (JSON):**

```
{
  "voter_id": "VTR123",
  "date_of_birth": "2000-01-01",
  "gender": "Male"
}
```

**Successful Response (JSON):**

```
{
  "success": true,
  "data": {
    "name": "John Doe",
```

```
  "voter_id": "VTR123",
  "gender": "Male",
  "date_of_birth": "2000-01-01",
  "constituency": "XYZ",
  "photo": "6647fa61bce34a6f..."
 }
}
```

🖼️ **Image Retrieval via GridFS:**

**Endpoint:** GET /photo/:id

**Backend Logic:**

```
const readstream = gfs.createReadStream({ _id });
readstream.pipe(res);
```

**Error Handling Includes:**

- Invalid ObjectId format
- Missing or corrupted image files

**Headers:**

- Content-Type: image/jpeg

📄 **PDF Generation:**

**Libraries:**

- Default: PDFKit (Node.js)
- Alternative: TCPDF (PHP-based)

**PDF Content Includes:**

- Voter name, ID, gender, DOB, and constituency
- Voter photo fetched from /photo/:id
- Basic styling (borders, font size, alignment)

**Sample PDF Code Snippet (PHP – TCPDF):**

```
$pdf->Cell(40,10,'Name: ' . $name);

$pdf->Image('http://localhost:3000/photo/'.$photo_id);
```

🛡️ **Security Measures:**

- Sanitization of all form inputs in PHP
- Validation of ObjectId format in Node.js backend
- Enable CORS for secure cross-origin requests
- Use HTTPS protocol in production environments
- Rate limiting to prevent brute-force API usage
- Support for JWT authentication (future scope)

🚀 **Future Improvement:**

- ✅ OTP-based verification for voter authenticity
- ✅ Admin portal for uploading voter records and images
- ✅ QR code embedding in generated voter cards
- ✅ Watermark and certificate-style design for better visual integrity
- ✅ Voter lookup dashboard with filters and search functionality

# VOTING EVM STRUCTURE

# &

# VOTING PROCESS

**Module:** Vote matrix web & Biometric – EVM

**Application:** Online Biometric Voting System (OBVS)
**Version:** 1.0
**Author:** [Group A]
**Last Updated:** [09.06.2024]

## 📜 Overview

The voting process is divided into two parts. In the current voting system, there are two modules: **i) EVM, ii) Polling officer module**. After the polling officer presses the allow button to activate the EVM module and then a voter can vote.
In our voting system, there is no physical module for a polling officer like in previous systems. It requires two things: one is a vote matrix login app on a laptop (the laptop should be provided). And the second one is a particular assigned EVM module for that voting center.

- **Vote matrix website**:
  - it is a secure and restricted website that can be accessed by a valid and registered polling officer. By using the Vote Matrix Officer login app, an officer can log in to the web.
  - It is basically a search engine website that provides a search engine interface. Here officer searches and verifies the voter online, and after clicking confirm voter, the voter data will be transmitted into the EVM. After confirming that a 60-second timer will be displayed on the screen and the voter should vote within the time. After giving a vote, the timer will be stopped, and the page will be refreshed into the search page again.

- **Security & features:**
  - ✓ Restricted access
  - ✓ Live voter data checking
  - ✓ Officer identity verification
  - ✓ browser-specific Web
  - ✓ Connectionless communication with the EVM module.
  - ✓ Restrict double votes

## ➢ EVM module:

- It is a machine that consists of fingerprint sensors for each party, and a separate sensor is allocated.
- One display that will show the vote has been completed
- **Security & features:**
  - ✓ Connectionless communication with the vote matrix
  - ✓ Check biometric only with the provided or confirmed voter (confirmed by the polling officer)
  - ✓ Success votes will be directly sent to the database with voter details (for future inspections and cross-checking)
  - ✓ Every machine will be preset with the specific polling center code

✅ **Key Features**:

- **Voter authenticity:** During the voter search, only registered and authenticated voter data will appear for the voting process.
- **Restrict Double Vote:** Once a voter casts a vote, their data will be restricted for again preceding for vote.
- **Display Double vote warning:** If a voter forcefully tries to cast a vote, they can't do that. A warning will be displayed on the search screen
- **Biometric Votes:** Every vote will be cast after verifying the fingerprint of the voter.
- **Stand-alone build:** The voter search engine or vote matrix and the EVM are both different and not connected modules that work together to complete the voting process

- **Restricted time:** After the polling officer logs into the Vote Matrix, there will be a 24-hour period to use that vote matrix. After the completion of the session time, the officer will be automatically logged out. This ensures that the vote is now completed and further votes can not be done.

## 📐 Technologies Used:

- **Vote Matrix:**
  - Python Flask
  - HTML
  - JavaScript
  - Ajax
  - Flask session
  - CSS
  - MongoDb
- **Biometric EVM:**
  - Rasberrypi
  - Fingerprint Scanners
  - Multi-USB hub
  - USB to TTL converter
  - SH1106 OLED display

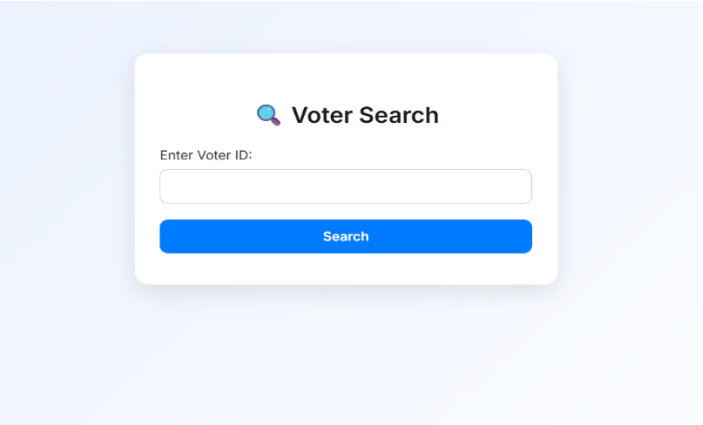
    & many more .....
- **FINGERPRINT-ENABLED VOTING MACHINE :**

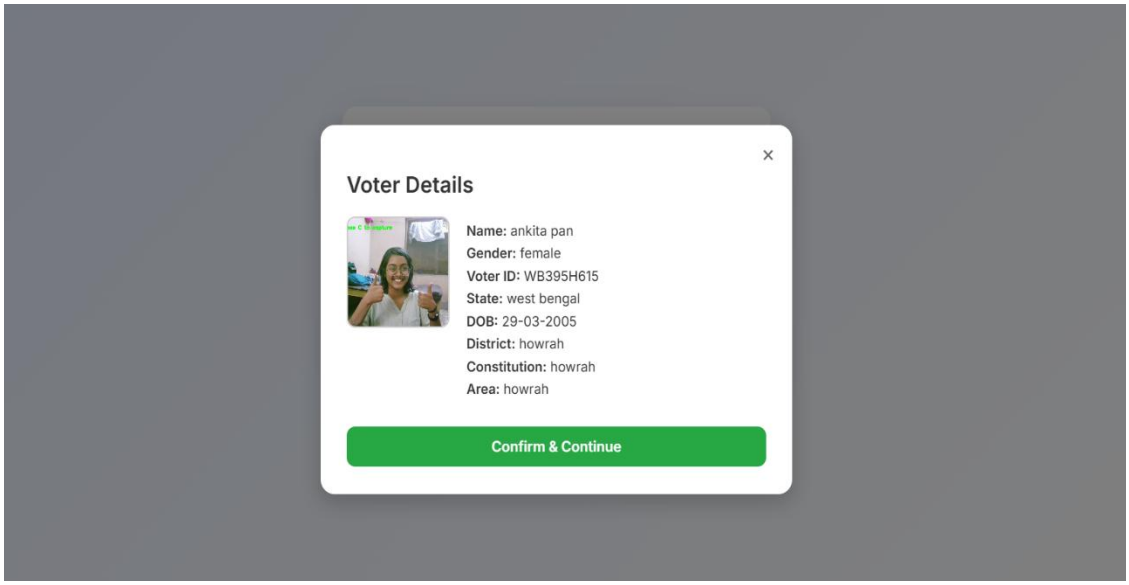
  This part is divided into two different sections

  ( i ) Software Section

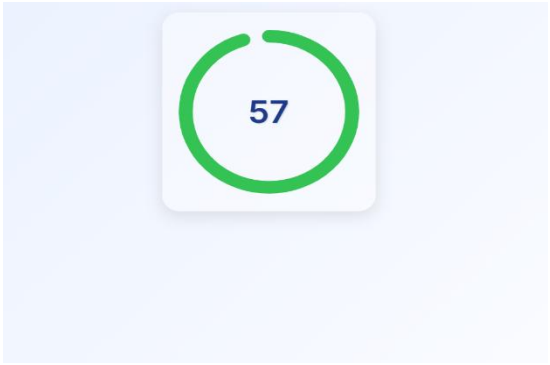
  ( ii ) Hardware Section

## 📷 Page Layout:

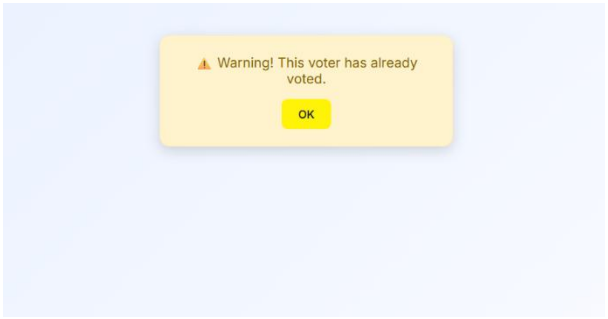




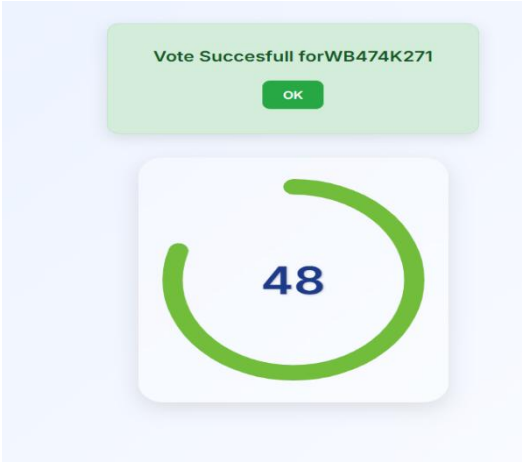

- **After confirm the voter identity the timer will appear**




- **If a voter tries to vote again server will deny permission**




- **If a vote is successfully done**



# SOFWARE COMPONENTS

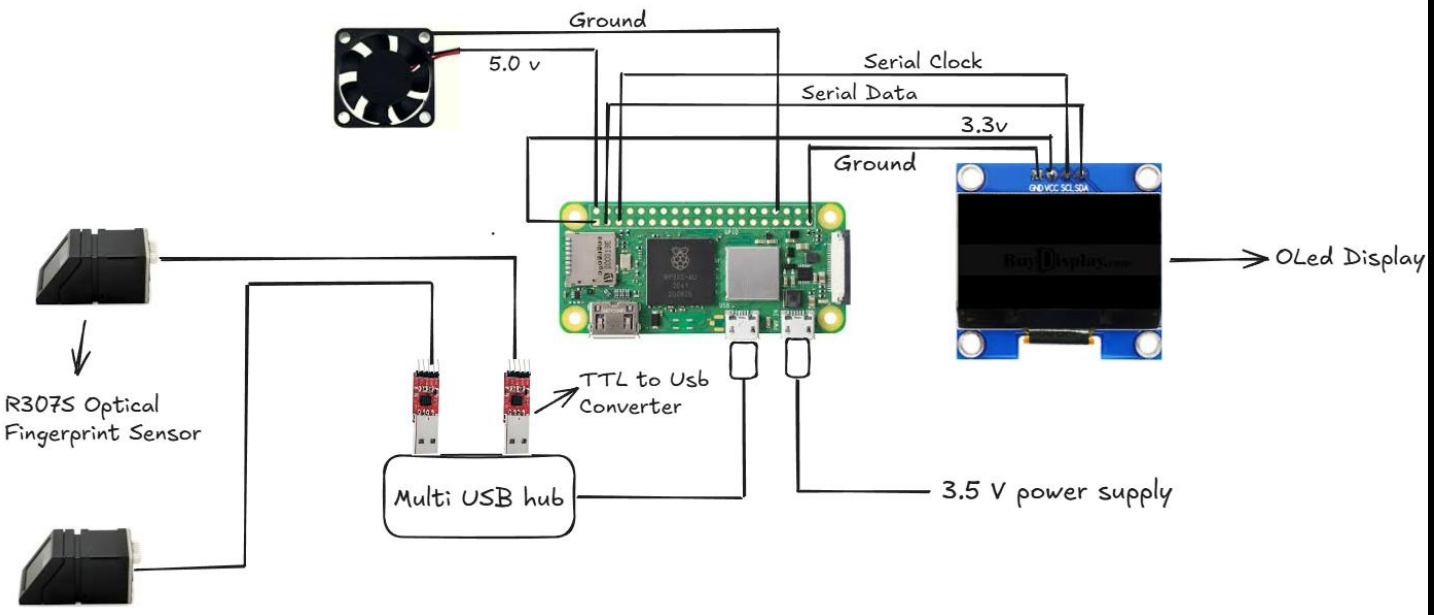
1. **Raspberry Pi OS (Lite or Full)**
   Lightweight Linux-based OS that serves as the base for all operations and interfaces.
2. **Python (3.x)**
   The main programming language used to control the fingerprint sensor, LCD display, and interaction logic.
3. **PyFingerprint Library**
   Python library used to communicate with the R307 fingerprint sensor.
4. **MongoDB Database (Local or Cloud)**
   Stores user details, fingerprints (in array format), voting results, and analytics.
5. **Serial Communication Libraries (like pyserial)**
   Used for communication between Python code and fingerprint hardware over UART.

# HARDWARE COMPONENTS

1. **Raspberry Pi (Model 3/4/Zero 2W)**
   Acts as the central controller for handling sensor inputs and communication with the backend server.
2. **R307 Fingerprint Sensor**
   Used to enrol and verify the identity of each voter using their fingerprint biometric data.
3. **I2C 16x2 LCD Display**
   Displays instructions, status messages, or voter confirmation on the screen.
4. **Breadboard & Jumper Wires**
   Required for building temporary circuit connections between modules and the Raspberry Pi.

5. **Power Supply / Adapter**
   To power the Raspberry Pi and other modules during operation.
6. **USB to TTL Converter**
   To provide the interface between the fingerprint sensors and the Raspberry pi
7. **CPU cooling Fan**
   To keep the Main Raspberry Pi CPU cool and workable.

Here is the Pin diagram with all hardware components mentioned earlier



# ELECTION RESULT SYSTEM

📄 **Documentation: Election Result System**
**Module:** Live Election Analytics Dashboard
**Application:** Online Biometric Voting System (OBVS)
**Version:** 1.0
**Author:** [Group A]
**Last Updated:** [09.06.2024]

🧩 **Overview:**
The Election Result System is a real-time, web-based analytics dashboard that displays live vote counts based on data stored in a MongoDB database. It is designed to provide public and administrative visibility into current election outcomes. The system uses a Node.js + Express backend to retrieve aggregated vote data and serves it via API to a responsive HTML/CSS/JS frontend styled with Bootstrap. The frontend continuously fetches vote data and dynamically updates the vote count per political party, along with a declaration of the winner.

✅ **Key Features:**
- Live vote tally from MongoDB displayed on a responsive web page
- Party-wise aggregation of votes using MongoDB aggregation pipeline
- Frontend styled with Bootstrap 4 for mobile and desktop responsiveness
- Real-time vote count updates using JavaScript fetch()
- Animated winner announcement with emoji confetti and .popper animation
- Tie-breaker logic and conditional winner display
- Separate backend (Node.js/Express) and frontend (HTML/JS) layers
- CORS enabled for cross-port or cross-origin deployment
- Deployed via Netlify (frontend) and Render/Heroku (backend)
- Cloud MongoDB Atlas integration for scalable data storage

⚠ **Known Issues / Limitations:**
- No vote casting mechanism included in this module

• No authentication or access restriction to the result API
• No admin dashboard or geo-level result filters in the current version
• API polling is used instead of real-time WebSocket updates

### 📐 Technologies Used:

• **Frontend:** HTML, CSS, JavaScript, Bootstrap 4
• **Backend:** Node.js, Express.js
• **Database:** MongoDB Atlas (Cloud)
• **Deployment Platforms:**

- Frontend: Netlify, GitHub Pages
- Backend: Render, Vercel, Heroku
  • **Other Tools:** CORS middleware, Mongoose ORM, Environment config via .env

### 🧱 System Architecture:

**Frontend:** index.html – includes party visuals, vote counts, and result display
**Backend:** Election_Result_server.js – Node.js + Express API server
**System Flow:**

1. The backend aggregates vote data from MongoDB
2. Total votes are grouped and counted by party name
3. The frontend fetches the vote result JSON using fetch()
4. Winner section is displayed with animations (confetti and text)

### 🗂️ Database Design:

**Collection:** votes
Each document structure:

```
{ "party": "Party A" }
```

*Votes are stored on a per-vote basis (1 document = 1 vote), allowing flexible querying and analytics.*

### 🧪 API Endpoints:

**Endpoint:** /vote-results
**Method:** GET
**Backend Code (Aggregation Example):**

```
const results = await Vote.aggregate([
 { $group: { _id: "$party", total: { $sum: 1 } } }
]);
```

**Sample Response Format:**

```
{
  "Party A": 10,
  "Party B": 8
}
```

*Returns vote counts grouped by party for live rendering in frontend.*

### 🖥️ Frontend Design (index.html):

Built using:
- Bootstrap 4
- Responsive card-based layout for each political party
- Displays image, candidate name, party name, and vote count

**Winner Section:**
- Conditionally renders winner or tie message
- Includes confetti animation with .popper class
- Javascript logic determines highest vote total

### 🧠 JavaScript Logic (Vote Count & Display):

fetch('http://localhost:3000/vote-results')

- Fetches result JSON and updates DOM vote counts
- Calculates winning party using conditional logic
- If a tie occurs, displays tie message
- Triggers confetti animation if winner is declared

### 🌐 CORS Enablement:

**Backend Code:**

app.use(cors());

- Allows frontend (e.g., hosted on port 5500) to access API on port 3000
- Mandatory for cross-domain or local development setups

### 🔧 Express Server Configuration:

**Initialization Code:**

const app = express();

app.listen(PORT, () => console.log(`Server running`));

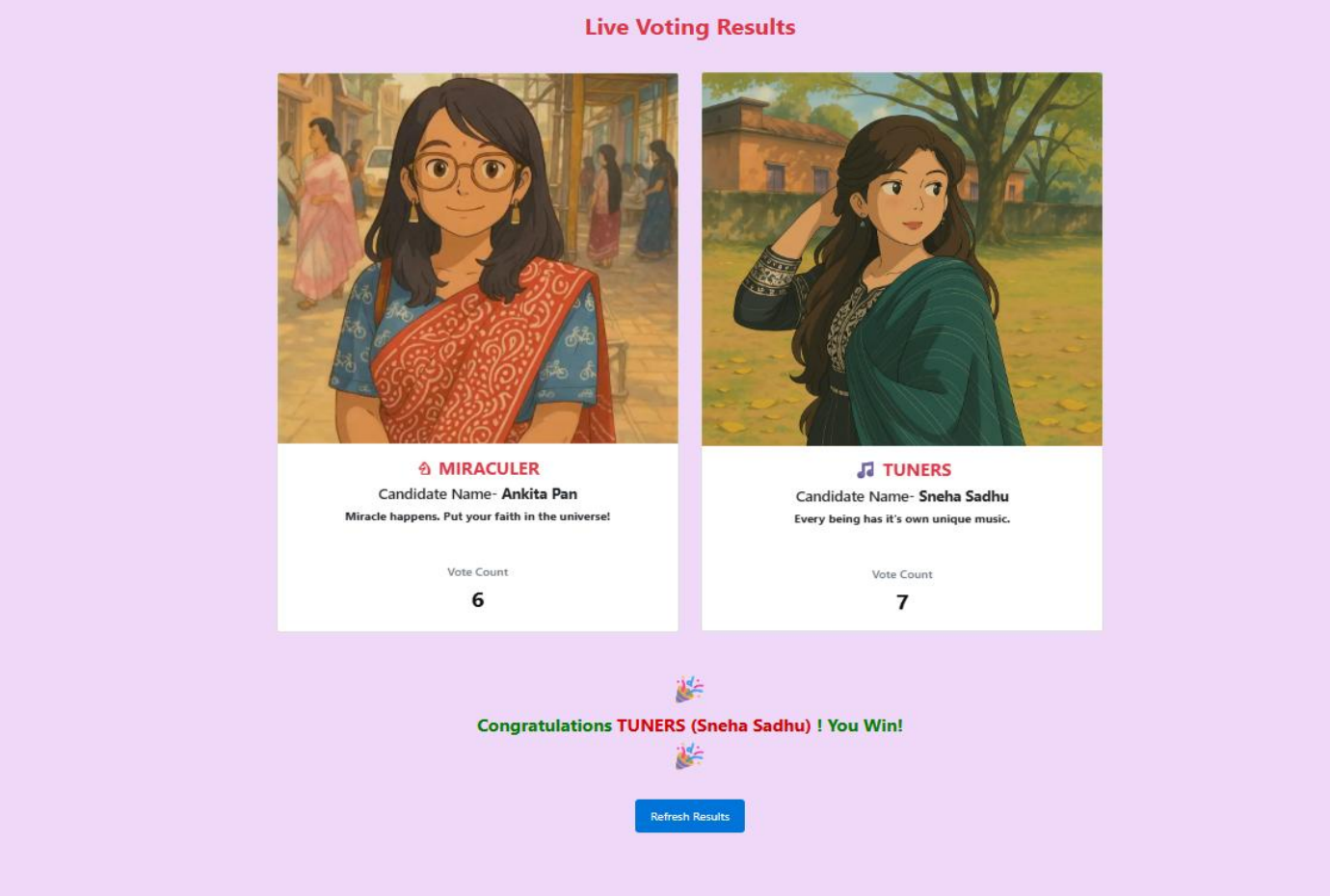
- Middleware used: cors()
- Database connection handled via Mongoose
- MongoDB URI secured using .env config file
- IP allowlisting recommended on MongoDB Atlas

### 🚀 Deployment Checklist:

- Frontend hosted on **Netlify** or **GitHub Pages**
- Backend deployed on **Render**, **Vercel**, or **Heroku**
- Environment variables managed through .env files
- MongoDB URI hidden in server files and restricted via **IP allowlisting**

📈 **Future Improvement:**

- ✅ Introduce real-time updates using WebSocket (e.g., Socket.IO)
- ✅ Integrate a secure vote casting module to simulate end-to-end flow
- ✅ Build an admin analytics dashboard for election authorities
- ✅ Add geo-location-based vote breakdown (e.g., by state or district)
- ✅ Export functionality for results in Excel or PDF format



# FUTURE IMPLEMENTATION

## ➢ Biometric Pattern Matching Algorithms:

**Understanding:**
Our system uses a fingerprint template for user authentication. These templates are stored as mathematical feature vectors (not actual images).

**Future Tech Upgrade:**

✅ Replace basic matching with **minutiae-based matching using neural networks**

✅ Use **OpenCV + CNN** to improve fingerprint image quality and pattern accuracy

✅ Implement **adaptive thresholding** for noisy fingerprints (elderly users)

**Impact:**
Enhances accuracy & reduces false accept/reject rate (FAR/FRR)

## ➢ Decentralized Voting via Blockchain:

**Understanding:**
Currently, votes are stored in a centralized MongoDB database, which can be tampered with if security is breached.

**Future Tech Upgrade:**

✅ Use the **Ethereum/Solana blockchain** to store each vote as a **transaction block**

✅ Every voter ID and vote hash is stored immutably

**Impact:**
Votes become **tamper-proof, traceable, and auditable** with no central point of failure

## ➢ Aadhaar / eKYC Integration

**Understanding:**
Currently, authentication is isolated to local DB fingerprint records.

**Future Tech Upgrade:**

- ✅ Link fingerprint or mobile verification to **UIDAI Aadhaar eKYC API**
- ✅ OTP and biometric validation can be cross-verified with government records

**Impact:**
Government-level security and legitimacy of voting

## ➢ Mesh Network for Multi-Booth Connectivity:

**Understanding:**
Each Raspberry Pi currently works independently and needs manual sync.

**Future Tech Upgrade:**

- ✅ Build a **LoRa or ESP-NOW based mesh network**
- ✅ All booths communicate and sync with a central Raspberry Pi master

**Impact:**
Scalable voting booths across large campuses with **wireless synchronization.**

## ➢ Multi-Factor Authentication:

Combine **fingerprint verification** with **OTP (One-Time Password)** sent to the voter's registered mobile number or Aadhaar-linked device.
This adds an extra layer of security, especially useful in high-stake elections.

## ➢ Face Recognition Integration:

**Add a camera module to include facial biometric verification along with the fingerprint.This supports multimodal biometrics, reducing false positives or spoofing attempts.**

# CONCLUSION

Our project ensures many features that improve the EVM-based election system.

The improvements are:

- Improved Voter data security
- Digital voter card
- Polling officer verification and authentication
- No unregistered voter is allowed to cast a vote
- Biometric authentication during voting
- Live identity checking
- Cryptographic Data Encryption
- Stand-alone modulation
- Restricted period for Voting
- Digital result publishing

# References:


- Fingerprint: https://robocraze.com/products/r307-optical-fingerprint-sensor-module?_pos=1&_psq=R307s&_ss=e&_v=1.0
- Raspberry Pi Zero 2w : https://robocraze.com/products/raspberry-pi-zero-2-w?_pos=35&_sid=6d619552f&_ss=r
- Google Chrome
- ChatGPT
- DeepSeek R1
- Replit
- npm: https://docs.npmjs.com/
- mongoose : https://mongoosejs.com/docs/guide.html
- express : https://expressjs.com/
- flet : https://flet.dev/blog