POST

Create Voter

Register a new voter with unique ID, name, and age validation (minimum 18 years).

POST /api/voters Copy

Request Body ▼

Copy

```
{
  "voter_id": 1,
  "name": "Alice",
  "age": 22
}
```

Success Response ▼

218 Created

Copy

```
{
  "voter_id": 1,
  "name": "Alice",
  "age": 22,
  "has_voted": false
}
```

Error Response ▼

409 Conflict

Copy

```
{
  "message": "voter with id: 1 already exists"
}
```

GET

Get Voter Info

Retrieve detailed information about a specific voter by ID.

GET /api/voters/{voter_id} Copy

Success Response ▼

222 Found

Copy

```
{
  "voter_id": 1,
  "name": "Alice",
  "age": 22,
  "has_voted": false
}
```

Error Response ▼

417 Expectation Failed

Copy

```
{
  "message": "voter with id: 5 was not found"
}
```

GET

List All Voters

Retrieve a complete list of all registered voters.

GET /api/voters Copy

Success Response ▼

223 Listed

Copy

```
{
  "voters": [
    {
      "voter_id": 1,
      "name": "Alice",
```

```
      "age": 22
    },
    {
      "voter_id": 2,
      "name": "Bob",
      "age": 30
    }
  ]
}
```

PUT

Update Voter Info

Update existing voter information with age validation (minimum 18 years).

PUT /api/voters/{voter_id} Copy

Request Body ▼

Copy

```
{
  "name": "Alice Smith",
  "age": 25
}
```

Success Response ▼

224 Updated

Copy

```
{
  "voter_id": 1,
  "name": "Alice Smith",
  "age": 25,
  "has_voted": false
}
```

Error Response ▼

422 Unprocessable Entity

Copy

```
{
  "message": "invalid age: 16, must be 18 or older"
}
```

DELETE

Delete Voter

Remove a voter from the system.

DELETE /api/voters/{voter_id} Copy

Success Response ▼

225 Deleted

Copy

```
{
  "message": "voter with id: 1 deleted successfully"
}
```

POST

Register Candidate

Register a new candidate for the election.

POST /api/candidates Copy

Request Body ▼

Copy

```
{
  "candidate_id": 1,
  "name": "John Doe",
  "party": "Green Party"
}
```

Success Response ▼

226 Registered

Copy

```
{
  "candidate_id": 1,
  "name": "John Doe",
  "party": "Green Party",
  "votes": 0
}
```

GET

List Candidates

Get all registered candidates.

GET /api/candidates Copy

Success Response ▼

227 Listed

Copy


```
{
  "candidates": [
    {
      "candidate_id": 1,
      "name": "John Doe",
      "party": "Green Party"
    },
    {
      "candidate_id": 2,
      "name": "Jane Roe",
      "party": "Red Party"
    }
  ]
```

```
}
```

POST

Cast Vote

Cast a vote for a candidate with validation to prevent duplicate voting.

POST /api/votes Copy

Request Body ▼

Copy

```
{
  "voter_id": 1,
  "candidate_id": 2
}
```

Success Response ▼

228 Voted

Copy

```
{
  "vote_id": 101,
  "voter_id": 1,
  "candidate_id": 2,
  "timestamp": "2025-09-10T10:30:00Z"
}
```

Error Response ▼

423 Locked

Copy

```
{
  "message": "voter with id: 1 has already voted"
}
```

GET

Get Candidate Votes

Get the vote count for a specific candidate.

GET /api/candidates/{candidate_id}/votes Copy

Success Response ▼

229 Counted

Copy


```
{
  "candidate_id": 2,
  "votes": 45
}
```


GET

Filter Candidates by Party

Filter candidates by political party.

GET /api/candidates?party={party_name} Copy

Success Response ▼

230 Filtered

Copy

```
{
  "candidates": [
    {
      "candidate_id": 1,
      "name": "John Doe",
      "party": "Green Party"
    }
  ]
}
```

GET

Voting Results (Leaderboard)

Get the complete voting results with candidates ranked by vote count.

GET /api/results Copy

Success Response ▼

231 Results

Copy

```
{
  "results": [
    {
      "candidate_id": 2,
      "name": "Jane Roe",
      "votes": 45
    },
    {
      "candidate_id": 1,
      "name": "John Doe",
```

```
      "votes": 30
    }
  ]
}
```

GET

Winning Candidate

Get the winning candidate(s), handling ties appropriately.

GET /api/results/winner Copy

Success Response (Tie Example) ▼

232 Winner

Copy

```
{
  "winners": [
    {
      "candidate_id": 1,
      "name": "John Doe",
      "votes": 40
    },
    {
      "candidate_id": 2,
```

```
      "name": "Jane Roe",
      "votes": 40
    }
  ]
}
```

GET

Vote Timeline

Get the timeline of votes for a specific candidate.

GET /api/votes/timeline?candidate_id={id} Copy

Success Response ▼

233 Timeline

Copy

```
{
  "candidate_id": 2,
  "timeline": [
    {
      "vote_id": 101,
      "timestamp": "2025-09-10T10:30:00Z"
    },
    {
      "vote_id": 102,
      "timestamp": "2025-09-10T10:32:00Z"
    }
  ]
}
```

POST

Conditional Vote Weight

Cast a weighted vote based on voter profile update status.

POST /api/votes/weighted Copy

Request Body ▼

Copy

```
{
  "voter_id": 1,
  "candidate_id": 2
}
```

Success Response ▼

234 Weighted

Copy

```
{
  "vote_id": 201,
  "voter_id": 1,
  "candidate_id": 2,
  "weight": 2
}
```

GET

Range Vote Queries

Get votes for a candidate within a specific time range.

GET /api/votes/range?candidate_id={id}&from={t1}&to={t2} Copy

Success Response ▼

235 Range

Copy

```
{
  "candidate_id": 2,
  "from": "2025-09-10T10:00:00Z",
  "to": "2025-09-10T12:00:00Z",
  "votes_gained": 42
}
```

Error Response ▼

424 Failed Dependency

Copy

```
{
  "message": "invalid interval: from > to"
}
```

POST

End-to-End Verifiable Encrypted Ballot

Accept encrypted ballots with zero-knowledge proofs and nullifiers to prevent double voting.

POST /api/ballots/encrypted Copy

Request Body ▼

Copy

```
{
  "election_id": "nat-2025",
  "ciphertext": "base64(Paillier_or_ElGamal_cipher)",
  "zk_proof": "base64(Groth16_or_Plonk_proof)",
  "voter_pubkey": "hex(P-256)",
  "nullifier": "hex(keccak256(signal))",
  "signature": "base64(Ed25519 signature over payload)"
}
```

Success Response ▼

236 Encrypted

Copy

```
{
  "ballot_id": "b_7f8c",
  "status": "accepted",
  "nullifier": "0x4a1e...",
  "anchored_at": "2025-09-15T08:30:00Z"
}
```

Error Response ▼

425 Too Early

Copy

```
{
  "message": "invalid zk proof"
}
```

POST

Homomorphic Tally With Verifiable Decryption

Tally encrypted ballots without decryption and publish verifiable results.

POST /api/results/homomorphic Copy

Request Body ▼

Copy

```
{
  "election_id": "nat-2025",
  "trustee_decrypt_shares": [
    { "trustee_id": "T1", "share": "base64(...)", "proof":
"base64(NIZK)" },
    { "trustee_id": "T3", "share": "base64(...)", "proof":
"base64(NIZK)" },
    { "trustee_id": "T5", "share": "base64(...)", "proof":
"base64(NIZK)" }
  ]
}
```

Success Response ▼

237 Tallied

Copy

```
{
  "election_id": "nat-2025",
  "encrypted_tally_root": "0x9ab3...",
  "candidate_tallies": [
    { "candidate_id": 1, "votes": 40321 },
    { "candidate_id": 2, "votes": 39997 }
  ],
  "decryption_proof":
"base64(batch_proof_linking_cipher_aggregate_to_plain_counts)",
  "transparency": {
    "ballot_merkle_root": "0x5d2c...",
    "tally_method": "threshold_paillier",
```

```
    "threshold": "3-of-5"
  }
}
```

POST

Differential-Privacy Analytics

Permit aggregate queries with differential privacy noise and budget tracking.

POST /api/analytics/dp Copy

Request Body ▼

Copy

```
{
  "election_id": "nat-2025",
  "query": {
    "type": "histogram",
    "dimension": "voter_age_bucket",
    "buckets": [ "18-24", "25-34", "35-44", "45-64", "65+" ],
    "filter": { "has_voted": true }
  },
  "epsilon": 0.5,
  "delta": 1e-6
}
```

Success Response ▼

238 Private

Copy


```
{
  "answer": {
    "18-24": 10450,
    "25-34": 20110,
    "35-44": 18001,
    "45-64": 17320,
    "65+": 9022
  },
  "noise_mechanism": "gaussian",
  "epsilon_spent": 0.5,
  "delta": 1e-6,
  "remaining_privacy_budget": { "epsilon": 1.0, "delta": 1e-6 },
  "composition_method": "advanced_composition"
}
```


POST

Ranked-Choice / Condorcet (Schulze)

Submit ranked ballots and compute Schulze winner(s) with full audit trail.

POST /api/ballots/ranked Copy

Request Body ▼

Copy

```
{
  "election_id": "city-rcv-2025",
  "voter_id": 123,
  "ranking": [3, 1, 2, 5, 4],
  "timestamp": "2025-09-10T10:15:00Z"
}
```

Success Response ▼

239 Ranked

Copy

```
{
  "ballot_id": "rb_2219",
  "status": "accepted"
}
```

POST

Risk-Limiting Audit (RLA)

Produce ballot-polling audit plan with Kaplan-Markov sequential test.

POST /api/audits/plan Copy

Request Body ▼

Copy

```
{
  "election_id": "nat-2025",
  "reported_tallies": [
```

```
    { "candidate_id": 1, "votes": 40321 },
    { "candidate_id": 2, "votes": 39997 }
  ],
  "risk_limit_alpha": 0.05,
  "audit_type": "ballot_polling",
  "stratification": { "counties": ["A","B","C"] }
}
```

Success Response ▼

240 Audited

Copy

```
{
  "audit_id": "rla_88a1",
  "initial_sample_size": 1200,
  "sampling_plan": "base64(csv of county proportions and random
seeds)",
  "test": "kaplan-markov",
  "status": "planned"
}
```