



Create Voter

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

POST /api/voters Copy

```
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

Сору

{

"message": "voter with id: 1 already exists"





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
Сору
 "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

```
Copy
{
    "name": "Alice Smith",
    "age": 25
}
Success Response ▼
```





```
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 Voter

Remove a voter from the system.

DELETE /api/voters/{voter_id} Copy

Success Response ▼

DELETE

```
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
```



List Candidates

Get all registered candidates.

GET /api/candidates Copy



}

POST

Cast Vote

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

```
POST /api/votes Copy
Request Body ▼
Сору
{
 "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
Сору
  "message": "voter with id: 1 has already voted"
}
```





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





Filter Candidates by Party

Filter candidates by political party.

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

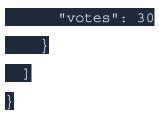
Success Response ▼

230 Filtered

Voting Results (Leaderboard)

GET

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





GET

Winning Candidate

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

```
"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"

    }

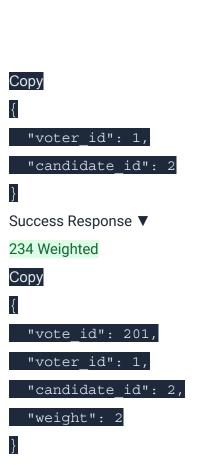
]
```

POST

Conditional Vote Weight

Cast a weighted vote based on voter profile update status.

POST /api/votes/weighted Copy









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

```
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"
}
```

Homomorphic Tally With Verifiable Decryption

Tally encrypted ballots without decryption and publish verifiable results.

```
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)" }
1
}
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"
}
```

Differential-Privacy Analytics

Permit aggregate queries with differential privacy noise and budget tracking.

POST /api/analytics/dp Copy

Request Body ▼

Success Response ▼

```
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
}
```





```
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"
}
```

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", <u>"voter_id": 123,</u> "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"
}
```

