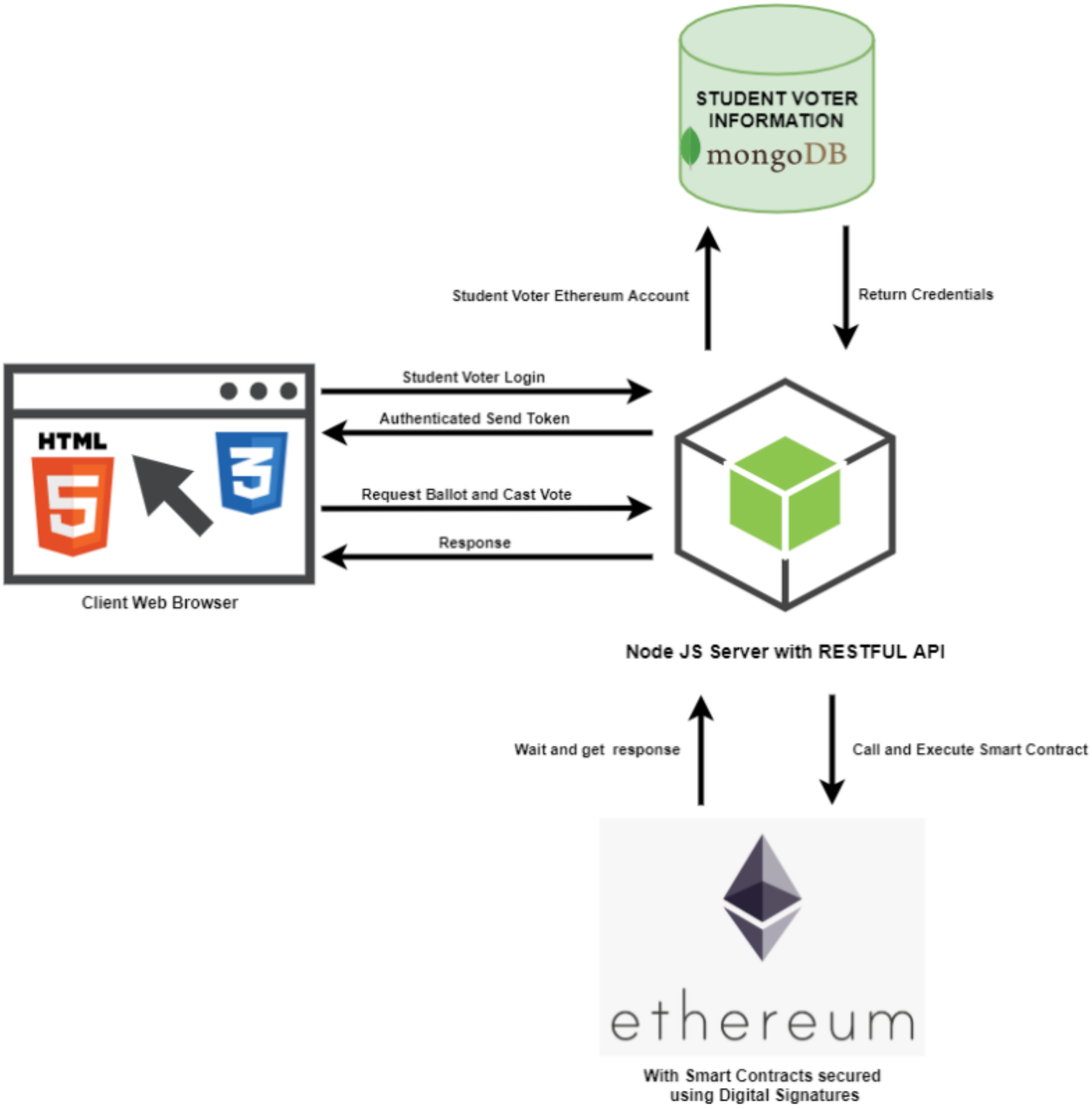
Application Overview

A database of student voters will be prepared beforehand using MongoDB as the database management system which will also serve to authenticate the validity of a student to cast their vote. The web application developed with HTML5, CSS3, JavaScript, and Handlebars will allow its users to authenticate themselves as well as cast their votes for the ballot. These requests will be handled by a Node.js server that implements RESTful API to authenticate student login and voting status, and Web3.js to interact with the Ethereum node which then requests a transaction to the Ethereum network and subsequently add the transaction to the block once validated by a smart contract. The node then returns the status of the transaction to the server which then updates the web application interface to display the transaction status on the client's side.


STUDENT VOTER INFORMATION
mongoDB
Student Voter Ethereum Account
Return Credentials
Student Voter Login
Authenticated Send Token
Request Ballot and Cast Vote
Response
HTML
Client Web Browser
Node JS Server with RESTFUL API
Wait and get response
Call and Execute Smart Contract
ethereum
With Smart Contracts secured using Digital Signatures

Roles

- Voter
- Election Administrator

Application Decomposition

Trust Boundaries

- The perimeter firewall.
- The database server trusts calls from the web application's identity.
- The web application sends transaction to the public Ethereum blockchain network.
- The web application includes dependencies such as secure password generators.

Data Flows

1. Voter Authentication
   1.1. The voter logs in to the system using the credentials provided by the election administrator.
   1.2. If the credentials entered are not valid, do not allow the voter to proceed and go back to step 1, else proceed to step 3.
   1.3. The application renders the voter dashboard.

2. Election Administrator Authentication
   2.1. The administrator logs in to the system.
   2.2. If the administrator's login credentials are invalid, display an error message and go back to step 1, else, proceed to step 3.
   2.3. The application renders the administrator dashboard.

3. Voter Registration
   3.1. If the administrator intends to create a single voter account, proceed to step 2, else load batch voter information .CSV file to the application and proceed to step 3.
   3.2. Administrator fills up the voter account creation form.
   3.3. The application generates a password for the voters/voter using a dependency called password-generator.js from Node.js.
   3.4. The application generates public and private keys for each voter using jsrsasign.js.
   3.5. The application enables the administrator to export the credentials, passwords, and keys.
   3.6. The application hashes the passwords using PBKDF2withHmacSHA256.
   3.7. The application encrypts the private key of the voter/voters using password-based encryption (PKCS #5).
   3.8. The application stores all the credentials and information to MongoDB.

4. Candidate Registration
   4.1. If the administrator intends to register a single candidate, proceed to step 2, else load candidates' information from a .CSV file to the application and proceed to step 3.
   4.2. The administrator fills up the candidate registration form.
   4.3. The system writes all the candidates' information to MongoDB.

5. Vote Casting
   5.1. The controller sends the filtered ballot to the voter based on their college and batch.
   5.2. Voter answers the ballot and submits it.
   5.3. The application requires the voter to re-authenticate using their password.
   5.4. If the password is correct, proceed to step 5, else, notify the voter with an error message then go back to step 3.
   5.5. The controller retrieves the encrypted private key and public key of the voter from the database.
   5.6. The controller decrypts the private key which will then be used to generate the digital signature.
   5.7. The controller generates a JSON file from the original ballot submitted by the voter.
   5.8. The controller hashes the original ballot to create a message digest.
   5.9. The controller encrypts the obtained message digest using the voter's private key to create a signed ballot.
   5.10. The controller encodes the signed ballot using Base64 format.
   5.11. The controller forwards the signed ballot, original ballot, public key, and Base64 encoding of the signed ballot to the election module.
   5.12. The controller sends the transaction to be recorded in the blockchain.