Tab 1

Codex of Collaboration

This is a temporary doc to log contributions

### **Aryan Pansare**

**(16-11-2024)** :

* Created and uploaded **Election.sol** smart contract to handle elections.
* Features added:
  + **Admin Role**:
    - Admins can host elections by specifying details such as the organization name, title, and other relevant information.
    - Admins have the authority to:
      * Add candidates by specifying their name and campaign slogan.
      * Verify voter eligibility based on provided details.
      * Start and stop voting and set the voting duration.
  + **Candidate Management**:
    - Candidate details, including name and slogan, are stored in an efficient mapping structure for quick access and management.
  + **Voter Management**:
    - Voters must register by providing their name, address, and phone number.
    - Verification by the admin is required before voters are eligible to participate.
    - Voter information is stored securely in a mapping structure.
  + **Voting Process**:
    - Verified voters can cast their vote for a candidate of their choice within the designated voting period.
    - Voters can check:
      * Their voting eligibility status.
      * The voting status (active/inactive).
      * The start and end times of the voting period.
* **Technologies Used:** Solidity, Ethereum, Remix IDE
* **Key Takeaways:**
  + Learned about Blockchain and Web3 Applications. Gained hands-on experience in writing robust smart contracts using Solidity.
  + Designed and implemented an efficient mapping data structure to manage real-time election data, including voters and candidates.
  + Conducted extensive testing of the smart contract on an Ethereum test network hosted on virtual machines via the Remix platform.

### **Chaitanya Kumar (10-11-2024):**

### Created a GitHub repository to manage the decentralized voting system project.

### Designed and developed a responsive landing page for the project website with Ayush Agarwal . **(16-11-2024):**

* **Enhanced and Updated a Decentralized Voting System Smart Contract**
  + Improved the **Election.sol** smart contract by implementing additional features and optimizing existing logic for better functionality and security.
  + Added the **declareWinner** functionality to determine and return the election winner with their ID, name, and vote count after the voting ends.
  + Created the **getDetails** method to fetch specific candidate details, such as their name and vote count, by index.
  + Introduced state management modifiers like **onlyWhenVotingActive** and **onlyWhenVotingEnded** to enforce contract rules during different phases of the election.
  + Incorporated **event logging** for actions like candidate addition, voter registration, vote casting, and winner declaration to ensure traceability and transparency.
  + Enhanced **security** by:
    - Preventing overwriting of existing voters and candidates.
    - Adding checks for valid candidate IDs and voter registration/verification statuses.
  + Improved the readability and maintainability of the smart contract through modularization and better organization of logic.
* **Technologies Used**: Solidity, Ethereum Blockchain, Remix IDE
* **Key Takeaways**:
  + Gained experience in **smart contract security** and Solidity best practices.
  + Implemented features to handle real-world voting scenarios while ensuring the integrity of data on the blockchain.
  + Successfully tested the updated contract on Remix IDE in a simulated Ethereum environment.

### **(17-11-2024):**

* Developed comprehensive technical documentation for the decentralized voting system's smart contract, providing detailed explanations for each line of code.