**Title: Simple Voting System**

**Tool Details:**

* **Backend:** Node.js with Express.js
* **Frontend:** Web Components (vanilla JavaScript, no frameworks)
* **Database:** Simulated in-memory storage (or optional database like MongoDB)
* **AI Integration:** Store and count votes dynamically

**Goal:** Candidates will learn how to build an Express.js backend that collects and counts votes and integrate it with a WebComponent-based frontend. They will practice API request handling, user input validation, and real-time vote updates.

**Assignment Description:** Candidates will create a Simple Voting System where users can cast votes for predefined options. The backend will handle vote submissions, update the vote count, and return the current vote tally. The frontend will dynamically display the voting interface and update results in real time.

**Tasks & Steps:**

1. **Backend Development:**
   * Set up an Express.js server.
   * Create an API endpoint to receive votes and update the vote count.
   * Store votes in memory (or use a database like MongoDB for persistence).
   * Provide an endpoint to retrieve the current vote tally.
2. **Frontend Development:**
   * Create a Web Component with buttons to vote for different options.
   * Send API requests when users submit votes.
   * Retrieve and display updated vote counts dynamically.
3. **Result Display:**
   * Show voting options and their respective counts.
   * Update vote counts dynamically when new votes are cast.

**Mathematical Calculation/Steps:**

* Store vote counts for each option in an object or database.
* Increment the count of the selected option when a vote is received.
* Return the updated vote tally when requested.

**Third-Party Packages (if required):**

* express (for backend server)
* cors (to allow frontend-backend communication)
* body-parser (to handle JSON requests)
* mongoose (optional, if using MongoDB for persistent storage)

**Acceptance Criteria:**

* The backend should correctly receive and store votes.
* The frontend should capture user input and dynamically update the UI.
* The API should return an accurate vote count.
* The Web Component should encapsulate voting logic and display updates.
* Users should see real-time updates when new votes are cast.

**Submission Guidelines:**

1. Fork the provided repository.
2. Create a folder with your name inside the repo.
3. Implement both backend and frontend in the designated folder.
4. Push the completed code to your forked repo.
5. Submit a pull request for review.

The assignment is complete when the backend correctly processes votes, maintains vote counts, and the frontend successfully updates the UI based on API responses.