**Title:** Scientific Calculator

**Tool Details:**

* **Backend:** Node.js with Express.js
* **Frontend:** Web Components (Vanilla JavaScript, HTML, CSS)
* **Database (if needed):** None (stateless calculation)

**Goal:**  
By completing this assignment, candidates will gain hands-on experience in developing a full-stack application, integrating an Express.js backend with a WebComponent-based frontend. They will also learn how to dynamically generate AI-driven calculations and process mathematical formulas through API calls.

**Assignment Description:**  
Candidates will develop a scientific calculator application where the frontend consists of a WebComponent-based form. Users will input mathematical expressions, and the backend will process these calculations using JavaScript’s built-in math functions. The backend should return the computed results dynamically. AI will generate example calculations and test cases to validate the system.

**Tasks & Steps:**

1. **Backend API Development (Express.js)**
   * Set up an Express.js server.
   * Create an API endpoint (/calculate) to process user input.
   * Implement logic to parse mathematical expressions securely.
   * Return the result as a JSON response.
2. **Frontend Integration (Web Components)**
   * Create a WebComponent-based UI with an input form for users to enter expressions.
   * Add a button to send the input to the backend.
   * Display the calculated result dynamically.
   * Ensure responsive design for different screen sizes.
3. **Result Display & Error Handling**
   * Display calculated results in real-time.
   * Handle invalid input cases with proper error messages.
   * Implement loading indicators for better user experience.

**Mathematical Calculation/Steps:**

* Candidates must support basic arithmetic (+, -, \*, /) and advanced functions (sin, cos, tan, log, sqrt, power, factorial).
* The backend should securely evaluate user inputs, avoiding security risks like eval().
* Example Expressions: 5+3, sqrt(16), log(100), sin(45), 3^4.

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

* express (for backend API)
* mathjs (for mathematical computation)
* cors (for enabling frontend-backend communication)

**Acceptance Criteria:**

* Backend API correctly processes and returns calculated results.
* WebComponent-based frontend successfully integrates with the backend.
* The UI is user-friendly, responsive, and displays results correctly.
* Proper error handling for invalid inputs and edge cases.
* AI-generated test cases validate correctness.

**Submission Guidelines:**

1. Fork the provided repository.
2. Create a new folder with your name (username\_scientific\_calculator).
3. Implement the backend and frontend within the folder.
4. Push the code to your forked repository.
5. Submit a pull request with a description of your implementation.

**Reference:**  
[Calculator.net](https://www.calculator.net/)