**Title: Image Size Compressor**

**Tool Details:**

**Technology Stack:**

* **Frontend:** Web Components (Lit or Vanilla JS)
* **Backend:** Express.js (Node.js)
* **Processing:** Image compression and resizing
* **Data Handling:** File uploads & API requests

**Goal:**  
By completing this assignment, candidates will:

* Learn how to build a **RESTful API** using **Express.js**.
* Implement **image compression and resizing** techniques.
* Work with **Web Components** to create an interactive frontend.
* Understand **client-server communication** via API calls.
* Gain experience in handling **file uploads and transformations**.

**Assignment Description:**

Develop an **Image Size Compressor** where users can upload an image via a **WebComponent-based form**. The backend, built with **Express.js**, will compress and resize the image based on user-selected settings and return the optimized file. The frontend should display the **original and compressed image details** along with a **download link**.

**Tasks & Steps:**

**1. Backend API Development (Express.js):**

* Set up an **Express.js** server to handle **image uploads**.
* Implement **image compression** to reduce file size while maintaining quality.
* Implement **image resizing** options (e.g., **small, medium, large**).
* Provide an API route that accepts an image, compresses/resizes it, and returns the optimized file.

**2. Frontend (WebComponent-based UI):**

* Create a **form** using Web Components that allows users to **upload an image**.
* Provide options for **image compression quality** (e.g., Low, Medium, High).
* Provide options for **image resizing** (e.g., 50%, 75%, 100%).
* Send the uploaded image and user-selected options to the backend using the **fetch API**.
* Display **before & after file sizes** and provide a **download link** for the compressed image.

**3. Integration & Testing:**

* Ensure the frontend **properly communicates** with the backend.
* Handle errors gracefully (e.g., **unsupported image formats, very large files**).
* Test the **image compression and resizing process** to ensure it works efficiently.

**Mathematical Calculation/Steps (if applicable):**

* **Compression Ratio Formula:**  
  CR=Original Image SizeCompressed Image SizeCR = \frac{\text{Original Image Size}}{\text{Compressed Image Size}}
* **Resizing Formula:**  
  New\_Width=Original\_Width×Scaling\_FactorNew\\_Width = Original\\_Width \times Scaling\\_Factor  
  New\_Height=Original\_Height×Scaling\_FactorNew\\_Height = Original\\_Height \times Scaling\\_Factor

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

* express (for backend server)
* multer (for handling file uploads)
* sharp (for image compression and resizing)
* lit (for WebComponent development)

**Acceptance Criteria:**

* The **Express.js backend** should successfully **compress and resize** uploaded images.
* The **WebComponent-based frontend** should have a **responsive form** for image upload and settings selection.
* Proper **error handling** should be in place for **unsupported image formats**.
* The application should work seamlessly across **modern web browsers**.

**Submission Guidelines:**

1. **Fork** the provided GitHub repository.
2. **Create a folder** named image-size-compressor-<your-name>.
3. **Implement the backend and frontend** in the respective subfolders.
4. **Push the code** to your forked repository.
5. **Submit a pull request** with a brief description of your implementation.

**Ensure that the backend correctly processes the image and integrates seamlessly with the WebComponent-based frontend.**