**Title: Audio Compressor**

**Tool Details:**

**Technology Stack:**

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

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

* Learn how to build a **RESTful API** using **Express.js**.
* Implement **audio compression** to reduce file size while maintaining quality.
* Work with **Web Components** to create an interactive frontend.
* Understand **client-server communication** via API calls.
* Gain experience in handling **file uploads and audio processing**.

**Assignment Description:**

Develop an **Audio Compressor** where users can upload an **audio file** via a **WebComponent-based form**. The backend, built with **Express.js**, will process the file to reduce its size while maintaining a good balance between **quality and file size**. The frontend should display **before & after file sizes** and provide a **download link** for the compressed audio file.

**Tasks & Steps:**

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

* Set up an **Express.js** server to handle **audio file uploads**.
* Implement **audio compression** using an appropriate library.
* Provide an API route that **accepts an audio file, compresses it, and returns the optimized file**.

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

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

**3. Integration & Testing:**

* Ensure the frontend **properly communicates** with the backend.
* Handle errors gracefully (e.g., **unsupported file types, large files**).
* Test the **audio compression process** to ensure it works efficiently.

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

* **Compression Ratio Formula:**  
  CR=Original Audio SizeCompressed Audio SizeCR = \frac{\text{Original Audio Size}}{\text{Compressed Audio Size}}
* **Bitrate Adjustment Formula:**  
  New\_Bitrate=Original\_Bitrate×Compression\_FactorNew\\_Bitrate = Original\\_Bitrate \times Compression\\_Factor

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

* express (for backend server)
* multer (for handling file uploads)
* ffmpeg or fluent-ffmpeg (for audio compression and format conversion)
* lit (for WebComponent development)

**Acceptance Criteria:**

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

**Submission Guidelines:**

1. **Fork** the provided GitHub repository.
2. **Create a folder** named audio-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 audio file and integrates seamlessly with the WebComponent-based frontend.**