**Title: Image Color Converter**

**Tool Details:**

**Technology Stack:**

* **Frontend:** Web Components (Lit or Vanilla JS)
* **Backend:** Express.js (Node.js)
* **File Processing:** Convert images to grayscale, sepia, or other color filters
* **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 processing** to apply color transformations.
* Work with **Web Components** to create a frontend for image upload.
* Understand **client-server communication** via API calls.
* Gain experience in handling **image manipulation techniques**.

**Assignment Description:**

Develop an **Image Color Converter** where users upload an **image file** (e.g., PNG, JPG) via a **WebComponent-based form**. The backend, built with **Express.js**, will process the image and allow users to apply different **color filters** (e.g., grayscale, sepia, invert). The transformed image will be sent back to the frontend and displayed alongside the original image.

**Tasks & Steps:**

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

* Set up an **Express.js** server to handle **file uploads**.
* Use a suitable **Node.js package** to process images and apply color transformations.
* Implement API routes to **process the file and return the modified image**.

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

* Create a **form** using Web Components that allows users to **upload an image file**.
* Provide options for different **color filters** (grayscale, sepia, invert, etc.).
* Send the uploaded file and selected filter to the backend using the **fetch API**.
* Display both the **original image** and the **processed image** in the UI.

**3. Integration & Testing:**

* Ensure the frontend **properly sends data** to the backend.
* Handle errors gracefully (e.g., **invalid file types, unsupported formats**).
* Test **end-to-end functionality** to ensure seamless integration.

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

* **Grayscale Formula:**  
  Y=0.3R+0.59G+0.11BY = 0.3R + 0.59G + 0.11B
* **Sepia Transformation:**  
  R′=0.393R+0.769G+0.189BR' = 0.393R + 0.769G + 0.189B  
  G′=0.349R+0.686G+0.168BG' = 0.349R + 0.686G + 0.168B  
  B′=0.272R+0.534G+0.131BB' = 0.272R + 0.534G + 0.131B
* **Invert Colors:**  
  R′=255−R,G′=255−G,B′=255−BR' = 255 - R, G' = 255 - G, B' = 255 - B

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

* express (for backend server)
* multer (for handling file uploads)
* sharp (for image processing and color manipulation)
* lit (for WebComponent development)

**Acceptance Criteria:**

* The **Express.js backend** should successfully process **uploaded image files** and apply **selected color transformations**.
* The **WebComponent-based frontend** should have a **responsive form** that allows users to upload files and select a **color filter**.
* 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 image-color-converter-<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 handles requests, processes the image file, and seamlessly integrates with the WebComponent-based frontend.**