**Title: Image to Text Converter**

**Tool Details:**

**Technology Stack:**

* **Frontend:** Web Components (Lit or Vanilla JS)
* **Backend:** Express.js (Node.js)
* **File Processing:** Extract text from images using Optical Character Recognition (OCR)
* **Data Handling:** File uploads & API requests

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

* Learn how to build a **RESTful API** using **Express.js**.
* Implement **OCR (Optical Character Recognition)** to extract text from images.
* Work with **Web Components** to create a dynamic frontend.
* Understand **client-server communication** via API calls.
* Gain experience integrating third-party OCR packages.

**Assignment Description:**

Develop an **Image to Text 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 extract readable **text** from it. The extracted text should be displayed on the frontend.

**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 the uploaded image and extract text.
* Implement API routes for **file processing** and **returning the extracted text**.

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

* Create a **form** using Web Components that allows users to **upload an image file**.
* Send the uploaded file to the backend using the **fetch API**.
* Display the **extracted text** from the image in a readable format.

**3. Integration & Testing:**

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

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

* OCR algorithms use **pattern recognition and machine learning** to detect characters in images.
* Ensure **high accuracy** by optimizing image processing before text extraction.

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

* express (for backend server)
* multer (for handling file uploads)
* tesseract.js (for OCR and text extraction)
* sharp (for image preprocessing, if needed)
* lit (for WebComponent development)

**Acceptance Criteria:**

* The **Express.js backend** should successfully process **uploaded image files** and return **extracted text**.
* The **WebComponent-based frontend** should have a **responsive form** that allows users to upload files and displays the extracted text.
* Proper **error handling** should be in place for **invalid inputs or unreadable text**.
* The application should work seamlessly across **modern web browsers**.

**Submission Guidelines:**

1. **Fork** the provided GitHub repository.
2. **Create a folder** named image-to-text-<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.**