```
// Function to read the PDF and extract text using PDF.js
async function extractTextFromPDF(pdfUrl) {
    const pdf = await pdfjsLib.getDocument(pdfUrl).promise;
    const numPages = pdf.numPages;
    let textContent = '';

    for (let pageNum = 1; pageNum <= numPages; pageNum++) {
        const page = await pdf.getPage(pageNum);
        const content = await page.getTextContent();
        const pageText = content.items.map(item => item.str).join(' ');
        textContent += pageText + '\n'; // Add text of each page
    }
    return textContent;
}

// Function to perform OCR on an image
async function performOCR(image) {
    try {
        const result = await Tesseract.recognize(image, 'spa+hin', {
            logger: (m) => console.log(m)
        });
        const extractedText = result.data.text;
        console.log("Extracted Text:", extractedText);
        return extractedText;
    } catch (error) {
        console.error("Error performing OCR:", error);
        return null;
    }
}

// Detect language using an API
```

```
async function detectLanguage(text) {
  try {
    const response = await axios.get('https://api.mymemory.translated.net/get', {
      params: { q: text, langpair: "en|es" }
    });
    let detectedLang = response.data.responseData.detectedLanguage;
    if (!['hi', 'es'].includes(detectedLang)) {
      detectedLang = 'hi'; // Default to Hindi if not Spanish or Hindi
    }
    console.log("Detected Language:", detectedLang);
    return detectedLang;
  } catch (error) {
    console.error("Error detecting language:", error);
    return "hi";  // Default to Hindi if detection fails
  }
}

// Translate the extracted text
async function translateText(text, detectedLang, targetLang) {
  try {
    const sourceLang = detectedLang || "hi";
    const response = await axios.get('https://api.mymemory.translated.net/get', {
      params: { q: text, langpair: `${sourceLang}|${targetLang}` }
    });
    const translatedText = response.data.responseData.translatedText;
    console.log("Translated Text:", translatedText);
    return translatedText;
  } catch (error) {
    console.error("Error translating text:", error);
    return null;
  }
```

```
}

// Main function to handle file upload, detect type, extract text, and translate
async function handleFileUpload() {
    const fileInput = document.getElementById("fileInput");
    const targetLang = 'en';  // Target language (English)

    if (fileInput.files.length === 0) {
        alert("Please select a file.");
        return;
    }

    const file = fileInput.files[0];
    const fileUrl = URL.createObjectURL(file);

    let extractedText = '';
    if (file.type === 'application/pdf') {
        extractedText = await extractTextFromPDF(fileUrl);
    } else if (file.type.startsWith('image')) {
        extractedText = await performOCR(fileUrl);
    }

    if (extractedText) {
        const detectedLang = await detectLanguage(extractedText);

        if (['hi', 'es'].includes(detectedLang) && detectedLang !== targetLang) {
            const translatedText = await translateText(extractedText, detectedLang, targetLang);
            if (translatedText) {
                document.getElementById("output").innerText = `Original Text:\n${extractedText}\n\nTranslated Text:\n${translatedText}`;
                speakText(translatedText, targetLang);
            } else {
                document.getElementById("output").innerText = "Translation failed.";
            }
        } else {
            document.getElementById("output").innerText = `No translation needed. Original Text:\n${extractedText}`;
        }
    } else {
        document.getElementById("output").innerText = "Failed to extract text.";
    }
}

// Function to read the text out loud using the SpeechSynthesis API
function speakText(text, targetLang) {
    const speech = new SpeechSynthesisUtterance(text);

    if (targetLang === 'es') {
        speech.lang = 'es-ES'; // Spanish
    } else if (targetLang === 'hi') {
        speech.lang = 'hi-IN'; // Hindi
    } else {
        speech.lang = 'en-US'; // English
    }

    if (window.speechSynthesis) {
        window.speechSynthesis.speak(speech);
    } else {
        console.error("SpeechSynthesis API is not supported in this browser.");
    }
}
```