PROTOCOL INSTRUCTIONS

For this task, you will create a new API endpoint to handle PDF report generation using HTML-to-PDF conversion. You will adhere to the following protocols:

1. **Primary Frameworks:** You will use Node.js with Express.js for the backend and Puppeteer for PDF generation.
2. **File Naming Convention:** All new files will be placed in a logical directory structure, such as /report-generation/ for the new route and /templates/ for the HTML template.
3. **Security First:** You will use Replit Secrets for any API keys or sensitive data. The HTML template must be sanitized to prevent injection attacks before rendering.
4. **Error Handling:** Implement robust error handling for all API calls and Puppeteer operations. Log errors to the console.
5. **Design Adherence:** You will strictly follow the layout, design style, fonts, and elements from the attached LCA_Report_31_2025-09-08.pdf for all CSS and HTML generation. Replicate the document's professional and clean appearance.

TASK: Create a High-Quality PDF Report Generation Endpoint

INSTRUCTIONS:

Build a new backend API endpoint that generates a high-quality, professional-looking sustainability report PDF from a single API call. This task involves creating a new Express route, an HTML template, and a Puppeteer script to perform the conversion.

CONTEXT & DETAILS:

1. **New API Endpoint:**
   - Create a new POST endpoint at the route /api/generate-report.
   - This endpoint must accept a JSON payload in the request body.
   - The JSON payload will contain the report data. A sample of this data is attached in a JSON file for your reference.
2. **HTML Template (report_template.html):**
   - Design a new HTML file that serves as the template for the sustainability report.
   - This template must be well-styled using CSS. The styling should adhere to the existing style of the platform for all visual elements. Pay close attention to the font selection, color palette, and the clean, organized layout of the original platform.
   - The template should include a header and footer with page numbers and a report date, matching the style from the attached document.
   - The layout must be responsive, ensuring it looks good when rendered for the PDF.
3. **Puppeteer Script (pdfGenerator.js):**
   - Create a reusable JavaScript module that handles the Puppeteer logic.
   - This module should contain a function, e.g., generatePDF(data).
   - The function will:
     - Take the JSON data as an input.
     - Use a templating engine (e.g., EJS, Pug) to inject the data into the HTML

template created in step 2.
    - Launch a Puppeteer browser instance.
    - Navigate to a page and set its content to the fully-rendered HTML.
    - Use Puppeteer's page.pdf() method with high-quality settings to generate the PDF. This should include:
      - format: 'A4'
      - printBackground: true
      - displayHeaderFooter: true
      - Margins for a clean look.
  - The function should return the generated PDF as a buffer or a stream.
4. **Integration (server.js or app.js):**
  - Update your main Express server file to:
    - Import the new pdfGenerator.js module.
    - Set up the /api/generate-report endpoint.
    - In the endpoint handler, extract the data from the request body.
    - Call the generatePDF function with the extracted data.
    - Send the resulting PDF buffer back to the client with the Content-Type: application/pdf header.

ATTACHED FILES:

(A JSON file with sample data and a LCA_Report_31_2025-09-08.pdf will be attached here.)