**AI Rooftop Solar Analyzer: Documentation**

**1. Project Overview**

This project implements an AI-powered rooftop analysis tool that uses satellite imagery to assess solar installation potential. The system analyzes a user-uploaded rooftop image and provides:

* Rooftop characteristics (area, suitable sections).
* Solar panel recommendations (type, count, capacity).
* Estimated energy generation.
* Identification of potential obstructions or shading.
* Installation considerations.
* An indicative Return on Investment (ROI) estimate.
* A confidence score for the assessment.

The tool is built as a single-page web application using HTML, Tailwind CSS, and JavaScript, leveraging the Gemini API for its vision and language understanding capabilities.

**2. Project Setup Instructions**

1. **Save the Code:** Copy the HTML code provided (from the AI Rooftop Solar Analyzer immersive) and save it as an .html file (e.g., solar\_analyzer.html).
2. **Internet Connection:** Ensure you have an active internet connection, as the application fetches Tailwind CSS from a CDN and makes API calls to Google's Gemini service.
3. **API Key (Important):**
   * The provided code uses gemini-2.0-flash. If you are running this in an environment where the API key is automatically provided for this model (like the current environment), you might not need to do anything further for the API key.
   * If you intend to use a different Gemini model or run this outside such an environment, you would typically need to obtain an API key from Google AI Studio (Google Cloud).
   * In the JavaScript code, locate the line: const apiKey = "";
   * If an API key is required, you would replace "" with your actual API key. **However, for gemini-2.0-flash in the current environment, leaving it as "" is often correct as the environment injects it.**
4. **Open in Browser:** Open the solar\_analyzer.html file in a modern web browser (e.g., Chrome, Firefox, Edge).
5. **Usage:**
   * Click the "Choose File" button to upload a satellite image of a rooftop (JPG, JPEG, or PNG format).
   * Optionally, enter the location (City, Country) and average electricity cost to help the AI provide a more tailored ROI.
   * Click the "Analyze Rooftop" button.
   * Wait for the analysis to complete. Results will be displayed on the page.

**3. Implementation Documentation**

**3.1. Frontend (HTML, Tailwind CSS, JavaScript)**

* **Structure (index.html):**
  + A single HTML file defines the page structure.
  + Tailwind CSS is used for styling, loaded via CDN.
  + Google Fonts (Inter) is used for typography.
  + The layout includes sections for image upload, optional parameters, the analyze button, and results display.
* **Image Handling:**
  + The <input type="file"> element allows image selection.
  + JavaScript FileReader API is used to read the image locally and display a preview.
  + The image is converted to a base64 encoded string to be sent to the Gemini API.
* **User Inputs:**
  + Optional input fields for location and electricityCost allow users to provide more context for the analysis, particularly for ROI calculations.
* **Dynamic Display:**
  + JavaScript manipulates the DOM to show/hide the image preview, loading indicators, error messages, and the final analysis results.
  + Results are parsed from the JSON response and formatted into distinct sections for readability.

**3.2. AI Implementation (Gemini API)**

* **Model:** The application uses the gemini-2.0-flash multimodal model from Google. This model can process both text and image inputs.
* **API Call:**
  + A fetch request is made to the Gemini API endpoint: https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent.
  + The payload includes the user's prompt and the base64 encoded image data.
  + The generationConfig specifies responseMimeType: "application/json" to request a JSON formatted response.
* **Prompt Engineering:**
  + A detailed prompt is constructed in JavaScript. This prompt guides the Gemini model to perform the rooftop analysis and structure its output.
  + The prompt explicitly asks for a JSON object with a predefined schema. This schema includes keys for:
    - rooftop\_overview: Description, total area, suitable sections, usable area.
    - solar\_panel\_recommendation: Panel type assumption, count, total capacity, layout suggestion.
    - energy\_generation\_estimate: Assumed sun hours, daily, monthly, and annual generation.
    - obstructions\_and\_shading: List of identified issues.
    - installation\_considerations: Roof type suitability, orientation/tilt advice.
    - roi\_estimate: Currency, electricity cost assumption, system cost range, annual savings, payback period, notes.
    - confidence\_score: Overall confidence and reasoning.
  + The prompt also incorporates the optional location and electricityCost provided by the user to tailor the analysis.
  + It emphasizes that the response should *only* be a valid JSON object.
* **Response Handling:**
  + The JavaScript code parses the JSON response from Gemini.
  + It includes a step to clean potential markdown formatting (e.g., ` json ... ) that the model might occasionally add around the JSON output, ensuring robust parsing.
  + The parsed data is then used to populate the analysisResults section of the webpage.
* **Error Handling:**
  + try...catch blocks are used to manage errors during image processing and API calls.
  + User-friendly error messages are displayed on the UI.
  + Console logs provide more detailed error information for debugging.

**3.3. Data Flow**

1. User uploads an image and optionally provides location/electricity cost.
2. JavaScript reads the image, converts it to base64, and constructs the prompt.
3. A POST request with the image and prompt is sent to the Gemini API.
4. Gemini analyzes the image based on the prompt and returns a JSON response.
5. JavaScript receives the response, parses the JSON, and updates the webpage to display the analysis.

**4. Example Use Cases**

* **Homeowners:** Can quickly get an initial assessment of their rooftop's solar potential and an idea of costs/savings before contacting installers.
* **Solar Installers (Preliminary Assessment):** Can use the tool for a rapid first-pass analysis of a potential customer's rooftop, saving time on initial site visits for unsuitable candidates.
* **Educational Purposes:** Demonstrates the capability of multimodal AI in practical applications like renewable energy assessment.

**Example Scenario:**

1. A user uploads a clear satellite image of a suburban house with a simple pitched roof.
2. They input "Austin, Texas" as the location and "0.14 USD/kWh" as electricity cost.
3. The AI analyzes the image and might return:
   * **Rooftop Overview:** South-facing pitched roof, approx. 120 sqm total, 70 sqm usable.
   * **Panel Recommendation:** 18 x 400W monocrystalline panels, 7.2 kW total capacity.
   * **Energy Generation:** Approx. 30 kWh/day, 10,950 kWh/year (assuming 4.2 peak sun hours).
   * **Obstructions:** Small vent pipe, minimal shading.
   * **ROI Estimate:** System cost $9,000-$12,000 USD, annual savings ~$1533 USD, payback ~6-8 years.
   * **Confidence:** High, due to clear image and simple roof.

**5. Future Improvement Suggestions**

* **Advanced Location Integration:**
  + Use a Geocoding API to get precise latitude/longitude from the user-entered location.
  + Integrate with APIs (e.g., NREL PVWatts or similar) to fetch accurate local solar irradiance data (GHI, DNI, DHI) and typical weather patterns for more precise energy generation estimates.
* **Detailed Financial Modeling:**
  + Incorporate local/national solar incentives, tax credits, and rebates.
  + Allow input for loan terms or financing options.
  + Calculate more advanced financial metrics like Net Present Value (NPV) or Internal Rate of Return (IRR).
* **Interactive Rooftop Drawing:**
  + Allow users to draw or outline their rooftop area directly on the uploaded image for more accurate area calculation.
  + Enable users to mark obstructions.
* **Panel Database & Customization:**
  + Integrate a database of commercially available solar panel models with their specifications (efficiency, dimensions, cost).
  + Allow users to select preferred panel types or brands.
* **Shading Analysis Over Time:**
  + If 3D data or multiple angle views were available (beyond current scope), simulate shading patterns throughout the day and year.
* **User Accounts & History:**
  + Allow users to create accounts to save their analyses and track projects.
* **Report Generation:**
  + Provide an option to download the analysis as a PDF report.
* **Enhanced Validation & Confidence:**
  + Implement more sophisticated checks on the AI's output.
  + If the AI expresses low confidence, provide more specific reasons or request a clearer image/more information.
* **Backend API Development:**
  + For a more scalable solution, develop a dedicated backend API (e.g., using Python with Flask/FastAPI) to handle the AI logic, API key management, and potential database interactions. The frontend would then call this backend API. This also helps protect the Gemini API key.
* **Batch Processing:**
  + For professional users, allow uploading multiple images or a list of addresses for batch analysis.
* **Integration with Mapping Services:**
  + Allow users to select a rooftop directly from a map interface (e.g., Google Maps, OpenStreetMap) instead of uploading an image.

This documentation provides a comprehensive overview of the AI Rooftop Solar Analyzer, its setup, implementation, and potential future enhancements.