# **Solar Rooftop Analysis Tool Documentation**

## **1. Project Overview**

The Solar Rooftop Analysis Tool is a full-stack web application designed to assess the solar potential of any given rooftop using AI-powered image analysis and financial calculations. Users can input an address (with latitude and longitude), and the system will provide a comprehensive report including rooftop characteristics, estimated solar energy production, cost analysis, and return on investment (ROI).

**Key Features:**

* **AI-Powered Image Analysis:** Utilizes Google Gemini Vision Pro to analyze satellite imagery of rooftops, identifying usable area, orientation, shading, and obstructions.
* **Solar Potential Calculation:** Estimates potential kilowatt-hour (kWh) production, system size, installation costs, and financial metrics like payback period and ROI.
* **User Authentication:** Allows users to register and log in.
* **Data Persistence:** Stores user information and rooftop assessment reports in a PostgreSQL database.
* **Modular Architecture:** Separates concerns into a React frontend, Node.js API gateway, and Python AI engine.

## **2. Architecture**

The application follows a microservices-like architecture, separating distinct functionalities into different services that communicate via HTTP requests.

+----------------+ +-------------------+ +-------------------+ +-------------+  
| Frontend | | Node.js Backend | | Python AI Engine | | PostgreSQL |  
| (React) | <---> | (Express/Sequelize)| <---> | (FastAPI/Gemini API)| <---> | Database |  
+----------------+ +-------------------+ +-------------------+ +-------------+  
 | ^ ^  
 | | (API Requests) | (Google Maps Static API)  
 | | |  
 +------------------------------------------------------+ (AI Model Requests)

**Components:**

* **Frontend (frontend/):** A React application providing the user interface for inputting data and displaying assessment results.
* **Node.js Backend (backend-node/):**
  + Acts as the primary API gateway for the frontend.
  + Handles user authentication (registration, login).
  + Orchestrates the solar assessment process by making requests to the Python AI Engine.
  + Manages data persistence (saving user data and assessment reports) in PostgreSQL using Sequelize ORM.
* **Python AI Engine (backend-python/):**
  + A FastAPI application responsible for the core AI logic and solar calculations.
  + Fetches satellite images from Google Maps Static API (or uses a local demo image).
  + Leverages the **Google Gemini Vision Pro API** for image analysis to determine rooftop characteristics.
  + Performs solar energy production and financial ROI calculations.
* **PostgreSQL Database:** The primary data store for user accounts, authentication details, and all generated rooftop assessment reports.

## **3. Technologies Used**

* **Frontend:** React, Vite, Tailwind CSS
* **Node.js Backend:** Express.js, Sequelize ORM, PostgreSQL, bcrypt (for password hashing), axios (for HTTP requests)
* **Python AI Engine:** FastAPI, Google Generative AI SDK (for Gemini API), Pillow (for image processing), httpx (for async HTTP requests), python-dotenv
* **Database:** PostgreSQL
* **APIs:** Google Gemini API (gemini-pro-vision, gemini-pro), Google Maps Static API

## **4. Setup Guide**

This section provides step-by-step instructions to set up and run all components of the application.

### **4.1. Prerequisites**

Before you begin, ensure you have the following installed:

* **Node.js & npm:** <https://nodejs.org/> (LTS version recommended)
* **Python 3.9+ & pip:** <https://www.python.org/downloads/>
* **PostgreSQL:** <https://www.postgresql.org/download/>
  + Remember your PostgreSQL username and password.
* **Google Cloud Platform Account:**
  + Enable the **Google Gemini API** (via Google AI Studio or Google Cloud Console). Obtain your **Gemini API Key**.
  + Enable the **Maps Static API** in the Google Cloud Console. Obtain your **Google Maps Static API Key**.
  + **Important:** For the Google Maps Static API, you generally need to link a **billing account** to your Google Cloud Project, even if you intend to stay within the free tier.

### **4.2. Project Structure**

First, clone the project (or ensure your directories match this structure if you built it manually):

solar-rooftop-analysis-tool/  
├── backend-node/  
│ ├── config/  
│ │ └── database.js  
│ ├── models/  
│ │ ├── index.js  
│ │ ├── user.js  
│ │ ├── rooftopAssessment.js  
│ │ ├── solarPanel.js  
│ │ └── incentive.js  
│ ├── .env  
│ ├── package.json  
│ ├── server.js  
│ └── ... (node\_modules, etc.)  
├── backend-python/  
│ ├── src/  
│ │ ├── ai\_core.py  
│ │ └── solar\_calculator.py  
│ ├── data/  
│ │ ├── default\_params.json  
│ │ └── sample\_rooftop.png  
│ ├── .env  
│ ├── main.py  
│ ├── requirements.txt  
│ ├── venv/  
│ └── ...  
└── frontend/  
 ├── public/  
 ├── src/  
 │ ├── App.jsx  
 │ └── index.css  
 ├── index.html  
 ├── package.json  
 ├── tailwind.config.js  
 ├── vite.config.js  
 └── ... (node\_modules, etc.)

### **4.3. Database Setup (PostgreSQL)**

1. Create a PostgreSQL Database:  
   Open your PostgreSQL client (e.g., psql command line, pgAdmin, DBeaver) and create a new database.  
   CREATE DATABASE solar\_analysis\_db;  
   -- Optional: CREATE USER your\_pg\_username WITH PASSWORD 'your\_pg\_password';  
   -- Optional: GRANT ALL PRIVILEGES ON DATABASE solar\_analysis\_db TO your\_pg\_username;  
     
   Replace solar\_analysis\_db, your\_pg\_username, and your\_pg\_password with your desired values.

### **4.4. Backend (Node.js) Setup**

1. **Navigate to Node.js backend directory:**  
   cd solar-rooftop-analysis-tool/backend-node
2. **Install dependencies:**  
   npm install  
   npm install axios bcrypt dotenv sequelize pg
3. Create .env file:  
   Create a file named .env in the backend-node/ directory and add the following:  
   NODE\_PORT=5000  
   DB\_HOST=localhost  
   DB\_PORT=5432  
   DB\_NAME=solar\_analysis\_db  
   DB\_USER=your\_pg\_username  
   DB\_PASSWORD=your\_pg\_password  
     
   # URL for your Python FastAPI backend  
   PYTHON\_API\_URL=http://localhost:8000  
     
   # Secret for JWTs (generate a strong, random string for production)  
   JWT\_SECRET=supersecurejwtsecretkey\_replace\_me
4. Verify/Create Model Files:  
   Ensure these files exist in backend-node/models/:
   * user.js
   * rooftopAssessment.js
   * solarPanel.js
   * incentive.js
   * index.js
5. **Start Node.js Backend:**  
   node server.js  
     
   This will connect to PostgreSQL and synchronize the database models (creating tables if they don't exist). You should see messages like "Database models synchronized successfully." and "Node.js Backend running on port 5000."

### **4.5. Backend (Python AI Engine) Setup**

1. **Navigate to Python backend directory:**  
   cd solar-rooftop-analysis-tool/backend-python
2. **Create and activate a virtual environment:**  
   python -m venv venv  
   # On macOS/Linux:  
   source venv/bin/activate  
   # On Windows (Command Prompt):  
   # venv\Scripts\activate  
   # On Windows (PowerShell):  
   # .\venv\Scripts\Activate.ps1
3. **Install dependencies:**  
   pip install fastapi uvicorn google-generativeai httpx pillow python-dotenv
4. Create .env file:  
   Create a file named .env in the backend-python/ directory and add your API keys:  
   GEMINI\_API\_KEY=YOUR\_GEMINI\_API\_KEY\_HERE  
     
   # Optional: Specify exact Gemini models if you prefer  
   GEMINI\_VISION\_MODEL=gemini-pro-vision  
   GEMINI\_TEXT\_MODEL=gemini-pro  
     
   # Your API key for Google Maps Static API (from Google Cloud Console)  
   # This key needs to have the "Maps Static API" enabled.  
   # If using local image demo, you can leave this empty.  
   GOOGLE\_MAPS\_STATIC\_API\_KEY=YOUR\_GOOGLE\_MAPS\_STATIC\_API\_KEY\_HERE  
     
   **Important for Google Maps Static API Key:**
   * Ensure the "Maps Static API" is enabled for the key in Google Cloud Console.
   * Ensure a **billing account is linked** to your Google Cloud project (even for free tier usage).
   * If you have "IP addresses (web servers, cron jobs, etc.)" restrictions, add 127.0.0.1 and ::1.
   * As a last resort for local testing, temporarily remove all API restrictions (then re-add them for security!).
   * Restart Python backend after .env changes.
5. Create data/ directory and default\_params.json:  
   Create a data folder inside backend-python.  
   Create backend-python/data/default\_params.json with the following content:  
   {  
    "panel\_efficiency\_percent": 20.0,  
    "system\_cost\_per\_watt\_usd": 3.0,  
    "federal\_tax\_credit\_percent": 30.0,  
    "average\_electricity\_price\_kwh\_usd": 0.15,  
    "system\_lifespan\_years": 25,  
    "degradation\_rate\_percent\_per\_year": 0.5,  
    "average\_daily\_peak\_sun\_hours": 4.5,  
    "shading\_impact\_factor": {  
    "low": 1.0, "medium": 0.85, "high": 0.70, "negligible": 1.0, "significant": 0.70, "unknown": 0.90  
    },  
    "orientation\_impact\_factor": {  
    "South": 1.0, "South-East": 0.95, "South-West": 0.95, "East": 0.85, "West": 0.85, "North": 0.60,  
    "Flat": 0.90, "Multi-directional": 0.90, "Unknown": 0.90  
    },  
    "panel\_wattage\_per\_sqm": 200,  
    "typical\_panel\_area\_sqm": 1.7  
   }
6. **Optional: Setup for Local Image Demo (if no Maps API key/billing)**
   * Take a screenshot of a rooftop, crop it to a square (e.g., 640x640px), and save it as sample\_rooftop.png inside backend-python/data/.
   * The ai\_core.py is configured to use this image if latitude and longitude are 0.0.
7. **Start Python AI Engine:**  
   uvicorn main:app --reload --port 8000  
     
   You should see output indicating FastAPI is running on http://127.0.0.1:8000.

### **4.6. Frontend (React) Setup**

1. **Navigate to Frontend directory:**  
   cd solar-rooftop-analysis-tool/frontend
2. **Install dependencies:**  
   npm install  
   npm install -D tailwindcss postcss autoprefixer  
   npx tailwindcss init -p
3. Configure tailwind.config.js:  
   Ensure frontend/tailwind.config.js has the correct content paths:  
   /\*\* @type {import('tailwindcss').Config} \*/  
   export default {  
    content: [  
    "./index.html",  
    "./src/\*\*/\*.{js,ts,jsx,tsx}",  
    ],  
    theme: {  
    extend: {},  
    },  
    plugins: [],  
   }
4. Add Tailwind directives to frontend/src/index.css:  
   Replace or add to the top of frontend/src/index.css:  
   @tailwind base;  
   @tailwind components;  
   @tailwind utilities;
5. **Start React Frontend:**  
   npm run dev  
     
   This will usually start the development server at http://localhost:5173.

## **5. API Endpoints**

### **5.1. Node.js Backend Endpoints (http://localhost:5000)**

* **GET /**
  + **Description:** Basic health check for the Node.js backend.
  + **Response:** {"message": "Node.js Backend is running!"}
* **POST /api/auth/register**
  + **Description:** Registers a new user.
  + **Request Body (JSON):**  
    {  
     "username": "testuser",  
     "email": "test@example.com",  
     "password": "strongpassword"  
    }
  + **Response (JSON):**
    - **Success (201):** {"message": "User registered successfully!", "user": {"id": 1, "username": "testuser", "email": "test@example.com"}}
    - **Error (400/409/500):** {"error": "..."}
* **POST /api/auth/login**
  + **Description:** Logs in an existing user.
  + **Request Body (JSON):**  
    {  
     "email": "test@example.com",  
     "password": "strongpassword"  
    }
  + **Response (JSON):**
    - **Success (200):** {"message": "Login successful!", "user": {"id": 1, "username": "testuser", "email": "test@example.com"}}
    - **Error (400/401/500):** {"error": "..."}
* **POST /api/solar-assessment**
  + **Description:** Triggers a comprehensive solar rooftop assessment. This is the main endpoint used by the frontend. It proxies the request to the Python AI Engine and saves the results to the database.
  + **Request Body (JSON):**  
    {  
     "address": "1600 Amphitheatre Pkwy, Mountain View, CA",  
     "latitude": 37.422,  
     "longitude": -122.084  
     // "userId": 1 // Optional, if user is logged in  
    }  
      
    **For Local Image Demo (if no Maps API key):**  
    {  
     "address": "DEMO ROOFTOP LOCATION",  
     "latitude": 0.0,  
     "longitude": 0.0  
    }
  + **Response (JSON):**
    - **Success (200):** Returns a comprehensive JSON object containing AI analysis results and solar financial calculations.
    - **Error (400/500):** {"error": "...", "details": "..."}
* **GET /api/assessments**
  + **Description:** Retrieves all saved rooftop assessments from the database. (For testing/admin)
  + **Response (JSON):** Array of RooftopAssessment objects.
* **GET /api/assessments/:id**
  + **Description:** Retrieves a specific rooftop assessment by its ID.
  + **Response (JSON):** A single RooftopAssessment object.

### **5.2. Python AI Engine Endpoints (http://localhost:8000)**

* **GET /**
  + **Description:** Basic health check for the Python AI Engine.
  + **Response:** {"message": "Python AI Engine is running! (FastAPI)"}
* **POST /api/analyze-rooftop**
  + **Description:** Internal endpoint called by the Node.js backend. Performs AI analysis and solar calculations.
  + **Request Body (JSON):**  
    {  
     "address": "...",  
     "latitude": 37.422,  
     "longitude": -122.084  
    }
  + **Response (JSON):** A structured JSON object containing:
    - request\_details: Original request parameters.
    - ai\_image\_analysis: Detailed AI insights (rooftop shape, usable area, shading, etc.).
    - solar\_financial\_analysis: Calculated solar potential, costs, and ROI.
    - overall\_message: A general status message.

## **6. How to Run the Application (All Components)**

To run the entire full-stack application, follow these steps in separate terminal windows/tabs:

1. Start PostgreSQL Database:
   * Ensure your PostgreSQL server is running.
2. Start Node.js Backend:
   * Open a terminal.
   * cd solar-rooftop-analysis-tool/backend-node
   * node server.js
3. Start Python AI Engine:
   * Open a new terminal.
   * cd solar-rooftop-analysis-tool/backend-python
   * source venv/bin/activate (or your OS's equivalent)
   * uvicorn main:app --reload --port 8000
4. Start React Frontend:
   * Open a new terminal.
   * cd solar-rooftop-analysis-tool/frontend
   * npm run dev

Once all three servers are running, open your web browser and navigate to the address provided by the React frontend (usually http://localhost:5173).

## **7. Troubleshooting Tips**

* "Cannot find module './solarPanel'" or similar in Node.js:
  + Ensure solarPanel.js and incentive.js (and user.js, rooftopAssessment.js) exist in backend-node/models/ and contain the correct code.
  + Check backend-node/models/index.js to ensure it correctly imports all model files.
  + Restart the Node.js backend.
* "name 'HTTPException' is not defined" in Python:
  + Verify that from fastapi import FastAPI, HTTPException is present at the top of backend-python/main.py.
  + Verify that from fastapi import HTTPException is present at the top of backend-python/src/ai\_core.py.
  + Deactivate and re-activate your Python virtual environment, then restart the Python backend.
* "Failed to fetch satellite image: The Google Maps Platform server rejected your request. The provided API key is invalid.":
  + Double-check your GOOGLE\_MAPS\_STATIC\_API\_KEY in backend-python/.env for typos.
  + Go to Google Cloud Console (APIs & Services -> Credentials -> Your API Key).
  + Ensure "Maps Static API" is enabled for that key.
  + Ensure a billing account is linked to your Google Cloud project (even for free tier usage).
  + If you have "IP addresses (web servers, cron jobs, etc.)" restrictions, add 127.0.0.1 and ::1.
  + As a last resort for local testing, temporarily remove all API restrictions (then re-add them for security!).
  + Restart Python backend after .env changes.
* "Could not connect to database..." (Node.js):
  + Ensure PostgreSQL is running.
  + Verify DB\_NAME, DB\_USER, DB\_PASSWORD, DB\_HOST, DB\_PORT in backend-node/.env exactly match your PostgreSQL setup.
* Frontend/Backend communication errors (e.g., Failed to fetch in browser console):
  + Ensure both Node.js (localhost:5000) and Python (localhost:8000) backends are running.
  + Check for CORS errors in the browser console. The Node.js backend has cors() enabled, but ensure it's not blocked by browser extensions or other configurations.
  + Verify the fetch URL in frontend/src/App.jsx matches your Node.js backend URL (http://localhost:5000).

## **8. Future Enhancements**

* User Authentication & JWTs: Implement JWTs for secure user sessions and authenticated API calls.
* Assessment History: Allow logged-in users to view their past assessment reports.
* Map Visualization: Integrate an interactive map (e.g., Google Maps JavaScript API, Leaflet) to display the rooftop being analyzed and potentially overlay suggested panel layouts.
* Detailed Report Generation: Create PDF or more structured reports from the assessment data.
* Advanced AI Analysis: Incorporate more sophisticated AI to detect roof material, age, and potential structural issues.
* Local Incentives: Integrate a database or API for state/local solar incentives.
* Panel Selection: Allow users to select specific solar panel models to refine calculations.
* 3D Model Reconstruction: (Advanced) Use AI to generate a 3D model of the roof for precise shading analysis.