Aerialytic Technical Assignment

Overview

You are tasked with building a full-stack web application that allows users to:

- Input geographic coordinates (latitude and longitude).
- Receive an estimate of the **annual optimal roof pitch** and **azimuth angle** for solar panel installation at that location.

This tool should simulate how an intelligent solar design assistant might behave when assessing a building site.

Objectives

Core Features

Layer	Requirements
Frontend (TypeScript)	• Simple UI for inputting latitude and longitude. • Ability to optionally select a point via an embedded Google Map. • Optionally accept an "offset angle" (angle between the ground surface and horizontal line) as a secondary input. • Display calculated optimal pitch and azimuth clearly.
Backend (Python - Django)	• Receive coordinate and optional offset angle. • Calculate optimal **tilt (pitch)** and **azimuth** using appropriate solar angle formulas (use NREL or equivalent logic, such as the Liu and Jordan model). • Return values as JSON. • Add stubbed comments for improvements or future enhancements (e.g., dynamic horizon modeling, PV simulation, terrain adjustment).
Build/ Deployment	• Both frontend and backend must be **Dockerized**. • Include Kubernetes (K8s) manifests for scalable deployment. • Helm chart is a bonus. • Environment variables should be easily configurable (use `.env` or ConfigMaps).

Bonus Points

Area	Bonus Tasks
UI/UX	Show selected coordinates and angles visually on a Google Map or canvas.
Backend	Use `pvlib` or `solpy` for realistic solar geometry (must be justified with citations in comments).
Modeling	Accept and use **offset angle** to adjust the pitch/tilt logic more accurately.
Data	Show solar insolation intensity estimates (even rough estimates via NREL or SolarAnywhere API integration).
Infra	Add autoscaling K8s configs or integrate with a cloud platform (GCP/AWS).

Submission Requirements

- Provide either:
  - A **GitHub repository** (preferred), or
  - A downloadable **.zip archive** with full code and deployment instructions.
- Include a **README.md** with:
  - Setup steps.
  - Architecture decisions.
  - Assumptions and known limitations.
  - Brief explanation of your solar modeling logic and any references used.
- Comment **any feature** that was skipped due to time but should be included in a production version.

Evaluation Criteria

Category	Description
Technical Accuracy	Correct usage of solar formulas for pitch/azimuth.
Code Quality	Type safety, modularity, appropriate comments, and documentation.
UX Consideration	Intuitive design, responsive layout, map integration bonus.
Docker/K8s Readiness	Correct Dockerfiles, deployment manifests, env handling.
System Design	Code organization, extensibility, and performance readiness.
Bonus Features	Any additional functionality, testing, monitoring, CI/CD, etc.