Tech Stack (PTC)

ChatGPT 4o

Frontend – Geospatial-First User Experience

- **Framework:** Next.js with TypeScript
  Enables reactive UI, rapid development, and dynamic routing for complex geospatial layers
- **Map Rendering:** Deck.gl + Mapbox
  Supports satellite overlays, intervention pins, region selection, and time-series playback
- **Authentication:** Supabase Auth
  Provides secure, scalable user access with operator-specific permissions and row-level controls

Backend – Simulation-Ready API Architecture

- **API Framework:** RESTful + GraphQL hybrid
  REST for ingestion and updates; GraphQL for flexible, query-efficient access to simulation results and intervention metadata
- **Climate Intelligence Engine:**
  - Python microservices for intervention modeling
  - Sci-kit learn / XGBoost for forecasting impact (e.g., local temperature shift, CO_2 drawdown)
  - Async queue processing (via Celery or equivalent) for large geospatial queries

Database – High-Resolution Spatiotemporal Modeling

- **Primary DB:** Supabase (PostgreSQL) with PostGIS
  Handles spatial queries and time-series intervention data with performance
- **Data Model:**
  - `climate_grid_cells`: Indexed by lat/long, with time-based CO_2/temp/biomass overlays
  - `interventions`: Track deployments, impact, cost, and validation outcomes
  - `operators`: Project owner accounts with API keys and usage logs

Tooling and Infrastructure

- **Dev Stack:** Node.js, Python, TypeScript, Supabase, GitHub
- **Containerization:** Docker (Planned)
  Supports isolated simulation runs and secure deployment across orgs
- **CI/CD:** GitHub Actions
  Automates linting, testing, and production pushes
- **Security:**
  - Row-level access control per intervention/operator
  - Secure API key handling for satellite and operator ingestion
  - Optional end-to-end encryption for sensitive project data

Why This Stack?

- **Geospatial-First:** Built for climate modeling and location-specific recommendations
- **Simulatable:** Designed for real-time and predictive ops modeling
- **Composable:** Modular services enable easy integration of new intervention types or regions

- **Scalable:** From single ecosystem use case to planetary-level deployment planning

---

Claude Sonnet 4

Frontend – Real-Time Climate Command Interface

Framework: Next.js with TypeScript – ensures fast development, server-side rendering, and seamless integration with geospatial APIs and real-time climate data feeds

Authentication: Supabase Auth – secure, scalable user management with role-based access control for climate operators, researchers, and government agencies

Mapping & Visualization: Mapbox GL JS – interactive global maps for climate intervention visualization, satellite data overlay, and real-time deployment tracking

Backend – Scalable Climate Data Architecture

API Structure: RESTful endpoints for satellite data ingestion, geospatial optimization, and coordination access — decoupled per data source (e.g., NASA, Copernicus, climate intervention operators)

Optimization Logic: Climate-specific algorithms for intervention placement, effectiveness modeling, and planetary cooling optimization

Geospatial Processing: PostGIS extensions for geographic optimization, intervention overlap detection, and regional cooling impact analysis

Database – Real-Time, Geospatial, and Climate-Optimized

Supabase (PostgreSQL) with PostGIS and full Row-Level Security – ensures operator-specific data access and government-grade auditability

Data Model: Split between `climate_data`, `interventions`, and `optimization_results` tables per data source – designed for real-time coordination and future climate AI pipelines

Time-Series Storage: Optimized for historical climate trends, intervention effectiveness tracking, and predictive cooling modeling

Climate Data Integration

Satellite APIs: NASA Earth Observing System, Copernicus Climate Change Service, NOAA Global Monitoring Laboratory

Real-Time Processing: Node.js workers for continuous satellite data ingestion and climate change detection

Data Pipeline: ETL processes for normalizing atmospheric CO_2, temperature data, and intervention effectiveness metrics

Optimization & Coordination Engine

Geographic Algorithms: Custom optimization algorithms for intervention placement based on climate effectiveness, cost, and coordination opportunities

Climate Modeling: Integration with open-source climate models for cooling impact prediction and intervention synergy detection

Coordination Logic: Real-time algorithms for detecting intervention conflicts and optimization opportunities

Tooling and Infrastructure

Dev Stack: Node.js, TypeScript, Supabase, PostGIS, GitHub

Planned: Docker containerization for government deployment, expanded real-time satellite data sync, Kubernetes for planetary-scale processing

Security: Built-in access controls through Supabase + auth layer, plus field-level visibility for international climate policy compliance

Monitoring: Real-time climate data validation, intervention tracking, and planetary cooling impact measurement

Why This Stack?

Low-friction pilots with high-leverage climate data integrations

Fast iteration cycles for feedback-driven development with climate operators

Ready to scale into a fully API-driven planetary climate coordination infrastructure with minimal refactoring

Government-grade security for international climate agency collaboration

Real-time processing capable of handling planetary-scale climate data and intervention coordination

Technical Architecture Priorities

Climate Data Processing

- **Satellite data ingestion:** Real-time feeds from multiple climate monitoring satellites
- **Geospatial optimization:** PostGIS for location-based intervention effectiveness analysis
- **Time-series climate modeling:** Historical and predictive climate data processing
- **Multi-operator coordination:** Support independent climate operators sharing coordination benefits

Scalability & Performance

- **Planetary-scale data:** Handle global climate monitoring data with minimal latency
- **Real-time coordination:** Instant intervention optimization and conflict detection
- **Government deployment:** Docker/Kubernetes ready for international climate agency infrastructure
- **API-first architecture:** Enable integration with existing climate management and monitoring systems

Security & Compliance

- **International standards:** Support for climate policy compliance and international coordination agreements
- **Audit trails:** Full intervention coordination lineage for policy and accountability
- **Role-based access:** Climate operators, researchers, and government agencies with appropriate permissions
- **Data sovereignty:** Support for regional climate data requirements and international cooperation frameworks

The technical goal: Build a robust, scalable system that can coordinate climate interventions at planetary scale while maintaining the speed and flexibility needed for rapid climate response.