EcoVision AI – Professional Developer Build Prompt (Condensed ~1000 words)

Overview: EcoVision AI is a sustainability-focused analytics platform combining a polished React-based dashboard, a secure backend API with user authentication, and a lightweight machine-learning microservice for consumption prediction. The main goal is to provide users with tools to visualize, predict, and reduce energy, water, and transportation-related carbon impact. This document provides clear build instructions suitable for fast development on a hackathon timeline while maintaining engineering standards.

Core Modules: 1. Authentication (Email + Password + Google Sign-In via Firebase Auth) 2. Dashboard Visualization (Energy, Water, and CO■ usage trends) 3. Prediction Models (Simple regression-based forecasting via Flask ML API) 4. Eco-Route Optimization (Rule-based suggestion logic and UI module) 5. AI Assistant (Placeholder text-based assistant to give sustainability guidance)

Recommended Tech Stack: Frontend: React (Vite), Tailwind CSS, Recharts, Framer Motion Backend: Node.js + Express, MongoDB (Atlas), Firebase Admin for token verification ML Service: Python + Flask + scikit-learn model for prediction Hosting/Dev Environment: Replit or GitHub → Vercel / Render

System Architecture Summary: The application is structured as a monorepo containing three major services: • client/ (React application) • server/ (Node.js REST API) • ml-service/ (Flask prediction microservice)

The client communicates securely with the backend using Firebase ID tokens. The backend verifies these tokens using Firebase Admin SDK and uses MongoDB to store user and usage history. The backend also proxies prediction requests to the ML service. This ensures clean separation between UI, business logic, and modeling.

Authentication Flow: 1. User signs in via Firebase Auth (email/password or Google). 2. The client calls getIdToken() to obtain a fresh ID token. 3. Client sends authenticated requests to backend APIs with Authorization: Bearer . 4. The backend verifies token signature and extracts user identity. 5. Once identity is confirmed, backend operations continue normally.

Frontend Structure: • Home Page – Introduction to EcoVision AI with eco-themed design, gradient background, and CTAs to log in or sign up. • Login & Signup Pages – Firebase authentication forms plus social login option. • Dashboard – Displays consumption trends, prediction summary, and sustainability indicators. • Predictions Page – Input form to submit temperature, day, and previous usage to generate usage forecast. • Eco-Route Page – Basic UI form for origin/destination and output recommended route with estimated CO■ savings. • Assistant Page – Simple chat-like UI that returns informational sustainability tips.

Styling Guidelines: Use Tailwind CSS utility classes for layout, spacing, and theming. Apply a "Neo-Glass" aesthetic: semi-transparent UI cards, subtle shadows, and smooth framer-motion transitions. Include a Dark/Light Mode toggle stored in client state. Keep typography clean and accessible, avoid clutter, and maintain consistent margins and spacing.

Backend Structure: • /auth/upsert – Stores or updates user profile information post-login • /predictions/energy – Proxies request to ML service, logs usage predictions • /eco/route – Returns eco-friendly route recommendation • /tips/daily – Returns a predefined list of sustainability advice messages

Data Model Summary: User { uid: string (Firebase UID), email: string, name: string, provider: string, preferences: { theme: string } }

UsageRecord { uid: string, date: Date, energy_kwh: number, predicted_energy_kwh: number }

RouteLog { uid: string, origin: string, destination: string, pickedRoute: string, savedCo2Kg: number }

ML Service Details: The ML service hosts a simple linear regression model trained on synthetic or real usage data. It accepts JSON input with fields (temp, day, usage_prev) and responds with predicted usage. If a trained model is missing, a fallback heuristic formula is used.

Deployment Notes: On Replit, run both Node server and client concurrently. Ensure the ML service runs on port 5000 locally. Set .env variables for Firebase, MongoDB, and ML_BASE_URL.

Conclusion: This specification defines a consistent way to integrate UI, backend logic, authentication, and a lightweight prediction model into a coherent sustainability dashboard. Follow the outlined file structure and API contracts to ensure compatibility between modules. The system is intentionally modular to allow future enhancements, such as improved model accuracy, live map routing, gamified user engagement, and integration with IoT energy monitoring devices.