SiteSage: Your personal AI camping scout

Project Objectives

1. **Develop a Centralized Campground Database:** To create and maintain a comprehensive database of campgrounds, sourced from Google Maps, that includes detailed information on amenities, location, and user reviews.
2. **Deliver a Seamless User Experience:** To build an intuitive, self-contained desktop application with a conversational chat interface, eliminating any need for technical setup by the end user. 🖥️
3. **Leverage a Hybrid RAG Architecture:** To provide accurate, context-aware answers by performing the "Retrieval" step on a local index on the user's machine and the "Augmented Generation" step in the cloud, ensuring both speed and intelligence.
4. **Build a Maintainable and Cost-Effective Solution:** To use Google Cloud for the heavy lifting (data ingestion, index creation) while keeping operational costs near zero by shifting the continuous query workload to the local client application.

Implementation Plan: Modular Phases

Phase 1: Data Foundation and Ingestion (Cloud Backend)

This phase focuses on creating the cloud infrastructure to gather and store the raw campground data. It remains the foundational first step.

- **Objective A: Provision Cloud Infrastructure:** Set up the Google Cloud project, billing alerts, networking, and necessary IAM permissions.
- **Objective B: Deploy Database:** Design and deploy a Firestore database to flexibly store detailed campground information.
- **Objective C: Develop Ingestion Pipeline:** Create a manually triggered Cloud Function that uses the Google Maps Platform API to search for campgrounds in a specified region and populate the Firestore database.

Phase 2: Knowledge Base Generation and Distribution (Cloud Backend)

This phase replaces the cloud-hosted index. Its goal is to process the raw data into portable index files and make them available for download.

- **Objective A: Develop an Indexing Pipeline:** Create a cloud-based process (e.g., a Cloud Run job) that reads data from Firestore, generates vector embeddings, and packages them into a compact, portable index file format using a library like **FAISS**.
- **Objective B: Store Indexes for Download:** Place the generated index files into a Google Cloud Storage bucket.
- **Objective C: Create a Versioning API:** Deploy a simple, serverless API endpoint (Cloud Function) that the desktop application can call to check the version of the latest available index file.

Phase 3: Desktop Application & Local Retrieval System (Client-Side)

This phase focuses on building the user-facing application that will run on their local machine.

- **Objective A: Develop the Desktop Application:** Build a cross-platform desktop application using a framework like **Electron**. This application will contain the chat UI and all necessary logic.
- **Objective B: Integrate a Local Search Server:** Bundle a lightweight Python server (using Flask or FastAPI) within the Electron app. This server's sole job is to load the downloaded FAISS index into memory and expose a local API endpoint (e.g., localhost:1234/search) for performing queries. 🔍
- **Objective C: Implement Sync Logic:** Code the functionality for the application to call the versioning API, download the index file from Google Cloud Storage on first launch, and handle future updates.

Phase 4: Hybrid RAG Integration and Cloud Agent Setup

This phase connects the local client application with the powerful generative model in the cloud to complete the RAG workflow.

- **Objective A: Configure the Generative Agent:** Set up the generative component in Vertex AI using a Gemini model. This agent will be responsible for taking structured data and generating a conversational response.
- **Objective B: Implement the Hybrid RAG Flow:** Wire up the end-to-end logic: The desktop app queries its local server for relevant campground IDs, then sends those IDs (as context) along with the user's original question to the Vertex AI agent for final answer generation.
- **Objective C: Test and Refine:** Conduct comprehensive testing of the full, hybrid workflow to ensure accuracy, speed, and high-quality conversational output.

Phase 5: Packaging, Deployment, and Maintenance

This final phase focuses on delivering the application to the user and maintaining the backend data pipeline.

- **Objective A: Package the Application:** Create user-friendly installers (.exe, .dmg) for the desktop application that handle the entire setup process automatically.
- **Objective B: Deploy Backend Services:** Deploy the minimal backend APIs (versioning, ingestion trigger) as cost-effective, serverless Cloud Functions or Cloud Run services.
- **Objective C: Establish a Maintenance Workflow:** Define a simple process for you, the developer, to periodically run the data ingestion and index generation pipelines to keep the campground information fresh for users to download. 🔄