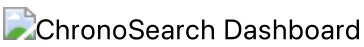
ChronoSearch 🕰️

AI-Powered Semantic Video Search Engine

ChronoSearch is a video hosting platform that lets you **search inside videos** using natural language. Instead of just matching titles or tags, it uses **Computer Vision** and **Vector Embeddings** to understand the visual content of every frame.

Search for *"Dragon"* or *"Red Car"*, and ChronoSearch will jump to the exact second that object appears, even if it's never mentioned in the title.



🚀 Key Features

- 🔍 **Hybrid Search Engine:** Combines **Metadata Search** (Titles/Tags) with **Deep Visual Search** (Frame-by-frame analysis) for distinct results.
- 🧠 **Visual AI Indexing:** Uses Google's **SigLIP** model to convert video frames into 1152-dimensional vectors.
- ⚡ **Serverless Backend:** Built on **Modal**, scaling GPUs (T4) on-demand to process uploads in parallel.
- 📽️ **Smart Streaming:** Custom-built streaming endpoint supporting **Range Requests (206 Partial Content)** for smooth playback.
- 📂 **Modular Architecture:** Clean separation of Extraction, Indexing, and Search logic.
- 🔐 **Secure Auth:** Integrated Google OAuth + JWT for secure user management.

🛠️ Tech Stack

Frontend

- **React + Vite:** High-performance UI.
- **Tailwind CSS:** Modern styling.
- **Axios:** API communication.

Backend (The Core)

- **Python & FastAPI:** REST API.
- **Modal:** Serverless Cloud Platform (GPU & Storage).
- **LanceDB:** Vector Database for billion-scale vector search.
- **OpenCV & PIL:** Frame extraction and processing.
- **HuggingFace Transformers:** SigLIP Model (`google/siglip-so400m-patch14-384`).

🏗️ Architecture

1. **Extraction:** When a video is uploaded, the backend extracts frames at **1 FPS**.

2. **Vectorization:** Each frame is passed through the **SigLIP AI model** to generate a vector embedding.
3. **Indexing:** Vectors are stored in **LanceDB** on a persistent Cloud Volume.
4. **Search:**
    - **Global:** Scans Titles & Tags first.
    - **Visual:** Scans Frame Vectors using Cosine Similarity.
    - **Hybrid:** Merges and ranks results to find the best match.

📂 Project Structure

```
├── backend/
│   ├── main.py             # Entry point (Run this to deploy)
│   ├── AI.py               # Modal App Orchestrator (GPU Logic)
│   ├── api.py              # FastAPI Routes (Stream, Upload, Search)
│   ├── auth.py             # Authentication Logic (JWT & Google Auth)
│   ├── common.py           # Configuration & Modal Image Definition
│   ├── database.py         # SQL Database Models (Users, Videos)
│   ├── extract.py          # Module: Frame Extraction (OpenCV)
│   ├── index.py            # Module: Vector Indexing (SigLIP + LanceDB)
│   ├── search.py           # Module: Deep Visual Search Logic
│   └── search_global.py    # Module: Hybrid Global Search Logic
│
├── frontend/
│   ├── src/
│   │   ├── components/
│   │   │   ├── Navbar.jsx        # Navigation & Search Bar
│   │   │   └── UploadModal.jsx   # Video Upload UI
│   │   ├── pages/
│   │   │   ├── Home.jsx          # Main Feed & Global Search Results
│   │   │   ├── VideoPlayer.jsx   # Video Streaming & Deep Search UI
│   │   │   ├── Profile.jsx       # User Dashboard & My Videos
│   │   │   └── Login.jsx         # Google Authentication Page
│   │   ├── services/
│   │   │   └── api.js            # Axios Setup & API Calls
│   │   ├── App.jsx               # Main Routing Layout
│   │   └── main.jsx              # React Entry Point
│   ├── public/                   # Static Assets
│   └── vite.config.js            # Frontend Proxy Configuration
│
└── .gitignore              # Git Ignore Rules
```

⚡ Setup & Installation

1. Clone the Repository

```
git clone [https://github.com/YOUR_USERNAME/ChronoSearch.git]
(https://github.com/YOUR_USERNAME/ChronoSearch.git)
```

```
cd ChronoSearch
```

2. Backend Setup (Modal)

You need a Modal.com account.

```
# Install Modal
pip install modal

# Authenticate
modal setup

# Deploy the App
modal deploy backend/main.py
```

*After deployment, copy the **URL** provided by Modal (ending in `.modal.run`).*

3. Frontend Setup

```
cd frontend
npm install
```

Configure Environment Variables: Create a `.env` file in the `frontend/` folder. This is **required** to connect the frontend to your backend.

```
# frontend/.env
VITE_API_URL=https://REPLACE_WITH_YOUR_MODAL_URL.modal.run
VITE_GOOGLE_CLIENT_ID=REPLACE_WITH_YOUR_GOOGLE_CLIENT_ID
```

Run the UI:

```
npm run dev
```

License

This project is open-source and available under the **MIT License**.

Built with ❤️ by Kishan Amaliya