Project Description and Purpose

This project is an **AI-powered Resume Optimizer** built with Next.js. Users can input a job description and their resume, and the app uses AI models to generate a tailored, ATS-friendly resume optimized for the job. The system highlights keyword optimization, provides improvement suggestions, calculates a match score, and allows users to save results to both MongoDB and Supabase for analytics and persistence.

Technologies Used

- **Frontend:**
  - Next.js (App Router, TypeScript)
  - React (functional components, hooks)
  - Tailwind CSS for styling
  - shadcn/ui component library
  - Lucide React for icons
  - jsPDF for PDF export
- **Backend/API:**
  - Next.js API routes (app/api/generate-resume/route.ts,app/api/save-resume/route.ts)
  - @xenova/transformers for free, local AI model inference (summarization, text2text-generation)
  - MongoDB for storing resumes and job descriptions
  - Supabase for user authentication and metadata storage
- **Other:**
  - ESLint, Prettier, TypeScript, PostCSS
  - Vercel for deployment

Challenges Faced and Solutions

1. AI Integration (Cost & Deployment)

- **Challenge:** Many AI APIs (OpenAI, HuggingFace Inference) are paid or have strict limits.
- **Solution:**
    - Integrated @xenova/transformers for free, on-device inference.
    - Used models like distilbart-cnn-6-6 and bart-large-cnn for summarization and resume tailoring.
    - Implemented fallback logic: If the main model fails, an alternative model or a keyword-based enhancement function is used.

2. Vercel Deployment Issues

- **Challenge:** Running heavy AI models/server-side code on Vercel can cause cold start delays or memory issues.
- **Solution:**
    - Kept model loading logic cached and lightweight (see let summarizer: SummarizerFunction | null = null; in app/api/generate-resume/route.ts).
    - Provided a fallback resume enhancement function that does not require AI inference, ensuring the API always responds even if the model fails to load on Vercel.

3. Keyword Extraction & Resume Structuring

- **Challenge:** Extracting relevant keywords and restructuring resumes for ATS compatibility.
- **Solution:**
    - Custom keyword extraction logic (extractKeywords) filters out common words and prioritizes technical terms.
    - Resume parsing and reconstruction functions (parseResumeIntoSections, reconstructResume) ensure output is well-formatted and sectioned.

4. Data Persistence & User Management

- **Challenge:** Storing user data securely and linking resumes to users.
- **Solution:**
    - Used Supabase for authentication and metadata.
    - Used MongoDB for storing large text data (resumes, job descriptions).

- API endpoints (app/api/save-resume/route.ts) handle saving to both databases and extracting job/company info for analytics.

Summary

I built a robust, full-stack AI resume optimizer using free, open-source AI models to avoid paid API costs, with careful handling of deployment and reliability challenges on Vercel. The system is user-friendly, secure, and designed for real-world ATS optimization.

Links

GitHub: **https://github.com/MianZainAllaudin/Nexium_ZainAllaudin_GrandProject**

Vercel: **https://airesumeoptimizer.vercel.app/**