Base Idea Used without brainstorm with AI

Problem:

Many freshly graduated or high school students struggle to find a job that's meaningful to them. Many young people never step foot into the field, this unfamiliarity makes them confused on what to put on their resume, what skill to highlight on their resume/portfolio, what necessary skill is required to grow in that field, and how to address ATS.

Target:

Freshly grad students

Features:

User profile: (understand forms) (store user data in json file can use seed data for template)

- Preferred job location
- Education level
- Ask user to put their experience with guided boxes (like a user survey to put their data, they can add project or experience guided not like a huge of text box)
- Ask user to put their skills

Resume analyzer(need AI then use hard coded prompt template personalized with user data)

- break the resume into parts. Like experience, profile, etc then change to json format (check notes stored in desktop stored in copilot) (tinyllma)
- Find relation between experience and goal job's description
    - Slice job desc into collection of skill sets using ai then compare with resume/user profile (can be done the same time with first point)
- Detect experience description impact
    - Use ai to determine if the description contain any valuable metric or is it detailed enough
- Proof read (stretch)
    - Detect any grammar error or typos
    - Detect repetition
    - Detect wordy points
- Provide score and cards that tell what need to be fixed
- ATS optimization (stretch)

- Career pathway guide
    - Quick stats (salary, required skill, industry, and work environment(remote or on site or hybrid), and employment rate up or down)
    - Provide a timeline of skillsets and experiences required to accomplish the target job
    - When card is clicked, provide info about course that can be taken if it's a skill or provide [get info using ai, ask ai to get info for specific metrics] if it's a job

Project Overview

- **Problem:** Many young students and recent graduates feel lost when trying to find a meaningful career path and struggle to build a resume that effectively showcases their skills. The unfamiliarity with a chosen field often leads to confusion about what to highlight and how to address modern hiring processes.
- **Target Audience:** Freshly graduated high school or college students.
- **Core Solution:** A local, all-in-one application that uses AI to analyze a user's resume, provide tailored feedback, and guide them on a clear career path.

MVP Technical Plan: Resume Analyzer, Local User Profile, & Career Pathway Guide

This plan outlines your three main components, all operating on the user's local machine.

WIREFRAME

User Profile (React Frontend + `localStorage`)

This component handles gathering and persisting user preferences and core data directly in the browser, eliminating the need for a separate backend for this data.

- **Frontend Tools:** React components, `useState` and `useEffect` hooks for data management.
- **Storage Mechanism:** Browser's `localStorage` API.
- **Process:**
    - **Data Collection (React):**
        - Create React components for the user profile, including guided input boxes for:
            - Preferred Job Location
            - Education Level
            - Experience (as guided inputs, not a large text box)
            - Skills
        - When the user inputs data, it's stored in the React component's local state.

    - **Data Persistence (React + `localStorage`):**
        - Upon saving (e.g., when a "Save" button is clicked or automatically on input change), the React application will convert the user's profile data (a JavaScript object) into a JSON string using `JSON.stringify()`.
        - This JSON string will then be saved to the browser's `localStorage` using `localStorage.setItem('userProfileData', jsonString)`.
        - When the application loads, a `useEffect` hook will check `localStorage` for `'userProfileData'`. If found, it will retrieve the string, parse it back into a JavaScript object using `JSON.parse()`, and use it to pre-fill the profile forms.
    - **Data Usage:** This stored `userProfileData` will be used as input for the Resume Analyzer's AI prompts, personalizing the analysis (e.g., by emphasizing skills relevant to the user's stated preferred job location or education level).
- **Premade Data for Testing:** For testing purposes, you can manually input data into the browser's `localStorage` through developer tools, or you can have a "load default" button that populates `localStorage` with a hardcoded JSON string for a sample user profile.

Resume Analyzer (Python Backend)

This is the core resume analysis functionality, executed by your Python scripts running on the user's machine. It takes a PDF resume, structures it, analyzes it against a job description, and generates feedback cards.

Stage 1: Text Extraction

Sources:

- https://www.youtube.com/watch?v=hqu5EYMLCUw
- https://www.youtube.com/watch?v=NoPS2YruQ1Y

This stage is the foundation of the entire process.

- **Tool: PyMuPDF (`fitz`)**. This is the best tool for this job due to its speed and ability to handle complex PDF layouts.
- **Process:** The Python backend uses PyMuPDF to extract all raw text from the user-uploaded PDF, combining it into a single string for the next stage.

Stage 2: Rule-Based Section Parsing

This stage is crucial for preparing the data for the LLM.

- **Tools:** Python's built-in string methods and the `re` (regular expressions) library.
- **Process:** The raw text from Stage 1 is scanned for common resume headers like 'Experience', 'Education', and 'Skills'. The text is then sliced into a dictionary where each key is a section name and the value is the corresponding text block.

Stage 3: AI-Powered Structured Parsing

This stage is dedicated to converting the raw text sections into a clean, structured JSON format using your tiny LLM.

- **LLM Tool: TinyLLaMA**.
- **Process:**
    - **Input:** The dictionary of text blocks from Stage 2.
    - **Prompt Engineering:** For each section (e.g., 'Experience', 'Education'), a highly specific prompt is sent to the TinyLLaMA model. This prompt includes the text block and a rigid JSON schema the model must follow.
    - **Execution:** The LLM parses each section and returns a JSON string, which your Python code then converts into a dictionary.
    - **Output:** A single, consolidated master JSON object that represents the entire resume's content in a structured, consistent format.

Stage 4: AI-Powered Analysis & Feedback Generation

This new, distinct stage takes the clean JSON from Stage 3 and the user's target job description to generate actionable feedback.

- **LLM Tool: TinyLLaMA**.
- **Process:**
    - **Input:** The master JSON object from Stage 3 and the target job description (which will also be provided by the user in the React frontend and passed to the Python process).
    - **Prompt for Analysis:** Targeted prompts are sent to the TinyLLaMA model to analyze specific parts of the structured resume data (e.g., individual achievement bullet points). The prompt asks for suggestions based on criteria like missing metrics or wordiness.
    - **Iterative Analysis:** Your Python code loops through relevant sections of the JSON object (e.g., 'achievements' in the 'experience' section). For each item, it sends a targeted prompt to the LLM.
    - **Generate Cards:** The output of each analysis prompt is used to generate the feedback "cards" in the specified JSON format (issue title, explanation, original text, and suggestion).
    - **Output:** A list of feedback "cards" (as a JSON array) sent from the Python process back to the React frontend for display.

Career Pathway Guide (Python Backend & React Frontend)

Backend Data Generation (Python & AI)

- **Process:** Your Python script will take a target job title from the frontend.
- **AI Queries:** Your script will send a series of simple, direct prompts to your local **TinyLLaMA** model. These prompts are crafted to retrieve specific information about the job.
    - **Quick Stats:** "What is the average salary range and top skills for a [Target Job Title]? Provide this as a JSON object."
    - **Timeline:** "Provide a simple, linear career path for a [Target Job Title]. List the job titles in the correct order, starting from an entry-level position."
    - **Skill Cards:** "For the skill '[Skill Name]', provide a short description of why it's important and list 3 free online resources to learn it. Return as a JSON object with 'title', 'summary', and 'resources'."
    - **Job Cards:** "What are the main responsibilities and a brief summary of a '[Job Title]'? Return this as a JSON object."
- **Output:** The script will consolidate all these JSON objects into a single file to be sent back to the React frontend. .

Frontend Presentation (React)

- **Process:** The React app sends the job title to the Python backend. When the backend's JSON output is received, the frontend will:
    - Display the **Quick Stats** from the `stats` field.
    - Render the timeline with a more detailed, skill-first structure: **skill 1 for job 1, skill 2 for job 1, job 1, skill 1 for job 2, etc.**
        - Your React frontend will receive a single, ordered list from the backend that contains both skills and jobs in the correct sequence. It will then loop through this list and render a card for each item.
    - Implement **Interactive Cards**: When a user clicks a card (either a skill or a job title), a pop up will appear with the detailed, bullet-pointed information from the corresponding section of the JSON object.