**Project Proposal: Smart AI Resume Screener & Career Advisor**

**Objective:**

To build an AI-powered platform that not only screens and ranks resumes based on job descriptions but also offers:

* Skill Gap Analysis
* AI Resume Enhancement Suggestions
* LinkedIn and GitHub Import Options
* Interview Readiness Scoring
* This tool not only benefits recruiters but also acts as a career advisor for job seekers.

**Key Features:**

**1. Resume Upload or LinkedIn/GitHub Import**

Allows uploading resumes in PDF/DOCX format.

Fetches profiles directly from LinkedIn or GitHub using APIs.

Parses GitHub for activity (commits, stars, repositories, languages).

**2. Smart Resume Matching & Ranking**

Uses NLP models like BERT or TF-IDF to analyze similarity with the job description.

Highlights matched and missing keywords.

Ranks candidates by relevance score.

**3. Skill Gap Analyzer**

Detects missing skills based on job requirements.

Recommends resources (courses, tutorials) to fill the gaps.

**4. AI Resume Enhancer**

Uses LLMs like GPT to suggest rewriting weak sections.

Offers tips like “Make this achievement more specific” or “Include project results.”

**5. Interview Readiness Score**

Analyzes overall strength of the resume, GitHub activity, skillset, and certifications.

Generates a readiness score and tips to improve further.

**Tech Stack:**

Frontend: React.js / HTML, CSS, Bootstrap

Backend: Flask / Django (Python)

NLP: spaCy, BERT, Sentence Transformers, GPT via OpenAI API

Parsing: pdfminer.six, docx2txt, BeautifulSoup (for LinkedIn), GitHub REST API

Database: MongoDB / SQLite

Others: HuggingFace Transformers, Fairness API (Fairlearn), GitHub Stats Analyzer

**Expected Output:**

Example Candidate A:

Match Score: 83%

Skill Gaps: Django, REST API

Resume Suggestion: "Add measurable outcomes to your project section."

Interview Readiness: 7.5/10

Example Candidate B (GitHub Import):

Match Score: 91%

GitHub Score: Active, 10+ repos, 3 with stars > 10

Final Interview Readiness: 9.2/10

**Visual Output:**

Includes pie charts, bar graphs, and highlighted feedback for both candidates and recruiters.

**Why This Project Stands Out:**

Integrates multiple modern technologies: ML, NLP, LLMs, REST APIs, and Visualization.

Solves real hiring and job-seeking problems.

Offers dual value: helps recruiters save time and helps candidates improve their chances.

Can evolve into a full product or startup MVP.

Innovative features (like GitHub analysis + AI resume enhancement) make it unique and stand out in hackathons or portfolio reviews.

**🔧 Core Modules & What You’ll Code**

**1. Frontend (React.js / HTML, CSS)**

**Code You'll Write:**

* Resume upload UI
* LinkedIn & GitHub import interface
* Dashboard for candidate match scores, charts, tips
* Visualization components (e.g., pie charts, bar graphs)

**Tools:**

* React.js or HTML/CSS
* Chart.js or Recharts for graphs
* Bootstrap or Tailwind for design

**FRONTEND DONE**

**2. Backend (Flask or Django in Python)**

**Code You'll Write:**

* REST API to:
  + Accept resume uploads
  + Call NLP functions
  + Return match scores, tips, readiness scores
* Authentication if needed (optional for MVP)

**Libraries:**

* Flask / Django
* Flask-RESTful or Django REST Framework

**3. Resume Parsing**

**Code You'll Write:**

* Parse resume files (PDF, DOCX)
* Extract:
  + Name, skills, experience, education, etc.

**Libraries:**

* pdfminer.six (PDF)
* docx2txt (DOCX)
* PyPDF2 (optional alternative)

**4. LinkedIn/GitHub Integration**

**Code You'll Write:**

* Fetch data using APIs
* Analyze:
  + GitHub repos, stars, commits
  + LinkedIn text (if accessible)

**Libraries/APIs:**

* GitHub REST API (use Python’s requests)
* LinkedIn API (difficult unless using user access)

**5. Resume Matching (NLP Module)**

**Code You'll Write:**

* Compare resume with job description
* Highlight missing/matched keywords
* Calculate match score

**Libraries:**

* spaCy, BERT via HuggingFace, sentence-transformers
* scikit-learn (for TF-IDF)

**6. Skill Gap Analyzer**

**Code You'll Write:**

* Identify missing skills
* Suggest learning resources (you can hardcode a few or integrate Coursera/edX APIs)

**7. AI Resume Enhancer (LLM Integration)**

**Code You'll Write:**

* Send resume text to OpenAI GPT API
* Get suggestions and rewrites

**Tools:**

* OpenAI GPT-4 API
* Python openai library

**8. Interview Readiness Scorer**

**Code You'll Write:**

* Use logic to assign a score (e.g., 0–10) based on:
  + Resume strength
  + GitHub activity
  + Match score
  + Skill coverage

**9. Database (MongoDB or SQLite)**

**Code You'll Write:**

* Store:
  + User profiles
  + Uploaded resumes
  + Scores and feedback

**10. Visual Output**

**Code You'll Write:**

* Graphs on dashboard (e.g., using Chart.js)
* Highlighted resume feedback (color-coded sections)

**✅ Suggested Order of Implementation:**

1. Resume Upload + Parsing
2. Job Description Matching (NLP)
3. GitHub Integration
4. Resume Matching + Score
5. AI Resume Enhancer (OpenAI)
6. Skill Gap & Course Suggestions
7. Interview Readiness Logic
8. Frontend Dashboard & Visuals
9. Polish UI + Export PDF Feature

**Step-by-Step Setup:**

1. **Created the Main Project Folder:**

mkdir resume-screener

cd my-resume-screener

1. **Create Frontend Folder (React App):**

npx create-react-app frontend/resume-screener

1. **Set Up Backend Folder (Flask):**

mkdir backend

cd backend

python3 -m venv venv

venv\Scripts\activate for Windows

1. **Install Python Dependencies for Flask:**
   * **Created a requirements.txt file in the backend folder with the following dependencies:**

flask

flask-cors

pdfminer.six

docx2txt

openai

spacy

sentence-transformers

* + **Installed them by running:**

pip install -r requirements.txt