**TalentMatch AI - MVP Blueprint & Execution Roadmap**

**🚀 1. Overview**

**TalentMatch AI** is an AI-powered hiring SaaS platform that deeply analyzes a candidate’s resume, GitHub activity, and competitive programming (CP) profiles to verify technical skills, reduce hiring bias, and provide data-backed candidate insights.

**🌟 MVP Objective:**

Build a Minimum Viable Product (MVP) that takes in a candidate's resume, GitHub profile, and CP profile URLs, runs analysis via LLM and scraping logic, and returns a verified skill report.

**🏢 2. Core Features for MVP**

**✅ User Flow:**

1. User signs up/logs in
2. User uploads resume (PDF/DOCX)
3. User enters GitHub + CP profiles (e.g., Leetcode, Codeforces)
4. Backend scrapes profile data and parses resume
5. LLM analyzes all data to generate a personalized, verified skill report
6. User can view/download report from their dashboard

**🔹 MVP Feature List:**

* JWT-based Auth System (Signup/Login)
* Resume File Upload + Text Parsing (DOCX, PDF)
* GitHub Repo Scraping (API + Public HTML)
* CP Profile Stats (Leetcode, Codeforces via scraping or APIs)
* AI-Powered Analysis (OpenAI API or local LLM)
* Report Generation (Summary, skill scores, insights)
* Dashboard UI (Upload resume, view past reports)

**💡 3. Tech Stack**

**Backend (FastAPI):**

* Python 3.10+
* FastAPI
* Pydantic
* Unicorn
* PyPDF2 / docx2txt
* OpenAI API / LLM wrappers
* Scrapping through official GitHub API

**Frontend (Next.js):**

* React + TypeScript
* Tailwind CSS
* Axios (API calls)

**Database:**

* PostgreSQL (via SQL Alchemy or Tortoise ORM)
* Optionally: Redis (for caching scrape results or queues)

**Auth:**

* JWT-based access and refresh token system

**Deployment:**

* Railway / Render for MVP
* GCP VM or Dockerized deployment for custom hosting

**🗃️ 4. Backend Folder Structure (Suggested)**

/app

/routers

- auth.py

- resume.py

- github.py

- cp\_profiles.py

- report.py

/services

- parser.py

- github\_scraper.py

- cp\_scraper.py

- ai\_analyzer.py

/models

- user.py

- report.py

/utils

- jwt\_handler.py

- file\_handler.py

main.py

**✨ 5. Stretch Goals (Post-MVP)**

* Recruiter-side dashboard to compare multiple candidates
* AI-based interview prep suggestions from skill gaps
* Job-matching engine using candidate skill graphs
* Admin dashboard to monitor users + feedback analytics
* Payment integrations (Stripe/Razorpay) for premium reports

**✨ 6. Notes for Dev Rhythm**

* Prioritize working features over perfect UI
* Timebox each module (ex: resume parser = 6 hrs max)
* Commit small and often; test each API route with Swagger
* Share MVP early with 3–5 dev friends for feedback

**⚡ Final Thought**

MVP is not about perfection. It's about proving *you can solve the problem*. Keep it lean, clean, and real. Let it cook 🔥