AI Resume Screener (Chatbot Style)

- Tech Stack: Flask | Python | SentenceTransformers | PyMuPDF | OCR | HTML/CSS
- Goal: Automatically screen and rank resumes based on Job Description (JD)

Problem Statement

- Challenges in Resume Screening:
- - HR spends hours manually reading resumes
- - Hard to compare multiple candidates efficiently
- - Keyword-based search misses semantic meaning
- - Scanned PDFs / DOCX files require extra effort

Solution

- Our AI Resume Screener:
- - Upload Job Description + multiple Resumes
- - System calculates meaning-based similarity between JD & resumes
- - Returns ranked candidates in chatbot-style interface
- - Handles PDF, DOCX, and OCR for scanned PDFs

How It Works (Flow)

- 1. Frontend:
-    - Chat-style UI
-    - Shows user uploads & bot responses dynamically

- 2. Backend (FastAPI):
-    - Extracts text from PDF/DOCX (PyMuPDF + OCR + python-docx)
-    - Encodes JD & resumes with SentenceTransformer embeddings
-    - Computes cosine similarity → % match
-    - Returns sorted results as JSON

Demo Output

- Example Output:
- - 📊 Top Candidates:
-   - ✅ Ramu_resume.docx — 49.24% match
-   - ✅ Raju_resume.docx — 49.11% match
-   - ✅ Bhanu Prakash Reddy.pdf — 47.06% match

Benefits / Advantages

- - Speeds up resume screening
- - Supports multiple file formats (PDF/DOCX)
- - meaning-based matching (understands meaning, not just keywords)
- - Chatbot-style interactive UI
- - Scalable — multiple resumes at once

Limitations

- - Only provides % match, no skill gap analysis yet
- - Scanned PDFs may need OCR
- - Doesn’t explain why a candidate is better
- - No ML classifier (just similarity scores)
- - Not integrated with ATS systems

Future Roadmap / Next Features

- 1. Skill Gap Analyzer → Highlight Matched ✅ & Missing ❌ Skills
- 2. Job Fit Prediction → Hire / Not Hire probability
- 3. Q&A HR Assistant → Ask questions like 'Who is best for Python Developer?'
- 4. Candidate Comparison → Side-by-side skill comparison
- 5. AI Cover Letter Generator → Auto-generate custom cover letter

Tech Stack

- - Backend: Python, FastAPI
- - AI: SentenceTransformers (all-MiniLM-L6-v2)
- - PDF / DOCX Handling: PyMuPDF, python-docx,
- Tesseract OCR
- - Frontend: HTML, CSS (chatbot-style UI), JS for dynamic updates

Takeaways

- - Saves HR time
- - Interactive & visual results
- - meaning-based understanding > keyword matching
- - Scalable & extensible for future AI HR features