AI-Powered ATS Resume Analyzer

🎯 Objective

To develop a streamlined web-based tool that automates resume-job matching using AI, delivering fast and insightful compatibility feedback between resumes and job descriptions.

🛠 Technologies Used

Component	Stack / Tool
Frontend UI	Streamlit
File Handling	pdfplumber, docx2txt
Text Analysis	Google Gemini 2.0 Flash (LLM)
API Interaction	google-generativeai, requests
Environment Mgmt	python-dotenv

✨ Features

- **Resume & JD Input**:
  - Upload resumes in **PDF** or **DOCX** format.
  - Paste job descriptions directly into a text area.
- **Automatic Resume Parsing**:
  - Extracts text using pdfplumber and docx2txt.
- **Gemini AI Evaluation**:
  - Sends resume and job description to **Gemini 2.0 Flash**.
  - Returns:
    - **Section-wise breakdown** of resume (Contact Info, Skills, etc.)

        - **Overall match score (0–100%)**
- **Match Score Display**:
    - Prominent, color-coded score with progress bar.

🧠 AI Analysis Metrics

- Semantic alignment between resume and job description.
- Section identification:
    - 📌 Contact Info
    - 💼 Experience
    - 🎓 Education
    - 🛠️ Skills
    - 📃 Certifications
    - 🏆 Achievements / Others
- Match score is a percentage based on semantic similarity, extracted via re parsing from AI response.

📂 Directory Structure

```
ats_resume_matcher/
├── app.py              # Main application
├── requirements.txt    # Dependencies
├── .env                # Stores Gemini API key securely
├── README.md           # Setup & usage instructions
```

⚙️ How to Run

```
# Clone the repo
```

```
git clone <your-repo-url>

# Setup virtual environment
python -m venv venv
source venv/bin/activate      # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Add Gemini API Key
echo GEMINI_API_KEY=<your_api_key> > .env

# Run the Streamlit app
streamlit run app.py
```

📜 Implementation Highlights

- ✅ Gemini prompt is crafted to extract both **resume structure** and a **match score** in a predictable format.
- 🔍 Regular expressions extract numeric score and split sections.
- 🔄 Analysis is automatically re-triggered on input change using st.session_state.
- 🧩 Modular design separates UI, parsing, AI interaction, and state logic.

🌐 High-Level Architecture

1. **Frontend (Streamlit)**:
    - Upload area for resume
    - Text box for job description

    - Real-time results display
2. **Backend Logic**:
    - Resume Parser (PDF/DOCX)
    - Gemini Prompt Constructor
    - Response Handler (Score + Sections)
3. **AI API**:
    - Google Gemini 2.0 Flash
    - Handles semantic analysis and structured output

🧪 Results

- 🚀 Fast and consistent AI-based screening
- ✅ Accurate parsing of diverse resume formats
- 📊 Clear match feedback for job seekers and recruiters
- 👩‍💼 Seamless one-click analysis experience

🔮 Future Scope

- 📥 Batch resume processing
- 🎯 Weighted score customization by section
- 💬 Sentiment and keyword highlighting
- 📤 Export analysis as downloadable PDF
- 🗂 Resume & JD library with user accounts

👨‍🏫 Contributors

- Charan Goriparthi
- GitHub Repository