Intelligent Resume Analysis & Career Guidance

Intelligent Resume Analysis and Career Guidance System

Project Description
This project is a local-only web application that analyzes a resume (PDF/DOCX) using NLP and ML, compares it with target job roles, identifies skill gaps, and recommends courses/certifications. It is designed for final-year students and academic evaluation with simple, explainable logic.

Features
- Resume upload (PDF/DOCX) with validation and local storage
- NLP extraction of skills, education, experience, projects
- Job-role matching using skill overlap and TF-IDF similarity
- Skill-gap analysis (matched vs missing skills)
- Career guidance and course recommendations
- Downloadable analysis report

Tech Stack
- Backend: Python 3.10+, Flask
- NLP: SpaCy, NLTK
- ML: Scikit-learn (TF-IDF + Cosine Similarity)
- Parsing: pdfplumber, docx2txt
- Database: SQLite
- Frontend: HTML, CSS, Bootstrap, basic JS

System Architecture (ASCII)
```
User -> Web UI (HTML/Bootstrap)
        |
        v
      Flask App
  / Upload + Analyze
        |
        v
 Resume Parser (PDF/DOCX)
        |
        v
 NLP Extractor (SpaCy + NLTK)
        |
        v
 Skill/Role Matcher + TF-IDF Similarity
        |
        v
 Recommendation Engine
        |
        v
 SQLite + Report Export
        |
        v
 Result Dashboard
```

UML Diagrams (Explained)

Use Case (Textual)
- Student uploads resume
- System extracts skills/education/experience
- Student selects job role
- System matches resume with role and JD
- System shows missing skills + recommendations
- Student downloads report

Class Diagram (Key Classes)
- `ResumeParser` -> extract text from PDF/DOCX
- `NLPExtractor` -> extract skills, education, experience, projects
- `Matcher` -> TF-IDF similarity + skill scoring
- `RecommendationEngine` -> course/certification recommendations
- `Database` -> store analysis results

Sequence Diagram (Simplified)
```
User -> UI: Upload resume + role
UI -> Flask: /analyze
Flask -> Parser: extract_text()
Parser -> NLPExtractor: build_summary()
NLPExtractor -> Matcher: scores
Matcher -> RecommendationEngine: recommend()
Flask -> DB: save_analysis()
Flask -> UI: render result
```

Data Flow Explanation
1. Resume file uploaded and stored locally.
2. Text extracted using pdfplumber/docx2txt.
3. NLP extraction identifies skills and sections.
4. Role comparison and TF-IDF similarity computed.
5. Skill gaps and recommendations generated.
6. Results stored in SQLite and shown in dashboard.

Folder Structure
```
resume-analyzer/
■■■ app.py
■■■ models/
■■■ utils/
■■■ templates/
■■■ static/
■■■ database/
■■■ datasets/
■■■ requirements.txt
■■■ README.md
```

Installation (Local Only)
1. Create a virtual environment (recommended).
2. Install dependencies:

```
pip install -r requirements.txt
```
3. Download SpaCy model:
```
python -m spacy download en_core_web_sm
```

How to Run
```
python app.py
```
Open browser at `http://127.0.0.1:5000`

Sample Input & Output
- Sample resumes are stored in `samples/`
- Example output: resume score, skill match %, missing skills, and course recommendations.

Screenshots
- After running the app, capture 2-3 UI screenshots (upload page and results page) and add them here for submission.

Datasets
- `datasets/job_roles.json` - predefined roles and skills
- `datasets/job_descriptions.csv` - sample job descriptions
- `datasets/skills.json` - skill dictionary
- `datasets/courses.csv` - course recommendations

Future Scope
- Add ATS-style formatting feedback
- Multi-language resume support
- PDF report export
- Integration with local job boards

Notes for Viva
- TF-IDF builds term vectors of resume and JD.
- Cosine similarity measures closeness of vectors.
- Skill match score is weighted by required vs preferred skills.
- No deep learning models used.