Candidate-Recommendation-Engine Outline

https://github.com/Jordan-Swartz/candidate-recommendation

- ***Stack Responsibility Break Down:***
    1. **Frontend/Deployment**
    - Streamlit → platform to receive input data and display results
        - Accept Job Description (text)
        - Accept list of candidate resumes (text or file upload)
        - Display top 5-10 most relevant candidates with data

    2. **Backend**
    - Python → backend language
        - Python 3.9.6

    3. **Libraries/Frameworks**
    - PyMuPDF →
        - Parse PDFs
    - Sentence-Transformers →
        - Generate Embeddings
    - Scikit-Learn →
        - Compute Cosine Similarity
    - Hugging Face Transformers →
        - Generate AI summary for persons fit (pros/cons)

- ***Deployment + Environment Setup: (Streamlit + Python):***
    1. **Set up environment**
        - Check python download:
            - `python3 -version`
        - Check pip download (package manager to install third-party libraries like Streamlit):
            - `pip3 -version`
        - Install PyCharm CE
        - Create Virtual Environment (VE) for project-specific packages in desired folder location:
            - `python3 -m venv venv` (1st venv creates the environment and 2nd venv is the subfolder inside of that environment that contains the VE python and package files)
            - This is best practice to avoid dependency issues between projects, similar to using a gradle wrapper for Java
            - To activate venv environment: `source venv/bin/activate`
        - Install packages in VE
            - Streamlit: `pip install streamlit`

- Sentence-Transformers: `pip install sentence-transformers`
- Scikit-learn: `pip install scikit-learn`
- Hugging Face: `pip install 'transformers[torch]'`
- PDF Reader: `pip install pymupdf`

- ***Research:***
  - Streamlit:
    - https://streamlit.io/playground?example=geospatial - Playground Examples
  - Generating Embeddings:
    - https://www.datastax.com/blog/how-to-create-vector-embeddings-in-python - Demo
    - Demo:

```
from sentence_transformers import SentenceTransformer


model = SentenceTransformer("all-MiniLM-L6-v2")
sentence = "A robot may not injure a human being or, through inac
embedding = model.encode(sentence)

print(embedding)
# => [ 1.95171311e-03  1.51085425e-02  3.36140348e-03  2.48030387
```

    - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html - Scikit Documentation
  - Cosine Similarity:
    - https://www.geeksforgeeks.org/dbms/cosine-similarity/
    - Given two inputs of text converted into non-zero vectors, calculate the cosine of the angle between those two vectors. The smaller the angle, the higher the cosine similarity value will be (-1 to 1, where 1 is an identical match).
    - Formula: $$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$
  - AI Generated Summary:
    - https://huggingface.co/docs/transformers/installation
      - Installation (transformers and pytorch)
    - https://huggingface.co/docs/transformers/pipeline_tutorial
      - Pipeline documentation for summary