Here is a detailed step-by-step guide to build the "Smart Resume Parser" Python project following the given objective and tools:

**1. Extract Text from Resumes**

* For PDFs: Use PyMuPDF (fitz) to open PDF files and extract all text.
* For DOCX files: Use the python-docx library to extract text from DOCX documents.

**2. Clean and Preprocess Text**

* Remove unnecessary newlines, extra spaces, and special characters.
* Convert text to lowercase for normalization (optional depending on extraction).
* Split text into lines or sections for easier parsing.

**3. Extract Information Using spaCy + Regex**

* Load a spaCy NLP model (e.g., en\_core\_web\_sm).
* Use spaCy Named Entity Recognition (NER) to extract entities like education (degrees, institutions), experience (organizations, dates), and others.
* Use regex patterns to locate sections like "Skills", "Education", "Experience" if labeled in text.
* Extract skills by comparing words in the text with a predefined skill list.

**4. Organize Output**

* Compile extracted information into a Python dictionary with keys like skills, education, experience.
* Convert this dictionary to JSON format for structured output.
* Optionally create a Pandas DataFrame for tabular representation.

**5. Build Streamlit UI**

* Use st.file\_uploader to upload PDF or DOCX resumes.
* Display extracted information in readable format (tables or JSON).
* Provide buttons to export extracted data to CSV or JSON files.

**6. Deliverables**

* Full Python codebase with modules for extraction, parsing, and UI.
* Streamlit app for interactive resume upload and parsing.
* Include 5 diverse test resumes in PDF and DOCX formats.
* Exported output files demonstrating results in CSV and JSON formats.

**Example Code Outline:**

python

**import** fitz *# PyMuPDF*

**import** docx

**import** spacy

**import** re

**import** json

**import** pandas **as** pd

**import** streamlit **as** st

*# Step 1: Extract text*

**def** extract\_text\_from\_pdf(file\_path):

doc = fitz.open(file\_path)

text = ""

**for** page **in** doc:

text += page.get\_text()

**return** text

**def** extract\_text\_from\_docx(file\_path):

doc = docx.Document(file\_path)

text = "\n".join([para.text **for** para **in** doc.paragraphs])

**return** text

*# Step 2: Clean text*

**def** clean\_text(text):

text = re.sub('\n+', '\n', text) *# Remove excessive newlines*

text = re.sub('[^a-zA-Z0-9\n,.]', ' ', text) *# Remove special chars except common punctuation*

**return** text.lower()

*# Step 3: Extract info using spaCy + regex*

nlp = spacy.load('en\_core\_web\_sm')

skills\_list = ['python', 'java', 'sql', 'machine learning', 'data analysis'] *# Example skills*

**def** extract\_skills(text):

skills\_found = [skill **for** skill **in** skills\_list **if** skill **in** text]

**return** skills\_found

**def** extract\_education(text):

education = []

edu\_pattern = re.compile(r'(bachelor|master|ph\.d|diploma|degree)', re.I)

**for** line **in** text.split('\n'):

**if** edu\_pattern.search(line):

education.append(line.strip())

**return** education

**def** extract\_experience(text):

experience = []

exp\_pattern = re.compile(r'(internship|engineer|developer|manager|consultant|experience)', re.I)

**for** line **in** text.split('\n'):

**if** exp\_pattern.search(line):

experience.append(line.strip())

**return** experience

*# Step 4: Organize output*

**def** parse\_resume(text):

cleaned = clean\_text(text)

**return** {

"skills": extract\_skills(cleaned),

"education": extract\_education(cleaned),

"experience": extract\_experience(cleaned)

}

*# Step 5: Streamlit UI*

**def** main():

st.title("Smart Resume Parser")

uploaded\_file = st.file\_uploader("Upload Resume (PDF or DOCX)", type=['pdf', 'docx'])

**if** uploaded\_file:

**if** uploaded\_file.type == 'application/pdf':

text = extract\_text\_from\_pdf(uploaded\_file)

**else**:

text = extract\_text\_from\_docx(uploaded\_file)

parsed = parse\_resume(text)

st.subheader("Parsed Information")

st.json(parsed)

**if** st.button("Export to JSON"):

**with** open("parsed\_resume.json", "w") **as** f:

json.dump(parsed, f)

st.success("Exported to parsed\_resume.json")

**if** st.button("Export to CSV"):

df = pd.DataFrame({k: pd.Series(v) **for** k, v **in** parsed.items()})

df.to\_csv("parsed\_resume.csv", index=False)

st.success("Exported to parsed\_resume.csv")

**if** \_\_name\_\_ == "\_\_main\_\_":

main()

This provides the core implementation to extract and parse resume contents and a UI for uploads and export.