Scope of Work: Named Entity Recognition (NER) for Resume Parsing Project

1. Objective:

The main goal of this project is to develop an NLP model that automates the extraction of key information from resumes using Named Entity Recognition (NER). The output will provide recruiters with structured data (e.g., Name, Education, Skills, Contact Info) from resumes, streamlining the resume screening process.

---

2. Project Deliverables:

- **Resume Text Extraction**: A script or application that extracts text from `.pdf` and `.docx` resume files.
- **NER Model**: A SpaCy-based NLP model to identify and extract entities like name, education, skills, and experience from resumes.
- **Contact Info Extraction**: A function to identify contact details (email, phone) using regular expressions (regex).
- **Structured Output**: Organize the extracted information into structured formats such as JSON or CSV.
- **Optional**:
  - Web-based interface for uploading resumes and displaying extracted information.
  - Integration with recruitment systems.
  - Model fine-tuning for domain-specific entities.

---

3. Tasks and Activities:

Phase 1: Requirements Gathering and Setup

- **Task 1.1**: Define the key entities to extract (e.g., Name, Education, Skills, Contact Info, Experience).
- **Task 1.2**: Identify tools and libraries for the project (SpaCy, Python-docx, PyPDF2, NLTK).
- **Task 1.3**: Set up the project environment with necessary dependencies (e.g., SpaCy, PDF/DOCX libraries).

Phase 2: Data Collection and Preprocessing

- **Task 2.1**: Collect a dataset of resumes in `.pdf` or `.docx` formats.

- **Task 2.2**: Build functions to convert resumes from PDF/DOCX to plain text.
- **Task 2.3**: Clean and preprocess the text (tokenization, removing noise).

Phase 3: NER Model Development

- **Task 3.1**: Build a basic NER pipeline using SpaCy's pre-trained model (`en_core_web_sm`).
- **Task 3.2**: Extract general entities like `PERSON` , `ORG` , `DATE` (Name, Education, Experience, etc.).
- **Task 3.3**: Fine-tune or customize the NER model to extract domain-specific entities (e.g., Skills, Certifications) if required.

Phase 4: Contact Information Extraction

- **Task 4.1**: Implement regex-based functions to extract email and phone numbers.
- **Task 4.2**: Integrate this into the pipeline with the NER model to consolidate extracted information.

Phase 5: Output Structuring

- **Task 5.1**: Organize the extracted data (Name, Contact Info, Education, etc.) into structured formats such as JSON, CSV, or database-friendly formats.
- **Task 5.2**: Build a function to display the extracted information clearly for recruiters or users.

Phase 6: Testing and Evaluation

- **Task 6.1**: Test the model on unseen resumes to evaluate entity extraction performance.
- **Task 6.2**: Evaluate precision, recall, and F1-score of the model for key entities (Skills, Education, Experience).
- **Task 6.3**: Refine the model based on feedback and errors (adjust regex or fine-tune the NER model).

Phase 7: Optional Features

- **Task 7.1**: Develop a web-based interface using FastAPI or Flask for resume upload and automatic parsing.
- **Task 7.2**: Allow users (recruiters) to upload resumes via a web form and view the parsed results.
- **Task 7.3**: Provide an API for integrating this system into larger recruitment platforms.

Phase 8: Documentation

- **Task 8.1**: Provide detailed documentation for:
  - Setup instructions for the resume parsing system.
  - Usage of NER functionality and the parsing interface.
  - Steps to customize or extend the system.

- **Task 8.2**: Include details on deployment (if deploying the web-based app).

---

4. Timeline and Milestones:

- **Week 1**:
  - Finalize requirements, environment setup, and data collection.
  - Complete resume text extraction and preprocessing.
- **Week 2**:
  - Build the basic NER model and integrate regex-based contact info extraction.
  - Organize the extracted data into structured formats (JSON/CSV).
- **Week 3**:
  - Test and evaluate the model on different resume samples.
  - Fine-tune the NER model for accuracy and performance.
- **Week 4** (Optional):
  - Develop the web-based interface and API.
  - Deploy the application to a cloud platform (Heroku, AWS, etc.).
- **Week 5**:
  - Final testing, documentation, and project delivery.

---

5. Dependencies:

- **Libraries**: SpaCy, NLTK (optional), python-docx, PyPDF2.
- **Python Environment**: Ensure all dependencies are compatible with the Python version being used.
- **Web Hosting**: For optional web-based features, cloud hosting (Heroku, AWS) may be required.

---

6. Risks and Mitigations:

- **Data Quality**: If resumes vary widely in format, entity extraction accuracy could decrease. Mitigate by training on diverse resumes.

- **PDF Parsing**: Some PDF files may not convert well to text. Use reliable PDF parsing libraries and test on different resume formats.
- **Regex Inaccuracy**: Regular expressions for contact info (email/phone) might fail on complex resumes. Regularly update and refine the regex patterns.

---

7. Success Criteria:

- Successful extraction of key resume entities with acceptable precision and recall (above 90% for Name, Education, Skills).
- The system can process resumes and output structured information in less than 5 seconds for each file.
- Optional: The system integrates smoothly with a web-based recruitment platform.