Subtask 1: Clearly Define the Problem Statement

Problem Statement:

Long resumes often contain irrelevant information when compared to a specific job description (JD), making them less effective in the job application process. The goal is to build a web application that can automatically analyze a long resume, match it with a given JD, and generate an optimized version of the resume that highlights the most relevant sections.

Subtask 2: Identify the Input and Output

Input:

1. **Resume:**
    - Format: PDF, DOCX, or plain text.
    - Content: Personal details, education, experience, skills, certifications, projects.
2. **Job Description (JD):**
    - Format: Plain text or DOCX.
    - Content: Required qualifications, skills, roles, and responsibilities.

Output:

1. **Optimized Resume:**
    - Highlighted sections that are most relevant to the JD.
    - Reduced content focusing only on key qualifications.
2. **Similarity Score:**
    - A percentage indicating how well the resume matches the JD.
3. **Downloadable File:**
    - Format: PDF or DOCX.
    - Contains both the original and the optimized version.

Subtask 3: List Features

Core Features:

1. **Resume and JD Upload:**
    - Drag-and-drop file upload functionality.
    - Support for multiple file formats (PDF, DOCX).
    - Clear error messages for unsupported formats.
2. **Text Extraction and Preprocessing:**
    - Extract text from PDF/DOCX using libraries like PyPDF2 and python-docx.
    - Clean and preprocess text (remove special characters, stopwords, etc.).

   - Tokenization and lemmatization to normalize the text.

3. **Text Matching and Relevance Calculation:**
   - Use NLP techniques (TF-IDF, cosine similarity) to find common phrases between resume and JD.
   - Calculate the similarity score to indicate relevance.
   - Highlight sections in the resume that match the JD.
4. **Results Visualization:**
   - Side-by-side display of the original and optimized resume.
   - Highlighted text sections to indicate relevance.
   - Display the similarity score on the result page.
5. **Download Optimized Resume:**
   - Allow users to download the processed resume.
   - Option to choose between PDF and DOCX formats.
6. **Data Management and Security:**
   - Store resumes securely in MongoDB.
   - Ensure data privacy and confidentiality.
7. **User Interface:**
   - Minimal and intuitive design using React and Tailwind.
   - Real-time progress indicator during processing.
   - Error handling and user guidance (e.g., tooltips).

Outcome:

With this detailed breakdown, you have a clear understanding of the problem, the inputs and outputs, and the essential features to be implemented.