Software Requirements Specification (SRS) for Large Language Model (LLM) Fine-Tuning Tool.

1. Introduction

   1.1 Purpose
   The purpose of this document is to define the software requirements for a Large Language Model (LLM) Fine-Tuning Tool. This tool is designed to generate human-like responses in a secure environment not connected to the internet. It includes AI/ML-based text summarization, grammar checking, and various other functionalities, aiming to extend the LLM's capabilities beyond existing models. This project aligns with the UN Sustainable Development Goal 9 (Industry, Innovation, and Infrastructure).

   1.2 Scope
   The LLM Fine-Tuning Tool will be developed as a stand-alone software application that provides text summarization, grammar checking, and content reformatting for Science and Technology documents, newspaper headlines, and editorial pages. It will also offer additional capabilities based on fine-tuned LLM models, ensuring contextual integrity in outputs. The system is evaluated on functionality, ease of use, flexibility, scalability, and compatibility with various open-source LLM models.

   1.3 Definitions, Acronyms, and Abbreviations
   - **LLM**: Large Language Model
   - **ML**: Machine Learning
   - **AI**: Artificial Intelligence
   - **NLP**: Natural Language Processing
   - **API**: Application Programming Interface
   - **SDG**: Sustainable Development Goals
   - **GDPR**: General Data Protection Regulation

   1.4 References
   - IEEE SRS Guidelines
   - Documentation for Pegasus and other open-source LLM models
   - UN Sustainable Development Goals, specifically SDG 9
   - Research papers on text summarization, grammar checking, and contextual NLP
   - Technical documentation for Huggingface transformers library

2. Overall Description

   2.1 Product Perspective
   The LLM Fine-Tuning Tool is a self-contained application aimed at deploying AI/ML models for offline text analysis. It leverages pre-trained LLMs fine-tuned for specific tasks like summarization, grammar checks, and reformatting, ensuring adaptability to various user needs.

   2.2 Product Features
   - **Text Summarization**: Fine-tuned models to generate concise summaries of Science and Technology documents.
   - **Grammar Checking**: Context-aware grammar checking for document-level integrity.
   - **Content Reformatting**: Reformat content to match specific style guidelines while maintaining context.
   - **Offline Functionality**: Operates on networks not connected to the internet for secure data handling.
   - **Model Flexibility**: Compatibility with multiple open-source LLMs.
   - **Customization**: Fine-tuning capability for user-specific needs.

   2.3 User Classes and Characteristics
   - **Data Scientists**: Users interested in experimenting with LLM fine-tuning.
   - **Researchers**: Users working with large datasets in science and technology.
   - **Editors and Journalists**: Users focusing on summarising and grammar-checking editorial content.
   - **Developers**: Users seeking to integrate LLM functionalities into offline systems.

   2.4 Constraints
   - Must operate offline without internet access.
   - Ensure GDPR compliance for user data processing.
   - Maintain high accuracy in summarization and grammar checking across multiple document types.

3. Specific Requirements

   3.1 Functional Requirements
   1. **Text Summarization**:
      - Fine-tune Pegasus and other models for summarising Science and Technology documents.

- Support bulk document processing.

2. **Grammar Checking**:
   - Enable grammar corrections with contextual awareness.
   - Allow user-defined style guidelines for grammar checks.
3. **Content Reformatting**:
   - Provide reformatting options while maintaining document integrity.
   - Support template-based content formatting.
4. **Model Flexibility**:
   - Allow users to switch between different open-source LLM models.
   - Support model updates and retraining based on user input.
5. **User Interface**:
   - Create a user-friendly interface using Flask for web-based interaction.
   - Allow for detailed feedback on model outputs for further fine-tuning.

3.2 Non-Functional Requirements

1. **Performance**:
   - Summarization and grammar check response times should not exceed 3 seconds per document.
   - Support up to 10,000 documents for batch processing.
2. **Security**:
   - Ensure secure storage and processing of documents.
   - Provide encryption for sensitive user data during offline operations.
3. **Usability**:
   - Intuitive UI with minimal learning curve.
   - Provide documentation and tutorials for users.
4. **Compatibility**:
   - Compatible with major operating systems (Windows, Linux, macOS).
   - Ensure flexibility to work with multiple LLMs.

4. System Models

4.1 Flowchart

A high-level flowchart illustrating:

1. User uploads document(s) for processing.
2. System processes the document(s) based on user-selected tasks (summarization, grammar check, reformatting).
3. Outputs are generated and displayed to the user.
4. User provides feedback for further fine-tuning.

4.2 System Architecture

- **Frontend**: Developed using Flask for an interactive UI.
- **Backend**: Python-based, leveraging Huggingface for model handling.
- **Model Layer**: Supports Pegasus and other open-source models for NLP tasks.
- **Storage**: Local file storage for documents, ensuring offline access.

5. Other Requirements

   5.1 Legal and Ethical Concerns

   - Ensure compliance with data protection regulations like GDPR.
   - Transparent handling of user data with a clear privacy policy.

   5.2 Assumptions and Dependencies

   - Users have access to machines capable of running fine-tuned models.
   - Availability of open-source LLM models for fine-tuning.
   - Users understand the basics of AI/ML for interaction with the system.