System Specification: LyrIAcs web service

1. **Introduction**
    1. **Purpose of the requirements document**

        The primary goal of this document is to outline the system specifications for an AI-powered web service that allows users to input some lines of text and a musical genre/emotion and generates the continuation of those lines as if they were the start of a song of that genre/emotion. This document serves as a comprehensive guide for the team of software developers and data engineering working on this project. It aims at defining all the key features of the web-service, as well as the little details concerning the user input and the user output and the non-functional requirements.

        Based on the team decision, this document may be updated to represent at any moment the current aim of the project.

    2. **Scope of the product**

        The product is an AI-powered web-service called “LyrIAcs”. It receives some lines of text and, based on this input, offers some genres and emotions from which the user can choose. After this selection, the web-service offers the continuation of the text as if the input lines were the first lines of a song of the selected genre and expressing the selected emotion. At this point, the user can choose to stop and save the lyrics or to generate the next chunk of lyrics or to generate again the lines proposed by the web-service.

    3. **Definitions, acronyms, and abbreviations**

        To ensure a clear communication throughout this document, the following recurring terms are defined here:

        - ***Emotion/Sentiment***: these terms are used interchangeably in the document, specifically they comprehend the following emotions:
        - ***Genre:*** musical genre specifically, the web-service takes into account the following genres:
        - ***AI***: Artificial Intelligence, part of the name of the web service and used as the main tool for offering the service.
        - ***Web API***: Application Programming Interface for either a web server or a web browser, specifically it is used to connect to a web server and send requests for a service.
        - ***NLP***: Natural Language Processing, a branch of machine learning that aims at extracting information from text.
        - ***LLM***: Large Language Models, a type of machine learning model designed for NLP tasks such as language generation.
2. **Overall Description**
    1. **Product perspective**

This web service should prove useful for emerging artists that may want to experiment with a more precise tool for lyric generation or casual users that want to play with non-existing lyrics. The genre/emotion option could help to have a more clear vision of the existing lyrics and to obtain a more pertinent result. By using our service instead of directly using a LLM, the user can be better guided towards its objectives.

This web-service is meant to be used as a form of inspiration and not for profit since there is no copyright control over the generated lyrics by the LLM.

2. **Product functions**

   The web-service will offer the following functions:

   - **Original text input**: Our web-service is able to receive a text input from the user, recognize the language, check if it respects the length requirements or if it contains hate speech. In case of a violation, the input is not accepted and an explanation of why is provided to the user. It provides a word count to show how many words it is still possible to write.
   - **Genres proposal**: Our web-service is able to extract possible genres from the input text using Natural Language Processing (NLP) and offer them back to the user as possible options to select.
   - **Sentiment proposal**: Our web-service is able to extract possible sentiments from the input text using NLP and offer them back to the user as possible options to select.
   - **Chunk of lyrics proposal**: Our web-service is able to generate a new chunk of lyrics based on the lyrics written up to that point, one musical genre and one emotion. When generating a new chunk the one generated before gets included in the original input, adding to the words count. This feature is not offered through a model implemented by us, but through the use of third party Large Language Models (LLM).

3. **User characteristics**

   This web-service can be appreciated by:

   - **Casual users**: users that want to experiment with casual lines they thought of. They don't need to know how to write a prompt to a LLM, since everything is mediated by our service.
   - **Artists**: users that want to experiment with AI tools for creativity, possibly to overcome the artist block. The high word limit is thought to make it possible to generate an ending to an almost complete lyric.

4. **General constraints**
   - **Compliance**: The platform must ensure that all input texts adhere to applicable legal and ethical standards

- **Storage Limitations**: The web service will need to handle many concurrent service requests which could impact platform performance
- **Text Generation Speed**: The system must maintain a balance between generating speed, quality and infrastructure cost

5. **Assumptions and dependencies**
    - **Internet Access**: it is assumed that users will have reliable internet access to interact with the platform.
    - **Scalability**: the infrastructure will need to scale according to the number of online users and AI tools in action, without causing delays.

3. **Specific Requirements**
    1. **Functional Requirements**

        The platform shall permit a text input. The platform shall accept the text input only if it's between 10 and 500 words, if it is recognizable as written in english and if it contains only characters from UTF-8. The platform shall communicate to the user if the text was accepted or not. In case the text was accepted, the platform shall extract 3 genres and 3 sentiments from the input text and provide them to the user as possible choices. The platform shall accept only one choice per genre and one choice per sentiment. Only after the user has selected genre and sentiment, the platform shall send a formatted prompt to a LLM API to generate some following text for the provided text input. The platform shall provide a text that is between 250 and 300 words. The platform shall make it possible for the user to generate the last chunk of lyrics again. The platform shall make it possible for the user to generate the next chunk of lyrics, in addition to the text generated up until that moment and while keeping the original information about the selected genre and emotion. The platform shall permit to conclude the lyrics generation process, redirecting the user to the home page. At any moment, the platform shall permit the user to go back to the home page to start the process from scratch.

    2. **Performance Requirements**

        The system should process 95% of transactions in less than 2 seconds. This includes actions like generating the choice of genre, generating the choice of emotions or generating the lyrics. The elapsed time between the submission of a text input and the determination of the validity of it should be minimal.

    3. **Interface Requirements**

        The user shall be able to use our web-service without any mandatory tutorial necessary. The user shall be able to access a page of "FAQ" where all admitted actions/input are listed and explained. From the main page it is possible to insert directly the text input, leading to the aforementioned process of genre/lyrics generation. The system shall provide clear instructions on what to do next during the generation process, without being excessively redundant.

4. **Operational Requirements**

   The machine learning kernel for text mining must efficiently process all requests and be always running. The LLM API must be correctly connected so that, even if there are third party requests, our web-service provides a seamless experience.

5. **Safety and Security Requirements**

   The input text must be subjected to the same privacy policy of the LLM to which it is sent. No additional information about the user will be passed to the LLM. Additionally, security audits and penetration testing should be conducted regularly to identify and address potential vulnerabilities.

6. **Software Quality Attributes**

   The platform must function correctly under defined conditions for an extended period of time. A minimum uptime of 99.9% should be maintained to ensure reliability, with automatic error detection and correction mechanisms in place. Failover strategies, including cloud-based redundancy, must be implemented to minimize downtime during server failures or maintenance.

   The platform should be scalable, enhancing server capacity and specifications. It must support a growing number of users, customizations and transactions without compromising the performance. Cloud infrastructure will enable dynamic scaling based on demand.

   Code documentation and use of design patterns will enable the development team to implement enhancements, bug fixes, and updates efficiently. Maintenance windows should be clearly scheduled and communicated to users, minimizing the impact on the user experience.

   The platform should integrate seamlessly with external systems, such as third-party LLMs. APIs should be designed to allow for easy integration with new services, ensuring the system can evolve alongside advancements in e-commerce technology.

4. **Appendices**
5. **Index**