Rhythm(13.6.25).docx

by Suprava Devi

Submission date: 13-Jun-2025 10:09AM (UTC+0530)
Submission ID: 2698110942
File name: Rhythm_13.6.25_.docx (1.09M)
Word count: 4087
Character count: 23056

1. INTRODUCTION

Production of rhythm patterns is a significant aspect of most music, and it has generally been created manually. However, this has been altered with the expansion of digital tools and artificial intelligence. The Rhythm Composer project presents an intelligent application that generates rhythm-based music automatically using lyrics, computer-generated vocals and beats to form finished audio pieces. The overall ambition is to simplify and quicken the procedure of rhythm composition through the generation of sound, the software modular design, and the interactive programming. The system is able to produce well-structured songs with minimal manual effort by imitating the way humans compose music. Designed to be versatile, the Rhythm Composer can be used by musicians, developers, and hobbyists alike to blend music and technology. It is more time-saving and less technical expertise is required to create rhythm tracks but it can be customized and can be upgraded in future.

The tool may be employed in various spheres such as music education, personal music projects, video game development, and music therapy. In the classroom it can teach students about how rhythms function and in the media, it can be used to create soundtracks and sample audio rapidly. The system consists of independent components that deal with lyrics, voice generation and beat synchronization. The different parts are independent of each other but integrate nicely with each other, which makes it simpler to construct, test and refine.

The Rhythm Composer uses music concepts plus programming, and it represents a large leap towards computer-aided creativity. It allows anyone to make music with little or no musical expertise required, and makes the process of music-making more transparent, interactive and contemporary. On the whole, this project demonstrates how technology can transform artistic work. It provides a quick, imaginative, and uncomplicated method of creating music with the assistance of code, sound, and clever design as compared to ancient procedures.

1.1 PROJECT OVERVIEW

Rhythm Composer is an artificially intelligent music generator that assists in the generation of rhythm-oriented music automatically, using lyrics, synthesized vocals, and beat patterns. It was written in Python and can create complete audio files with minimal manual intervention. It is designed in detachable components, which means that each of them can be easily worked on, and thus, it is an excellent choice in creative endeavours, learning to play music, or just creating some fast music samples.

The music creation process is also simplified through the system as the input and output are handled clearly, thus allowing users to create rhythmically balanced songs with ease. It is editable and can be modified to any extent, which is helpful both to a novice and an advanced user. Rhythm Composer combines musical expertise with intelligent code to offer a contemporary and accessible means of making digital music.

1.2 MOTIVATION

Rhythm Composer project was initiated in order to simplify and quicken the process of music creation. Usually, rhythm making needs special techniques and expensive equipment, which may be prohibitive to novices and hobbyists. This system solves that problem by using AI to automatically create beats and vocals, allowing anyone to make music with little effort. Key drivers for the project include:

- **Accessibility**: Lets people without musical training compose rhythms easily.
- **Time-Saving**: Reduces the effort spent on manual sequencing and synchronization.
- **Educational Use**: Offers a hands-on way to learn and explore rhythm patterns
- **Creative Freedom**: Supports trying out different lyrics, genres, and music ideas.
- **Broad Applications**: Can be used in schools, games, media, therapy, and personal music projects
- **Cost Efficiency**: Removes the need for expensive software or studio setups.

  In short, the goal of this project is to make music creation open to everyone by using AI and audio tools in a simple and user-friendly platform.

1.3 UNIQUENESS OF THE WORK

Rhythm Composer stands out in the field of automatic music creation because of its simple design, modular setup, and smooth user experience. Unlike many existing systems that focus only on making beats or generating text, this project brings together lyrics, vocal sounds, and rhythm patterns into one easy-to-use system.

Key features that make it unique include:

- **Everything under One Roof:** The system does it all- text to complete audio output. Therefore, users do not have to move between various apps or tools**.**
- **Modular Design:** All of the components, such as generating lyrics with GPT-Neo and voice generation with TTS, are independent of each other. This simplifies customizations, bug fixes and new feature additions in the future.
- **Light and Open Source:** It is developed in Python and free libraries, thus it can be simply executed in personal computers without using expensive music software and hardware.
- **Good for Learning and Creativity:** It not only allows users to learn about rhythm but also provides the room to experiment with new music styles, and hence it is ideal both in teaching and creative applications.

1.4 REPORT LAYOUT

This project report is properly structured so as to give a full and detailed description of the development process of the Rhythm Composer system as well as the way it works. Its structure makes it possible to guarantee that technical and non-technical audiences can learn the aims and methods of the project, its results, and prospects.

Section 1: Introduction starts with an explanation of the fundamentals of creating rhythms and why it is such an important element of music production. It describes the primary goals of the project- primarily to simplify, accelerate, and democratize the process of music creation with the help of artificial intelligence (AI) and audio processing technology. The introduction also covers the reason why the Rhythm Composer was built particularly how

music composition using the traditional methods can be time-consuming, involve the use of costly equipment as well as involving musical expertise which not every user might possess.

Section 2: In this section, the drawbacks of the existing tools are identified, including the systems that produce only beats or only lyrics. It emphasizes the necessity of the one-stop platform where lyrics, vocals, and rhythm can be processed.

Section 3: Materials and Methods gives details on the technical instruments involved in the development of the system, such as Python programming language, GPT-Neo to generate lyrics, TTS to generate voices, and other models to generate beats. It defines the step-by-step processing of the input text information to generate, ultimately, rhythm-based audio output.

Section 4: System Architecture gives a global view of the Rhythm Composer and the way various modules interact. It describes the flow of data as lyrics are entered through to the resulting music, and the algorithms that perform the tasks of matching the vocals to the beats as well as managing the timing problems. This renders the system efficient and flexible.

Section 5: Experimentation and Results speaks about the way the system was tested, user interface design, the quality of generated music and user feedback. It covers human judgment and technical performance measures to demonstrate the quality of the performance of the system in practice.

Section 6: Applications and Future Work discusses the possible areas of use of the Rhythm Composer, including schools, games, therapeutic applications and personal music creation. It also recommends the future enhancements such as multi-language support, emotional tone detection, and on-command music generation.

Section 7: Conclusion is a summary of the project implication, which demonstrates how the project helps in the simple creation of music using AI, and most importantly, to the users who are not trained in music theory.

Section 8: References The complete list of the books, articles, and websites utilized during the work is presented.

All these sections taken together make the report self-sufficient, informative, and logically structured so that the readers could grasp the overall scope and significance of the Rhythm Composer project.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEMS

Table 1. Existing Models

Topic	Authors	Model used	Limitations
Transformer architecture[1]	Vaswani, et al.	Transformer (original architecture with self-attention mechanism)	High computational and memory requirements for long sequences; lacks inductive bias for locality (unlike RNNs/CNNs)
Music Transformer: Addressing long-term structureinmusic generation [2]	Huang, et al.	Music Transformer (an adaptation of Transformer architecture optimized for music generation)	Requires large amounts of training data; struggles with capturing expressive nuances like dynamics or emotion in music beyond structure
GPT family of models for text generation [3]	Radford, et al	GPT (Generative Pretrained Transformer) series, including GPT-1, GPT-2, and GPT-3	Can produce factually incorrect or biased outputs; lacks fine-grained controllability over generated content and coherence over very long passages.
Jukebox: End-to-endmusic generationwith vocals[4]	Dhariwal, et al.	Jukebox (Transformer + VQ-VAE)	High computational requirements and long inference times; limited control over specific lyrical or musical structure, often generating incoherent vocals
Advances in text-to-audio models like AudioLM and suno-ai/bark[5]	AudioLM Authors: Borsos, et al. Suno AI's Bark Authors: Shulman, et al.	AudioLM (Audio Tokenizer + Language Model) Bark (GPT-style Transformer)	Models like Bark can produce inconsistent prosody or unnatural transitions; AudioLM's reliance on pretrained models may limit generalization across languages or genres.

MusicLM: High-quality music generation fromtext descriptions [6]	Agostinelli, et al.	MusicLM (w2v-BERT + SoundStream + MuLan)	Struggles with fine control over musical elements like melody or rhythm; potential copyright concerns with generated content
MusicGen: Simple and Controllable Music Generation [7]	Jean, et al.	MusicGen (Transformer Decoder + EnCodec)	Limited by the model's training dataset diversity; struggles with generating culturally nuanced or genre-blending compositions
MusiCoder: Controllable Symbolic Music Generation with Codebook Tokens[8]	Tian, et al.	MusiCoder (VQ-VAE + Codebook Transformer)	Focuses on symbolic music, not raw audio; lacks realism and requires additional conversion steps for audio playback.
Noise2Music: Text-to-Music Generation with Latent Diffusion[9]	Jang, et al.	Noise2Music (Latent Diffusion Model)	Requires large training data and struggles with long-form composition and complex emotional tones.

2.2 PROBLEM IDENTIFICATION

Rhythm Composer is an intelligent system designed to automatically create rhythm-focused music. It brings together three core elements written lyrics, AI-generated vocals, and background beats to produce full music tracks. Built with a modular approach, each component such as text creation, voice synthesis, and beat generation operates independently but integrates seamlessly with the others. By processing simple text input, the system uses text-to-speech (TTS) technology along with digital audio tools to generate complete songs. Its primary goal is to make music creation more accessible and efficient for everyone. Users do not need musical training or expensive software both beginners and experienced creators can easily craft rhythm-rich tracks using this tool.

2.3 PROPOSED SYSTEM

Rhythm Composer system is an artificial intelligence and modular audio tool aimed at simplifying the process of creating rhythm-based music. Conventionally music production can be time consuming and requires expertise which is eliminated with this system since the user is only required to input lyrics or even short text messages. These recordings are then automatically transformed into finished music pieces. The system consists of three primary components: a lyric generation module, which takes either GPT-Neo or text written by the user, a voice module, which converts the lyrics into a sound by means of Text-to-Speech (TTS), and a beat component, which adds rhythm to the song. These components are compatible with each other and hence the users do not require manual editing and mixing. The backend is powered by FastAPI in order to make the system user-friendly. This enables users to communicate with it in real time using a straightforward web interface. It is constructed in a modular way, which implies that all the components of the song, lyrics, vocals, and beats, could be renewed or changed without influencing the others. It also maintains all input and output files well organized thus users have easy time finding and managing music they produce. Overall, the Rhythm Composer makes music creation more accessible, especially for beginners and non-musicians. Designed with scalability and flexibility in mind, the proposed system can be extended in the future to support dynamic tempo changes, genre-specific styles, AI-generated melodies, and real-time music previews. By automating and streamlining the rhythm composition process, Rhythm Composer offers an accessible, efficient, and innovative solution for music enthusiasts, educators, and developers alike.

3 MATERIALS AND METHODS

3.1 ARCHITECTURE LAYOUT AND DIAGRAM

3.1.1 SCHEMATIC LAYOUT

A text prompt is taken as an input from the user. This input is used to generate lyrics using GPT Neo Model and stored in a text file. This lyric is used to generate music using Bark model. Schematic layout for "RHYTHM COMPOSER: MUSIC GENERATOR" is shown in **Figure 1**.

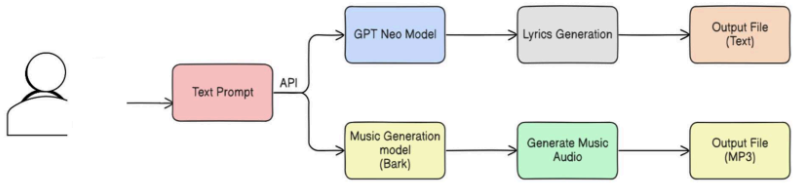

Figure 1. Schematic Layout for the Rhythm Composer

3.1.2 TRANSFORMER MODEL ARCHITECTURE

The transformer model is a type of deep learning architecture that works well with language and sequence data. It uses a technique called **self-attention**, which helps the model understand the relationship between words in a sentence, no matter their position. Unlike older models, it processes all words at once, making it faster and more accurate. This is why transformers like GPT-Neo are good at generating meaningful and well-structured text. **Figure 2** shows the architecture of the Transformer model that is depicted the project.

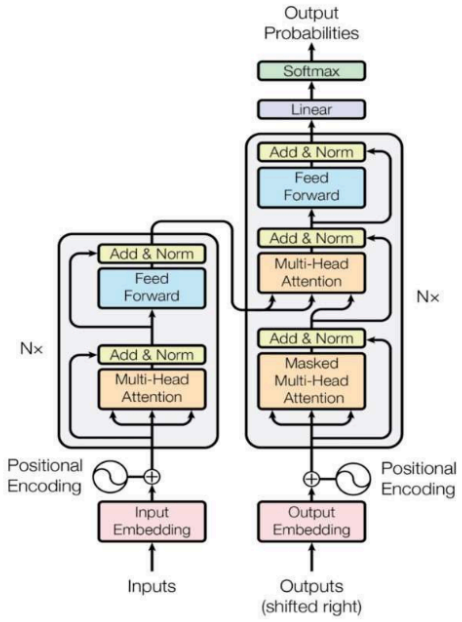

Figure 2. Architecture of Transformer model

3.1.3 BERT MODEL ARCHITECTURE

The BERT model which is explained in **Figure 3**, is a transformer-based architecture designed to understand the full meaning of a sentence by looking at all the words at the same

time. It uses a method called **bidirectional attention**, which means it reads text from both left to right and right to left to get deeper context. BERT is mainly used for understanding and analyzing language rather than generating it. This makes it highly useful in activities such as question answering, sentence classification and sentiment analysis.

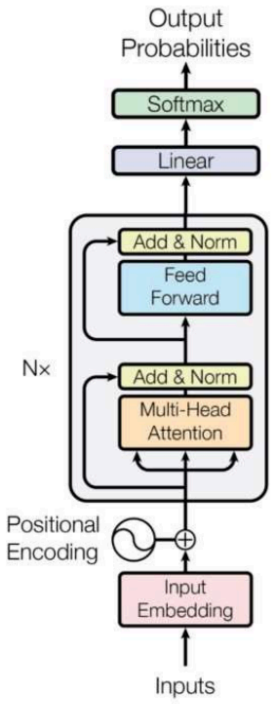

Figure 3: Architecture of Bert Model

3.1.4 GPT MODEL ARCHITECTURE

The GPT model that is explained in **Figure 4**, is a transformer-based architecture mainly used for generating text. It employs **unidirectional attention**, or in other words, it reads left to

right and can predict the following word due to the preceding ones. This aids it in producing sentences that are grammatically acceptable and sound natural. GPT particularly excels at text completion, storytelling and conversation.

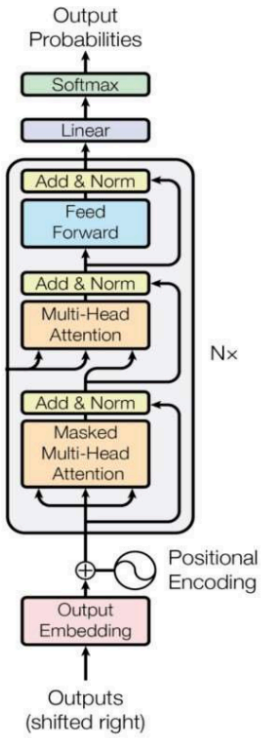

Figure 4. Architecture of GPT Mode

3.1.5 BARK MODEL ENCODER-DECODER ARCHITECTURE

Bark model is an **encoder-decoder** model that is depicted in **Figure 5** and is used in generation of audio and speech. Input text or prompts are encoded into a feature set that

represents meaning and style by the encoder. These features are in turn used by the decoder to produce lifelike audio, such as speech, music, or sound effects. Such a configuration enables Bark to manipulate tone, pitch and expression, and so can be used to generate natural, expressive speech to text.

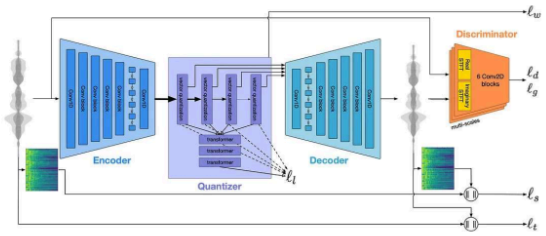

Figure 5: Architecture of Bark Model (Encoder-Decoder)

3.1.6 BARK MODEL VECTOR QUANTIZATION

Bark model quantization that is represented in **Figure 6** is to encode audio into little codes that can be learned. Rather than operating on raw audio data, it divides the sound into small pieces and encodes each piece by a fixed code taken out of a codebook. This assists the model to learn and reproduce complicated sounds more effectively.

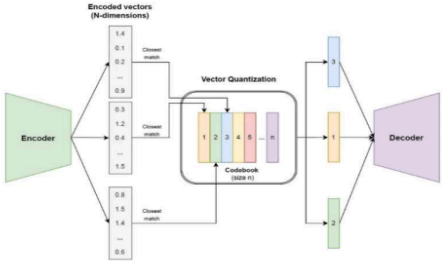

Figure 6: Architecture of Bark Model (Vector Quantization)

3.2 ALGORITHMS USED

This section describes the main algorithm of the Rhythm Composer system that combines AI text generation, audio synthesis, and beat alignment to create musically sensible results.

1. **Python Control System**

   Python is used as the workhorse of the whole system as it coordinates every process, involved in this system, starting with receiving user input and ending with the creation of a final audio file. The rich libraries of the language qualify it to be used in AI integration, audio processing, and fast prototyping.

2. **Lyrics Generation with GPT-Neo**

   Lyrics are produced with the help of GPT-Neo, a text-generation model based on the transformer. When provided with a prompt or seed phrase, GPT-Neo can produce context-aware and rhythmically appropriate lyrics that can follow musical structures like verses and choruses.

3. **Contextual Flow with Transformer Architecture**

   GPT-Neo is constructed based on the transformer architecture that employs self-attention. This assists the model to find connections between words, and it can produce sensible lyrics that have a feel of rhyme, rhythm and thematic consistency.

4. **FastAPI for Real-Time User Interaction**

   FastAPI is employed to build a web interface that allows real-time interactions. Users can input lyrics or prompts, select musical styles, and trigger the music generation process. FastAPI handles requests asynchronously, enabling fast and responsive interactions.

5. **Text-to-Music Conversion Pipeline**

   Once lyrics are available (either generated or user-provided), the system initiates a conversion pipeline:

   - Segmentation: Lyrics are tokenized and segmented into song parts like verses and choruses.
   - Timing Assignment: Each line is mapped to beat intervals based on syllable count and estimated tempo.
   - Voice Synthesis: TTS (Google Text-to-Speech) converts the lyrics into vocal audio clips.

6. **Music Structuring and Synchronization**

   - The structured lyrics are aligned with beat patterns using pydub. This involves:
   - Matching vocal snippets with corresponding instrumental loops.

- Ensuring rhythmic alignment by adjusting silence, overlap, or repetition as needed.
- Preserving song structure (Intro → Verse → Chorus → Verse...).

7. **Composition Logic**

   Composition follows a set of predefined musical rules:

   - Maintain consistent syllabic rhythm across lines.
   - Repeat choruses for familiarity.
   - Introduce variation in intensity across song parts.
   - Ensure alignment between lyrical emotion and musical tone.

8. **Neural Architecture for Quality Assurance**

   A neural validator checks:

   - Rhyme consistency
   - Alignment between syllables and beats
   - Musical phrasing and flow

   If any inconsistencies are found, the model adjusts audio fragments or regenerates sections to ensure musical harmony.

9. **Final Output Generation**

   The audio segments—composed of synthesized vocals and beat loops—are merged into a final track. This audio file (in .mp3 or .wav) is returned to the user via the FastAPI interface for download or further use.

3.3 TOOLS/TECHNOLOGIES

The Rhythm Composer project utilizes cutting-edge technology in AI, deep learning, and web engineering to transform input text from users into meaningful music compositions. The system integrates large language models, music structuring using neural networks, and a web interface with high performance to provide an end-to-end music generation experience.

Hardware Requirements

- Processor: Intel Core i7 or AMD Ryzen 7 processor (quad-core or above) as a minimum for model inference and real-time processing. For quicker audio generation and parallel processing, a multi-core (8+) processor is ideal.
- GPU: A dedicated NVIDIA GPU (e.g., GeForce RTX 3060 or higher with 6 GB or more VRAM) is required for efficient neural network-based text generation and audio synthesis, especially when using transformer models like GPT-Neo.
- RAM: At least 16 GB RAM is recommended. For larger-scale input handling and model loading, 32 GB or higher is ideal.
- Storage: A 512 GB SSD (or larger) is recommended to hold large pre-trained models (e.g., GPT-Neo), music generation libraries, and synthesized audio files.
- Clock Speed: As a minimum, 2.5 GHz; however, 3.5 GHz or more is optimal for quicker response times and multi-threaded operation.

Software Requirements

- Operating System: Windows 10/11, Ubuntu 20.04+, or macOS Monterey or higher.
- Programming Language:Python 3.10 or later – Utilized for creating backend logic, GPT-Neo integration, and structuring music.
- Key Python Packages & Frameworks:

  transformers – For GPT-Neo model integration.

  torch – PyTorch framework for deep learning support.

  fastapi – For building RESTful API endpoints to communicate with the application.

  uvicorn – ASGI server for hosting FastAPI apps.
- Development Tools:

  Visual Studio Code or PyCharm – For debugging and coding.

  Postman – For debugging API endpoints.

  Git – For collaboration and version control.
- Model & Hosting:

  GPT-Neo 1.3B or 2.7B – Hosted locally or using Hugging Face.

Hugging Face Transformers Library – Used to load and fine-tune GPT-Neo.

- Music & Audio Tools

  Text-to-Music Integration Tools

  Application of neural architectures or template-based composition reasoning to synchronize lyrical content with melody and rhythm.

  Generation of song structure (intro, verse, chorus) by AI from GPT-generated material.

- Audio File Output:

  Music composition is output as .mp3 or .wav files based on MIDI synthesis and audio rendering libraries.

  In short, the Rhythm Composer system brings together large AI models and audio processing functionality to provide a hassle-free experience from text prompt to AI-generated music within a web-accessible, responsive application environment.

3.4 Evaluation Measures

Assessing the quality of AI-generated music is a complex task that requires a combination of objective metrics, subjective listening, and foundational music theory analysis. Since the Rhythm Composer system blends natural language processing with audio composition and synthesis, it is essential to evaluate both the linguistic and musical aspects of the output. One important part of testing the Rhythm Composer system is checking how well the music follows common song structures, like verse–chorus–bridge formats. This entails considering factors such as whether the beats are even, the rhythm is constant and whether the key and tempo are well synchronized throughout the song. In order to achieve that, we may employ the use of rule-based or music theory-based checks or tools, to make sure that the end result sounds musically correct, and adheres to the normal patterns of music.

Human listening tests are another important method of assessing the system. These tests assist us in artistic and emotional effect of the music. Humans listen to the music created and provide feedback as to whether the music sounds natural, smooth, and well-produced. They can also inform us whether the emotions in the music suit the meaning and tone of the words, which is relevant to producing songs that can relate to the listeners. Furthermore, the

originality of the music is considered to ensure that the output is not just a replication of known musical patterns but instead introduces fresh and creative elements. Lastly, the **fit between lyrics and music** is reviewed to confirm that the rhythm, mood, and pacing of the audio align well with the lyrical content. These comprehensive evaluations help validate that the Rhythm Composer not only produces technically sound music but also delivers emotionally engaging and creatively unique outputs, ensuring both coherence and expressiveness in AI-generated compositions.

4. RESULTS

The user interface of our AI music generator is shown in **Figure 7**and **Figure 8**. User provides the prompt in the box. After clicking the generate lyrics button, lyrics is generated.

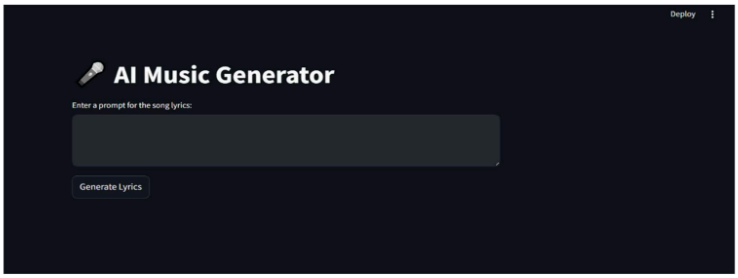

Figure 7. User Interface of the AI Music Generator

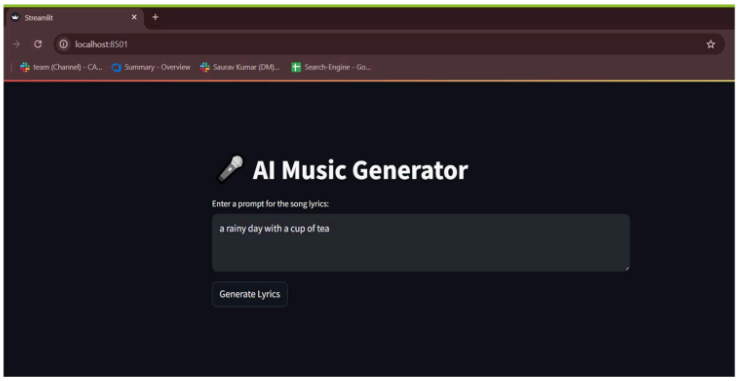

Figure 8. User Interface of the AI Music Generator with prompt

After executing lyrics generation button, response body containing lyrics is generated. The response body of lyrics generation on the basis of prompt provided by the user is shown in **Figure 9.**

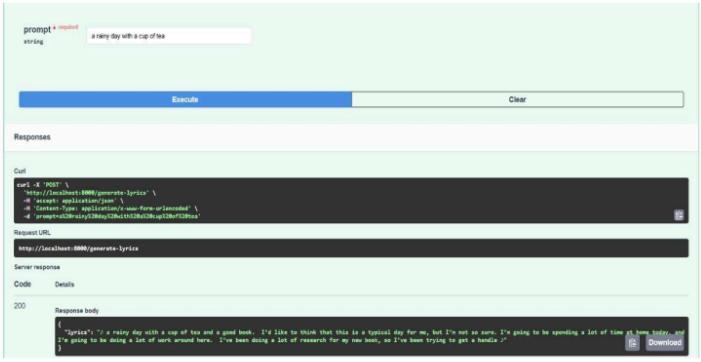

Figure 9: Response Body for Lyrics Generation

A key value pair in json format generated is stored in a text file. Output of lyrics generated in json format is shown in figure **Figure 10.**

Figure 10: Lyrics generated in json format

Pair after lyrics generation, music is generated on the basis of the lyrics. This process is done in multiple chunks and the last is by combining all of them to generate music audio. Generated music audio is shown in **Figure 11**.

Figure 11: Generated Music Audio

5. CONCLUSIONS AND FUTURE WORK

The Rhythm Composer project demonstrates the potential of AI to automate music creation by transforming text into structured musical outputs. The system uses GPT-Neo to create lyrics and Text-to-Speech (TTS) technology to turn those lyrics into vocals. This lets users make rhythm-based music even if they do not have any musical training. It is modular, that is, each part lyrics, voice, and rhythm works independently, so the system is versatile and simple to enhance. Generally, the platform simplifies music creation and makes it accessible, which is helpful in learning, exploring new ideas, and personal music projects.

KEY FINDINGS

The idea of the Rhythm Composer project is to demonstrate clearly that artificial intelligence may contribute to automatizing the process of creating music on the rhythmical basis. It runs on GPT-Neo to generate lyrics and Text-to-Speech (TTS) to convert the generated lyrics into vocal sound. This enables the system to be able to come up with a fully formed and well-organized music with only the simple input of text. Individuals who listened to the generated tracks indicated that they were musically attractive and suited the lyrics very well. The system was also effective as a learning tool and to allow users to explore music ideas.

FUTURE SCOPE

The possible future updates of Rhythm Composer are the rules of other music styles, real-time music creation, and multi-language support to embrace more users. The other potential enhancements include identifying the emotion in lyrics to align with the tone of the vocals, implementing harmony capabilities, and integrating the system with digital audio workstations (DAWs) to use it professionally. Explainable AI techniques might also assist users in gaining insight into the system to make musical choices. Such updates would help the tool to become more innovative, potent, and practical both in the creative and work contexts.

Rhythm(13.6.25).docx

ORIGINALITY REPORT

2%	1%	1%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Kutub Thakur, Helen G. Barker, Al-Sakib Khan Pathan. "Artificial Intelligence and Large Language Models - An Introduction to the Technological Future", CRC Press, 2024 Publication	1%
2	Submitted to APJ Abdul Kalam Technological University, Thiruvananthapuram Student Paper	<1%
3	techpoint.org Internet Source	<1%
4	shehrozpc.com Internet Source	<1%
5	Zhihai Yang, Guangjun Wang, Lei Feng, Yuxian Wang, Guowei Wang, Sihai Liang. "A Transformer Model for Coastline Prediction in Weitou Bay, China", Remote Sensing, 2023 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		