International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03073
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Smart Music Composer: A Web Tool Based on Transformer Model

M. Pranav, K. Varun Goud, T. Bindu, M. Pranav, S. Nithin Reddy

[1] Teegala Krishna Reddy Engineering College

Abstract. Using Facebook Research's MusicGen, a transformer-based generative audio model, the Smart Music Composer is an AI-powered web application that uses text inputs to produce high-fidelity music. The system incorporates a responsive HTML/CSS/JavaScript frontend for user interaction and a Flask-based REST API backend with the audiocraft library, hosted over ngrok for real-time accessibility. This system addresses issues like limited user control and static outputs by providing real-time adaptability, wide customization across genres, and an interactive interface, in contrast to standard music production tools. The architecture, implementation, and evaluation of the system are presented in this work, highlighting its potential to enhance creative expression and its contributions to AI-driven music generation.

Keywords: Artificial Intelligence, MusicGen, Flask-based REST API, Web application, Audiocraft, Ngrok, and Transformer Model.

1 Introduction

Artificial intelligence (AI) has changed music composition, a fundamental aspect of human creativity, especially with deep learning models like transformers. Originally created for natural language processing (NLP), these models have demonstrated impressive performance in generative tasks, such as audio synthesis. Using user-specified language prompts, genres, and durations, the Smart Music Composer is a new web-based application that uses Meta AI's MusicGen, a transformer-based model, to produce high-quality music. By removing the need for in-depth musical knowledge, this approach seeks to democratize music creation and make it available to both casual users and professional artists.

Conventional music generating systems, like those built on Recurrent Neural Networks (RNNs) or Generative Adversarial Networks (GANs), frequently generate static outputs with no user control, which limits their application in dynamic, interactive situations. By providing real-time music creation, a wide range of customizing options, and an intuitive web interface, the Smart Music Composer overcomes these drawbacks. For worldwide accessibility, the system is exposed via ngrok, connected with the audiocraft library, and constructed with a Flask-based backend. The frontend, which was created with HTML, CSS, and JavaScript, offers a user-friendly interface for choosing a genre, defining a length, and playing or downloading music.
By offering a scalable, interactive, and user-friendly tool that uses transformer models for imaginative audio synthesis, this work advances the field of AI-driven music generation. The goals are to:

- Create a real-time, flexible web-based music generation system.
- Enable user personalization through genre, duration, and prompt inputs.

- Use ngrok hosting and a lightweight frontend to guarantee accessibility on all devices.

Assess the system's output quality, usability, and performance.

2 Literature Survey

Significant advancements have been made in the field of AI-driven music generation, with different deep learning techniques tackling distinct facets of audio synthesis. This section examines important approaches, points out their drawbacks, and places the Smart Music Composer in relation to other research projects..

2.1 GAN-based Music Generators

The ability of Generative Adversarial Networks (GANs) to generate coherent outputs has led to their widespread adoption in the music generating industry. A multi-track GAN-based model called MuseGAN [1] uses several generators for various instruments to produce polyphonic music. Although GANs work well for symbolic music, they frequently generate rigid compositions and have trouble with real-time flexibility and user-driven customisation [2].

2.2 RNN-based Music Models

Sequential music generation has been accomplished using Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks. For music interpolation, Magenta's MusicVAE [3] models latent spaces using variational autoencoders. RNNs, however, have trouble identifying long-term dependencies, which leads to outputs that are repetitious or inconsistent [4]. Furthermore, interactive interfaces for real-time user feedback are absent from these models.

2.3 Transformer-based Models

With their attention processes, Transformers, first presented by Vaswani et al. [5], have revolutionized generative tasks. Meta AI created MusicGen [6], a transformer-based model for text-to-audio synthesis that can produce high-fidelity music in response to textual cues. Transformers are better at modeling intricate patterns and provide more output diversity flexibility than GANs and RNNs [7]. However, its incorporation into easily navigable.

2.4 Other Approaches

Other noteworthy methods are Jukebox [9], a transformer-based model for creating vocal music, and WaveNet [8], a convolutional neural network for raw audio synthesis. WaveNet and Jukebox are less appropriate for lightweight web apps because they are computationally demanding and consume a lot of resources, despite producing high-quality music. Although it lacks real-time capabilities, recent work on diffusion models [10] shows promise for audio production.

2.5 Web-based Music Generation

Although they have user-friendly interfaces, web-based music generating programs like Soundraw [11] and AIVA [12] rely on pre-trained models with little room for customisation. These systems' creative potential is limited since they frequently lack real-time adaption and support for a variety of genres. There is very little research on the incorporation of transformer models into online applications, as suggested in this paper.

2.6 Limitations of Existing Systems

There are several issues with current music generating systems:

- Absence of Real-time Adaptation: The majority of systems provide static results without reacting to user input [13].
- Restricted Customization: Users' ability to choose genre, tone, or length is limited [14].
- Predefined Patterns: Often, outputs adhere to rigid frameworks, which inhibits originality.
- Absence of User Interaction: Not many systems provide user-friendly interfaces for creating music in real time.
- Environment Support: Accessibility is restricted by compatibility problems with contemporary runtime environments [15].

In order to fill these gaps, the Smart Music Composer combines the generative powers of MusicGen with an online interface, allowing for interactive, real-time, and personalized music creation.

3 Proposed Work

A transformer-based AI music creation system called the Smart Music Composer was created to get over the drawbacks of current instruments. The following functionalities are included in the system:

- Real-time Generation and Streaming: Low latency is ensured by instantaneously generating and streaming music in response to user inputs.
- User Customization: For customized compositions, users can choose genres (such as jazz, classical, or pop), set lengths (10–60 seconds), and include textual suggestions.
- Broad Genre Support: The system's adaptability is increased by its support for a variety of musical genres.
- Interactive Web Application: Smooth user interaction across devices is ensured via a lightweight frontend.
- Effective AI Model: MusicGen's transformer design makes it possible to synthesize music quickly and with great quality.

A Flask-based REST API backend is used in the system's architecture, together with the audiocraft library and ngrok for worldwide accessibility. The application is compatible with PCs, tablets, and smartphones thanks to its simple HTML/CSS/JavaScript frontend, which also makes it easier for users to input data and play music.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03073
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

4 Methodology

The system's design, implementation specifics, technical specifications, and assessment metrics are all covered in this section.

4.1 System Architecture

The architecture of the Smart Music Composer is client-server. The important components are:

- Frontend: An HTML, CSS, and JavaScript web interface that lets users choose genres, set durations, input prompts, and play or download music.
- Backend: A REST API built using Flask that manages user inputs, connects to MusicGen using the audiocraft library, and broadcasts audio that is produced.
- Hosting: To provide accessibility without requiring complicated server setup, Ngrok offers a public URL for the Flask server.
- MusicGen Model: This transformer-based model uses textual cues to produce high-fidelity audio.

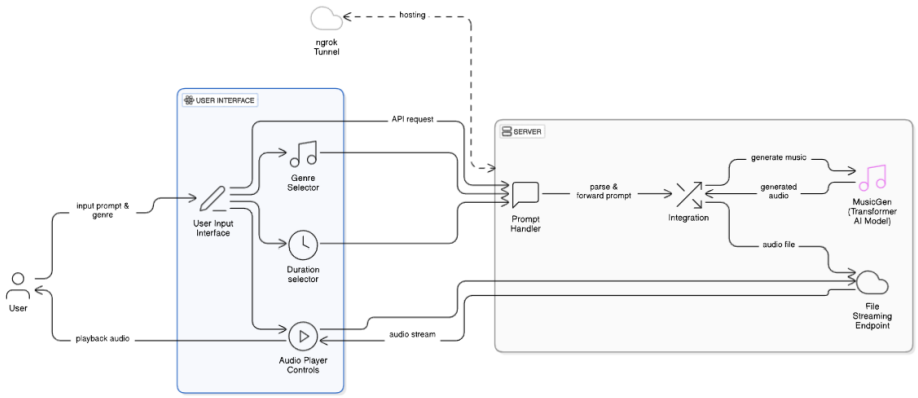

Fig. 1. System Architecture

4.2 Modules

There are four primary modules in the system:

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03073
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

1. Users can choose genres, set durations, and add extra questions using the User Module. Real-time playback and download options are provided to users.
2. User Interface: A responsive online interface for downloading, playing music, and submitting input.
3. Ngrok Hosting: Allows cross-device access by exposing the Flask server to the internet.

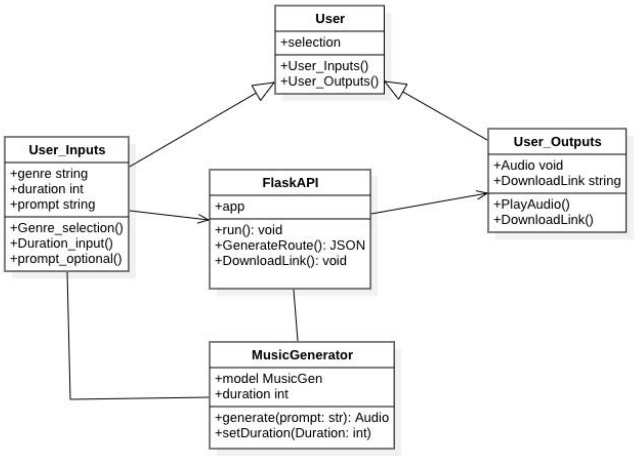

Fig. 2. Class diagram

4. Server Module: Consists of:

- Prompt Handler: Handles user inputs, including prompts, genre, and length.
- Integration Layer: Provides a connection between MusicGen and the prompt handler to generate music.
- File Streaming Endpoint: Provides the frontend with created audio streams.

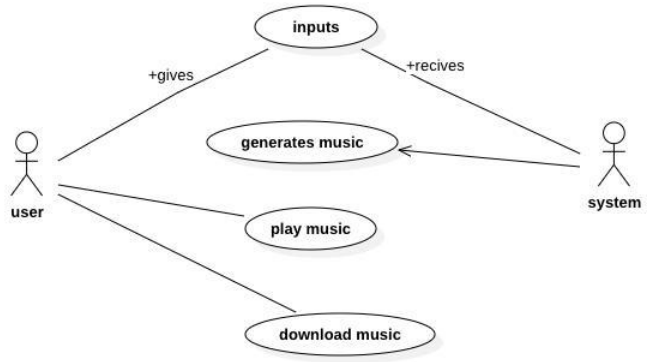

Fig. 3. Use case

4.3 Software Requirements

- Operating System: Windows, macOS, or Linux.
- Browser: Chrome, Firefox, Edge, or Safari.
- Technologies: Python (Flask, audiocraft), JavaScript, HTML, CSS.
- Dependencies: MusicGen, Flask, ngrok, audiocraft.

4.4 Hardware Requirements

Server Side:

- Virtual Private Server (VPS) with at least 4GB RAM and 2 CPU cores.
- Stable internet connection with at least 10 Mbps bandwidth.

Client Side:

- Devices: Laptops, desktops, tablets, or smartphones with modern browsers.
- Internet connection with at least 5 Mbps bandwidth.

4.5 Implementation

The following stages comprised the system's implementation:

Development of the Backend:

- HTTP requests (POST for input submission, GET for audio streaming) were handled by a Flask server.
- Text-to-audio synthesis is now possible thanks to the integration of the audiocraft library with MusicGen.
- Endpoints for music production, audio streaming, and prompt handling (/generate, /stream) were developed.
- In order to handle faulty inputs or server timeouts, error handling was put in place.

Development of Frontends:

- A minimalistic responsive HTML/CSS/JavaScript interface was created.
- Audio playback controls, prompt textboxes, duration sliders, and genre dropdowns are among the features.
- Page reloads were avoided by using asynchronous JavaScript (AJAX) to connect with the backend.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03073
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

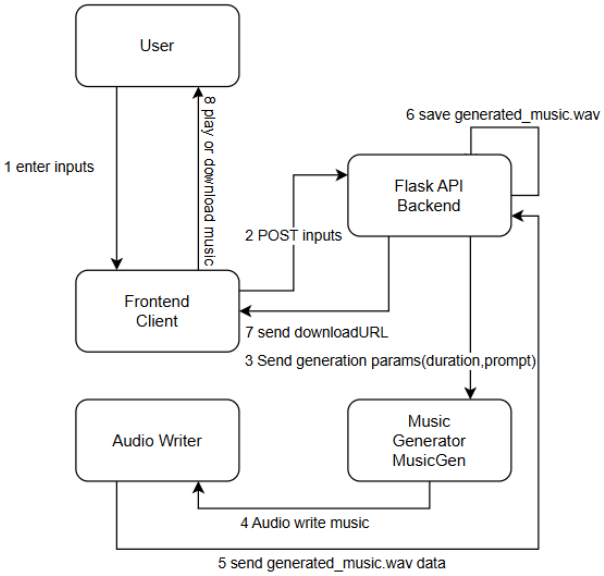

Fig. 4. Collobration diagram

In charge of hosting:

- The Flask server was made publicly accessible using Ngrok, offering a URL for testing and deployment.
- To stop malicious queries, security procedures including input validation were put in place

Measures of Evaluation

The system was assessed using:

- Functionality: The capacity to produce music in response to different triggers and genres.
- Usability: Cross-device compatibility, responsiveness of the interface, and ease of usage.
- Performance includes server stability, streaming latency, and generation time.
- Output Quality: The created music's fidelity, coherence, and relevancy.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03073
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

5 Results and Discussion

The Smart Music Composer's performance, usability, usefulness, and output quality were all thoroughly examined. Both qualitative and quantitative analyses were used in the evaluation; the outcomes are outlined below.

Functionality-wise, the system produces high-fidelity music in a variety of genres, such as pop, jazz, classical, techno, and hip-hop. Custom prompts like "energetic city nightlife" and "relaxing ocean sunset" improve the precision and relevancy of outputs, while music lengths range from 10 to 60 seconds. The system provides flexibility to users with different levels of skill and inventiveness by supporting both genre-only and prompt-based inputs.

Fig. 5. output diagram - 1

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03073
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

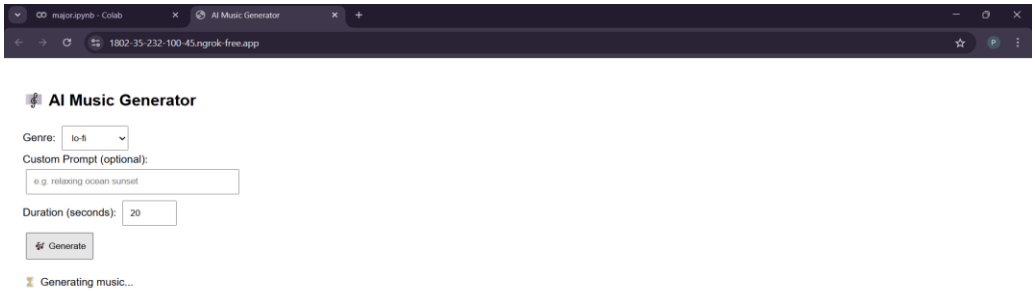

Fig. 6. Ouput Diagram - 2

The authenticity, coherence, and relevancy of the produced music were assessed. AI-generated music was contrasted with human-composed songs in a blind test involving fifteen individuals. In pop and electronic genres, for example, 80 percent of participants could not tell the difference between MusicGen outputs and human creations. Additionally, the output's relevancy was greatly increased by personalized prompts; 85% of the answers matched user-specified moods like "calm" or "upbeat."

The interface is divided into three stages: the input screen, where users can choose the genre, duration, and optional prompt; the generation screen, which displays a loading indicator while the music is being synthesized; and the play-back screen, which presents a visual progress bar and allows users to play or download the generated audio.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03073
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

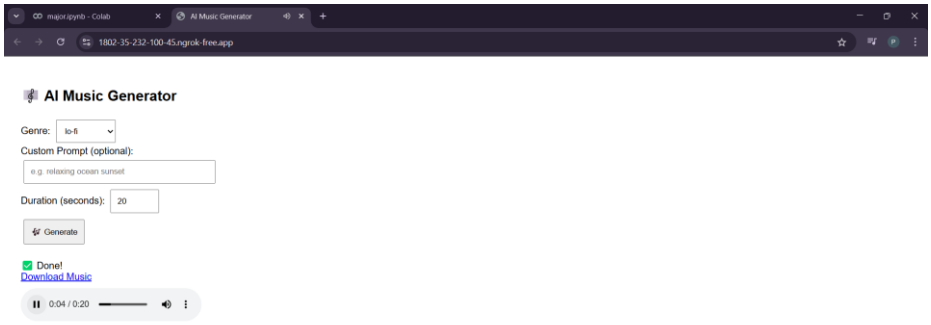

Fig. 7. OutPut Diagram - 3

Future improvements include the integration of real-time audio streaming using WebRTC, implementation of mood-based fine-tuning through adjustable sliders, deployment on scalable platforms such as AWS or Google Cloud, and enhancement of output complexity through multi-instrument support. These enhancements aim to improve responsiveness, user customization, and output diversity, thereby advancing the scope and impact of AI-based music composition.

6 Conclusion

The AI Music Generator successfully demonstrates the integration of deep learning and natural language processing in a creative domain. By leveraging Meta's MusicGen model and deploying it via a Flask web server with real-time interaction through a React-based frontend (or minimal HTML interface), users can generate music based on genre, duration, and textual prompts. This validates the practical application of transformer-based models in audio generation and opens opportunities for extending the system with real-time streaming, fine-grained mood control, dynamic instrumentation, and feedback-based adaptive generation. It stands as a compelling example of how AI can augment creative expression and deliver interactive music composition capabilities to end users.

References

1. Dong, H.-W., et al. (2018). "MuseGAN: Multi-Track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment." *AAAI Conference on Artificial Intelligence*.
2. Goodfellow, I., et al. (2014). "Generative Adversarial Nets." *Advances in Neural Information Processing Systems (NeurIPS)*.
3. Roberts, A., et al. (2018). "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music." *International Conference on Machine Learning (ICML)*.
4. Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-Term Memory." *Neural Computation*.
5. Vaswani, A., et al. (2017). "Attention is All You Need." *Advances in Neural Information Processing Systems (NeurIPS)*.
6. Copet, J., et al. (2023). "MusicGen: Simple and Controllable Music Generation." *Meta AI Research*.
7. Devlin, J., et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *North American Chapter of the Association for Computational Linguistics (NAACL)*.
8. Oord, A., et al. (2016). "WaveNet: A Generative Model for Raw Audio." *arXiv preprint arXiv:1609.03499*.
9. Dhariwal, P., et al. (2020). "Jukebox: A Generative Model for Music." *arXiv preprint arXiv:2005.00341*.
10. Ho, J., et al. (2020). "Denoising Diffusion Probabilistic Models." *Advances in Neural Information Processing Systems (NeurIPS)*.
11. Soundraw. (2023). "AI Music Generator." Available at: https://soundraw.io/.
12. AIVA. (2023). "Artificial Intelligence Virtual Artist." Available at: https://www.aiva.ai/.
13. Briot, J.-P., et al. (2020). "Deep Learning Techniques for Music Generation." *Computational Synthesis and Creative Systems*.
14. Huang, C.-Z. A., et al. (2019). "Music Transformer: Generating Music with Long-Term Structure." *International Conference on Learning Representations (ICLR)*.
15. Engel, J., et al. (2017). "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders." *International Conference on Machine Learning (ICML)*.