**DATA 675**

**Data Science Project**

**Generative Artificial Intelligence (AI)**

**Jean Paul Melendez Villanueva**

**Professor Vahe Heboyan**

**AI MusicGen Project**

**May 25th, 2025**

**Executive Summary**

The AI Music Generator project is a full-stack generative AI application that empowers users to create original instrumental music using natural language prompts. This tool simplifies music production by eliminating the need for musical knowledge, expensive software, or complex installations. It integrates Meta’s MusicGen model for text-to-music generation, allowing users to input a brief prompt, such as “a cinematic metal battle with electric guitars and choir” and receive a custom .wav file as output.

The front end of the system is built with Streamlit, offering an intuitive and user-friendly interface where users can enter their music prompt, choose track duration (5 to 30 seconds), and optionally input lyrics for future vocal synthesis integration. The backend runs on Google Colab and is implemented using Flask, which processes generation requests and streams the output. Ngrok is used to connect the frontend and backend securely over the internet.

The project aims to democratize access to music creation, enabling content creators, educators, indie developers, and hobbyists to quickly generate high-quality music without traditional barriers. The system’s modularity supports future enhancements, including AI-generated vocals through Suno’s Bark model, genre presets, and user authentication for personalized song libraries. The application's scalability and reliance on free-tier services also make it suitable for students and researchers interested in prototyping music AI systems without financial or infrastructure overhead.

The application is open-source and hosted on GitHub, enabling developers to replicate, modify, or expand the tool for custom use cases. This project exemplifies how cutting-edge generative AI can be transformed into a practical, real-world application that merges creativity with accessibility. The generative AI project that I am creating is a user-friendly, end-to-end system that brings together MusicGen and future Bark integration to enable fast, creative music generation from simple text inputs. The entire system is developed and maintained in Visual Studio Code, which serves as the primary environment for writing, organizing, and deploying all project components including the Streamlit frontend and Flask backend.

**Table of Contents**

[Project Scope 3](#_Toc247472791)

[Problem Description 4](#_Toc1817051028)

[Project Importance 4](#_Toc817639565)

[Proposed Solution 4](#_Toc1064524227)

[Generative AI Development Tools 4](#_Toc640636557)

[Project Milestones-Lessons Learned 4](#_Toc2096878179)

[Generative AI System Requirements 5](#_Toc1800046692)

[Requirements Listing 6](#_Toc1353401213)

[System Inputs 6](#_Toc1874953460)

[System Parameters 6](#_Toc1704600171)

[System Outputs 6](#_Toc2020071251)

[Generative AI System Implementation 6](#_Toc1691915430)

[Development Tools 7](#_Toc770848128)

[Programming Strategy 7](#_Toc1768765609)

[Programming Modules/Subroutines/API Calls 7](#_Toc758344571)

[Next Steps 7](#_Toc1712306980)

[Generative AI System Review 7](#_Toc658545852)

[System Description in Words for Typical User 8](#_Toc487476669)

[Scenario 1 8](#_Toc486480617)

[Scenario 2 8](#_Toc214125766)

[Final Report 8](#_Toc1238170751)

[Findings 9](#_Toc1908815632)

[Review of Ethical, Privacy and Security Aspects for Your Selected Project 9](#_Toc776451771)

[Review of Success or Completion 9](#_Toc1093772876)

[Recommendations for Future Development 9](#_Toc1305500001)

[References 9](#_Toc1473871078)

# Project Scope

**Unit 2 Assignment**

# Problem Description

Creating original music typically requires a combination of artistic skill, music theory knowledge, technical equipment, and access to professional software. For individuals without formal musical training such as educators, small content creators, game developers, or hobbyists, producing customized tracks for videos, games, podcasts, or personal projects can be both time-consuming and expensive. Traditional music production workflows also involve complex digital audio workstations (DAWs), which can be overwhelming for non-musicians and inaccessible for those working with limited resources.

In addition, the rising demand for short-form content on platforms like YouTube, TikTok, and Instagram has increased the need for fast, flexible, and royalty-free music generation. However, existing tools often require pre-recorded libraries, expensive licensing, or subscription fees. These limitations create a significant gap for individuals and teams seeking an easy, affordable, and creative way to generate unique, on-demand soundtracks. Generative AI offers a promising solution to this problem by enabling users to produce music simply by describing what they want. Yet, many generative music models remain restricted to research environments or require technical know-how to set up. This project addresses the accessibility gap by providing a user-friendly, web-based AI music generation platform that operates on free-tier tools and delivers immediate, downloadable results. The generative AI project that I am creating is a streamlined application that uses Meta’s MusicGen model to convert natural language prompts into custom music compositions, with future plans to integrate AI vocals via Bark.

# Project Importance

This project was chosen because it represents an exciting intersection of innovation, accessibility, and creative freedom. Rather than relying on traditional music production methods that require years of training or costly equipment, I wanted to explore how artificial intelligence can empower anyone to become a music creator. The idea of transforming simple text prompts into expressive musical compositions is both powerful and deeply relevant in today’s fast-paced, content-driven world. I was inspired by the challenge of building a tool that merges state-of-the-art AI with a simple, user-friendly design, allowing creativity to flourish regardless of technical background.

The project is important because it responds directly to the needs of modern content creators, educators, and hobbyists who require unique, royalty-free music on demand. As short-form video and interactive media platforms continue to grow, so does the need for personalized audio that can enhance storytelling and audience engagement. Unfortunately, most AI music tools are either locked behind paywalls, limited in functionality, or too complex to set up. Generative AI systems like MusicGen offer a breakthrough by generating music directly from text prompts, making it possible to skip the traditional production process. According to Briot, Hadjeres, and Pachet (2019), such AI systems are reshaping the way music is composed by automating style-aware, high-quality outputs across various musical genres.

This project benefits a wide range of users, content creators, indie game developers, students, teachers, and even small marketing teams. It democratizes music production by making it accessible, fast, and flexible. The tool also supports learning and experimentation for those interested in generative AI, creative coding, or audio engineering. By releasing the system as open-source, it encourages collaboration and iterative improvement, helping advance the field of creative AI in both educational and professional settings.

# Proposed Solution

The proposed solution is a full-stack, web-based generative AI system that enables users to create original music compositions simply by typing a natural language prompt. This system utilizes Meta’s pretrained MusicGen model, which is capable of generating coherent and stylistically relevant instrumental tracks from textual descriptions. Users access the tool through an interactive Streamlit interface, where they can enter a music prompt, select the desired duration, and optionally provide lyrics for future vocal integration. Once submitted, the prompt is sent to a backend hosted on Google Colab using a Flask API, which processes the request and returns a downloadable .wav audio file. The connection between front end and backend is managed using ngrok, allowing real-time communication over the web.

The dataset powering this system is embedded within Meta’s MusicGen model, which was trained on a large collection of licensed music and public-domain audio data curated by Meta AI. The model includes access to high-quality multitrack music datasets with associated captions, enabling it to learn relationships between musical features and natural language descriptions. Because MusicGen weights and training data are publicly available for research and creative use under Meta’s license terms, this project complies with those guidelines. The model is accessed directly from Meta’s official GitHub repository, ensuring alignment with licensing and citation standards.

# Generative AI Development Tools

This project leverages a combination of cutting-edge AI frameworks, development libraries, and cloud-based tools to deliver seamless user experience from prompt to music generation. At its core, the system uses Meta’s MusicGen, an open-source generative model developed by Meta AI for text-to-music synthesis. MusicGen is implemented in PyTorch and accessed through the Audiocraft repository, which provides a flexible interface for running the model locally or in a Colab environment. MusicGen processes a user’s text input and generates corresponding instrumental music in high-quality .wav format. For deployment, the backend runs in Google Colab, using Flask to create a lightweight API that handles user requests and returns generated audio files.

On the front end, the application is built using Streamlit, a Python framework designed for building data applications quickly and intuitively. Streamlit enables the user interface to be interactive and visually accessible, eliminating the need for frontend web development expertise. Users can input prompts, control duration, and preview audio with just a few clicks. To bridge communication between the backend and front end, ngrok is used to expose the Colab server to the web securely and reliably. Additional libraries like requests, soundfile, and torchaudio handle tasks such as audio playback, file writing, and model support. Together, these tools form an end-to-end pipeline for AI music generation that is portable, modular, and easy to extend, perfect for both educational and creative applications. The coding, debugging, and file management across the project are conducted in Visual Studio Code, chosen for its GitHub integration, terminal access, and seamless development experience across Python-based tools.

# Project Milestones – Lessons Learned

*Describe the major milestones and the lessons you learned for each assignment.*

|  |  |  |
| --- | --- | --- |
| **Unit 2 Assignment**  **Project Scope** | I completed the initial planning phase of my generative AI music project, which included writing the executive summary, problem description, project importance, proposed solution, and the list of development tools. I defined the scope of the system, identified the target users, selected Meta’s MusicGen model, and reviewed licensing requirements. | I learned the importance of clearly defining the project’s goals and technical structure before implementation. This step helped me understand the critical need for aligning user needs with AI capabilities. I also gained insights into responsible dataset use and how to integrate licensing compliance into project planning. |
| **Unit 3 Assignment**  **System Requirements** | For this assignment, I documented the functional requirements of my generative AI music system using 15 “The system shall…” statements. I also wrote detailed reviews of the system’s inputs, parameters, and outputs. This included identifying how user text prompts, duration sliders, and optional lyrics fields feed into the MusicGen model, and how configuration settings like sampling rate, model size, and temperature affect performance. | I learned how essential it is to define and separate the various components of a generative AI system. Thinking through each requirement helped me better understand how user interaction flows into backend logic and how output is produced and delivered. It also emphasized the importance of modular design and scalability for future enhancements like vocal integration using Bark. |
| **Unit 4 Assignment Presentation** | *Describe the tasks that you completed.* | *Reflect on what you learned as part of this assignment.* |
| **Unit 5 Assignment**  **Implementation** | *Describe the tasks that you completed.* | *Reflect on what you learned as part of this assignment.* |
| **Unit 7 Assignment**  **System Testing** | *Describe the tasks that you completed.* | *Reflect on what you learned as part of this assignment.* |
| **Unit 8 Assignment Final Report**  **Findings** | *Describe the tasks that you completed.* | *Reflect on what you learned as part of this assignment.* |

# Generative AI System Requirements

**Unit 3 Assignment**

### Requirements Listing

1. **The system shall** accept natural language prompts from users describing the desired music style or mood.
2. **The system shall** allow users to select the desired duration of the generated music (5–30 seconds).
3. **The system shall** provide an optional text field for users to input lyrics for future vocal integration.
4. **The system shall** send user inputs to a backend API for music generation.
5. **The system shall** process generation requests using Meta’s MusicGen model.
6. **The system shall** return a downloadable .wav file containing the generated music.
7. **The system shall** use ngrok to create a secure tunnel between the frontend and the Colab backend.
8. **The system shall** display the generated audio using an embedded player for immediate playback.
9. **The system shall** notify users of generation status using a loading spinner and status messages.
10. **The system shall** handle API request errors gracefully with clear error messages.
11. **The system shall** be accessible through a Streamlit-based web interface.
12. **The system shall** validate input prompts to ensure they meet basic text length requirements.
13. **The system shall** use Flask as the backend framework for routing generation requests.
14. **The system shall** be designed to operate on Google Colab using free-tier GPU resources.
15. **The system shall** be modular and extendable to include future features such as genre presets and Bark-based voice generation.

### System Inputs

The primary input for the generative AI music system is a natural language text prompt provided by the user. This prompt is a short description that conveys the desired style, mood, or theme of the music, such as "epic orchestral battle theme" or "relaxing lo-fi hip hop beat." The system interprets this text to guide the MusicGen model in composing a corresponding instrumental track. This flexible input format allows for creative exploration across musical genres without requiring technical knowledge of music theory or production. The system also includes basic input validation to ensure that the prompt is not empty and meets a minimum length requirement, enhancing model reliability.

In addition to the main prompt, the user must select a duration value using a slider interface, which currently ranges from 5 to 30 seconds for now. This input informs the MusicGen model of how long the generated track should be. A third optional input is a text area for lyrics, designed for future integration with a voice generation model such as Bark. Although lyrics are not currently processed, this input is retained for scalability and future development. All inputs are collected through the Streamlit frontend and transmitted to the backend via a JSON-formatted POST request, ensuring a lightweight and readable data transfer format. These structured inputs form the foundation of the system’s interactive and user-friendly design.

### System Parameters

The generative AI music system uses several internal parameters to control the behavior and performance of the MusicGen model during inference. One key parameter is the sampling rate, typically set at 32 kHz, which determines the audio resolution of the generated track. Higher sampling rates produce clearer, more realistic sound but require more processing power and memory. Another parameter is the model type or size—in this case, the “medium” or “melody” version of MusicGen can be selected depending on whether the focus is on melody conditioning or general prompt-based generation. These configurations influence model response time and output fidelity, especially on hardware-constrained platforms like Google Colab.

Additional parameters include top-k sampling, temperature, and beam width, which affect how the model explores the space of possible outputs. For example, a higher temperature introduces more randomness into the generation process, which can result in more creative but less predictable music. These values are typically set to defaults optimized by Meta for public use, but they can be adjusted manually in the backend code for experimentation or advanced control. Lastly, the system uses the duration selected by the user to set the number of audio tokens the model should generate, effectively controlling the length of the final output. These internal parameters work behind the scenes to ensure the generated music aligns with user expectations while maintaining computational efficiency and quality.

### System Outputs

The primary output of the generative AI music system is a downloadable .wav audio file that contains the music generated by Meta’s MusicGen model. This file is streamed back to the user through the Streamlit interface, where they can play it directly using an embedded audio player or download it for external use. The audio reflects the characteristics described in the user's text prompt, such as the genre, instruments, or emotional tone, and adheres to the duration specified in the input slider. This seamless output flow allows users to go from concept to custom soundtrack in a matter of seconds, making the experience both efficient and creatively rewarding.

In addition to the audio file, the system produces several secondary outputs aimed at improving user interaction and debugging. These include on-screen status indicators (e.g., "Summoning Colab magic...") and success or error messages that inform the user of the request status. For example, if the backend fails or if ngrok is disconnected, the frontend displays an appropriate error message like “Connection error” or “Backend failed to generate music.” These feedback messages are essential for maintaining usability and transparency, especially in a multi-component system involving third-party tools and real-time processing. The combination of clear audio output and real-time visual feedback ensures the system is both functional and user-friendly.

# Generative AI System Implementation

**Unit 5 Assignment**

*In at least two (2) well-written paragraphs, provide a summary of the generative AI system implementation.*

### Development Tools

*In at least two (2) well-written paragraphs, provide a summary of the generative AI system development environment you will use.*

### Programming Strategy

*In at least two (2) well-written paragraphs, provide a summary of the generative AI system development approach you will utilize.*

### Programming Modules/Subroutines/API Calls

*List and provide a 2-sentence description for each component of the program provided. Review each routine, subroutine, function, and API call. If there is only one routine, then provide source code and comment on each line of the routine.*

### Next Steps

*In at least two (2) well-written paragraphs, provide a review of the next steps to complete the programming of the system.*

# Generative AI System Review

**Unit 7 Assignment**

*In the assignment you will review the functions of your created system and then show how the system works. Please provide screen shots showing how the system works. Please provide two scenarios walking through the use of the system.*

### System Description in Words for Typical User

*Provide a concise, updated description of your system describing what it does, the inputs it uses, and the intended output using words that your typical user will understand.*

### Scenario 1

*Review the use of your system. Identify the inputs and the output for this scenario. Provide screenshots with annotations showing and describing the execution of Scenario 1.*

### Scenario 2

*Review the use of your system. Identify the inputs and the output for this scenario. Provide screenshots with annotations showing and describing the execution of Scenario 2.*

# Final Report

**Unit 8 Assignment**

### Findings

*In at least six (6) well-written paragraphs, describe the results of your development and execution of the system.*

*Visuals and tables are welcomed.*

### Review of Ethical, Privacy, and Security Aspects for Your Selected Project

*In at least three (3) well-written paragraphs, evaluate your execution of the project.*

### Review of Success or Completion

*Provide at least two (2) paragraphs evaluating your execution of the system.*

### Recommendations for Future Development

*In at least four (4) well-written paragraphs, identify future development.*

# References

Briot, J. P., Hadjeres, G., & Pachet, F. D. (2019). *Deep learning techniques for music generation*. Springer. https://doi.org/10.1007/978-3-319-70163-9

Copet, J., Defossez, A., Gelly, S., Lam, J., Leader, R., Lonce, S., Teoh, J., & Usunier, N. (2023). *Simple and controllable music generation* [Computer software]. Meta AI. <https://github.com/facebookresearch/audiocraft>

Streamlit Inc. (2023). *Streamlit (Version 1.x)* [Computer software]. <https://streamlit.io/>

Ngrok. (2023). *ngrok: Secure introspectable tunnels to localhost* [Computer software]. <https://ngrok.com/>

Microsoft. (2023). *Visual Studio Code* [Computer software]. <https://code.visualstudio.com/>