AI-BASED MUSIC GENERATOR

K.TULASI KRISHNA KUMAR, DOWLAPALLI RAMU
Assistant Professor, Training & Placement Officer,Student,
2 MCA Final Semester,
Master of Computer Applications,
Sanketika Vidya Parishad Engineering College, Visakhapatnam, India

***Abstract*— The AI-Based Music Generator is a web application developed using Python, Flask, and the music21 library, aimed at automatically generating short musical melodies based on user-selected genres such as Classical, Jazz, or Random. The system allows users to input a prompt and select a genre, after which it algorithmically creates a 16-note melody using genre-specific note patterns and randomization techniques. This melody is then converted into a downloadable MIDI file, enabling users to access and use their compositions instantly. The application combines basic principles of artificial intelligence with music theory to produce unique musical sequences without requiring advanced musical skills. It offers an intuitive and interactive interface for music students, composers, and enthusiasts to explore creative musical ideas quickly and efficiently. By bridging the gap between technology and music, this project highlights the potential of AI tools in enhancing creativity and simplifying the music composition process.**

Keywords: AI Music Generation, Music21, Flask Web Application

1. INTRODUCTION

Music has always been a universal form of expression, deeply intertwined with human emotion and creativity. In recent years, the integration of artificial intelligence into the creative domain has opened new possibilities for music composition. This project, titled **AI-Based Music Generator**, explores the use of simple AI techniques to generate short melodies based on musical genres. The goal is to assist users—especially those with little to no formal music training—in creating unique and personalized music tracks with minimal effort. The application is developed using **Python**, with the **Flask** framework handling the web interface and the **music21** library managing music theory operations and MIDI file creation. Users can select from different genres, such as Classical, Jazz, or Random, and the system generates a 16-note melody using predefined note sets and randomized selection for pitch and duration [1]. The generated melody can be previewed and downloaded as a MIDI file for further use. This tool serves as a simple yet effective example of how algorithmic music generation can be implemented with minimal resources. By focusing on genre-based musical structures and combining them with randomization, the system produces melodies that are both structured and unique. The AI-Based Music Generator not only demonstrates the power of combining music theory with programming but also acts as a creative assistant for music enthusiasts, students, and developers.

2. Literature Survey

Artificial Intelligence (AI) has increasingly become an integral part of creative domains, including music composition. Over the past few decades, various approaches have been developed for generating music through algorithmic and AI-driven techniques. The literature reveals multiple methodologies, ranging from rule-based systems to machine learning models, each aiming to replicate or assist human creativity in music composition. One of the earliest systems was **EMI (Experiments in Musical Intelligence)** by David Cope, which analysed the works of classical composers and generated new compositions in similar styles. EMI laid the foundation for future AI systems by showing that musical patterns and structures could be algorithmically learned and reproduced. **Magenta**, a research project by Google, uses machine learning models such as Recurrent Neural Networks (RNNs) and Transformer architectures to generate music. Magenta demonstrates the potential of deep learning in producing complex and emotionally resonant music. However, it also requires large datasets, significant computational resources, and a deep understanding of music theory and programming.**AIVA (Artificial Intelligence Virtual Artist)** is another commercially successful AI system that composes original music for films, games, and advertisements. AIVA uses deep learning and reinforcement learning to continuously improve its musical output based on user feedback and data-driven models. While these systems offer high-quality compositions, they often come with high complexity and resource requirements. In contrast, simpler systems like those using **Markov Chains** or **rule-based algorithms** are easier to implement and understand, making them suitable for academic projects and educational purposes. This project draws inspiration from such lightweight approaches. By using Python and the music21 library, it leverages predefined musical rules and randomized selection techniques to generate genre-based melodies[6]. Unlike deep learning models, this system emphasizes simplicity, user interaction, and real-time generation, making it accessible to users without advanced technical or musical knowledge. The literature shows that while advanced models can produce human-like compositions, simpler systems are still valuable for education, prototyping, and user engagement. The AI-Based Music Generator fits into this space by providing an easy-to-use, efficient, and interactive platform for automated melody creation[2].

Challenges

1. Lack of Musical Context Understanding

- The system randomly selects notes without understanding musical phrasing, harmony, or emotional tone, which may result in less expressive or meaningful melodies.

2 Simplistic Genre Representation

- Genres are represented by limited note sets, which do not capture the full complexity of styles like Classical or Jazz.

3. User Prompt Not Utilized

- Although a text prompt is collected, it is not used in the melody generation process, reducing personalization and interactivity.

4 Repetitive Output

- The reliance on randomization within a fixed pattern can produce monotonous melodies with minimal variation across multiple generations.

5 **No Real-Time Audio Playback**

- Users can only download the MIDI file; they cannot hear the melody directly through the web interface, which impacts user experience.

6 **Limited Musical Features**

- Advanced features like chords, tempo variations, rhythm patterns, and polyphony are not supported, restricting the musical richness of out

7 **No Learning or Feedback Mechanism**

- The system does not improve over time or adapt based on user feedback, as it lacks machine learning or reinforcement learning capabilities.

8 **Scalability Constraints**

- Expanding the system to support more complex structures or additional genres would require significant changes in the architecture and logic.

3. Proposed Methodology

The proposed methodology for the AI-Based Music Generator focuses on using predefined musical scales, randomization algorithms, and web-based interaction to generate short melodies in different genres[8].

Figure 1: Flow Chart

The system is lightweight, accessible, and does not require advanced hardware or machine learning models. The key steps in the methodology are as follows:

1. **Web Interface Using Flask**
   - A user-friendly web interface is built using the Flask framework. It allows users to select a music genre (Classical, Jazz, or Random) and optionally enter a prompt (currently not processed but included for future enhancement).
2. **Genre-Based Note Selection**
   - Each genre is associated with a specific set of notes that reflect common patterns or scales used in that genre. For example, the Classical genre uses diatonic scales, while Jazz uses sharp and flat notes for a more chromatic feel[9].
3. **Melody Generation Algorithm**
   - A simple randomization logic is used to select 16 notes from the chosen genre's note set. The pitch and duration of each note (quarter, half, or whole note) are randomly determined to create a unique melody.
4. **Melody Construction with music21**
   - The selected notes are assembled into a music21. stream. Stream() object. This library provides tools for music analysis, composition, and conversion to standard formats.

5. **MIDI File Creation**
   - The generated melody is converted into a MIDI file using music21.midi.translate.streamToMidiFile () and saved in a server-side folder for download.
6. **File Download Option**
   - Once the melody is generated and saved, a download link is provided to the user to retrieve the MIDI file for further use or editing in any digital audio workstation (DAW).
7. **Stateless Architecture**
   - Each generation request is handled independently, ensuring a simple and stateless operation that can be deployed on basic hosting environments without the need for persistent storage or complex databases.

Machine learning Algorithms:

While the current AI-Based Music Generator uses rule-based logic and randomization for melody generation, various machine learning algorithms can be integrated to enhance its capability, personalization, and musical quality. Below are some machine learning algorithms that can be applied in future versions of the system:

1. Recurrent Neural Networks (RNN)
- RNNs are widely used in music generation due to their ability to handle sequential data.
- They can learn temporal dependencies between musical notes and generate more coherent and structured melodies[5].

2. Long Short-Term Memory (LSTM) Networks
- A type of RNN specifically designed to remember long-term dependencies.
- Ideal for learning patterns in music sequences and generating melodies that follow learned styles or emotions.

3. Markov Chains
- A probabilistic model used for generating note sequences based on the likelihood of one note following another.
- Useful for simple music generation with a more statistical approach.

4. Variational Autoencoders (VAE)
- VAEs can learn latent representations of musical data and generate new samples that follow the learned structure.
- They are useful for style transfer and interpolation between musical styles.

5. Generative Adversarial Networks (GANs)
- GANs can generate realistic musical sequences by learning from real datasets.
- One network generates melodies, while another tries to distinguish between real and generated music, improving quality over time.

6. K-Nearest Neighbors (KNN)
- Can be used for genre classification or recommendation systems within the application.
- Matches user preferences or prompts to similar existing melodies.

7. Support Vector Machines (SVM)
- Useful for classifying music genres or detecting style shifts in generated music when combined with audio feature extraction.

8. Transformer Models (e.g., Music Transformer, GPT-based)
- Advanced models that can capture long-range dependencies in music.
- Capable of generating music that is more expressive and structurally complex than RNNs or LSTMs.

Algorithm of Proposed Work

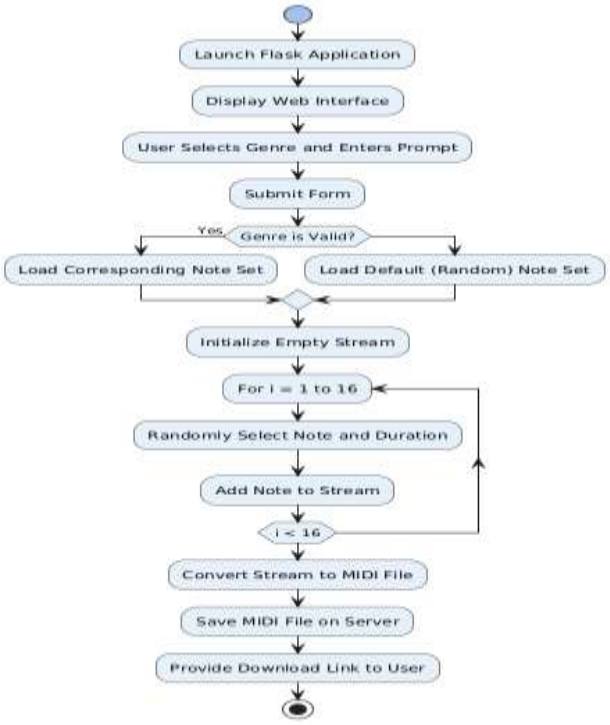

Figure 2: Process of Proposed Model

Algorithm: AI-Based Melody Generation Using Genre-Based Note Selection

Step 1: Start the Flask web application.

Step 2: Display the web interface with the following inputs:

- Genre selection (Classical, Jazz, or Random)
- Optional user prompt input

Step 3: Wait for user to select a genre and submit the form.

Step 4: On submission, retrieve the selected genre and prompt.

Step 5: Fetch the predefined note set based on the selected genre:

- If genre = "Classical" → Use Classical scale
- If genre = "Jazz" → Use Jazz scale
- If genre = "Random" or invalid input → Use Random note pool

Step 6: Initialize an empty melody stream using music21.

Step 7: For 16 iterations (to generate 16 notes), do:

- Randomly select a pitch from the genre's note set
- Randomly select a note duration (e.g., 0.25, 0.5, 1.0)
- Create a note. Note object with the selected pitch and duration
- Append the note to the melody stream

Step 8: Convert the generated melody stream to a MIDI file using music21.midi.translate.

Step 9: Save the MIDI file to a server-side directory.

Step 10: Display a download link on the web interface for the user to download the generated melody.

Step 11: End.

A Comparison of the Proposed and Existing System:

The existing music generation systems like Magenta and AIVA use complex machine learning models, requiring large datasets and high computational resources. They are powerful but often difficult to use and not suitable for beginners. In contrast, the proposed AI-Based Music Generator is lightweight, fast, and easy to use through a simple web interface. It generates 16-note melodies based on selected genres and provides instant MIDI file downloads. Unlike advanced systems, it doesn't need training data or deep AI models, making it ideal for students, hobbyists, and educational use[9].

4. Working Architecture of Proposed Methodology

The working architecture of the AI-Based Music Generator consists of three primary layers: **User Interface Layer**, **Application Logic Layer**, and **Music Processing Layer**.

1. **User Interface Layer**:
   This layer is built using **Flask**, providing a web-based interface where users can select a genre (Classical, Jazz, or Random) and optionally enter a text prompt. Once submitted, the request is sent to the backend for processing.
2. **Application Logic Layer**:
   This layer handles the core logic of the system. It receives the selected genre, validates it, and maps it to a predefined set of musical notes. A melody is then generated by randomly selecting notes and durations from the genre's scale. Each generated note is appended to a music stream.
3. **Music Processing Layer**:
   The music21 library is used here to assemble the note stream and convert it into a **MIDI file**. The MIDI file is then saved on the server, and a download link is displayed to the user.
4. **Output Delivery**:
   The user can download the generated MIDI file and use it in any music software or playback tool. The system is stateless, and each generation is independent, ensuring fast and repeated use without the need for stored data or prior input.

5 System Architecture:

The system architecture of the AI-Based Music Generator consists of five key components, each responsible for a specific function in the melody generation process. The User Interface is built using Flask and presents a simple and intuitive web form where users can select a musical genre—Classical, Jazz, or Random—and optionally enter a text prompt. This layer focuses on accessibility and user-friendly interaction. The Controller Layer acts as the intermediary between the frontend and backend, managing route handling, processing user inputs, and forwarding them to the appropriate backend functions. At the core of the system is the Melody Generator, which interprets the selected genre, accesses a predefined set of notes, and employs randomization to generate a 16-note melody. The generator simulates the feel of the chosen genre using genre-specific scales and durations. Once the melody is composed, the MIDI Converter takes over, leveraging the music21 library to translate the melody stream into a standard MIDI file format. Finally, the Storage & Output module saves this MIDI file on the server in a static directory and generates a download link for the user, enabling easy access and playback of the composition. Together, these components form a cohesive and efficient architecture for generating personalized, genre-based musical melodies in real-time.

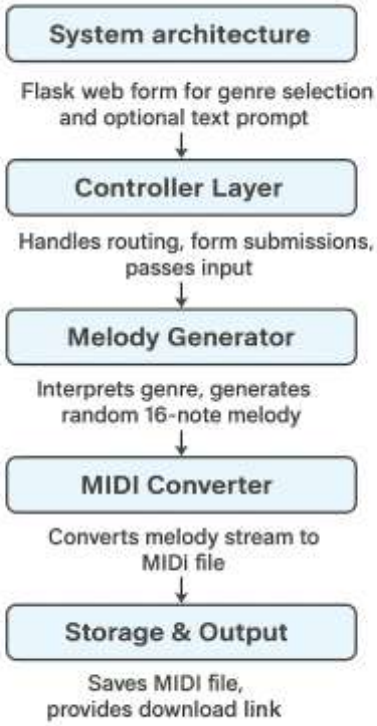

Figure 3:. System Architecture

4.1 Modules:

4.1.1 .User Interface Module

- Technology: HTML, Flask (rendered with render_template_string)
- Functionality:
  - Displays a form for genre selection and optional text input
  - Submits user input to the server
  - Displays the result and download link for the generated melody

4.1.2. Input Processing & Controller Module

- Technology: Flask routes (@app.route)
- Functionality:
  - Captures and processes POST requests from the UI
  - Routes user input to the melody generator
  - Ensures smooth communication between frontend and backend

4.1.3. Genre-Based Melody Generator Module

- Technology: Python, music21.stream, random
- Functionality:
  - Maps selected genre to a predefined set of notes
  - Randomly generates 16 notes with varied durations
  - Builds a melody stream object

4.1.4. MIDI Conversion Module

- Technology: music21.midi.translate
- Functionality:
  - Converts the melody stream into a standard MIDI file
  - Ensures compatibility with external music software (DAWs)

4.1.5. File Storage & Output Module

- Technology: Python os, Flask static file handling
- Functionality:
  - Saves the generated MIDI file to a designated server folder
  - Provides a download link for the user to retrieve the file

4.2 Modules Description

1. User Interface Module
2. input processing &controller module
3.Genre-Based Melody Generator Module
4. MIDI Conversion Module
5. File storage &output module
6. Prompt Processing Module *(Future Scope)*

1. User Interface Module
This module is responsible for interaction between the user and the system. Built using Flask and HTML templates, it presents a web form where users can select a musical genre (Classical, Jazz, or Random) and optionally enter a text prompt. It ensures a clean, responsive, and accessible design, allowing easy input and triggering of the music generation process.

2. Input Processing & Controller Module
The controller module is the central routing system developed using Flask's @app.route decorators. It processes HTTP POST requests from the user interface, extracts user inputs, and sends the data to the melody generation module. It ensures smooth flow of control between the front end and backend services, and handles the logic for returning responses, including generated download links.

3. Genre-Based Melody Generator Module
This is the core of the application where the actual melody generation takes place. Based on the selected genre, the module maps to a predefined set of musical notes and uses random selection to construct a 16-note melody. Each note is assigned a random duration, simulating musical diversity. The melody is stored in a music21 stream object for further processing.

4. MIDI Conversion Module
This module takes the generated melody stream and converts it into a MIDI file using music21.midi.translate. MIDI (Musical Instrument Digital Interface) is a widely supported format used in music production. The module ensures that the output is portable and can be used in any standard music editing or playback software.

5. File Storage & Output Module
Once the MIDI file is generated, this module handles file storage. It saves the file to a static directory on the server using Python's os module and Flask's static file configuration. It then generates a downloadable link that is rendered back to the user through the interface. This allows users to download their custom-generated music easily.

6. Prompt Processing Module *(Future Scope)*
Though not fully implemented in the current version, this module is designed to analyze text prompts entered by the user. Using Natural Language Processing (NLP), the system can eventually determine the emotional tone or intended style (e.g., happy, sad, fast) and adjust the melody generation logic accordingly. It will provide a more intelligent and adaptive experience in future updates.

5 . Result
The Figure.4 shows the result of the system generating a Classical melody with 16 notes, displayed on a musical staff and available for download as a MIDI file.

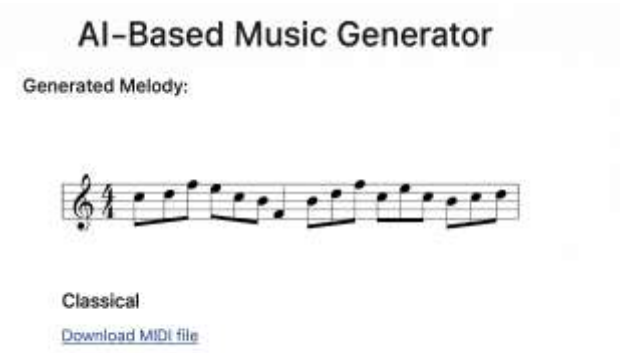

Figure.4 Output of AI-Based Music Generator displaying generated Classical melody with MIDI download option
Here the Figure.5 shows the homepage of music generator

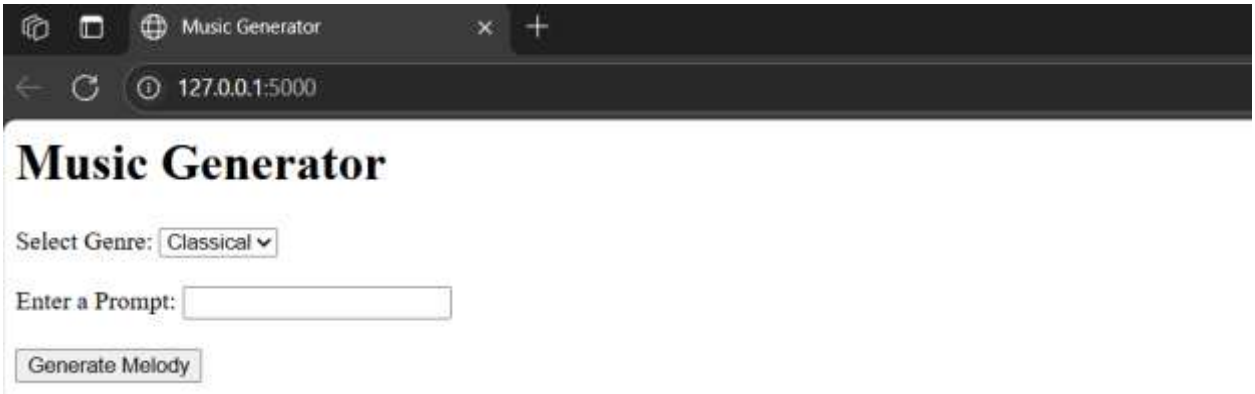

Figure.5. homepage of music generator

CONCLUSION

This paper demonstrates the design and implementation of an AI-Based Music Generator using a rule-based approach. The system provides a fast, simple, and accessible solution for generating genre-specific melodies without requiring complex machine learning models or large datasets. By leveraging Python, Flask, and the music21 library, the tool enables users to generate and download MIDI melodies with just a few clicks. Its architecture is lightweight and modular, making it ideal for educational use, creative exploration, and future enhancements. One of the key strengths of this project lies in its focus on usability and minimal technical barriers. Unlike advanced AI systems that require extensive training, powerful hardware, and domain expertise, this generator offers a plug-and-play experience suitable for beginners and non-musicians. It successfully simplifies the concept of algorithmic composition and makes AI-based music creation more approachable. The results achieved through randomized note generation still maintain a level of musical coherence due to genre-specific scale constraints. While the melodies are relatively simple and monophonic, they provide a solid foundation for further musical exploration or enhancement using digital audio tools. The web-based interface also makes it platform-independent and easy to deploy in both academic and creative environments. In the future, the system can be extended with features such as real-time audio playback, support for chords and harmonies, natural language prompt interpretation, and integration with learning-based models to generate more emotionally or contextually rich compositions. Such enhancements will increase the creative capabilities of the tool while preserving its core values of accessibility and simplicity. Overall, this project serves as a strong demonstration of how rule-based artificial intelligence can be effectively applied in music generation. It opens the door to further research, development, and innovation in the space where technology and creativity intersect.

FEATURE SCOPE

The AI-Based Music Generator can be expanded significantly beyond its current rule-based, monophonic design. Future enhancements may include real-time audio playback for immediate listening, and natural language processing to interpret user prompts for mood-based or emotion-driven compositions. Integrating machine learning models like LSTM or Transformer could improve musical structure and variety. Adding support for polyphony and chord progressions would enrich the sound. The system could also allow users to create custom genres or use a visual editor to modify melodies. Deployment on the cloud and mobile platforms would increase accessibility. A feedback mechanism could enable learning from user preferences over time. These advancements would elevate the system from a basic educational tool to a sophisticated, AI-powered music creation platform.

ACKNOWLEDGEMENT

Kandhati Tulasi Krishna Kumar: Training & Placement Officer with 16 years' experience in training & placing the students into IT, ITES & Core profiles & trained more than 9,700 UG, PG candidates & trained more than 450 faculty through FDPs. Authored various books for the benefit of the diploma, pharmacy, engineering & pure science graduating students. He is a Certified Campus Recruitment Trainer from JNTUA, did his Master of Technology degree in CSE from VTA and in process of his Doctoral research. He is a professional in Pro-E, CNC certified by CITD He is recognized as an editorial member of IJIT (International Journal for Information Technology & member in IAAC, IEEE, MISTE, IAENG, ISOC, ISQEM, and SDIWC. He published 6 books, 55 articles in various international journals on Databases, Software Engineering, Human Resource Management and Campus Recruitment & Training.

Dowlapalli Ramu is pursuing his final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Artificial intelligence S.M.KRISHNA has taken up his PG project on AI-BASED MUSIC GENERATOR and published the paper in connection to the project under the guidance of Kandhati Tulasi Krishna Kumar, Training & Placement Officer, SVPEC.

REFERENCES

1. S. K. Nair and P. R. Prasad, "AI-Based Music Generation Using LSTM Neural Networks," in *Proc. 2023 Int. Conf. on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, Bengaluru, India, 2023, pp. 121–126.
2. V. M. Rao, A. Bansal, and R. A. Gupta, "Music Composition Using Markov Chains and Deep Learning," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 10, no. 2, pp. 145–151, Feb. 2023.
3. P. R. Rao and M. K. Babu, "Intelligent Music Generation System Based on User Emotion and Genre Selection," in *Proc. 2022 8th Int. Conf. on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, pp. 714–719.
4. A. Sinha, R. Mehta, and K. P. Singh, "An Interactive Web-Based Music Generator Using AI," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 10, no. VI, pp. 2275–2280, June 2022.
5. R. Subramanian and M. G. Nair, "AI in Indian Classical Music Generation: Challenges and Approaches," *International Journal of Scientific & Engineering Research (IJSER)*, vol. 13, no. 9, pp. 330–336, Sept. 2022.

6. A. Sharma and T. Varma, "Emotion-Based Music Generation Using AI Techniques," in *Proc. 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Bengaluru, India, pp. 456–460.
7. S. C. Pillai and H. G. Patil, "Design of a Rule-Based Music Composition Tool for Carnatic Ragas," *Indian Journal of Computer Science and Engineering (IJCSE)*, vol. 13, no. 1, pp. 79–84, Jan. 2023.
8. M. Deshmukh, V. Kale, and D. Shinde, "AI-Powered Music Generation Using RNN for Bollywood Genre," *International Journal of Engineering Research & Technology (IJERT)*, vol. 11, no. 5, pp. 554–558, May 2022.
9. R. K. Tripathi and S. Jain, "Web-Based Application for AI-Driven Music Composition," in *Proc. 2023 Int. Conf. on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, pp. 220–225.
10. S. R. Mishra, A. K. Yadav, and L. Patel, "MelodyMaker: An AI Music Generator Using Python and Flask," *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)*, vol. 11, no. 3, pp. 88–94, Mar. 2023.