International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Aurora – Revolutionizing Human-AI Interaction – "Hey Luna"

P. Deepthi[1], A. Likhitha[2], A. Dhanush[3], K. Yeswanth[4], G. Hasha Vardan Reddy[5], Ms. P.V.S. Manisha[6]

[1],[2],[3],[4],[5] Computer Science and Information Technology, Lendi Institute of Engineering and Technology

[6] Associate Professor, Computer Science and Information Technology, Lendi Institute of Engineering and Technology.

Abstract - In the rapidly evolving world of artificial intelligence, voice assistants have become a pivotal element of human-computer interaction. Aurora is an intelligent AI-powered voice assistant designed to deliver a personalized and interactive experience by combining emotion detection, facial recognition, chatbot integration, and mobile automation. Utilizing machine learning algorithms like LBPH for facial recognition and neural networks for emotion-based music recommendation, Aurora adapts its responses based on the user's mood and commands. The assistant is equipped with hot word detection, WhatsApp messaging, making it a versatile tool for everyday tasks. Aurora also integrates ChatGPT-like capabilities via Hugging Face, offering intelligent and context-aware conversations. Built with Python, OpenCV, MediaPipe, and Streamlit, Aurora redefines digital interaction, blending real-time vision, voice, and automation into a seamless user experience.

***Key Words*:** Voice Assistant, Emotion Detection, Facial Recognition, LBPH Algorithm, Neural Network Music Recommendation, Hotword Detection, Chatbot Integration

1.INTRODUCTION

In today's digital world, artificial intelligence has revolutionized how humans interact with technology. Voice assistants have emerged as powerful tools that enable hands-free operation and intelligent response to user queries. However, most traditional voice assistants lack emotional intelligence and human-like adaptability. Addressing this gap, Aurora is designed as a smart voice assistant that not only listens and responds but also understands the user's mood and intent.

Aurora combines various advanced technologies such as facial recognition, emotion detection, and natural language processing to provide a personalized experience. It uses MediaPipe to extract facial landmarks and a trained neural network to detect the user's emotional state. Based on this emotion, it suggests appropriate music, making the assistant emotionally responsive. Additionally, Aurora can recognize users through face recognition using the LBPH algorithm, allowing secure and personalized access.

To extend usability, Aurora integrates ChatGPT-like chatbot features using Hugging Face and automates tasks on mobile devices using ADB commands. It also includes hotword detection with Porcupine, enabling hands-free activation. Built using Python, Streamlit, and OpenCV, Aurora brings together machine learning, vision, and automation to create a next-generation AI assistant that's both intelligent and emotionally aware.

2. System Architecture

The system architecture of Aurora is designed to facilitate seamless, intelligent, and emotionally-aware interaction between users and the AI assistant. It follows a modular, multi-layered approach, allowing each functional component to work both independently and collaboratively. At the core of this system lies the hotword detection mechanism, which listens for the activation phrase "Hey Luna" to initiate communication. The architecture combines real-time audio processing, natural language understanding, emotion detection, and personalized content delivery.

The process begins with the input layer, where the user interacts with the system via voice or text commands. The wake word detection engine, built using Porcupine by Picovoice, listens passively for the phrase "Hey Luna." Once detected, it activates the assistant and passes control to the command interpretation system.

The next critical component is the speech recognition and NLP module, where the user's spoken commands are converted into text using the SpeechRecognition library. This textual data is then processed using NLP tools like

SpaCy or Hugging Face to extract intent and context. In parallel, the system evaluates emotional cues from the user's voice or facial expressions using deep learning models. These models include Convolutional Neural Networks (CNNs) for analyzing speech spectrograms and Recurrent Neural Networks (RNNs)/LSTMs for handling time-series emotional data.
Once emotion and intent are identified, Aurora proceeds to deliver personalized responses. For instance, the system uses APIs such as NewsAPI or Google News to curate and present news articles tailored to the user's current emotional state. Similarly, mood-based music recommendations are fetched through services like the Spotify API, aligning song characteristics (tempo, genre, key) with the detected emotion.

The user interface developed using modern front-end frameworks such as HTML, CSS, JavaScript, and Bootstrap—acts as a bridge between the AI backend and the user. It offers voice input options, visual feedback through animations like Siri waveforms, and output displays for news and music content. Backend communication between the front-end and AI logic is handled via Node.js or through Eel for Python-JavaScript interaction.

To ensure reliability, the system employs multithreading using helper modules, allowing continuous hotword detection while handling backend tasks. The entire application is containerized using Docker for consistent deployment, and it can be hosted locally or on cloud platforms like AWS or Google Cloud.
In essence, the Aurora system architecture is a powerful combination of voice interaction, emotional intelligence, and personalized services, triggered effortlessly through the natural invocation phrase "Hey Luna."

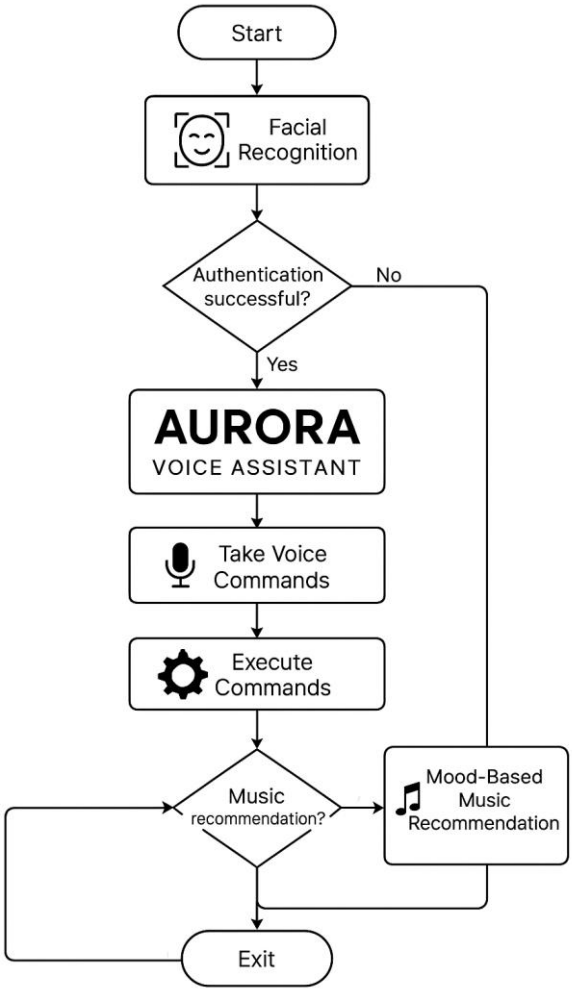


Fig 1. Flow chart

3. Technologies and Tools Used

- **Python:**
  Python is the core programming language used in Aurora due to its simplicity and wide range of libraries. It connects all components like AI models, automation tools, and APIs efficiently.
- **OpenCV:**
  OpenCV is used for real-time image and video processing tasks like facial recognition and emotion detection. It helps Aurora analyze visual data from the webcam or mobile camera.
- **MediaPipe:**
  MediaPipe provides fast and efficient solutions for face detection and hand tracking. It's used in Aurora for lightweight, real-time facial landmark detection.
- **TensorFlow/Keras:**
  TensorFlow and Keras are used to build and train deep learning models in Aurora. They support tasks like emotion classification and chatbot training with neural networks.

- **Streamlit:**
  Streamlit is used to create an interactive web interface for Aurora. It allows users to interact with the assistant visually and test features through a user-friendly dashboard.
- **Hugging Face:**
  Hugging Face provides advanced NLP models like GPT for chat interactions. It powers Aurora's chatbot, enabling natural and intelligent conversations.
- **Porcupine:**
  Porcupine is a lightweight wake word engine used to detect the hotword "Hey Luna." It ensures fast and offline hotword detection with minimal CPU usage.

4. Modules and Algorithms

Voice Control (SpeechRecognition + pyttsx3): Aurora listens to voice commands using the SpeechRecognition library, which converts speech to text. It then responds using pyttsx3, a text-to-speech engine that speaks out the answers. This creates a fully voice-interactive system without needing a screen.

Emotion Detection (Neural Network, MediaPipe landmarks):
Aurora uses MediaPipe to detect facial landmarks like eyes, mouth, and eyebrows in real time. These landmarks are passed to a neural network model trained to classify emotions such as happy, sad, or angry. This helps Aurora adapt its responses based on the user's emotional state.

Facial Recognition (LBPH algorithm): Aurora uses the LBPH (Local Binary Patterns Histograms) algorithm for facial recognition. It analyzes local facial textures and converts them into histograms to identify individuals. LBPH is fast and works well even in low-light conditions, making it ideal for real-time recognition.

Chatbot Integration (HugChat via Hugging Face): HugChat, powered by Hugging Face models, enables natural and intelligent text-based conversations in Aurora. It uses pre-trained GPT models to understand context and generate human-like replies. This makes the assistant capable of answering queries and holding meaningful chats.

Hotword Detection (Porcupine Engine): Porcupine Engine detects the custom hotword "Hey Luna" using lightweight and offline processing. It listens continuously for the hotword with very low CPU usage. Once detected, Aurora becomes active and ready to respond to commands.

Live Server:
Aurora uses a live server during development to instantly reflect code changes in the browser. This tool automatically reloads the web interface whenever HTML, CSS, or Python scripts are updated. It helps speed up testing and debugging by providing real-time feedback during development.

Feature	Algorithm/Model
Emotion Detection (music)	Feedforward Neural Network (via Keras, model.h5)
Facial Recognition	LBPH – Local Binary Pattern Histogram
Hotword Detection	Porcupine Engine (.ppn file + PyAudio)
Voice Commands	SpeechRecognition + pyttsx3
Chatbot Responses	Hugging Face's transformer models (HugChat)
Live server	Real-Time Web Preview

Table-1. Algorithms Used

5. Implementation

The implementation of *Aurora*, an AI-based voice assistant, involves integrating multiple smart modules into a unified system. Built primarily using Python, Aurora combines functionalities like voice recognition, facial analysis, chatbot communication, and real-time automation. The interface is developed using Streamlit to provide users with an interactive and visually friendly platform. Throughout development, a live server is used to ensure that any changes in code are immediately reflected, making testing and debugging faster and more efficient.

As soon as the appliaction is started, Aurora immediately recognizes the user's face for verification. This facial recognition process, powered by the LBPH (Local Binary Patterns Histograms) algorithm, helps identify the user and secure the system. Once the user is verified, the

assistant proceeds to activate the core functionalities. If the user's face is not recognized, Aurora can prompt for re-verification, ensuring that only authorized users can access personalized features.

Aurora remains in an idle state until it detects the custom hotword "Hey Luna." This hotword detection is powered by the Porcupine Engine, which is lightweight and works offline, constantly listening with minimal resource usage. Once the hotword is recognized, the assistant activates and becomes ready to interact with the user. This approach ensures the system is responsive and energy-efficient, while also respecting user privacy by activating only when needed.

After activation, users can communicate with Aurora through voice commands. SpeechRecognition is used to convert spoken words into text, which the assistant processes to understand the user's intent. Aurora then responds using pyttsx3, a text-to-speech engine that speaks out answers in a natural-sounding voice. This hands-free interaction allows users to control tasks, ask questions, and receive responses without touching a screen or keyboard.

Aurora also observes the user's facial expressions to detect emotions. Using MediaPipe, the assistant identifies key facial landmarks in real time, which are then analyzed by a trained neural network model. The model, built with TensorFlow/Keras, classifies the user's mood (such as happy, sad, or neutral), allowing Aurora to tailor its responses. For example, it might suggest upbeat music when the user appears sad, creating a more emotionally intelligent experience.

To further personalize interactions, Aurora uses facial recognition powered by the LBPH (Local Binary Patterns Histograms) algorithm. This algorithm captures unique facial textures and compares them with stored patterns to recognize individuals. Once the user is identified, Aurora can adapt its behavior, settings, or recommendations based on previous interactions, making the experience more customized and secure.

For intelligent conversation, Aurora integrates a chatbot using HugChat via Hugging Face. This component allows the assistant to understand and respond to user inputs in a natural and human-like way. By leveraging GPT models, the chatbot can answer questions, continue dialogues, or offer assistance across various topics. This adds a social and conversational dimension to Aurora, making it feel more like a companion than a tool.

Finally, the entire development is supported by tools like Streamlit for interface design and a live server for real-time updates. These tools ensure smooth coordination between front-end interaction and back-end processing. Aurora, with its combination of advanced modules and intelligent behavior, serves as a powerful and user-friendly voice assistant capable of understanding, interacting, and adapting to the user's needs.

6. Results:

The results of Aurora's implementation demonstrate its ability to seamlessly detect and respond to user inputs through facial recognition, hotword activation, and voice commands. The system accurately recognizes emotions and adapts its responses accordingly, enhancing user interaction. Overall, Aurora provides an intelligent, secure, and responsive voice assistant experience, tailored to the user's preferences and emotional state.

Fig 2. Initializing

The diagram shows the initialization animation of Aurora, where the system begins by verifying the user's face and preparing for interaction. This animation signifies the system's readiness to detect the hotword and activate the assistant seamlessly.

Fig 3. Facial recognition

The diagram represents Aurora detecting the user's face during initialization, ensuring secure verification before activation. This process is visualized as an animation, indicating the system's readiness to proceed with interaction once the face is recognized.

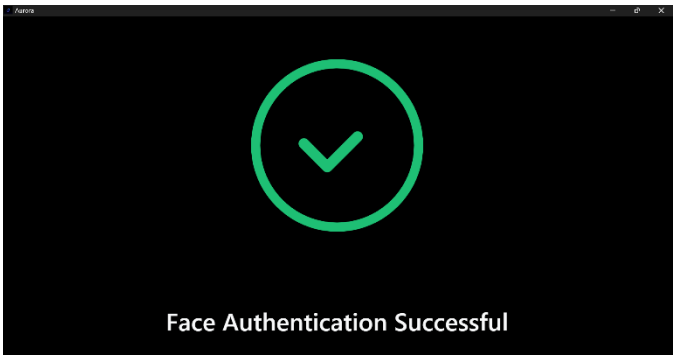


Fig 4. Authentication Successful

The diagram illustrates successful facial authentication, where the system recognizes the user and grants access. A green checkmark and confirmation message indicate secure and verified login to Aurora.

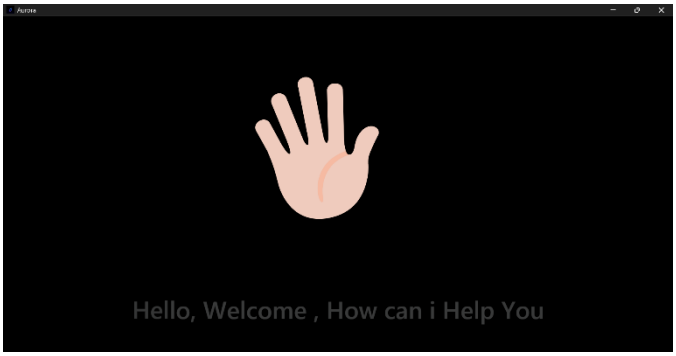


Fig 5. Welcoming to application

The diagram shows Aurora welcoming the user after successful facial authentication, marking the start of interaction. A friendly greeting message is displayed, creating a personalized and engaging experience.

Fig 6. Main page

The main page diagram displays a chat box with a prompt saying “Ask me anything,” inviting the user to start a conversation. This interface makes interaction with Aurora intuitive and user-friendly right from the start.

Fig 7. Siri-like wave animation

After calling “Hey Luna,” a Siri-like wave animation appears on the main page, indicating that Aurora is actively listening. This visual cue confirms that the assistant is ready to receive and process the user's voice query.

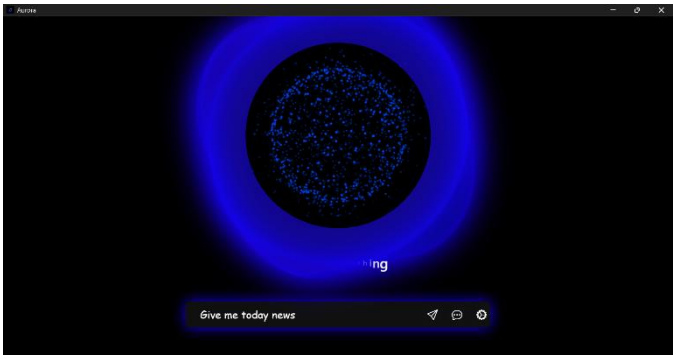


Fig 8. Text chat

The above diagram shows a message in the chat box, indicating that the user can interact with Aurora by typing or speaking. It reflects a responsive interface where queries and replies appear in real-time for a smooth conversational flow.

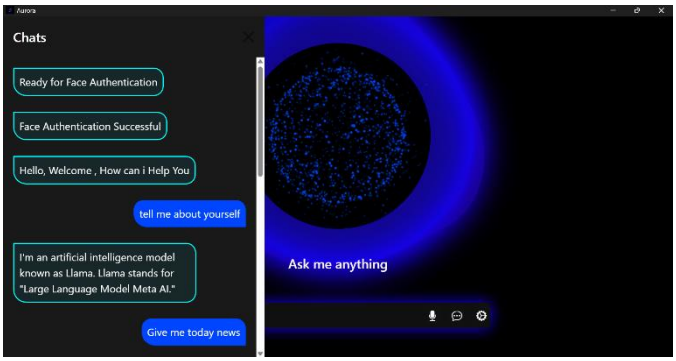


Fig 9. History

The diagram shows the chat history between the user and Aurora, displaying previous queries and responses. This feature helps users keep track of the conversation and continue interactions seamlessly.

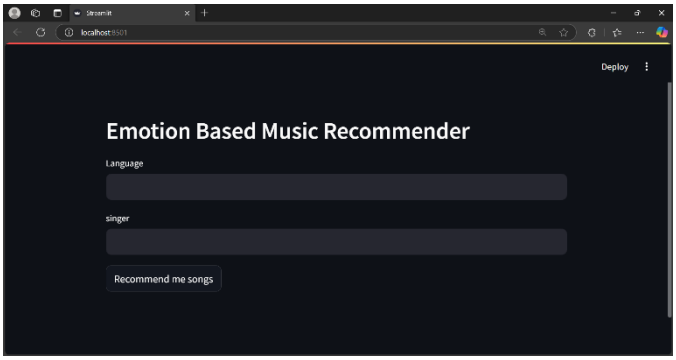
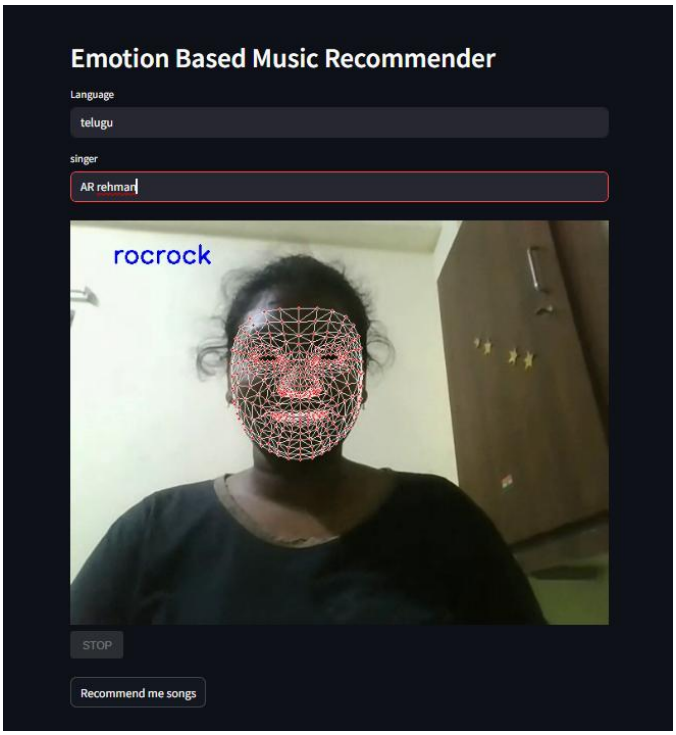
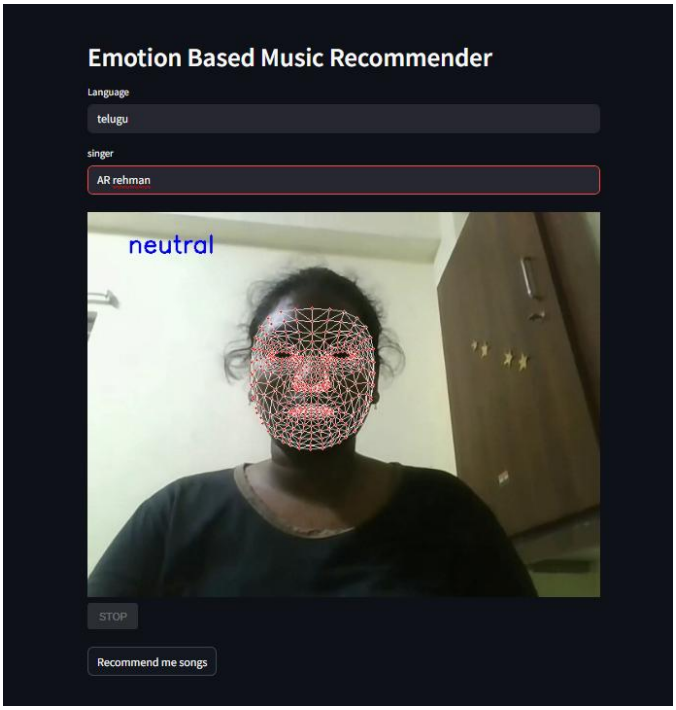


Fig 10. Music recommendation

After the user asks “Recommend me music,” the system redirects to a music recommendation page. This page presents curated song suggestions, allowing the user to explore personalized music options based on their preferences.





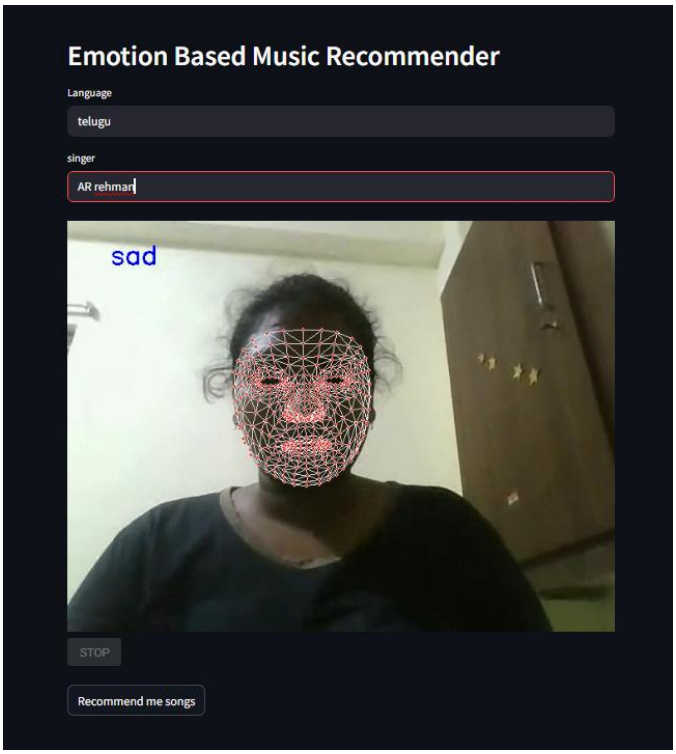


Fig 11. Emotion Detection

The diagram shows the process after entering the preferred language and favourite singer, where the system then detects the user's face for verification. This ensures personalized recommendations and security before proceeding with the next steps.

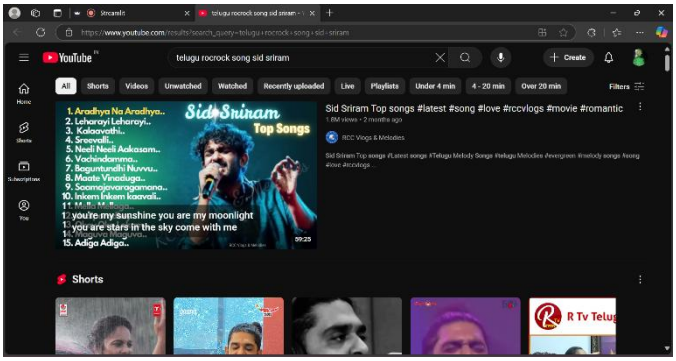


Fig 12. Music recommended

After the face detection and verification, the system recommends music based on the user's preferences, such as language and favourite singer. The assistant then redirects to YouTube, where the recommended music video plays, allowing the user to enjoy the suggestions.

After the face detection and verification, the system not only recommends music based on the user's preferences but also handles other requests like calling on WhatsApp. When asked to "Call WhatsApp," Aurora redirects to the app and initiates the call, providing a seamless integration of both music recommendations and communication tasks.

7. CONCLUSIONS

In conclusion, Aurora stands as an innovative and intelligent voice assistant that integrates multiple advanced technologies to provide a personalized user experience. By combining facial recognition, emotion detection, and voice control, it adapts to the user's needs in real-time, ensuring seamless and intuitive interactions. The system's ability to understand voice commands, process facial inputs, and provide accurate responses sets it apart as a highly interactive and dynamic assistant.

The integration of external services, like music recommendations and WhatsApp calling, further enhances its versatility, making it not just a virtual assistant but a smart, all-in-one platform for managing everyday tasks. Whether it's recommending mood-based music or initiating calls, Aurora is designed to cater to a wide range of user needs. The system ensures a smooth transition between various tasks and makes the overall user experience engaging and efficient.

Overall, Aurora combines state-of-the-art technologies into a cohesive system, transforming the way users interact with virtual assistants. With its personalized touch, real-time responses, and smooth integration with other apps, it proves to be a reliable and innovative assistant, ready to handle both simple and complex tasks in a user-friendly manner.

8. FUTURE SCOPE:

1. Enhance emotional intelligence with advanced machine learning for recognizing emotional cues and offering mood-based recommendations.
2. Expand compatibility with smart devices and platforms to control home automation and seamlessly integrate across devices.
3. Support multiple languages and dialects to increase accessibility and adapt responses based on cultural contexts.
4. Offer personalized experiences by analyzing user behavior and preferences for tailored responses and recommendations.

5. Integrate a news feed feature to provide users with real-time updates on current events, tailored to their interests and preferences.

REFERENCES

1. Reddy, V., & Kumar, S.: *Building Virtual Assistants: Siri, Alexa, and Beyond*. O'Reilly Media, Inc., Sebastopol (2019).
2. Johnson, R.: *Practical Guide to Alexa Skills Development*. Packt Publishing, Birmingham (2021).
3. Singh, A., & Gupta, P.: *Voice-Based Virtual Assistants: Design and Implementation*. Springer, Berlin (2020).
4. Lee, J., & Lee, K.: *Emotion Recognition and Music Recommendation Systems*. Springer, Berlin (2018).
5. Zhang, S., & Wang, H.: *Personalized Music Recommendation: Machine Learning Techniques for Emotion Detection*. Elsevier, Amsterdam (2017).
6. Bhattacharya, S., & Roy, A.: *Emotion-Based Music Recommendation System Using Deep Learning*. Springer, Berlin (2019).
7. Tzirakis, P., et al.: *Deep Learning for Emotion Recognition: A Survey*. IEEE Transactions on Affective Computing 10(1), 2020.
8. Zhang, Y., & Liu, L.: *Music Recommendation Systems Based on Emotion and Contextual Information.* Elsevier, Amsterdam (2016).
9. Sharma, A., & Kapoor, S.: *AI-Powered Music Recommendation: Understanding Emotion and Context.* Springer, Singapore (2021).