MUSIC PLAYER APPLICATION

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

**SRM University – AP, Andhra Pradesh**

Submitted by

AP22110010660 – N. Eswar Chowdary

AP22110010661 – V. Siva Karthik

AP22110010687 – N. Pavan Reddy

AP22110010693 - N. Mahesh Babu

ABSTRACT

This project presents a simple yet functional **Music Player application** developed using **Java AWT (Abstract Window Toolkit)**. The primary objective of the application is to enable users to play .wav audio files directly from their local system through a graphical interface. The music player supports basic functionalities such as **Play**, **Pause**, **Stop**, **Next**, and **Previous**, along with a list of available songs for user selection.

The user interface is built entirely using **Java AWT components**, providing a lightweight desktop application that is platform-independent. The core of the audio playback functionality is powered by the **Java Sound API**, which allows seamless handling of audio streams and clip controls. The program reads all .wav files from a designated "music" folder, lists them in a scrollable UI component, and enables real-time interaction using event-driven programming.

This project demonstrates key Java programming concepts including **GUI design**, **event handling**, **file I/O**, and **audio processing**, making it a valuable educational exercise in building desktop-based media applications. It can be extended further to support additional formats (like MP3), playlists, volume control, and modern UI frameworks like JavaFX for enhanced user experience.

INTRODUCTION

In today’s digital age, music has become an integral part of our lives, offering relaxation, motivation, and entertainment. With the increasing demand for multimedia applications, building a simple and interactive music player provides a practical way to understand key programming concepts in software development.

This project titled **"Java AWT-Based Music Player"** is a desktop application developed using **Java's Abstract Window Toolkit (AWT)** and **Java Sound API**. The goal of this project is to create a user-friendly music player that allows users to play, pause, stop, and navigate through .wav audio files stored locally on their system. The application features a basic graphical user interface (GUI) that displays the list of available songs and provides control buttons for playback operations.

By leveraging Java’s core libraries and event-driven programming model, this project demonstrates essential skills in **GUI design, audio file handling, file system access, and real-time event management**. It serves as a foundation for understanding how multimedia applications work at a fundamental level and can be extended with more advanced features in the future, such as MP3 support, volume control, playlists, and enhanced visual design using modern frameworks like JavaFX.

**OBJECTIVE**

The primary objective of this project is to develop a simple and interactive desktop-based music player using Java. The application aims to provide users with basic audio playback functionalities such as Play, Pause, Stop, Next, and Previous through a clean and user-friendly graphical interface built using Java AWT components. It is designed to automatically load .wav audio files from a local directory and display them in a selectable list, enabling users to choose and play songs of their choice. The project leverages the Java Sound API to handle audio streaming and clip control operations efficiently. Additionally, it emphasizes the use of event-driven programming to manage user interactions with various GUI elements. Through this project, the goal is to strengthen understanding of core Java concepts including GUI development, file handling, event management, and audio processing, while building a lightweight, cross-platform media application that can serve as a foundation for more advanced multimedia systems in the future.

**METHODOLOGY**

The development of the Java AWT-Based Music Player followed a structured approach to ensure functionality and simplicity. The user interface was built using Java AWT components such as Frame, Panel, Button, Label, and List to organize the layout. The interface includes a song list loaded from a local "music" folder, control buttons for playback, and a label to show the currently playing track.

Audio playback was managed using the Java Sound API, specifically with Audio System, Audio Input Stream, and Clip classes to handle operations like play, pause, and stop. Event-driven programming was applied through ActionListener and Item Listener interfaces to respond to user actions like button clicks and song selection.

Additionally, a Window Listener was implemented to stop playback and exit the application gracefully. The overall coding approach focused on modularity, making the application easy to understand, maintain, and extend. Proper exception handling was added to manage unexpected errors such as missing audio files or unsupported formats. The playback logic ensures that only one song plays at a time, avoiding overlapping audio. Default song selection and automatic UI updates were implemented to enhance the user experience. The design also allows future enhancements like volume control, playlist creation, and support for more audio formats.

**SYSTEM ARCHITECTURE**

The system architecture of the Java AWT-Based Music Player follows a modular and layered design, where each component is responsible for a specific functionality. At the core, the application is divided into three main layers: the User Interface Layer, the Event Handling Layer, and the Audio Playback Layer. The User Interface Layer is developed using Java AWT components like Frame, Panel, Button, Label, and List, which together create a simple and interactive GUI. This layer displays the list of available songs and provides buttons for Play, Pause, Stop, Next, and Previous functionalities.

The Event Handling Layer manages all user interactions with the GUI. It listens to button clicks and song selections using ActionListener and ItemListener interfaces and triggers the corresponding actions in the application. This ensures smooth control of the music player based on user input.

The Audio Playback Layer is responsible for handling the playback of audio files using the Java Sound API. It uses classes such as AudioSystem, AudioInputStream, and Clip to load, play, pause, and stop .wav files. These three layers work in coordination to deliver a seamless and responsive user experience. The architecture is designed to be lightweight, easy to understand, and extendable for future enhancements like volume control, playlist support, and additional audio formats.

**TECHNOLOGIES**

* **Java (JDK 8 or higher)** – Core programming language used for application development.
* **Java AWT (Abstract Window Toolkit)** – For designing the graphical user interface (GUI) components such as Frame, Panel, Button, Label, and List.
* **Java Sound API** – To handle audio playback using classes like AudioSystem, AudioInputStream, and Clip.
* **File I/O (Input/Output)** – For reading .wav files from the local "music" directory.
* **Event-Driven Programming** – Using interfaces like ActionListener, ItemListener, and WindowListener to manage user interactions.
* **Exception Handling in Java** – To catch and manage runtime errors such as missing or unsupported files.
* **Command Line / Terminal** – To compile and run the Java application.
* **Any Java-supported IDE**  – For writing, debugging, and running the code efficiently.

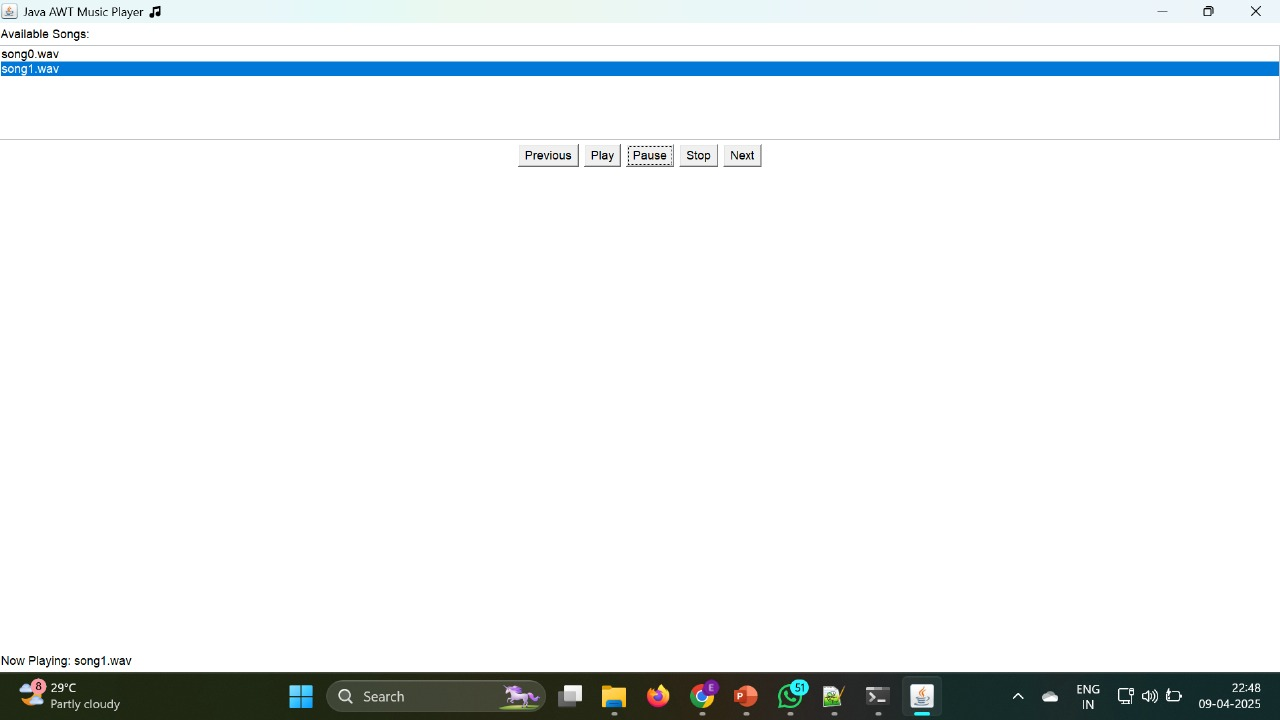
**IMPLEMENTATION**

* **Designed the main window using Java AWT's Frame and BorderLayout.**
* **Loaded .wav files from the local "music" folder using File I/O.**
* **Displayed songs in a List component with dynamic loading.**
* **Added playback buttons (Play, Pause, Stop, Next, Previous) with ActionListener.**
* **Used Java Sound API (AudioSystem, Clip) to handle audio playback.**
* **Implemented song selection via ItemListener to play chosen tracks.**
* **Managed Play/Pause toggle using clip's microsecond position.**
* **Used WindowListener to stop audio and close the app gracefully.**
* **Included exception handling for missing or unsupported files.**

**RESULTS AND OUTPUTS**

**The Java AWT Music Player was successfully developed and tested, providing a functional and interactive desktop-based audio playback application. Upon launching the application, all .wav files from the designated "music" folder are dynamically loaded and displayed in a scrollable list. Users can select any song and use the provided buttons to play, pause, stop, skip to the next track, or return to the previous one. The currently playing song is shown in a status label at the bottom of the interface, enhancing the user experience. The play/pause toggle works smoothly, resuming the song from where it was paused. The system handles file errors gracefully and exits cleanly upon window close. The outputs demonstrate that the music player is stable, responsive, and performs all intended operations correctly across different machines with Java installed.**

**OUTPUT:**

****

**CONCLUSION**

**The Java AWT-Based Music Player project successfully demonstrates how core Java concepts can be combined to build a simple yet effective multimedia application. By leveraging Java AWT for the graphical user interface and the Java Sound API for audio playback, the project delivers a fully functional desktop application capable of playing .wav files with basic controls like play, pause, stop, next, and previous. The implementation showcases event-driven programming, file handling, and sound processing in a modular and user-friendly design. Overall, the project meets its objectives by providing an interactive and responsive experience, while also laying a strong foundation for further enhancements such as volume control, playlist support, and compatibility with more audio formats.**