UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Mini Project

Student Name: Vivek Kumar **UID: 22BCA10237**
Branch: BCA **Section/Group: 22BCA-3A**
Semester: 5th **Date of Performance:**
Subject Name: Web Development **Subject Code: 22CAH-301**

Q. Create a Music Player desktop Application using c#.

1. Aim/Overview of the practical:

The Music Player is a software multimedia player implemented as an application ona desktop application with the user interface developed for the basic audio players that are made to perform very straightforward functionality. On one hand, the interface enables the loading of music songs into the playlist which the user then views at his/her instance in list boxes. Finally, different controls have been implemented around it to manipulate the listenability of those songs which he or shehas opted for. This includes

Play: Start Playing of the Selected Song.
Next/Previous Navigation: Let you browse songs in the playlist.
User-friendly interface: streamlined layout with a list box for song choice andbutton for the basic actions.

2. Task to be done:

- **Set Up the Development Environment**

Install the necessary development tools and libraries (e.g., Visual Studio for C# orPython IDE for Play Game).
Set up the project framework and structure.

- **Design the User Interface**

Create a List Box to display and select songs from the playlist.

Add Play/Pause, Next, and Previous buttons.
Optionally, add a Picture Box to display album art or a logo.

- **Implement File Loading Functionality**

Develop a feature to load audio files from the system into the list box. Enable multiple file selection if supported by the language or library used.

- **Develop Playback Controls**

Implement the Play and Pause functionality to start and stop playback. Code the Next and Previous buttons to navigate through the songs in the list.

- **Handle Media Playback**

Write code to manage media playback, ensuring smooth transitions between songs.

Ensure the application can handle various audio formats (e.g., MP3, WAV).

- **Test Functionality**

Test each feature individually to confirm functionality (play/pause, next/previous,file loading).
Conduct usability testing to make sure the interface is intuitive and responsive.

- **Optimize for Performance**

Optimize the application for efficient memory use and smooth playback. Handle exceptions (e.g., file not found or unsupported format) to improve userexperience.

- **Final Review and Documentation**

Review the entire application to ensure all tasks are complete and working asintended.
Write documentation, including a user guide and comments within the code forclarity.

3. Algorithm/Flowchart :

1. Initialize Application

- **Start the application.**
- **Set up the user interface with components like list box,buttons (Play, Pause, Next, Previous), and picture box.**

2. Load Songs

- **When the user selects the "Load" option:**
  - **Open a file dialog to select audio files.**
  - **Add selected files to the list box as a playlist.**

3. Play Song

- **When the "Play" button is clicked:**
  - **Check if a song is selected in the list.**
  - **If yes, start playback of the selected song.**
  - **If the song is already playing, resume playback if itwas paused.**

4. Pause Song

- **When the "Pause" button is clicked:**
  - **Pause the current song playback.**
  - **Retain the current playback position.**

5. Next Song

- **When the "Next" button is clicked:**
  - **Check if there is a next song in the playlist.**
  - **If yes, play the next song in the list.**
  - **If at the end of the list, loop back to the first song (optional).**

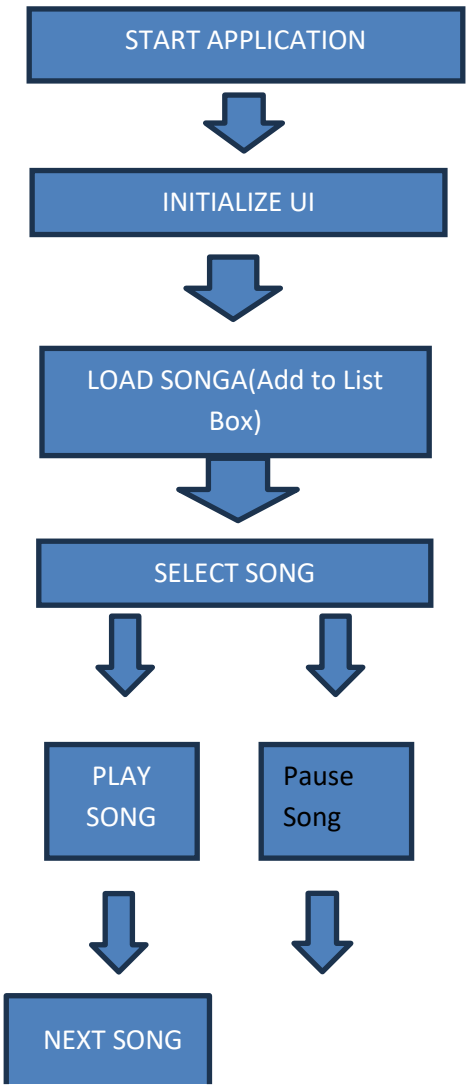
6. Previous Song

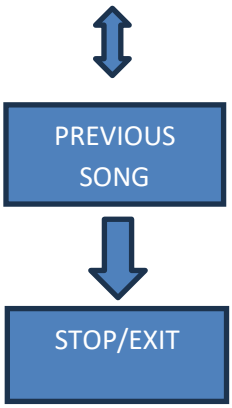
- **When the "Previous" button is clicked:**
  - **Check if there is a previous song in the playlist.**
  - **If yes, play the previous song.**
  - **If at the start of the list, loop back to the last song (optional).**

7. Stop or Exit Application

- **Stop all playback and exit the application when the usercloses the window.**

Flowchart:





4. Dataset:

- **File Name: The name of each audio file (e.g., song1.mp3).**

- **File Path: The directory path where the audio file is stored, allowing theapplication to locate and load the file.**

- **File Format: The audio file format (e.g., MP3, WAV, AAC) to help ensurecompatibility with the player.**

5. Code for experiment/practical:

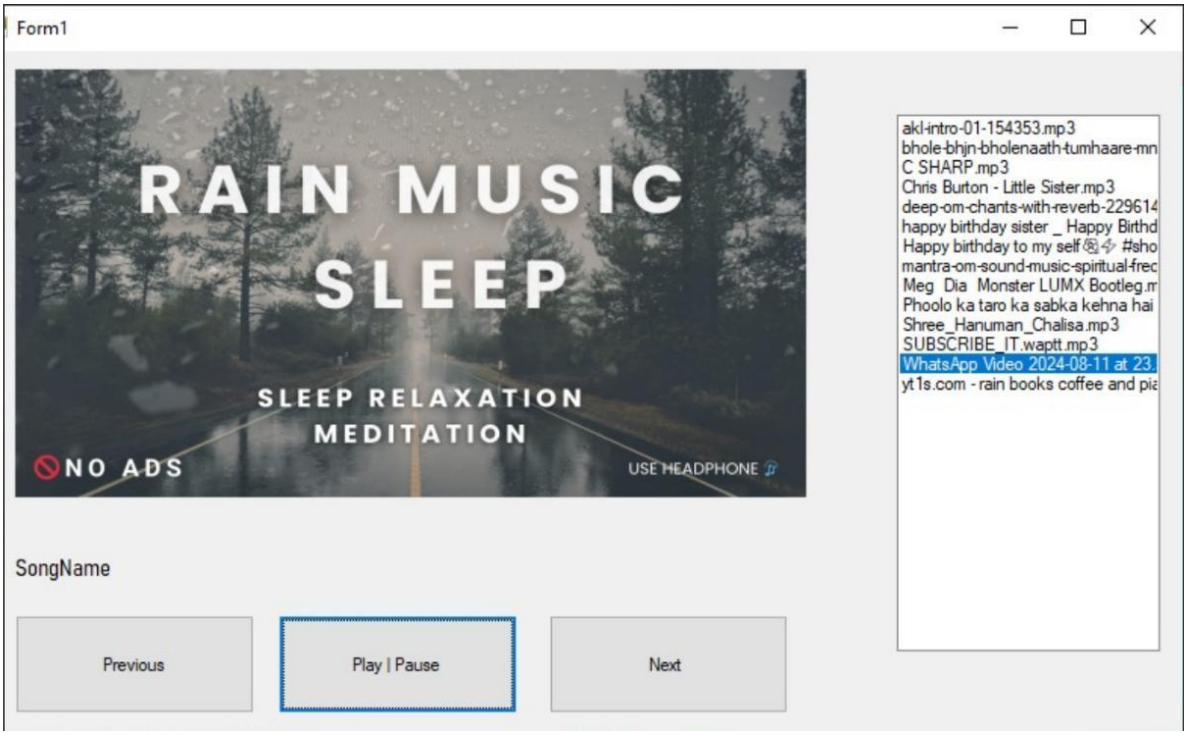
```
using System;
using System.Collections.Generic;using
System.IO;
using System.Linq;
using System.Windows.Forms;using
NAudio.Wave;

namespace Music_Player
{
    public partial class Form1 : Form
    {
        private List<string> songs;
```

```
private IWavePlayer waveOut;
private AudioFileReader audioFileReader;private
int currentSongIndex;

public Form1()
{
    InitializeComponent();
    LoadSongs();
}

private void LoadSongs()
{
    // Get all MP3 files from the Downloads folder
    string downloadsPath = @"C:\Users\dell\Downloads";
    songs = Directory.GetFiles(downloadsPath, "*.mp3").ToList();

    // Populate the ListBox with song names
    foreach (var song in songs)
    {
        Songs.Items.Add(Path.GetFileName(song));
    }
}

private void Songs_SelectedIndexChanged(object sender, EventArgs e)
{
    if (Songs.SelectedIndex != -1)
    {
        currentSongIndex = Songs.SelectedIndex;
        PlaySelectedSong();
    }
}

private void PlaySelectedSong()
{
    if (waveOut != null)
    {
        waveOut.Stop();
```

```
            waveOut.Dispose();
        }

        string selectedSongPath = songs[currentSongIndex];
        waveOut = new WaveOutEvent(); // Initialize the audio player audioFileReader = new
        AudioFileReader(selectedSongPath); // Read the
selected MP3 file
        waveOut.Init(audioFileReader); // Initialize the player with the audio file
        waveOut.Play(); // Play the audio
    }

    private void PlayPause_Click(object sender, EventArgs e)
    {
        if (waveOut != null)
        {
            if (waveOut.PlaybackState == PlaybackState.Playing)
            {
                waveOut.Pause(); // Pause if currently playing
            }
            else
            {
                waveOut.Play(); // Play if currently paused
            }
        }
    }

    private void Previous_Click(object sender, EventArgs e)
    {
        if (currentSongIndex > 0)
        {
            currentSongIndex--;
            Songs.SelectedIndex = currentSongIndex;
        }
    }

    private void Next_Click(object sender, EventArgs e)
    {
```

```
            if (currentSongIndex < songs.Count - 1)
            {
                currentSongIndex++; Songs.SelectedIndex =
                currentSongIndex;
            }
        }

        private void SongName_Click(object sender, EventArgs e)
        {

        }
    }
}
```

6. Result/Output/Writing Summary:



Learning outcomes (What I have learnt):

1. **Understanding Media Handling – Learn to manage audio files programmatically, including loading, playing, and pausing tracks.**

2. **User Interface Design – Gain experience in creating a user-friendly interfacewith interactive elements like buttons and list boxes.**

3. **Event-Driven Programming – Develop skills in handling user-driven events suchas play, pause, and navigation between songs.**

4. **Problem-Solving and Debugging – Enhance ability to troubleshoot and resolveissues related to file handling and playback control.**

5. **State Management – Learn how to manage application states (e.g., playbackstatus, song index), essential for building smooth, responsive apps.**

Evaluation Grid:

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Demonstration and Performance (Pre Lab Quiz)		5
2.	Worksheet		10
3.	Post Lab Quiz		5