SEMESTER PROJECT

MUSIC PLAYER IN WINFORMS

TERÉZIA MACHAROVÁ

SUMMER SEMESTER I. YEAR

2022/2023

Contents

1. Summary

This application is written in a programming language C# using the graphical interface WinForms. The application works with files and its purpose is to be able to play MP3 and WAV audio files, that the user has on their computer. They upload their files to the application and then they can listen to them, with functions such as stopping, pausing, skipping or going back to the previous file. There is also a function where they can search for a specific file (the file must be uploaded in the application) which gives the user an option for sorting by an artist or a song title.

2. User part

This part is for users who want to use this application on their device. It is supposed to be a summarized user guide. The goal of this part is to let the user know how to run the application, what data are needed and what to expect.

2.1. How to run the application

To run the application, it's necessary, to have some kind of IDE downloaded, for example, Visual Studio from Microsoft. The application is written in C#, so any appropriate .NET runtime or SDK installed is necessary to compile and run the application. The application uses libraries such as NAudio (version 2.1.0) and Newtonsoft.Json (version 13.0.3) which are also necessary to have when running this application. These can be installed through NuGet. The code is also intended to run with GUI, to be exact WinForms, so it's also necessary to be able to run this framework. It's also necessary to have a device that has audio output capabilities such as speakers or headphones.

2.2. Input data

As input data, we could consider *MP3* and *WAV* files that the application takes. If a new user runs this application for the first time, they need to upload some audio files to the application first. This is done with the *browse button*. After clicking on this button, new window is opened, and the user can choose the files they want to upload. After this, the audio files are saved into file *playlist.json*, where they are kept until removed by the user with the *remove button* in the application (or if the user removes them from their device), even after closing and reopening the application. The user can see what files are already added and can rename them in the application by double-clicking on them, if they are renamed and then removed and added again, the user needs to rename them again.

2.3. Output data

The output data are shown after the user interacts with the application in some way. It can be by clicking on a button to start or stop the playback or interact with the file in any other way. It also can be by opening a window for browsing the files, renaming them, or removing them.

2.4. Controlling the application

The app is controlled mainly with a cursor, the user can click on any button or file to get some response from the application, depending on what they want. The keys enter and the arrows work as well, but their functions are very limited. In the applications, there are multiple buttons, such as *Play, Stop, Pause, Next* and *Previous* to interact with the audio file playback. If the play button is pressed and no song is chosen, then the last song that was added will start playing. When the pause button is clicked, the playback will stop and by pressing the play button the playback will be resumed. By pressing the stop button, the playback will stop and pressing the play button will make it start from the beginning. The buttons Next and Previous work in a similar way, so that when one is pressed the next or previous file will be played. If it's the last or the first song and the buttons are pressed the playback is moved in a loop, so depending on what button was pressed, the first or last file will be played.
There are also buttons for managing the files, the *Browse* and *Remove song* buttons. For the remove button, the song needs to be selected to be removed. After removing, the next song will start playing.
If a song is double-clicked, the user can rename it and confirm it by clicking the *Save button* or cancel the action with the *Cancel button*. These two buttons are only visible after the double-click.
The user can also move to a specific time in a song with the help of a *track bar*.

3. Programming part

This part is for the users who want to look into the source code. It's supposed to help the user understand how the source code works, and what was used while creating the application. There are also comments in the source code to help with the orientation in the code.

3.1. Division

The source code has two main parts, one where is the logic of the Music player and the other for the GUI. The logic can be found in the *mainForm.cs* file, while the graphics are in the *mainForm.Designer.cs.* The *mainForm.cs* is then separated into multiple

classes, those are then separated into methods. Each class and method have a different purpose.

3.2. Libraries

There are multiple libraries used in this application. The first one is *NAudio*, which is used to help with the audio files. From this library, there are a few interfaces used such as *IWavePlayer* that represent the device playing the audio. From this interface, the class WaveOut has been used. This class has methods such as *Play*, *Pause* and *Stop* that help with the audio playback. Another class from this library used in this application is *AudioFileReader* which is used to read audio files specified by its file path. Another class used from NAudio is StoppedEventArgs and it's used in event handlers for when audio playback stops.
Another library is *Newtosoft.Json*. The thing that is used from this library is *JsonConvert,* used to serialize and deserialize objects to or from JSON format. This is used in two methods, *SavePlayslist()* and *LoadPlaylist().*

3.3. Data structures and classes

The source code is separated into multiple classes, each with different functionality. The first one is *MainForm* which has all the application UI-related methods. The next one is Song class, this class represents a song with its properties such as artist, title and file path. Class playlist represents a collection of songs and has methods that contain logic that manages them. Class Artist represents artist and their collection of songs. Another class, *TrackBarUpdateEventArgs* updates the track bar with the audio file playback progress. The last class is the *MusicPlayer* class and it's the main class for the logic behind the application and provides functionality to play, stop and manage the music playback. The summarized description of each method can be found in the comments in the source code.
In the application, there are a couple of data structures such as enums used for representing the playback states. Another data structure used is struct, which is used to store current song information, including artist and title. Next, there is a list used to store songs in the playlist.