Willian Bernatzki Woellner

Student ID 30021175

Programming 3

Product Specification Design and Test Documentation

04/11/2020

Version 2.0

Table of Contents

[Introduction 3](#_Toc55399882)

[Purpose 3](#_Toc55399883)

[Requirements 3](#_Toc55399884)

[Functional Requirements 3](#_Toc55399885)

[Non-Functional Requirements 3](#_Toc55399886)

[Software Quality Attributes 3](#_Toc55399887)

[Software Requirements 4](#_Toc55399888)

[Hardware Requirements 4](#_Toc55399889)

[Agile Methodology 4](#_Toc55399890)

[System Architecture 5](#_Toc55399891)

[Layer Pattern 5](#_Toc55399892)

[Design and Implementation 5](#_Toc55399893)

[Class Diagram 5](#_Toc55399894)

[Use Case Diagram 6](#_Toc55399895)

[User Interface Design 7](#_Toc55399896)

[Test Documentation 8](#_Toc55399897)

[Black Box Testing Table 8](#_Toc55399898)

[Figure 1 9](#_Toc55399899)

[Figure 2 9](#_Toc55399900)

[Figure 3 10](#_Toc55399901)

[Figure 4 10](#_Toc55399902)

[Figure 5 11](#_Toc55399903)

[Figure 6 11](#_Toc55399904)

[Figure 7 12](#_Toc55399905)

[Figure 8 12](#_Toc55399906)

[Figure 9 13](#_Toc55399907)

[Figure 10 13](#_Toc55399908)

[Figure 11 14](#_Toc55399909)

[Figure 12 14](#_Toc55399910)

[White Box Testing Table 15](#_Toc55399911)

[Figure 1 16](#_Toc55399912)

[Figure 2 16](#_Toc55399913)

[Figure 3 17](#_Toc55399914)

[Figure 4 17](#_Toc55399915)

[Product Design Specification Approval 18](#_Toc55399916)

Version History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Implemented By | Revision Date | Approved By | Approval Date | Reason |
| 1.0 | Willian Bernatzki Woellner | 22/10/2020 |  |  | Initial Design Definition draft |
| 2.0 | Willian Bernatzki Woellner | 04/11/2020 |  |  | Test Documentation and update Class Diagram |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Introduction

The Advance Music Player system is a desktop application developed for the Jupiter Mining Corporation. The main objective of this project is to provide a solution where the users can add songs to play and save the tracklist on a CSV file using a 3rd party library. Its solution should have an interface and the songs can be sorted and searched.

## Purpose

The purpose of this project is to provide a solution for the Jupiter Mining Corporation that users can play songs and save the tracklist on a CSV file. This document will outline the software design and system specifications. In addition to system architecture, system components, and software requirements as agreed upon by the customer and the project team.

# Requirements

## Functional Requirements

The Function Requirements for this project are listed below:

* The solution should have an interface.
* The user can add various songs to play and each song should be in a music format. E.g. mp3.
* All songs added should be displayed sorted.
* All songs should be stored on a Binary Tree.
* The tracklist can be saved on a CSV file.
* The user can search for a song to play.

## Non-Functional Requirements

## Software Quality Attributes

* Availability – The system should be available on a windows desktop after the installation.
* Usability – The system interface and operation must be easy to learn and use.
* Reliability – The software must be tested with each version.
* Performance – The desktop or notebook where the software is installed must have high performance.

## Software Requirements

* C# (front and back-end)
* .Net Framework 2.0
* Operating System: Windows 10
* Microsoft Excel or other to open the CSV file

## Hardware Requirements

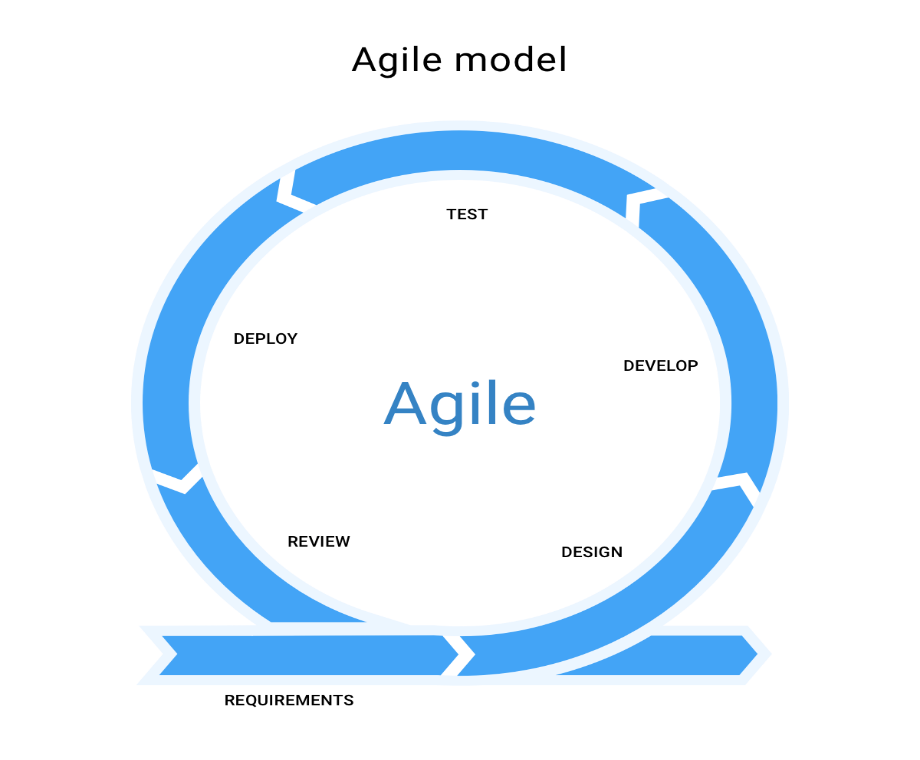
These requirements are to use the solution on the Windows 10 operating system:

* Processor: 1 gigahertz (GHz) or faster processor or SoC.
* RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit.
* Hard disk space: 16 GB for 32-bit OS or 20 GB for 64-bit OS.
* Graphics card: DirectX 9 or later with WDDM 1.0 driver.

## Agile Methodology

Agile methodology is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model, both development and testing activities are concurrent.

This project is small, all the project activities are carried out using the Agile Methodology phases.



# System Architecture

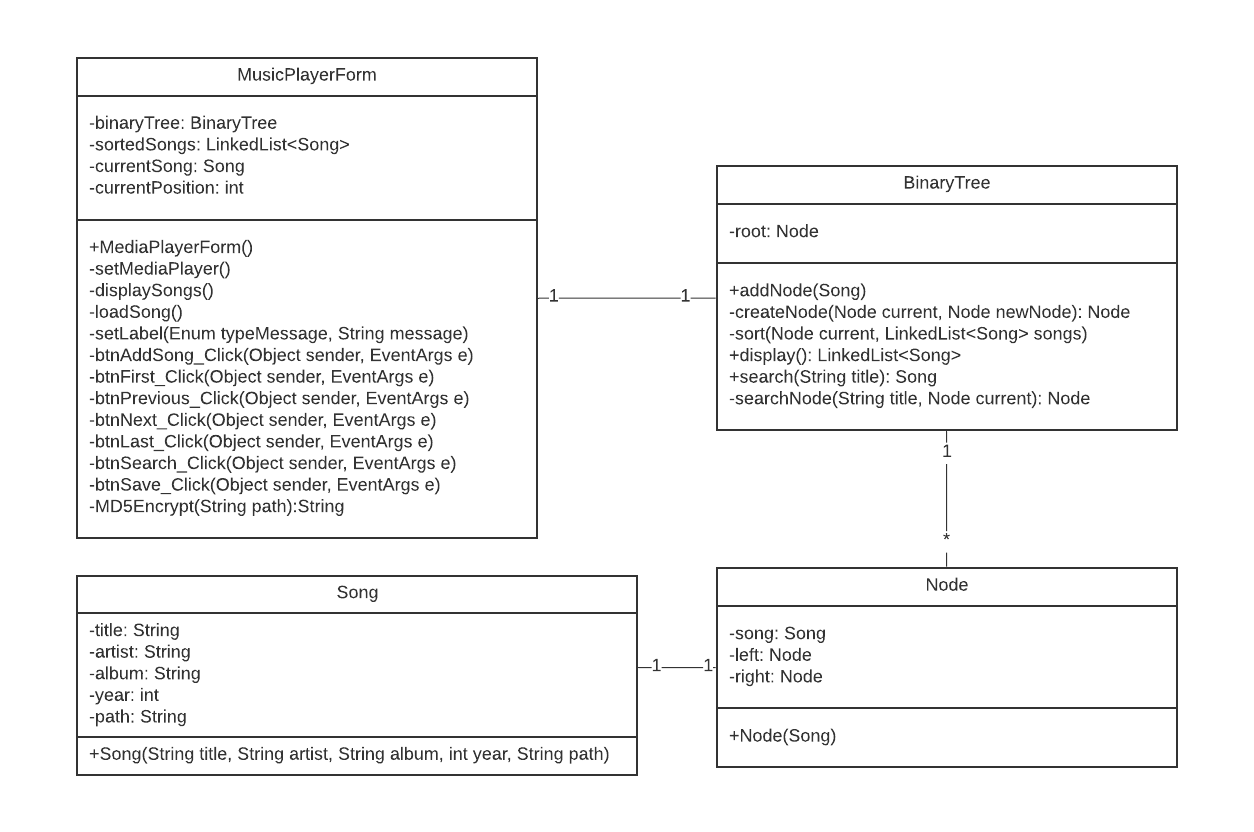
## Layer Pattern

The Layer pattern will be used to structure this program. It can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer.

* Presentation layer (UI layer)
* Application layer (Service layer)
* Business layer (Domain layer)
* Data Access layer (Persistence layer)

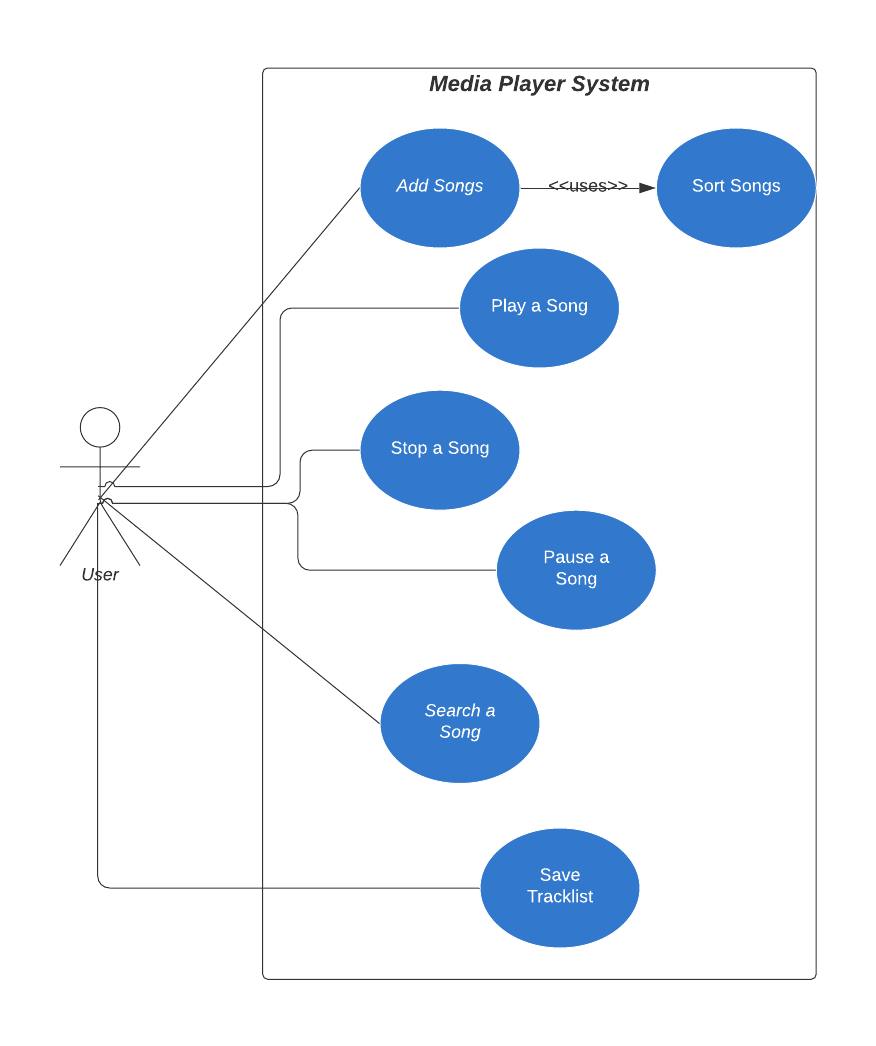
# Design and Implementation

## Class Diagram



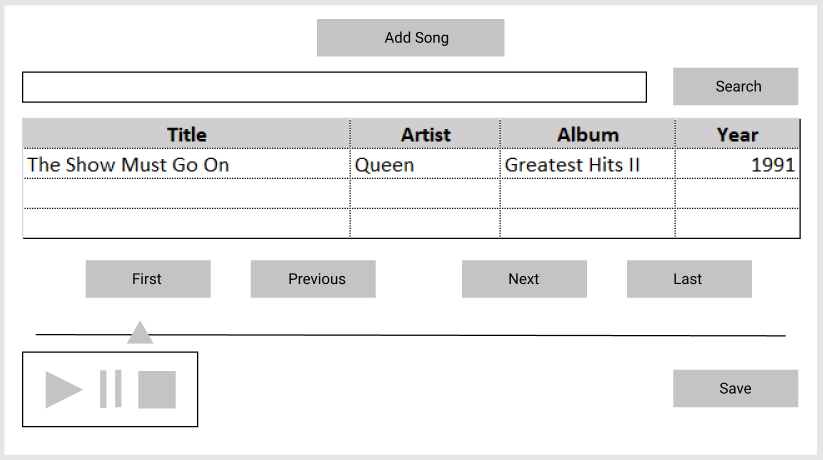
* Song: This class defines a song object with title, artist, album, and year fields.
* Node: This class defines a node object to a Binary Tree.
* Binary Tree: This class contains methods to manage the nodes.
* Music Player: This is a Main class of the Form, it contains methods of the Interface.

## Use Case Diagram



The system user can add songs to play, stop or pause a song that is played, search a song by title, and save the tracklist on a CSV file.

## User Interface Design



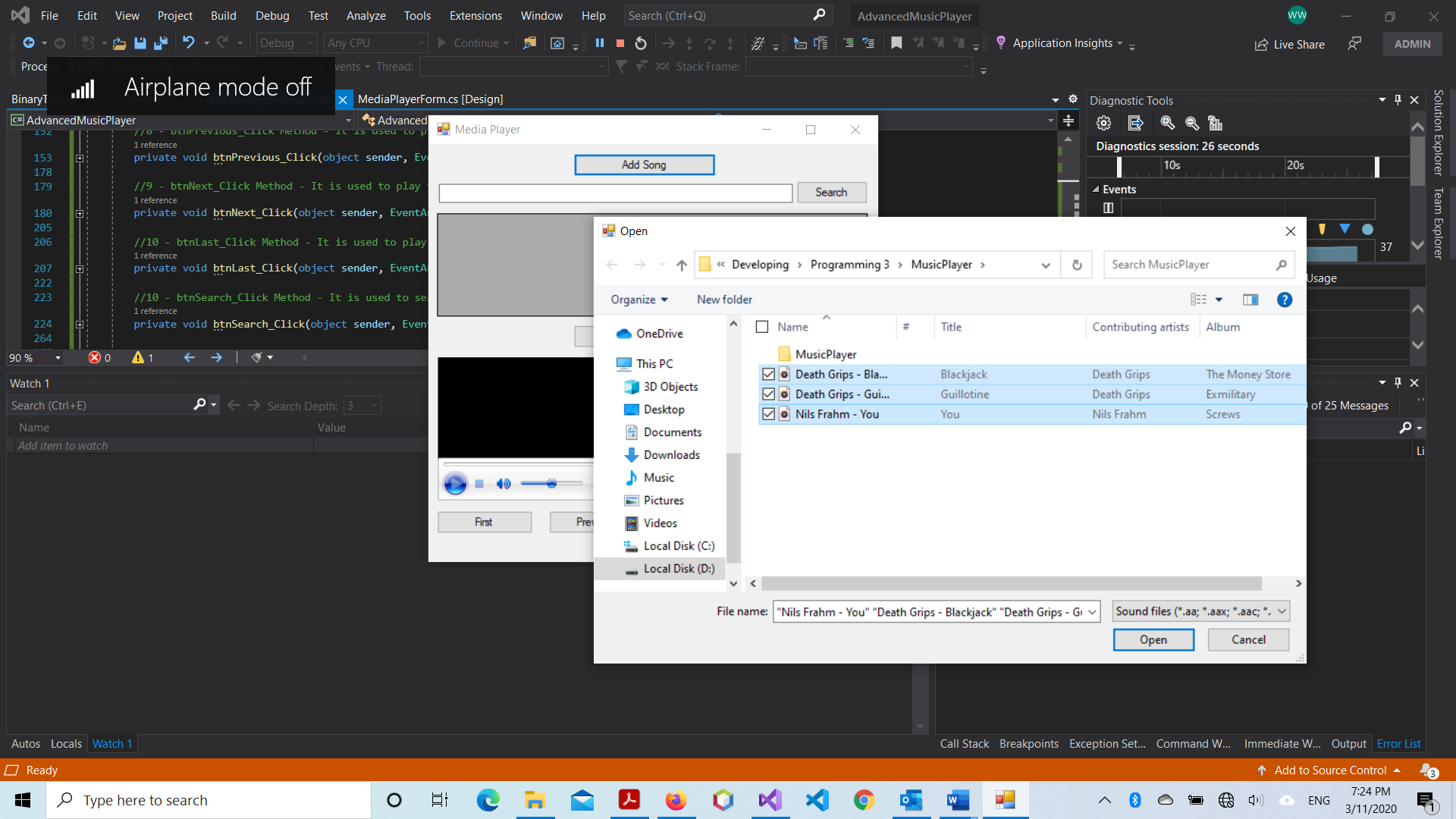
* The Search will be done by the song title.
* All songs added will be displayed on the table.
* All songs will be displayed sorted by title.
* The Save button will be used to save the tracklist on the CSV file.

# Test Documentation

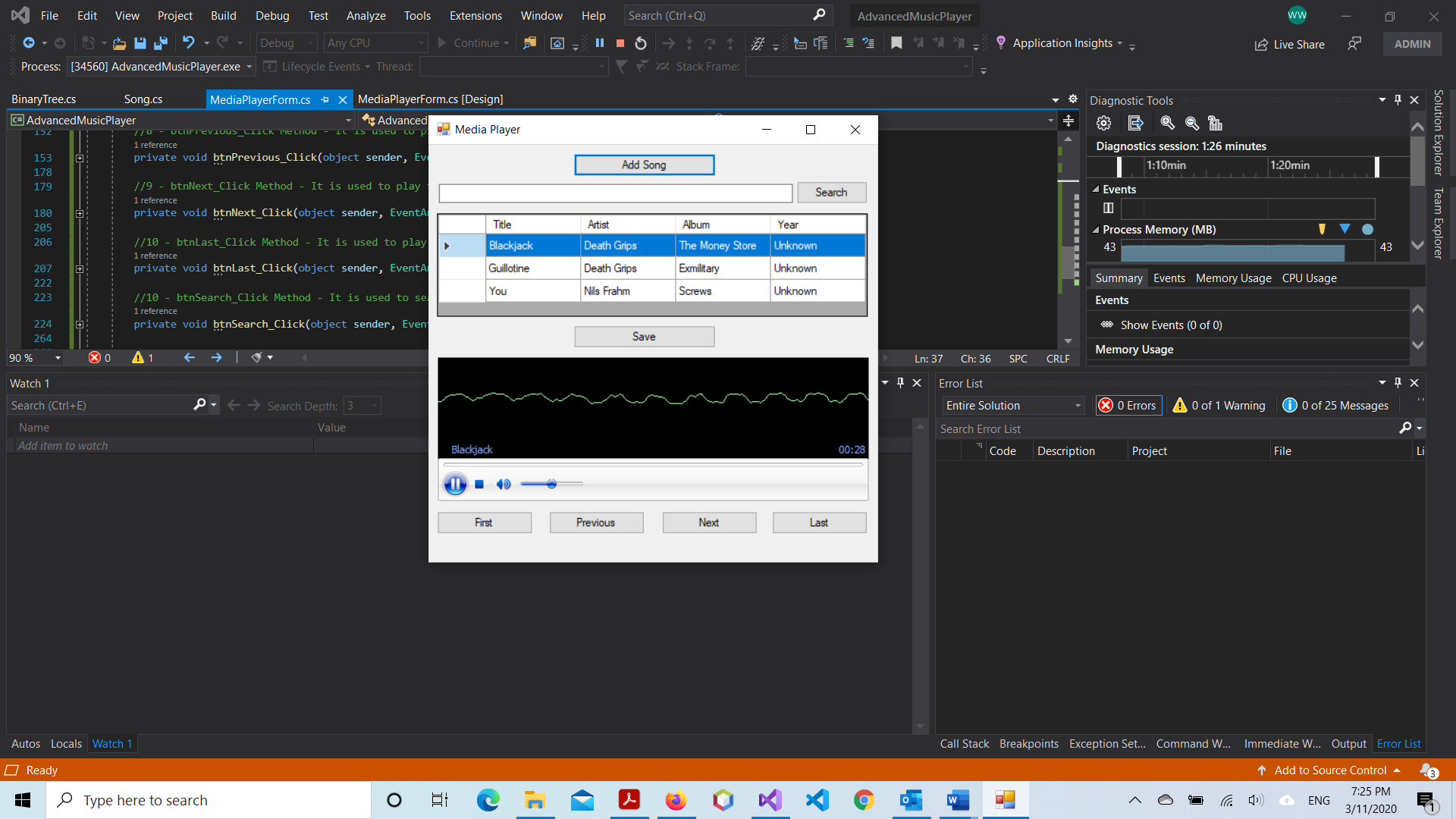
## Black Box Testing Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Actual | Comment |
| Add Songs to play | When clicking on the add Song button the user will select the files to be played and after the selection, the songs will be displayed in a table sorted by title. | The songs have been added to play and displayed on a table sorted by title. The first song started to play. | Figure 1  Figure 2 |
| Save the tracklist on a CSV file | When clicking on the Save button the system must save the tracklist on a CSV file, the file will be saved on the path configurated on the app.config file system. | The tracklist has been saved on a CSV file. | Figure 3  Figure 4 |
| Search a Song | When clicking on the search button the system must search a song by the title on the tracklist, if the song will be found, it must start to play. | The song has been found and it started to play. | Figure 5 |
| Search an invalid Song | When clicking on the search button the system must search a song by the title on the tracklist, if the song will not be found, the system must display a message. | The message has been displayed to alert the system user. | Figure 6 |
| Play the First Song | When clicking on the First button the system must play the first song by the tracklist. | The First Song has been started to play. | Figure 7 |
| Play the Last Song | When clicking on the Last button the system must play the last song by the tracklist. | The Last Song has been started to play. | Figure 8 |
| Play the Previous Song | When clicking on the Previous button the system must play the previous song by the tracklist. | The Previous Song has been started to play. | Figure 9 |
| Play the Next Song | When clicking on the Next button the system must play the next song by the tracklist. | The Next Song has been started to play. | Figure 10 |
| Pause the current Song | When clicking on the Pause button the system must pause the current Song. | The current Song has been paused. | Figure 11 |
| Stop the current Song | When clicking on the Stop button the system must stop the current Song. | The current Song has been stopped. | Figure 12 |

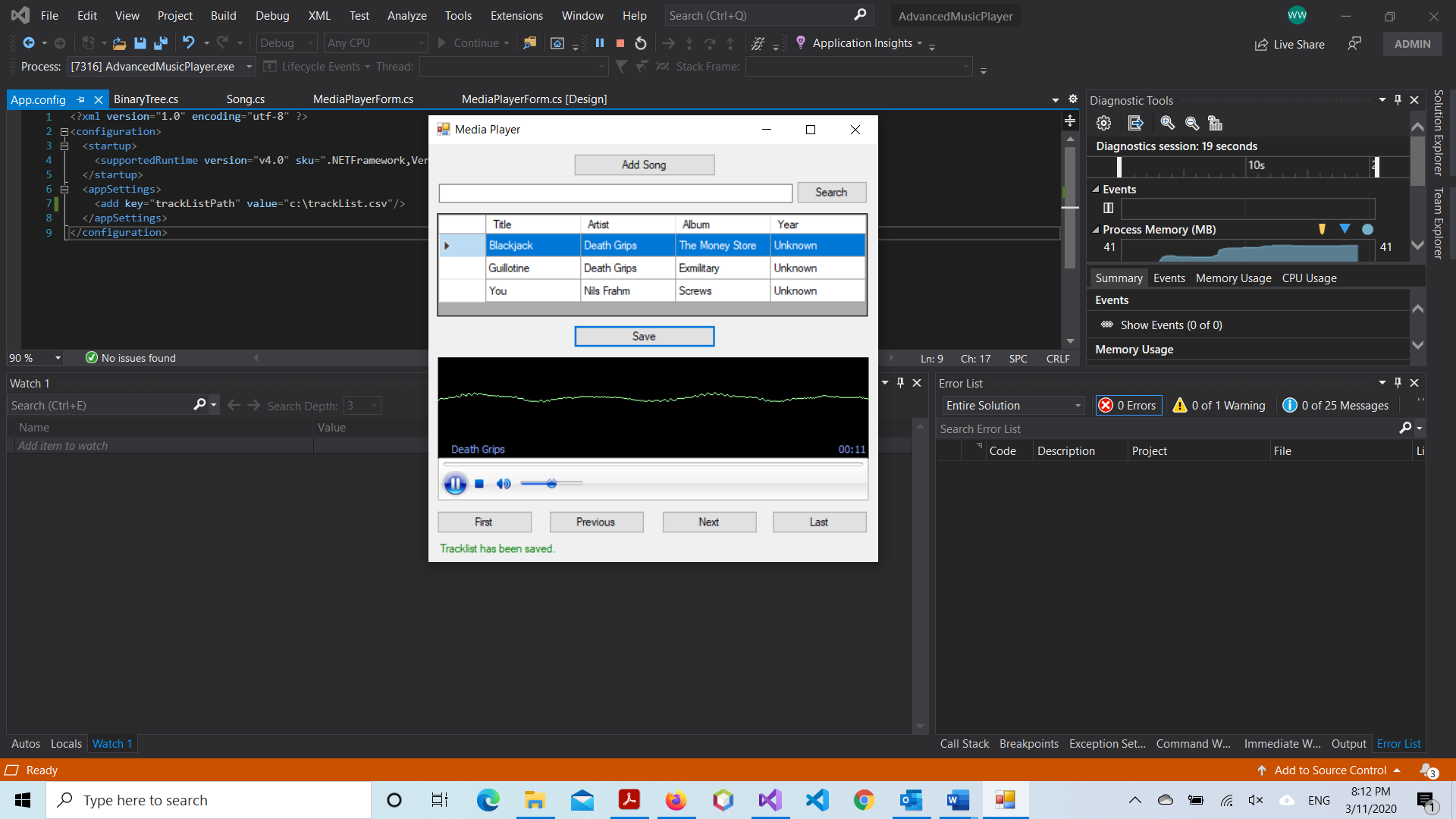
### Figure 1



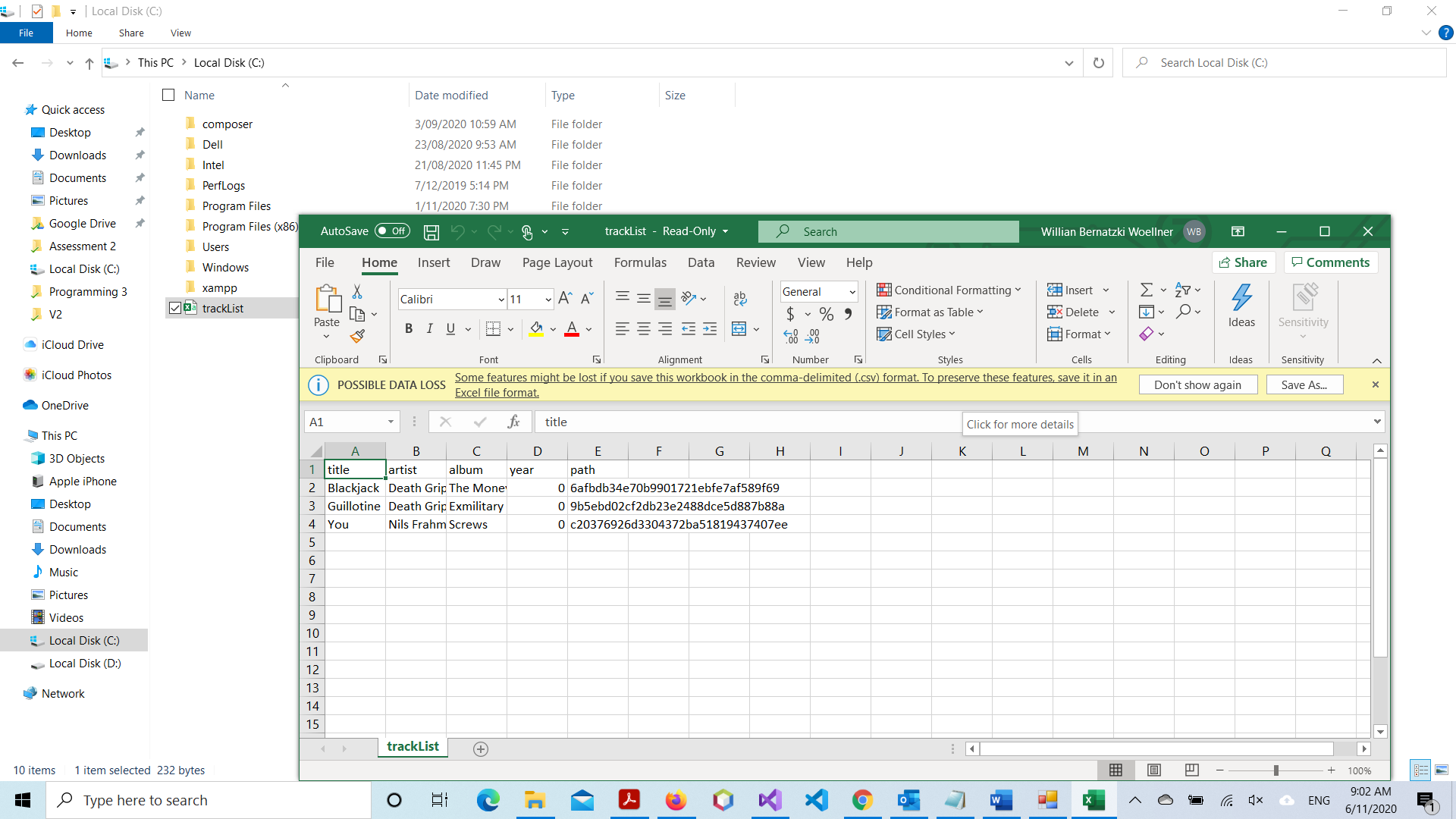
### Figure 2



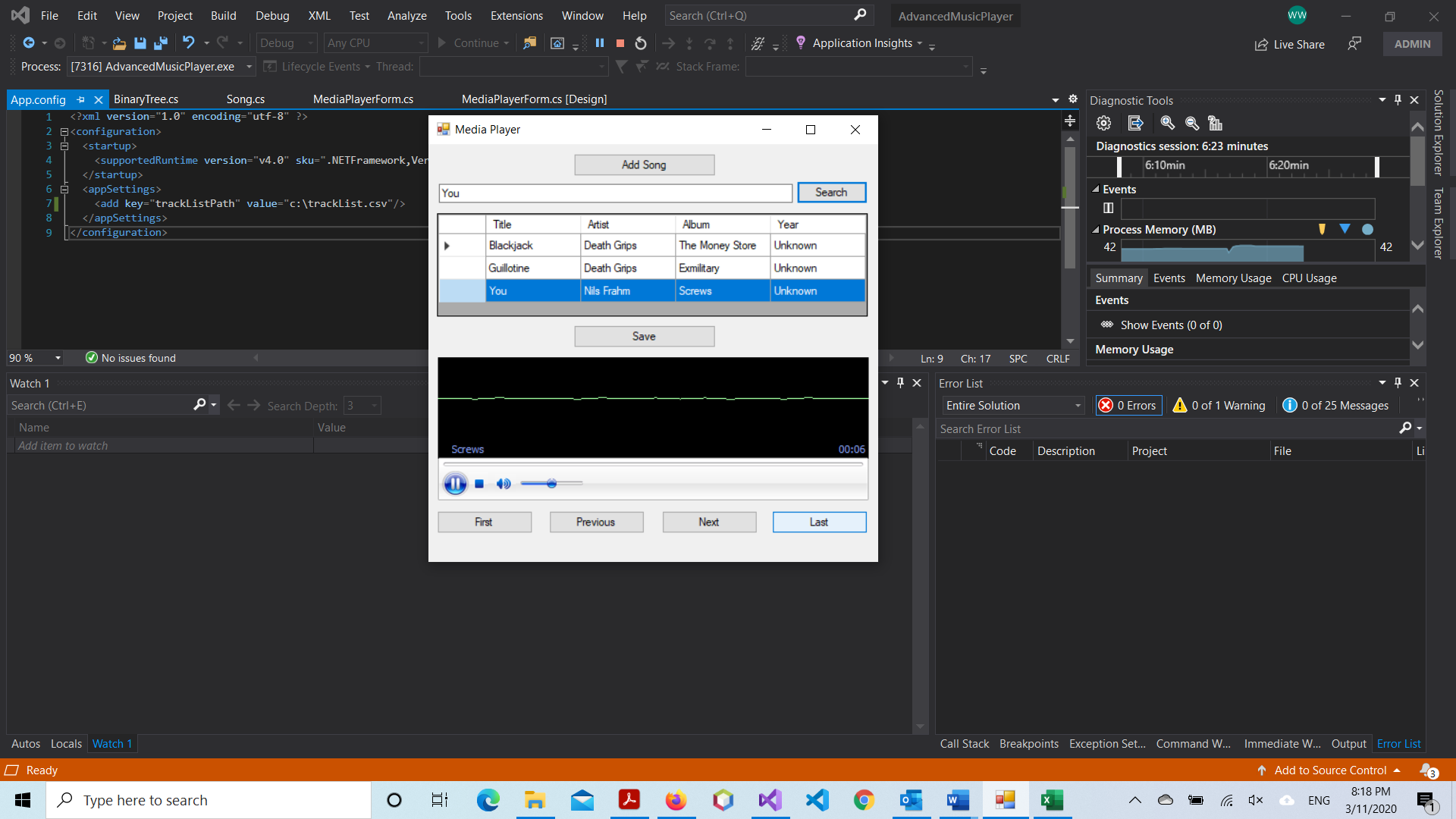
### Figure 3



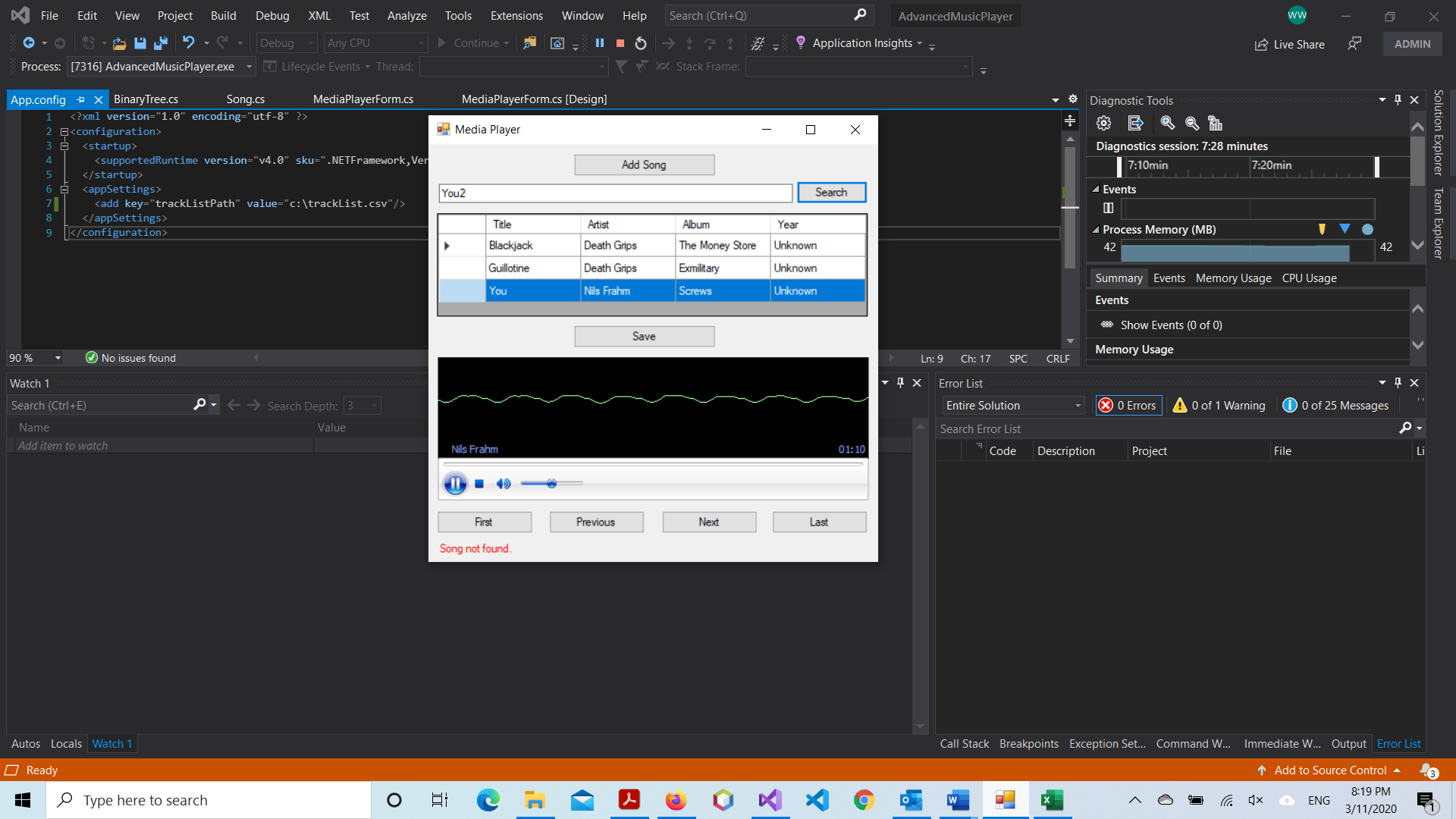
### Figure 4



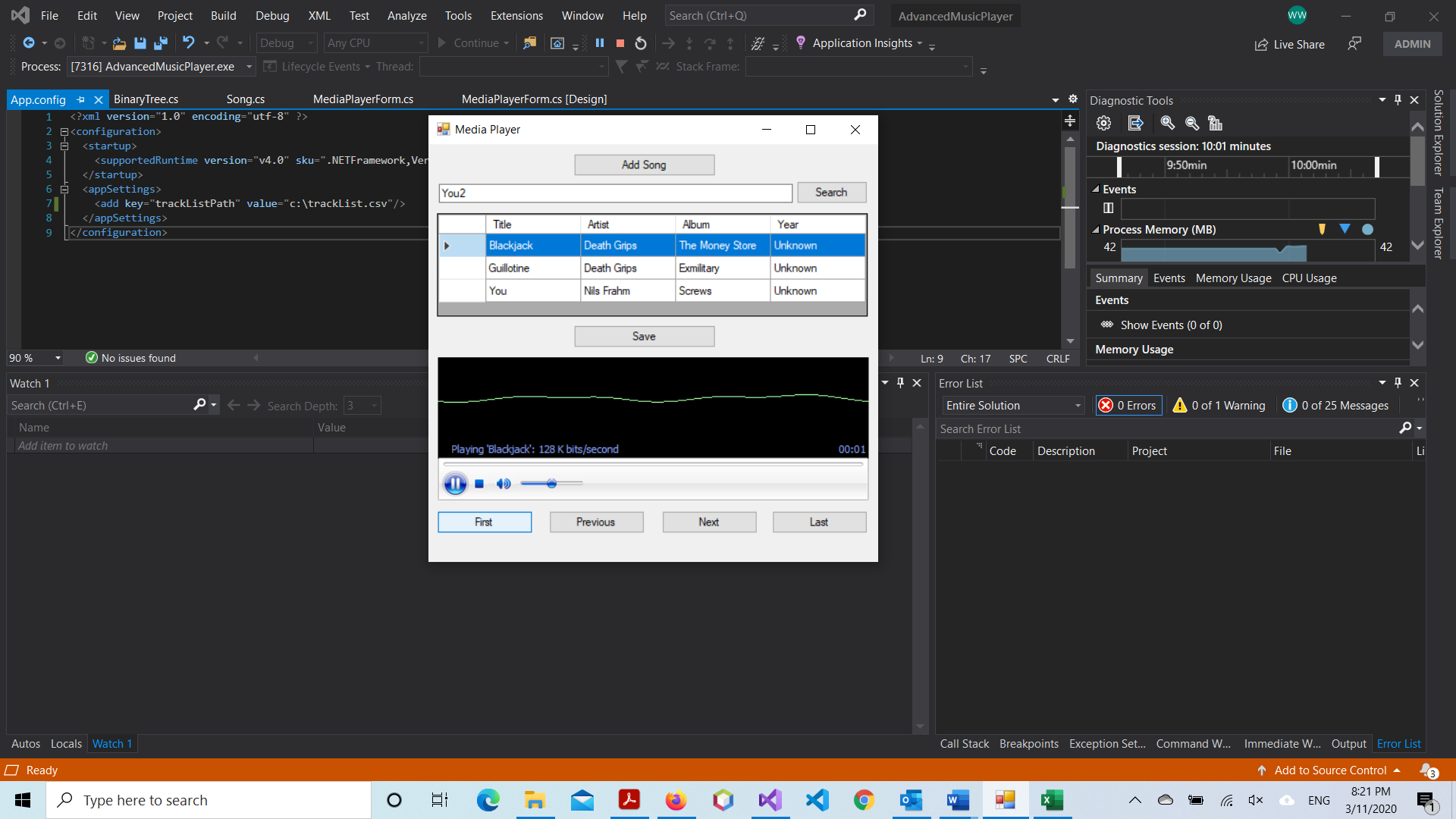
### Figure 5



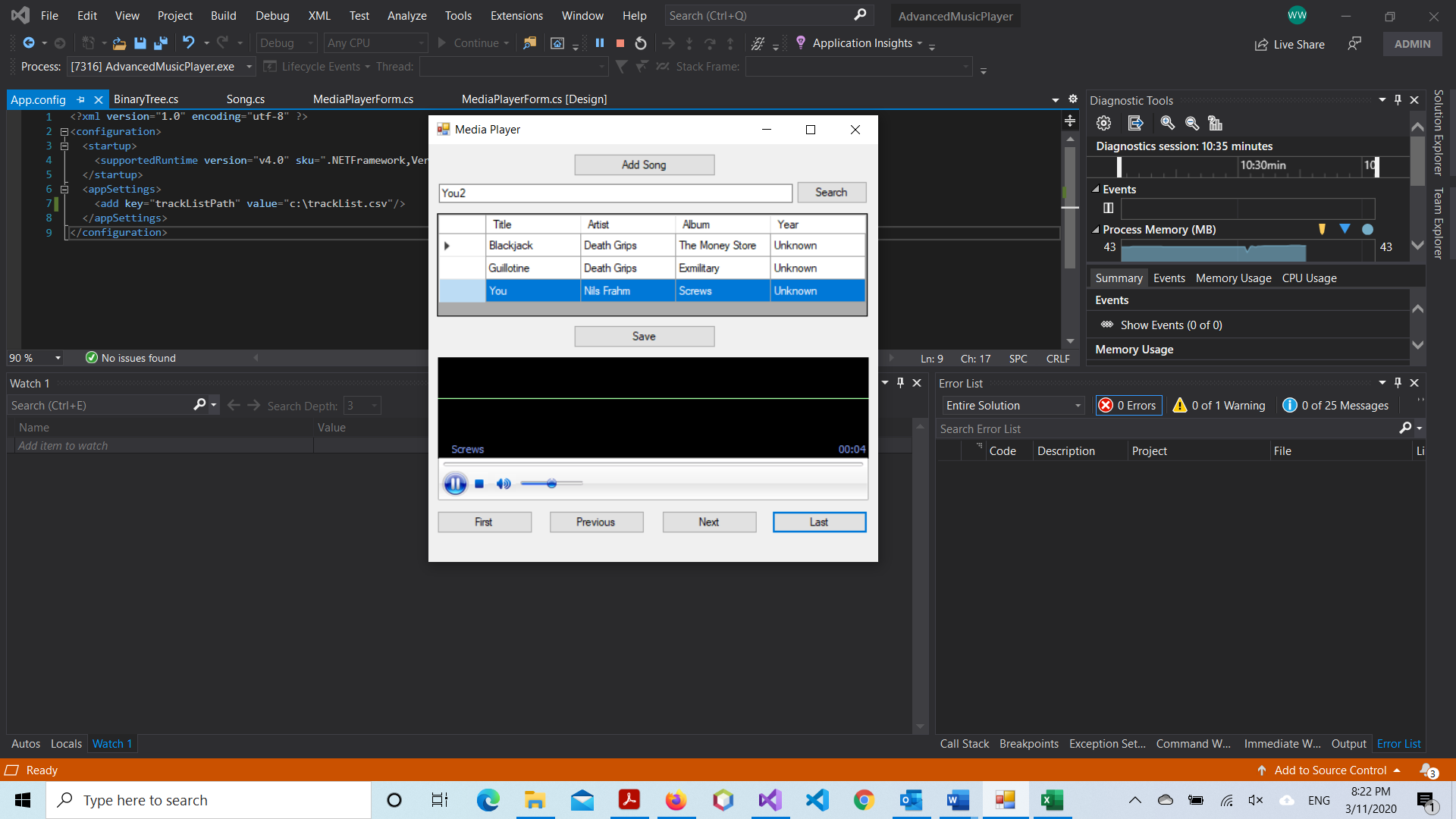
### Figure 6



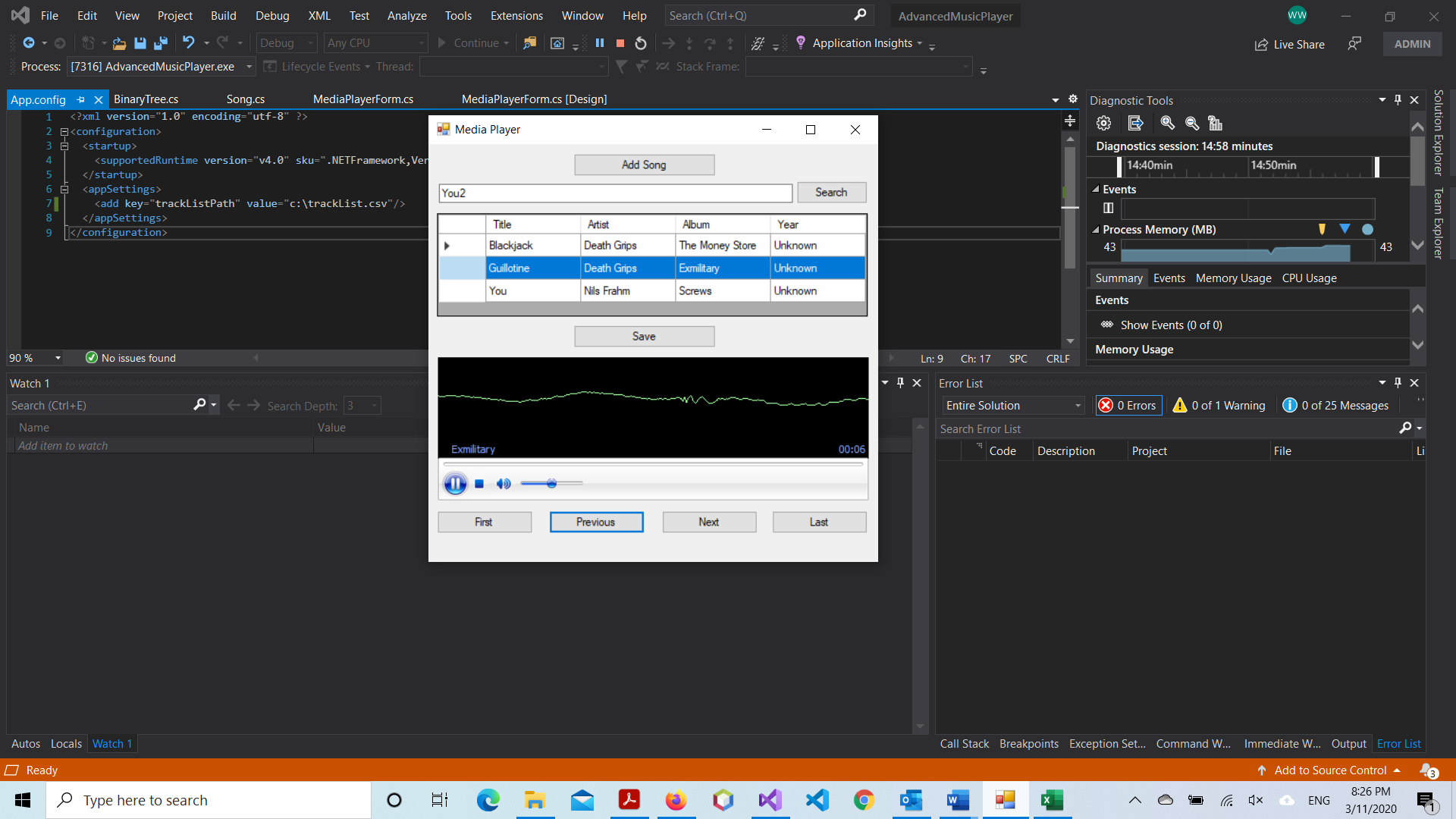
### Figure 7



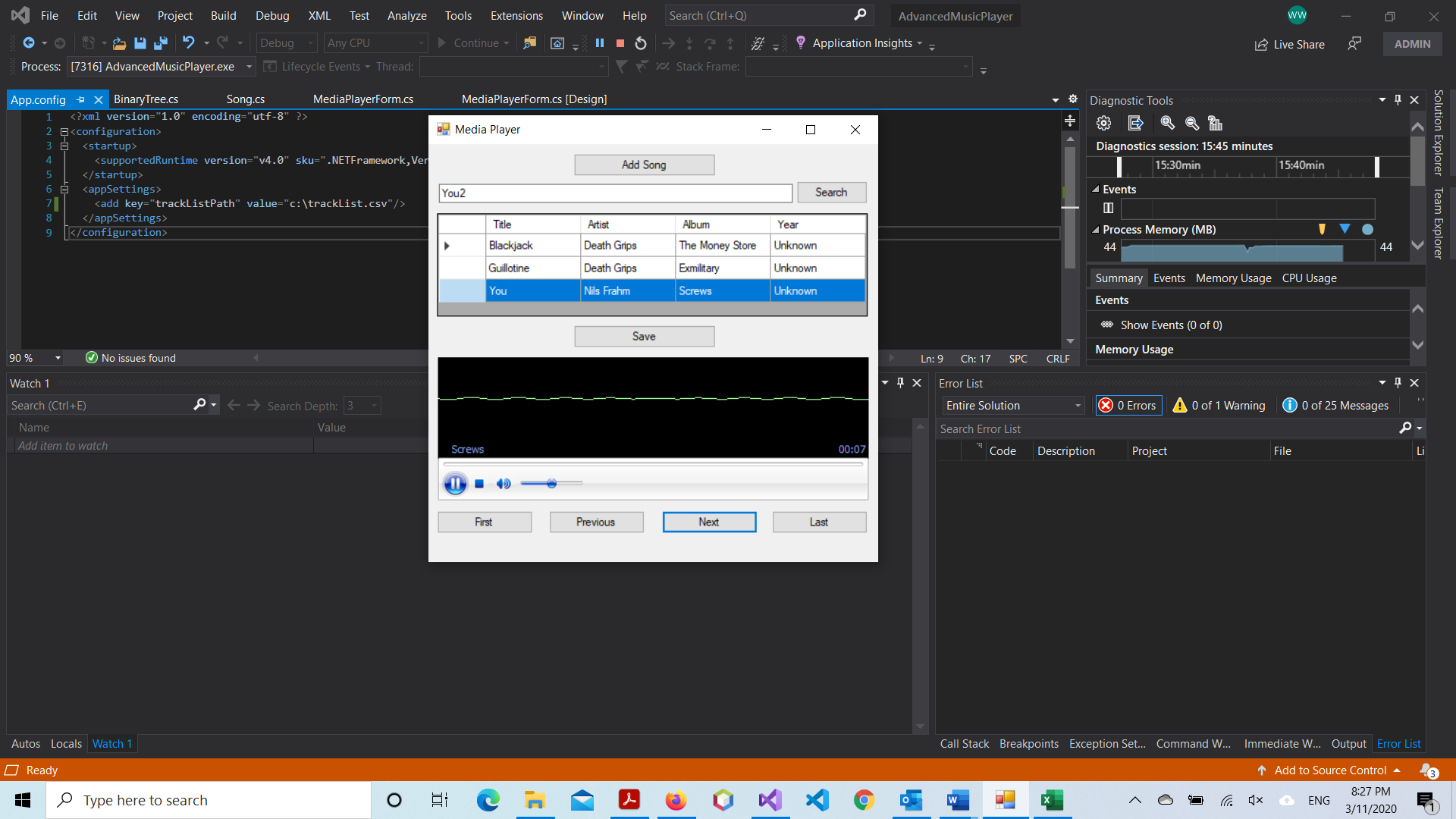
### Figure 8



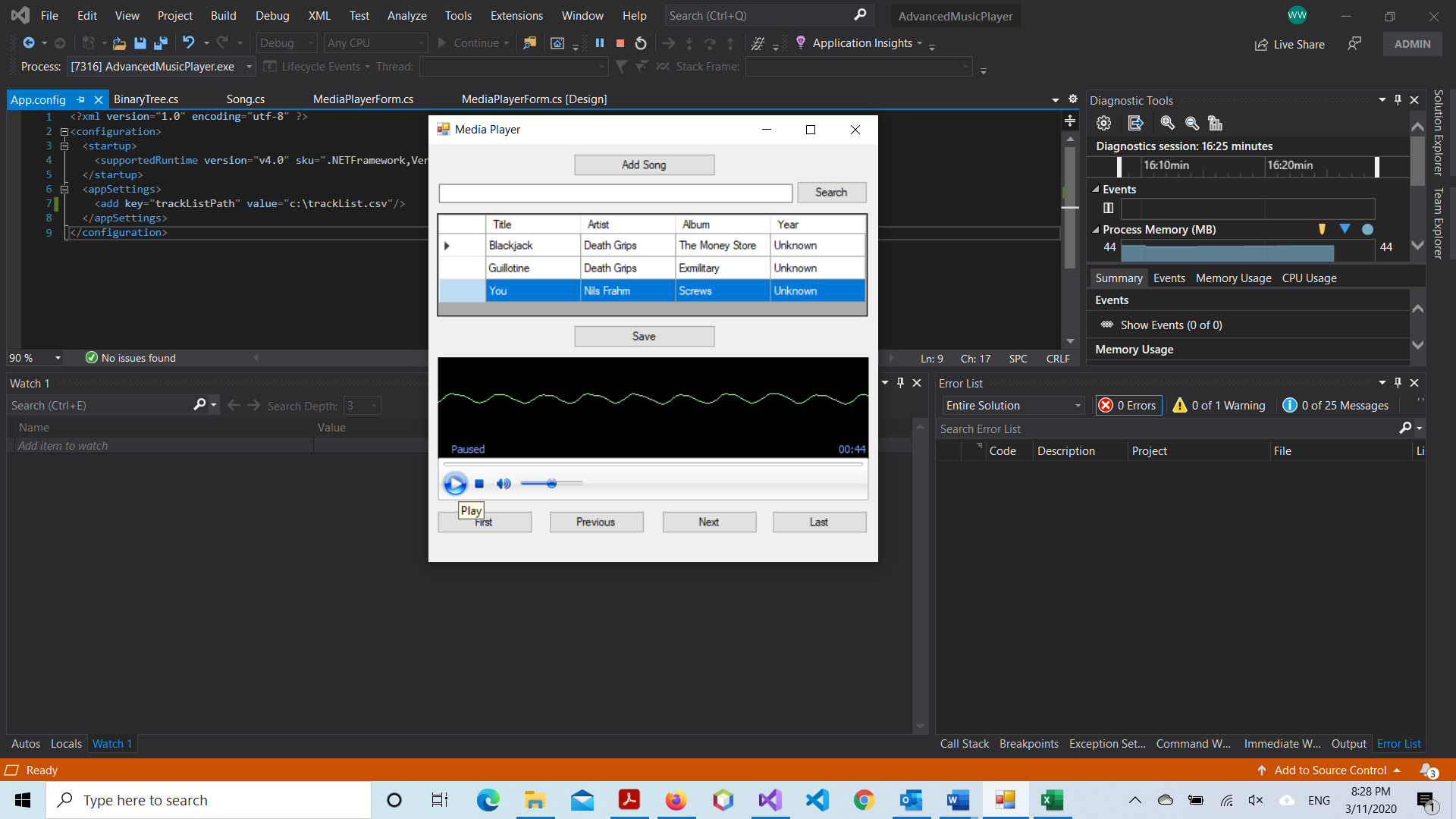
### Figure 9



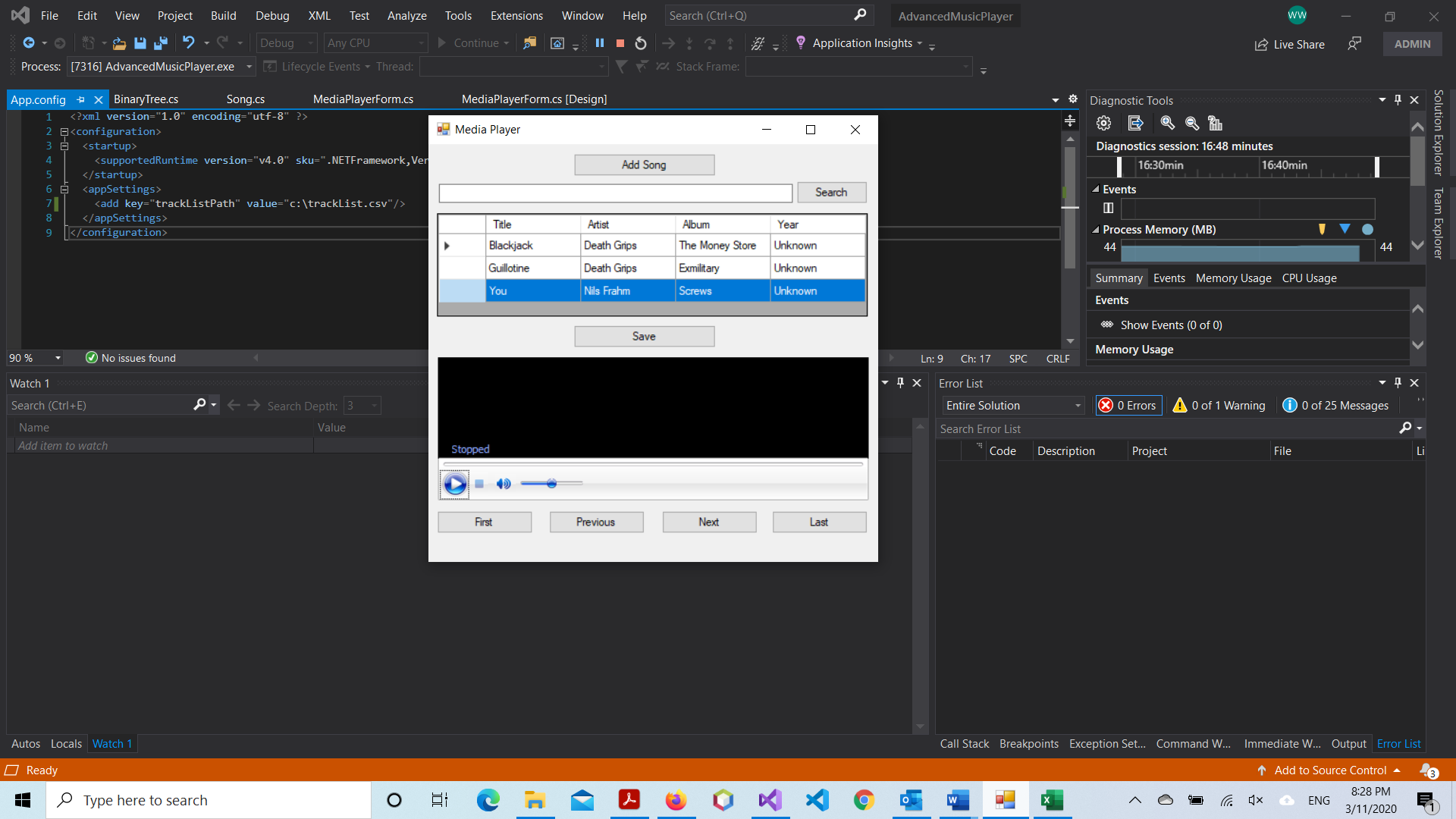
### Figure 10



### Figure 11



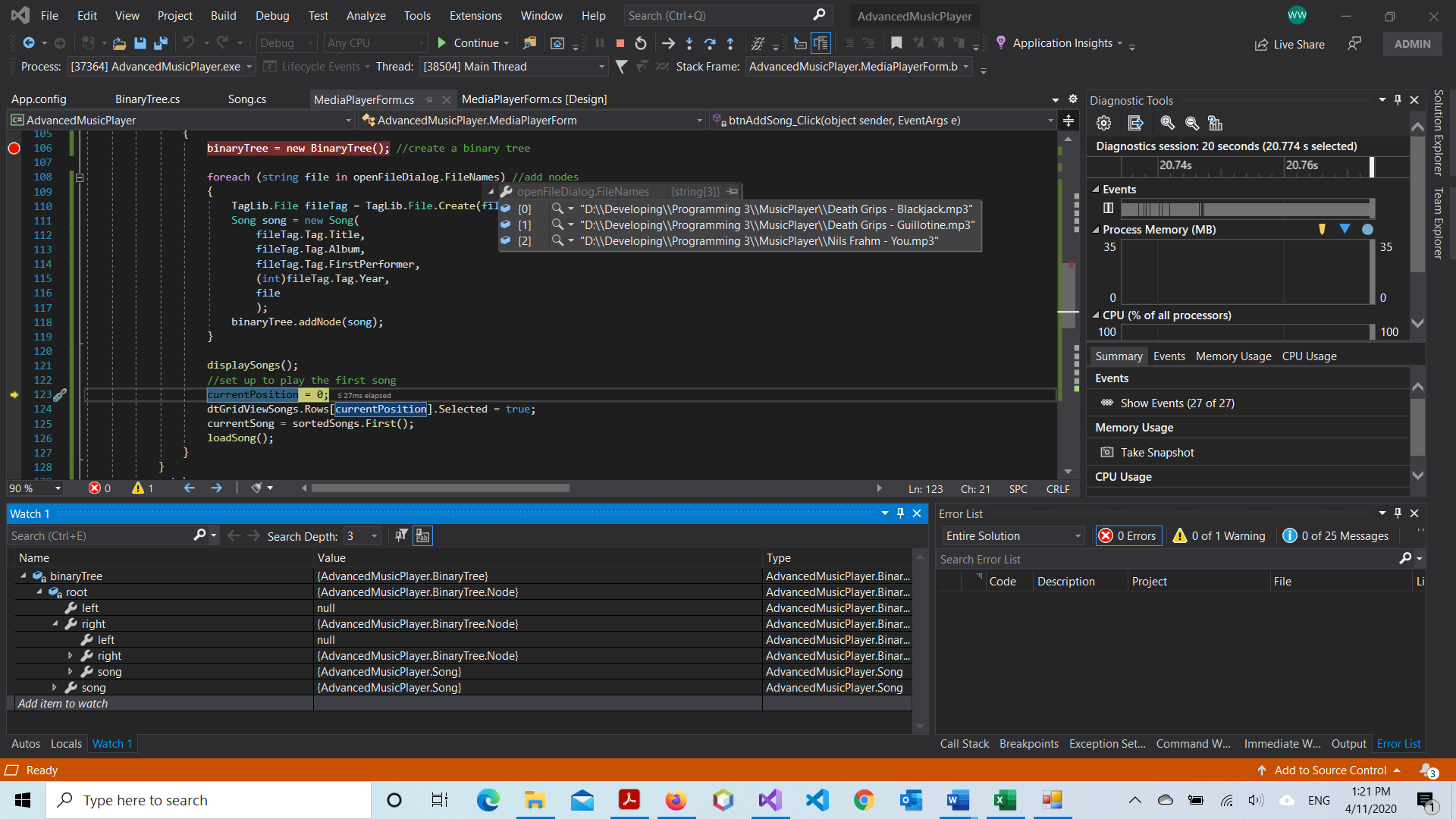
### Figure 12



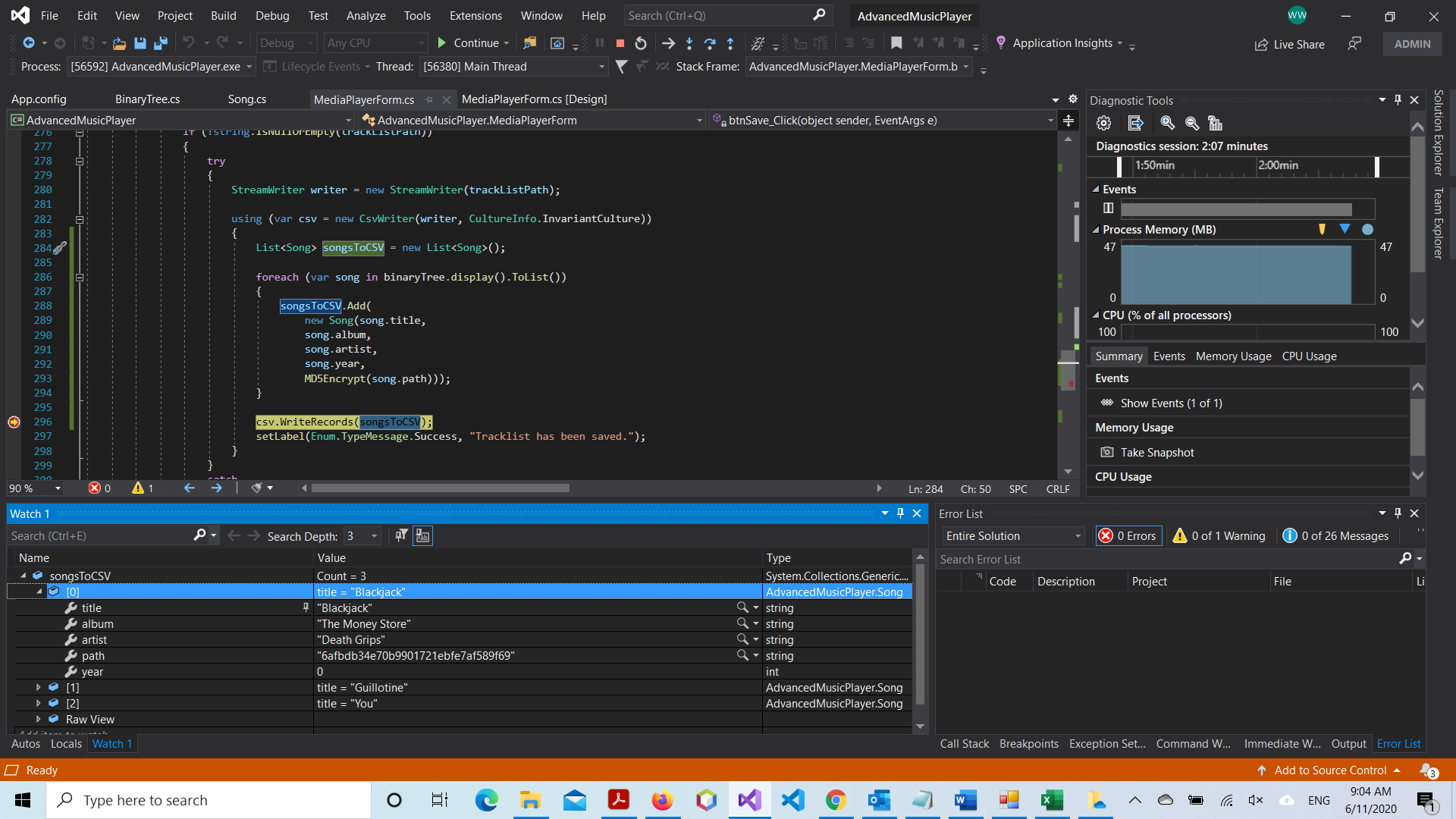
## White Box Testing Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Actual | Comment |
| Add Songs to play | All songs must be added in a binary tree and must be displayed sorted by title in a table. | Results as expected. | Figure 1 |
| Save the tracklist on a CSV file | The tracklist must be saved in a CSV file using a 3 party library. | Results as expected. | Figure 2 |
| Search a Song | The Song must be found on the binary tree. | Results as expected. | Figure 3 |
| Sort the Songs when they are added in the Binary Tree | Display all songs sorted by title. | Results as expected. | Figure 4 |

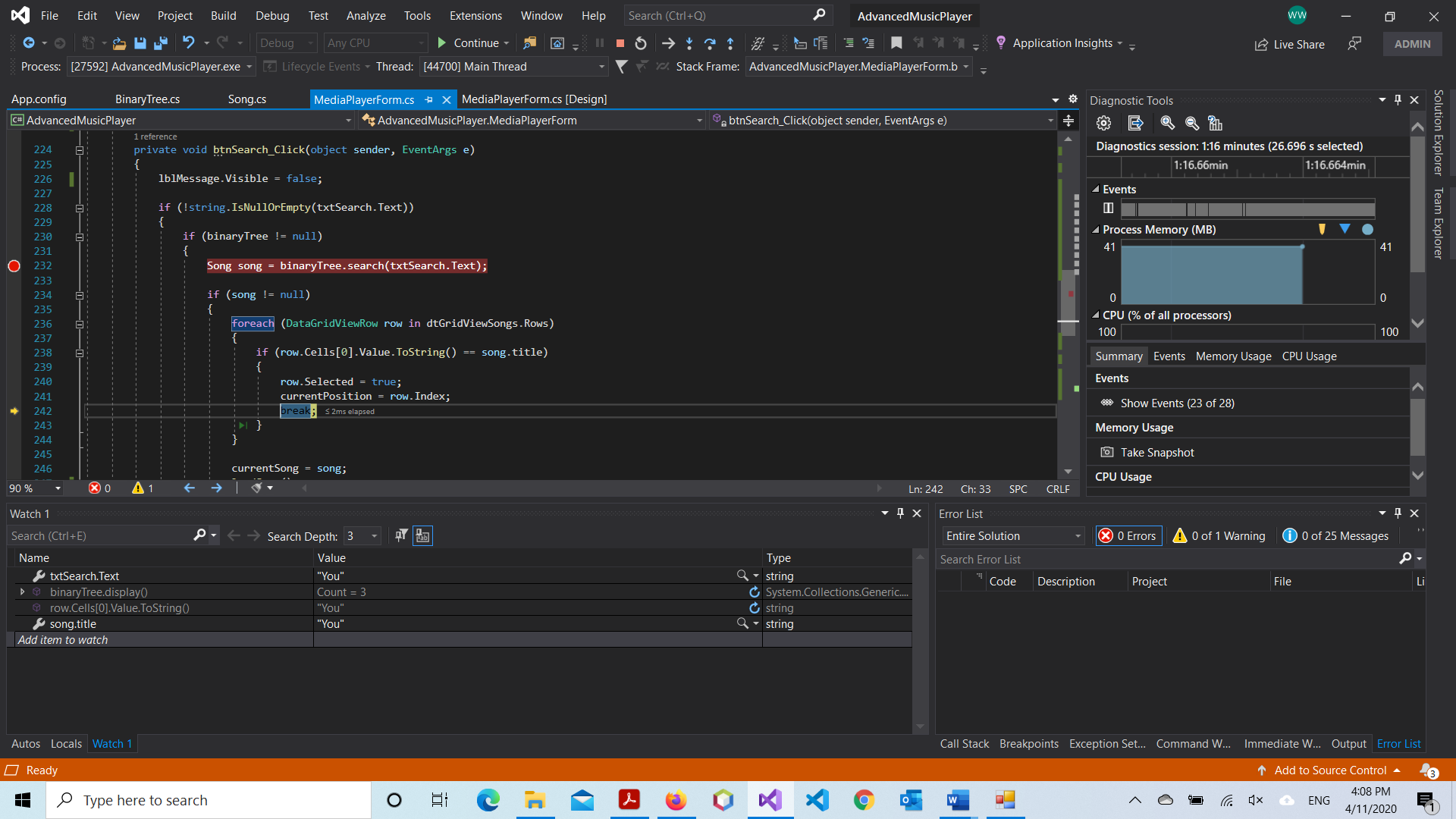
### Figure 1



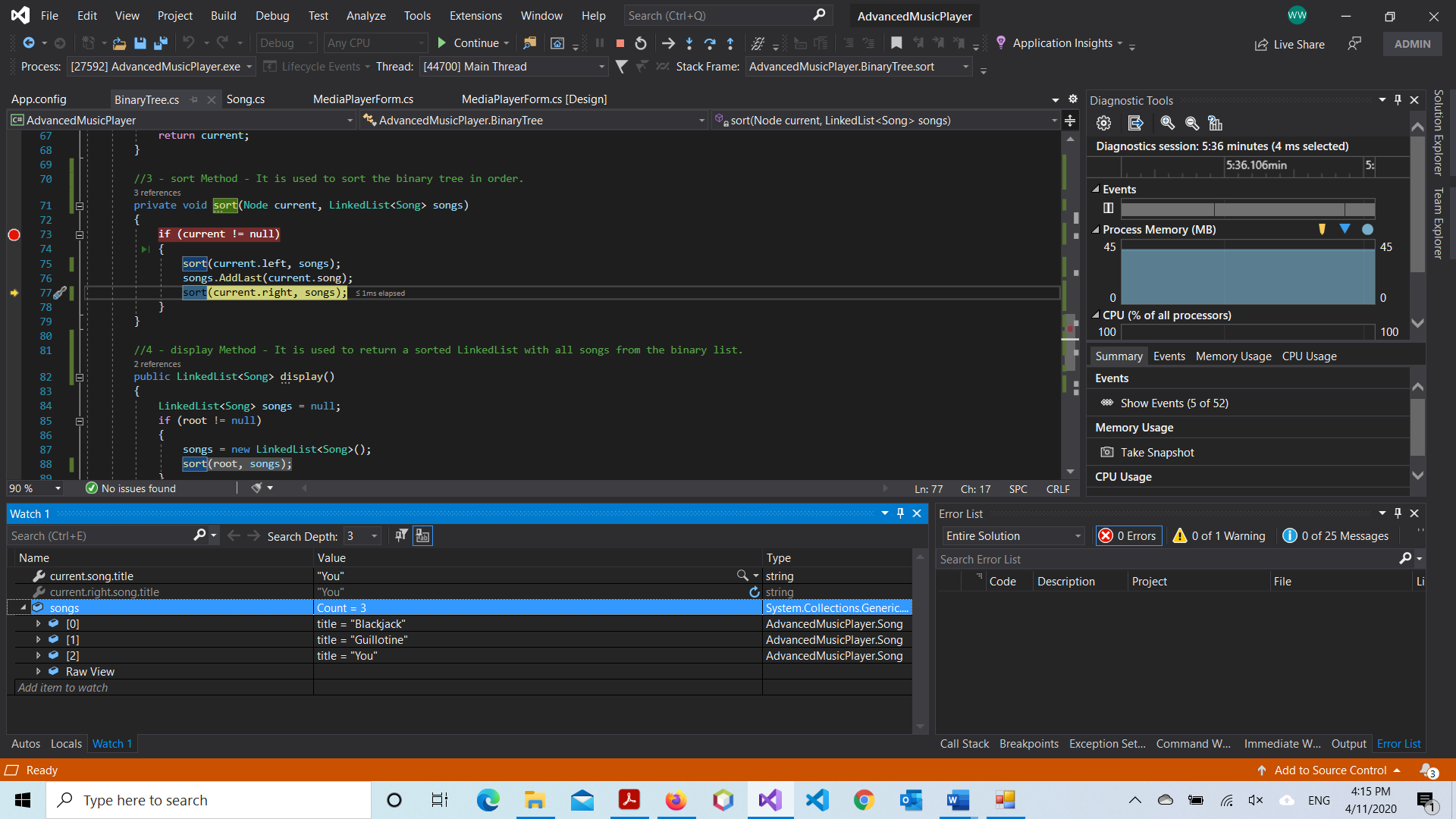
### Figure 2



### Figure 3



### Figure 4



# Product Design Specification Approval

The undersigned acknowledge they have reviewed the Music Player Project Product Design Specification document and agree with the approach it presents. Any changes to this Requirements Definition will be coordinated with and approved by the undersigned or their designated representatives.

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Title: Project Manager

Role: Project Manager

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Title: Business Administrator

Role: Business Administrator

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Title: Technical Administrator

Role: Technical Administrator