Willian Bernatzki Woellner

Student ID 30021175

Java 3

Product Specification Design and Test Documentation

13/11/2020

Version 1.0

Table of Contents

[Introduction 3](#_Toc56153993)

[Purpose 3](#_Toc56153994)

[Requirements 3](#_Toc56153995)

[Functional Requirements 3](#_Toc56153996)

[Non-Functional Requirements 4](#_Toc56153997)

[Software Quality Attributes 4](#_Toc56153998)

[Software Requirements 4](#_Toc56153999)

[Hardware Requirements 4](#_Toc56154000)

[Agile Methodology 5](#_Toc56154001)

[System Architecture 6](#_Toc56154002)

[Layer Pattern 6](#_Toc56154003)

[Design and Implementation 7](#_Toc56154004)

[Class Diagram 7](#_Toc56154005)

[Use Case Diagram 8](#_Toc56154006)

[Graphical Interface Design 9](#_Toc56154007)

[Test Plan 11](#_Toc56154008)

[Unit Test 11](#_Toc56154009)

[Black Box Test 15](#_Toc56154010)

[Product Design Specification Approval 22](#_Toc56154011)

[References 23](#_Toc56154012)

Version History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Implemented By | Revision Date | Approved By | Approval Date | Reason |
| 1.0 | Willian Bernatzki Woellner | 23/10/2020 |  |  | Initial Design Definition draft |
| 2.0 | Willian Bernatzki Woellner | 13/11/2020 |  |  | Test Documentation |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Introduction

The 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. The 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. To develop the sort function will be used the Merge Sort Algorithm.
* 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. To develop the search function will be used Binary Search Algorithm.

## 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

* Java JDK (back-end)
* Java FX (front-end)
* Java FX Media API to play songs
* Java JRE to run the application
* Operating System: Windows 10
* Java OpenCSV API to save the tracklist on CSV file
* Microsoft Excel or other to open the CSV file

## Hardware Requirements

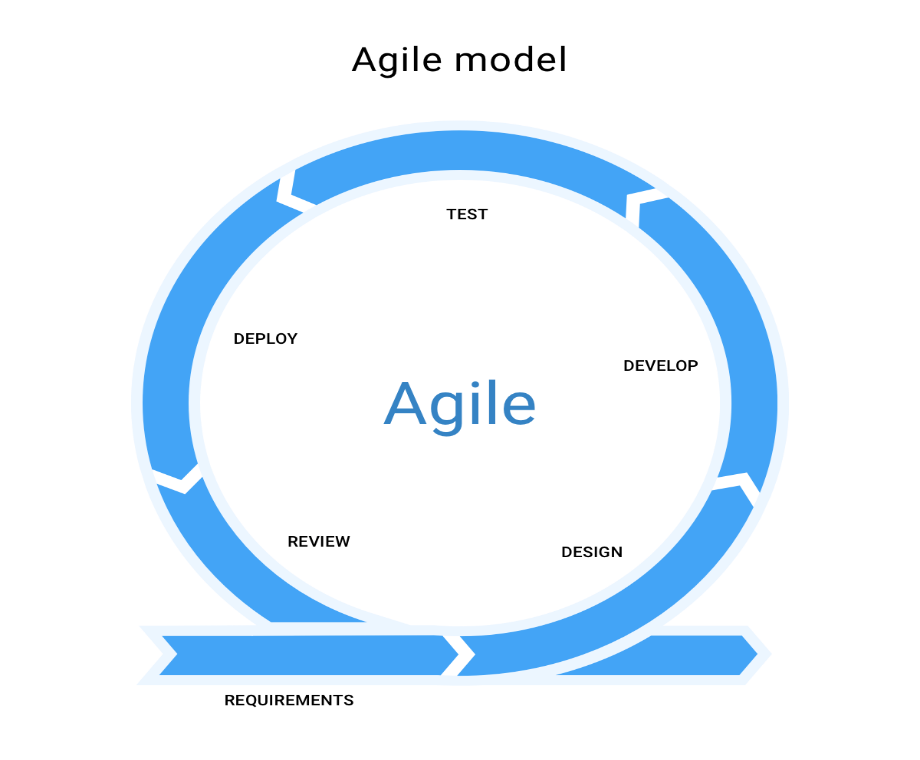
These requirements are to use the solution on the Windows 10 operating system:

* 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. (Guru 99, 2020)

All project activities are carried out using the Agile Methodology phases.



# System Architecture

## Layer Pattern

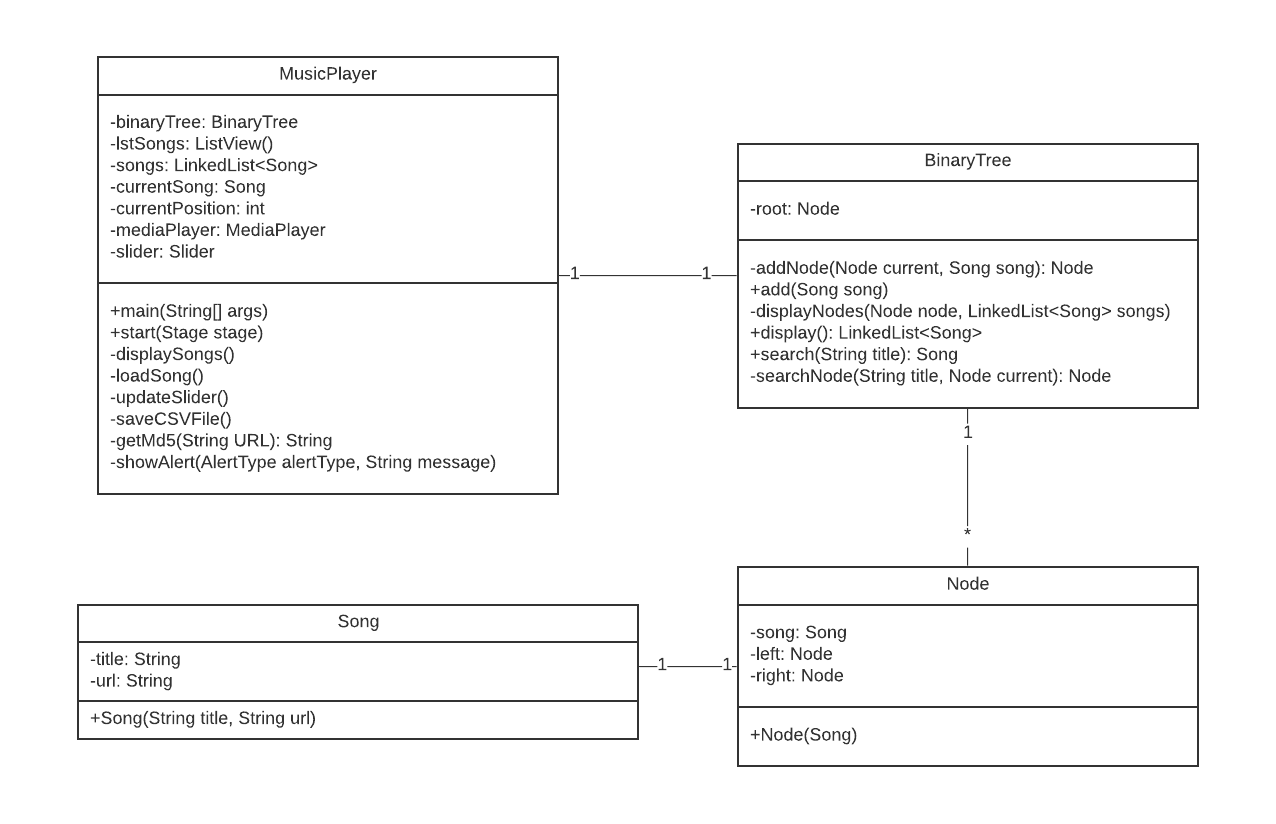
The Layer pattern can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer. (Mallawaarachchi, 2017)

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

The Media Player solution will use this concept with some alterations, The Data Access layer and the Business layer will be joining on a Binary Tree class.

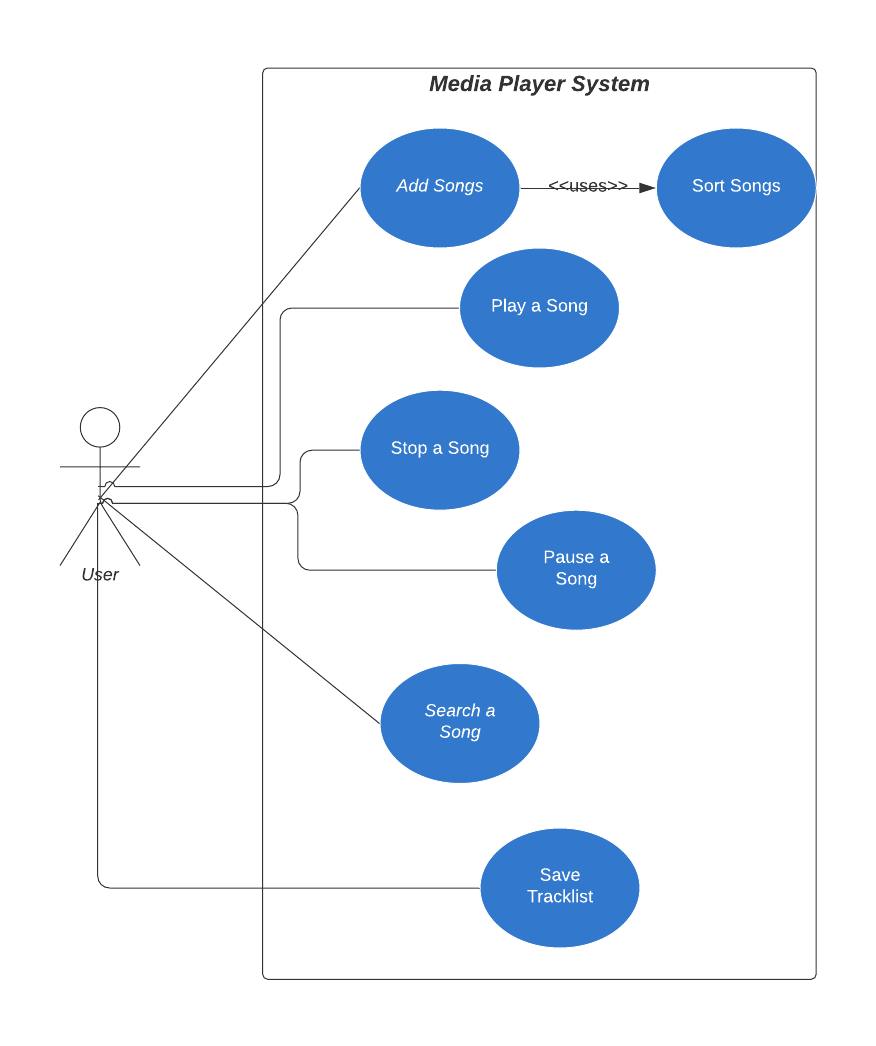
# Design and Implementation

## Class Diagram



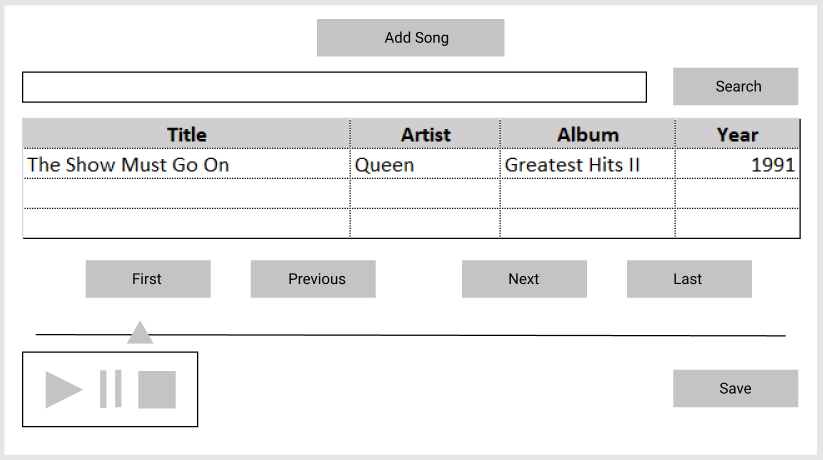
* Song: This class defines a song object with title and URL.
* Node: This class defines a node object to a Binary Tree.
* Binary Tree: This class contains methods to manage the nodes.
* Music Player: This is a Main class of the Form, it contains methods of the Interface.

## 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.

## Graphical Interface Design



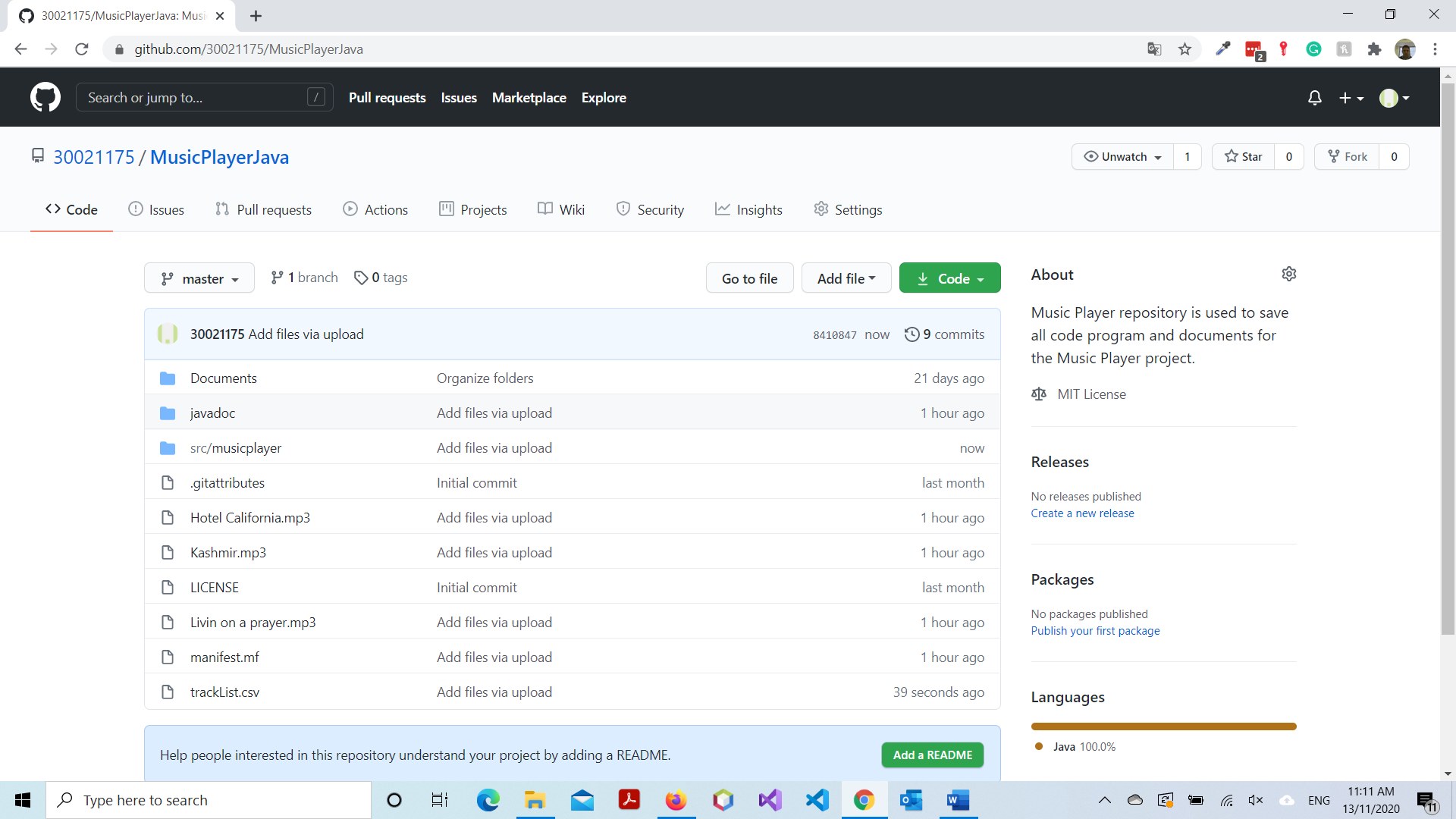
* The Search will be done by the song title.
* All songs added will be displayed on the table.
* All songs will be displayed sorted by title.
* The Save button will be used to save the tracklist on the CSV file.

Version Control Repository

The Media Player project will be used GitHub to control the versions of the system.

* Src Folder has the system code.
* Documents: All project documents, for example, Product Specification Design.

It is available on <https://github.com/30021175/MusicPlayerJava>



## Test Plan

The Test Plan for the Media Player project consists of two phases:

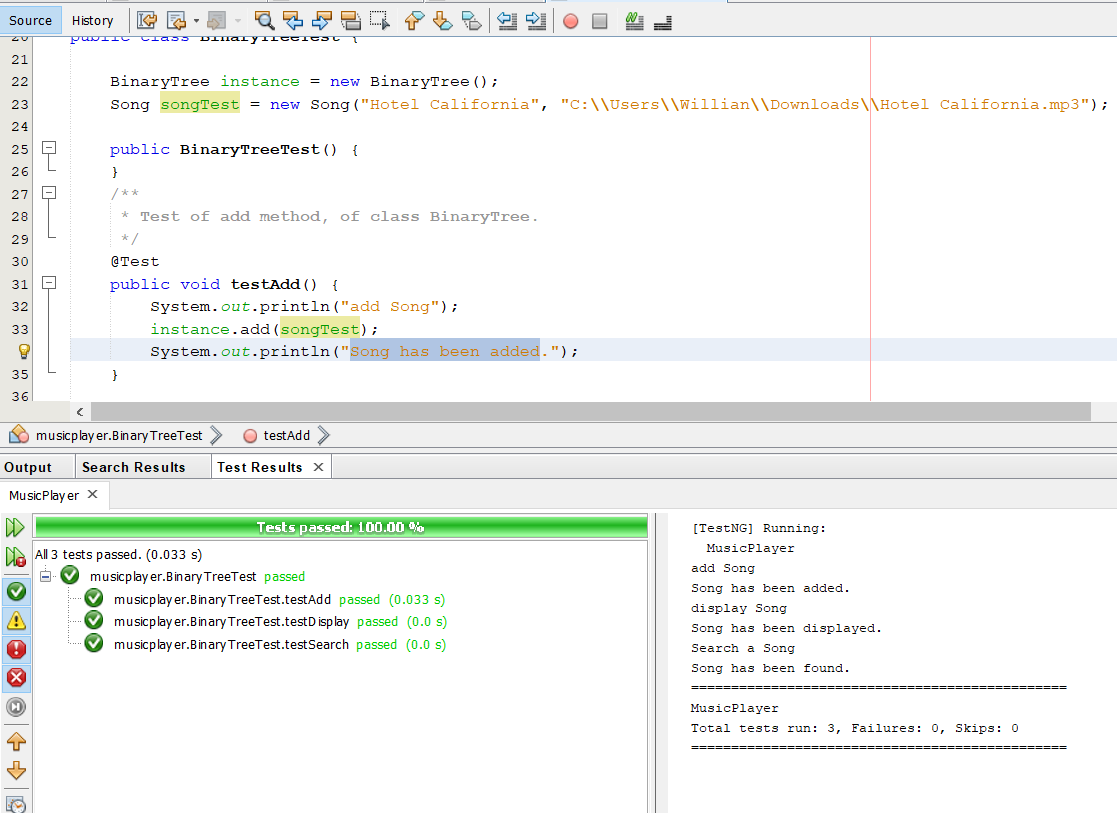
* Unit Tests to test the main functions of the code.
* Black Box Tests to test all functions.

### Unit Test

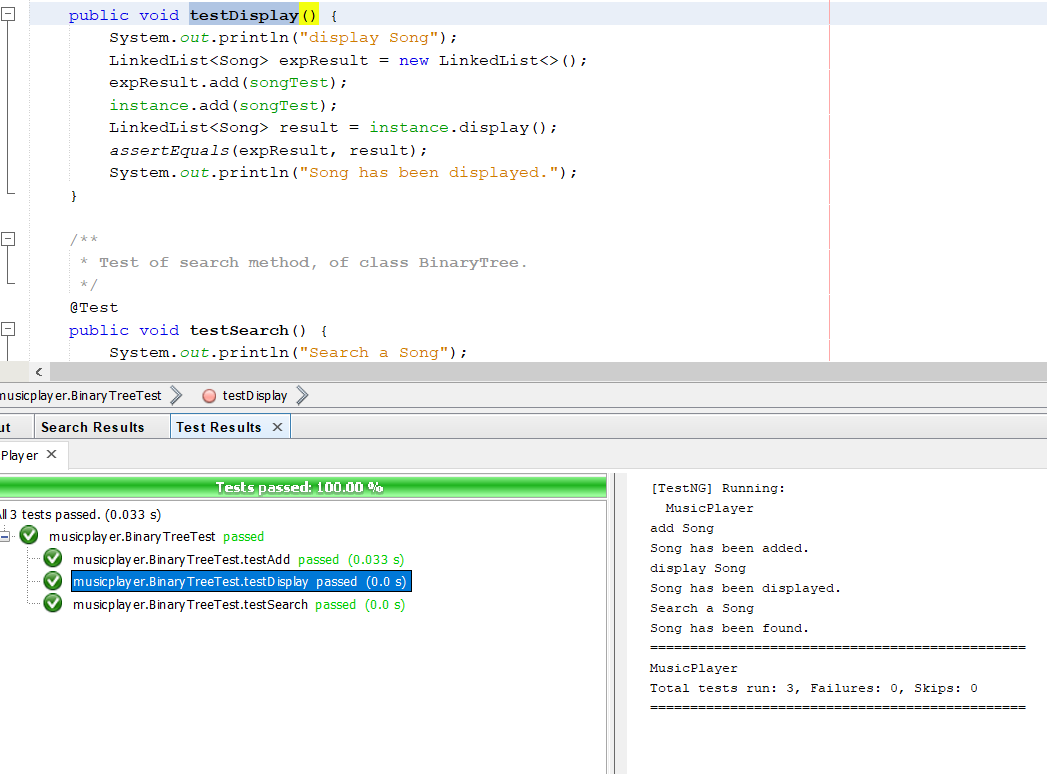
JUnit Framework has been used to automate the Unit Tests. It is an open-source framework hosted at GitHub. All Unit Tests are listed below:

|  |  |  |  |
| --- | --- | --- | --- |
| Unit Test | Expected | Result | Comment |
| testAdd function | Add a new Song on the Binary Tree. | As expected. | Ref. Figure 1 |
| testDisplay function(sort and display) | Display a Song from the Binary Tree. This test also tests the sort function, because the display function is used the algorithm to sort the songs to display. | As expected. | Ref. Figure 2 |
| testSearch function | Search a Song on the Binary Tree. | As expected. | Ref. Figure 3 |

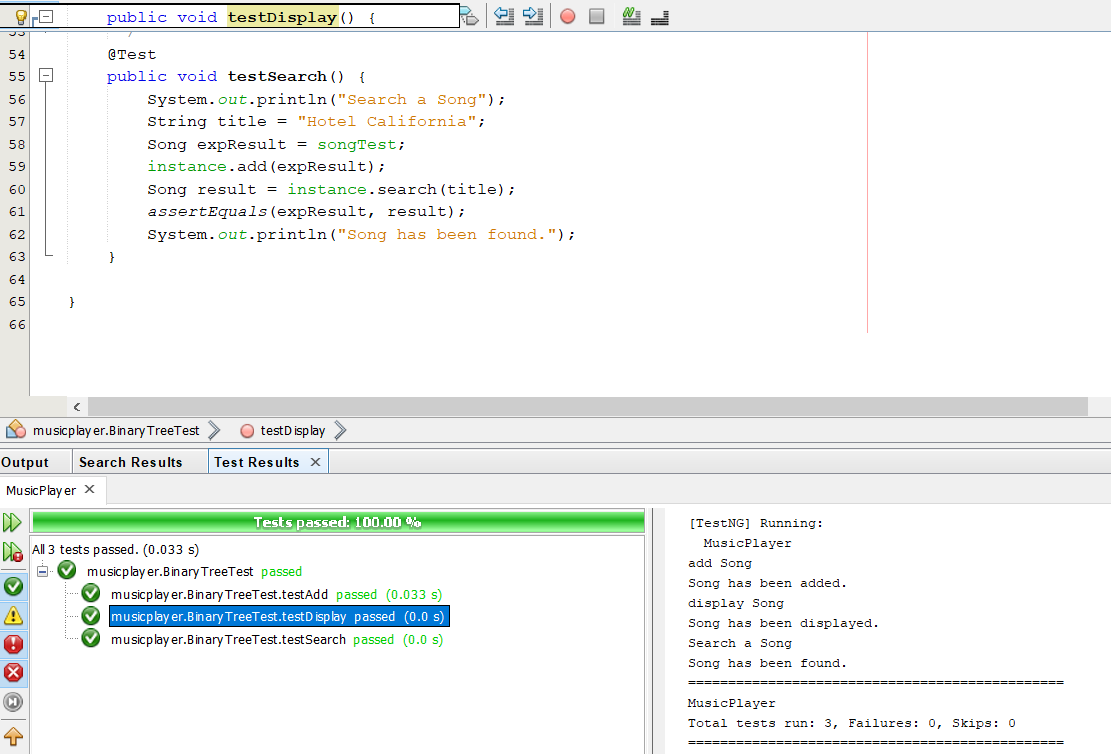
#### Figure 1



#### Figure 2



#### Figure 3

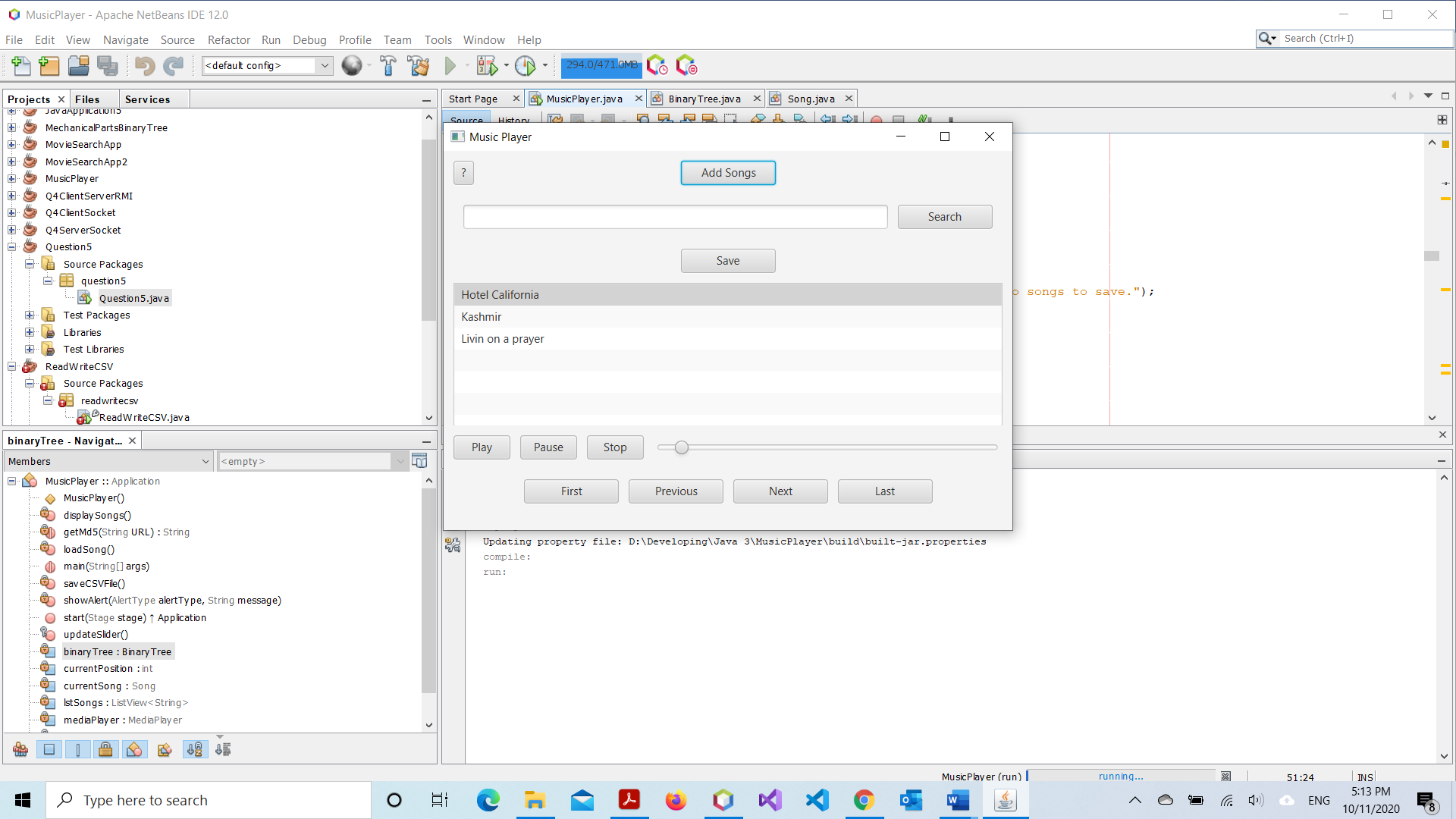


### Black Box Test

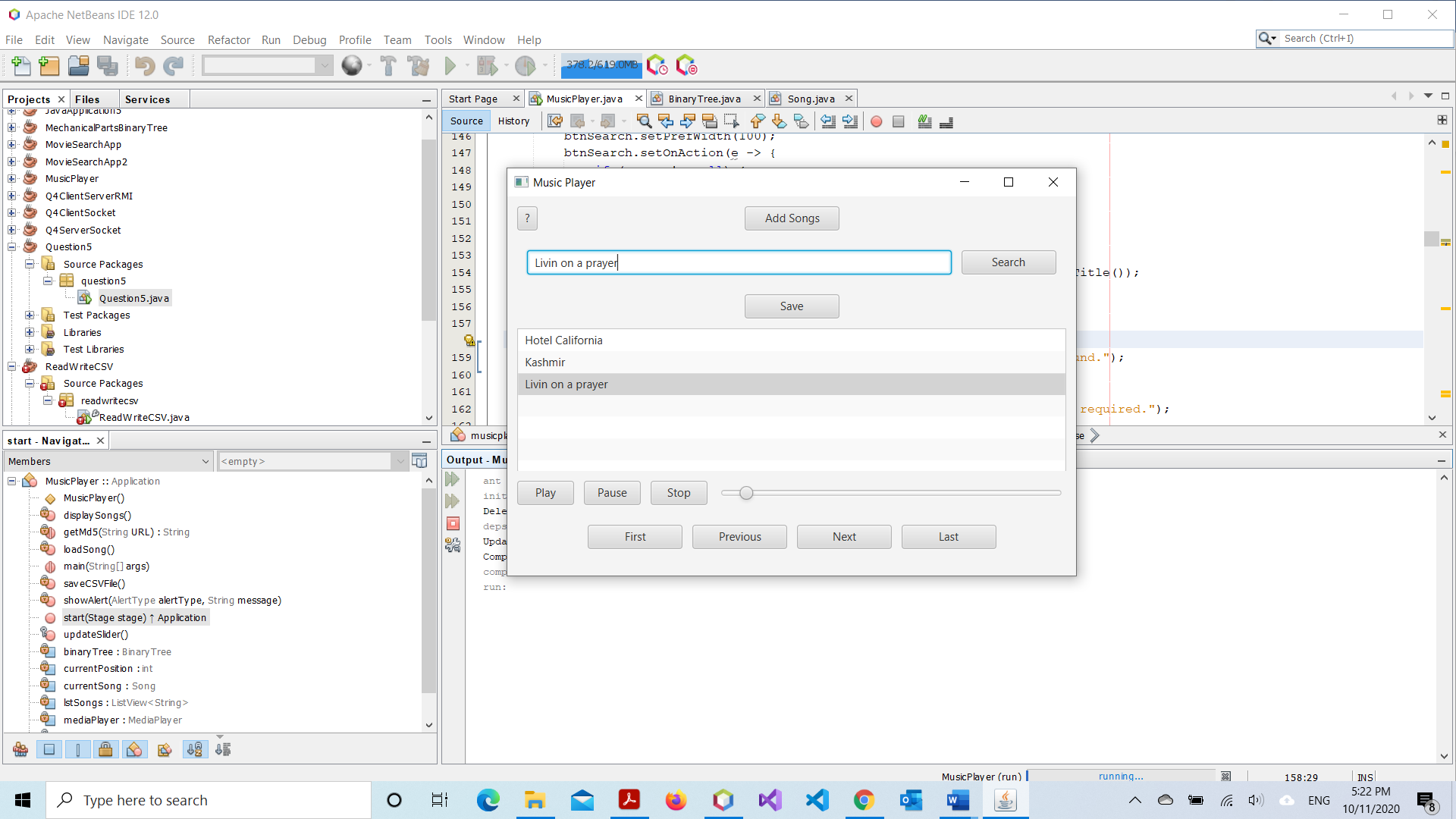
#### Test Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Actual | Comment |
| Add Songs to play | When the songs are added to the tracklist, the first song should start playing. | As expected. | Ref. Figure 1 |
| Sort the songs by title | When the songs are added to the tracklist, the first song sorted by title should start playing. | As expected. | Ref. Figure 1 |
| Search a song by title to play | The user should enter the song title and clicks on the Search button for the song to start playing. | As expected. | Ref. Figure 2 |
| Play the next song | When the user clicks on the Next button the next song on the tracklist should start playing. | As expected. | Ref. Figure 3 |
| Play the previous song | When the user clicks on the Previous button the previous song on the tracklist should start playing. | As expected. | Ref. Figure 4 |
| Play the last song | When the user clicks on the Last button the last song on the tracklist should start playing. | As expected. | Ref. Figure 5 |
| Play the first song | When the user clicks on the First button the first song on the tracklist should start playing. | As expected. | Ref. Figure 6 |
| Pause a song | When the user clicks on the Pause button the song that was playing should pause playing. | As expected. | Ref. Figure 7 |
| Stop a song | When the user clicks on the Stop button the song that was playing should stop playing. | As expected. | Ref. Figure 8 |
| Save tracklist on CSV file | The user should click on the Save button to save the tracklist on the CSV file. | As expected. | Ref. Figure 9  Ref. Figure 10 |
| Access the help file | The user should click on the Help button to access the Help file. | As expected. | Ref. Figure 11 |

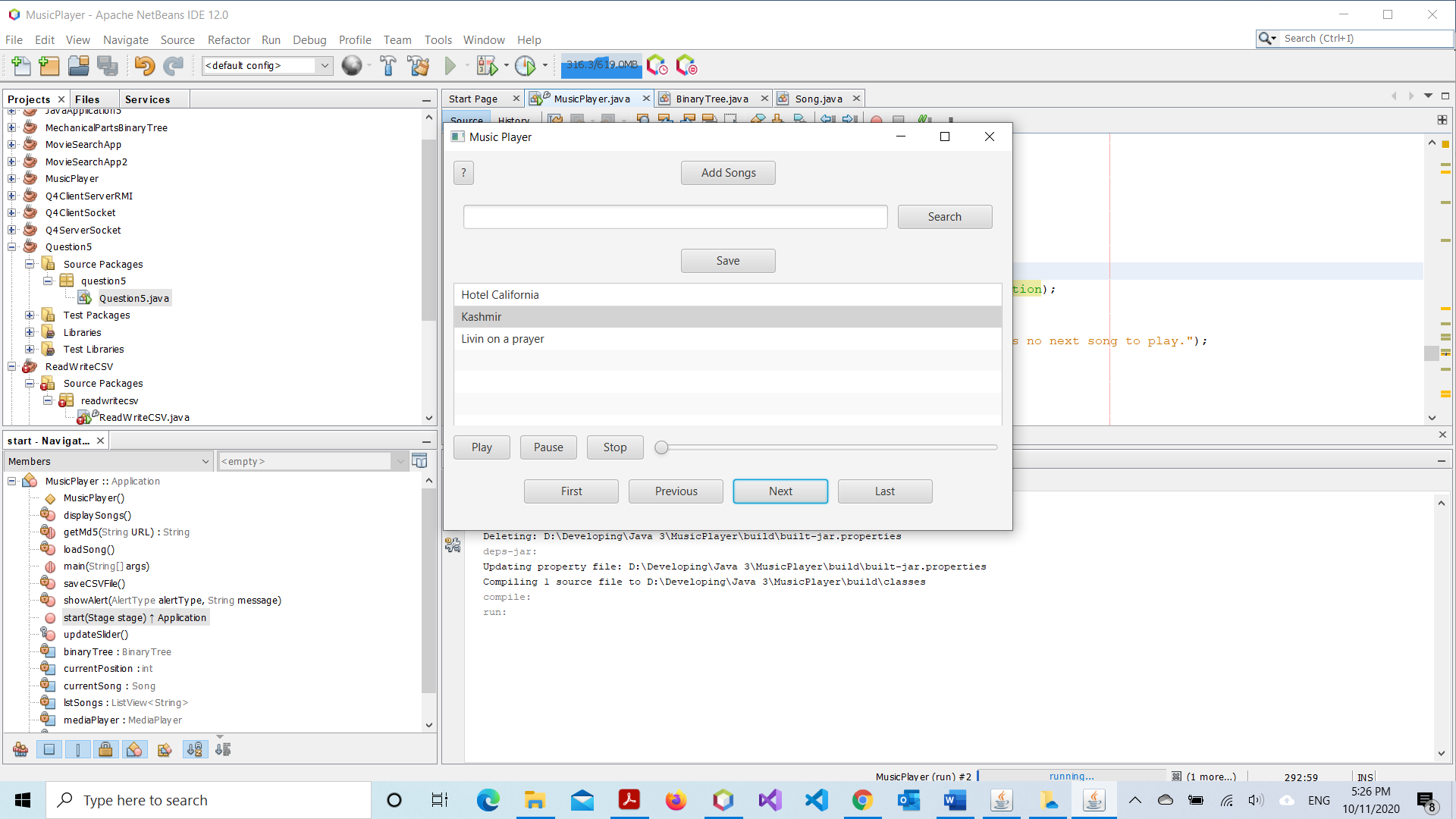
#### Figure 1



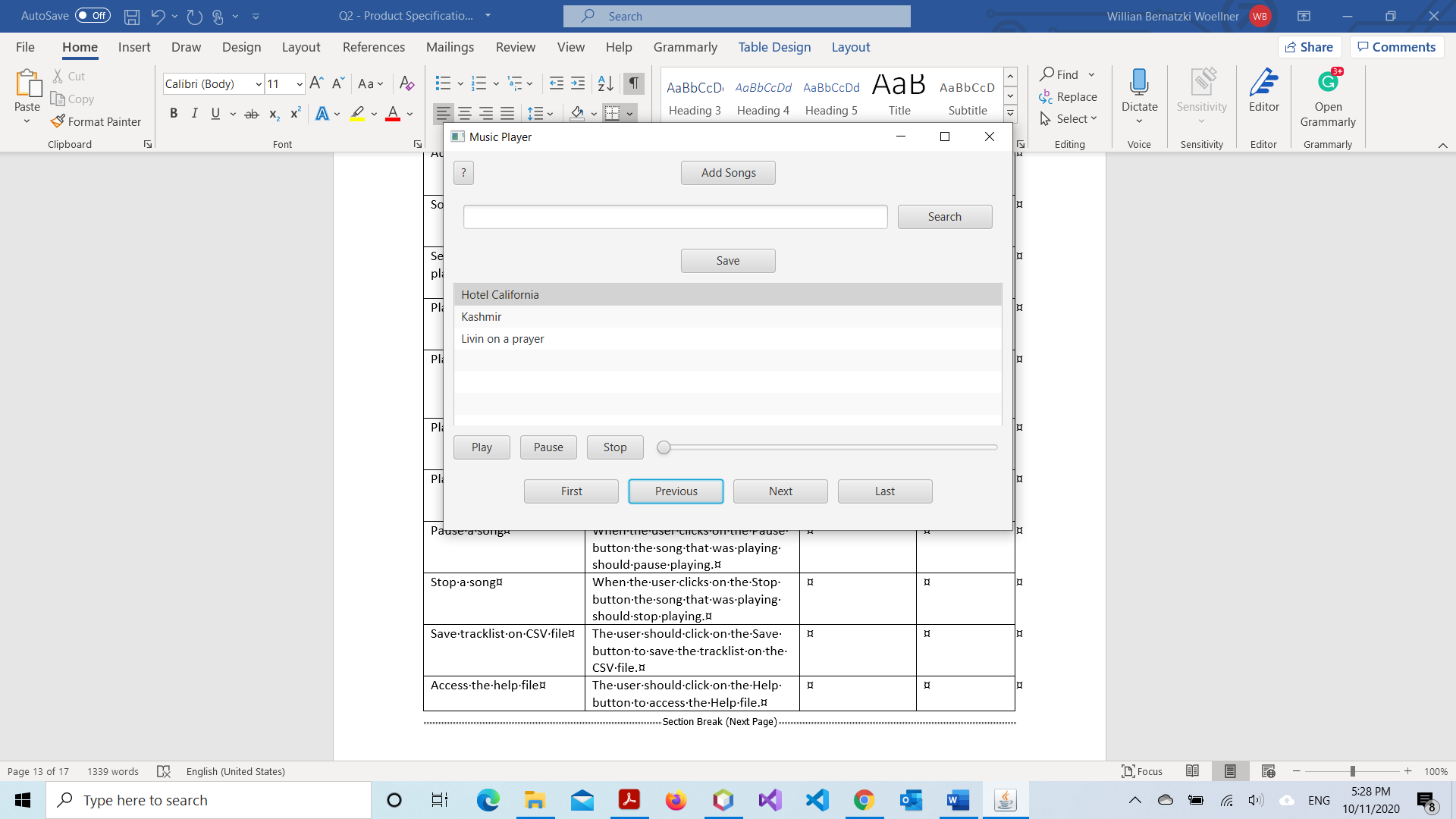
#### Figure 2



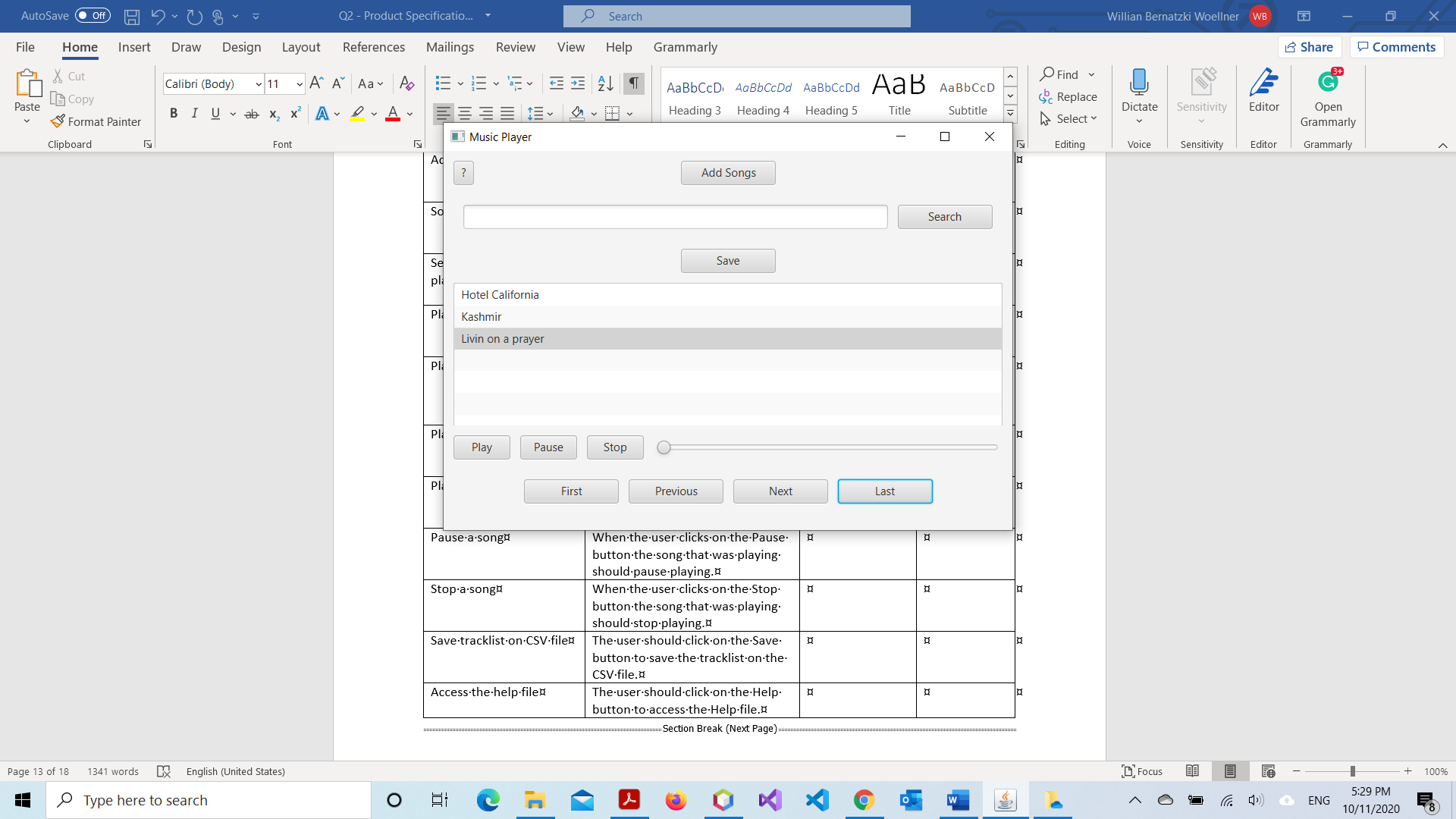
#### Figure 3



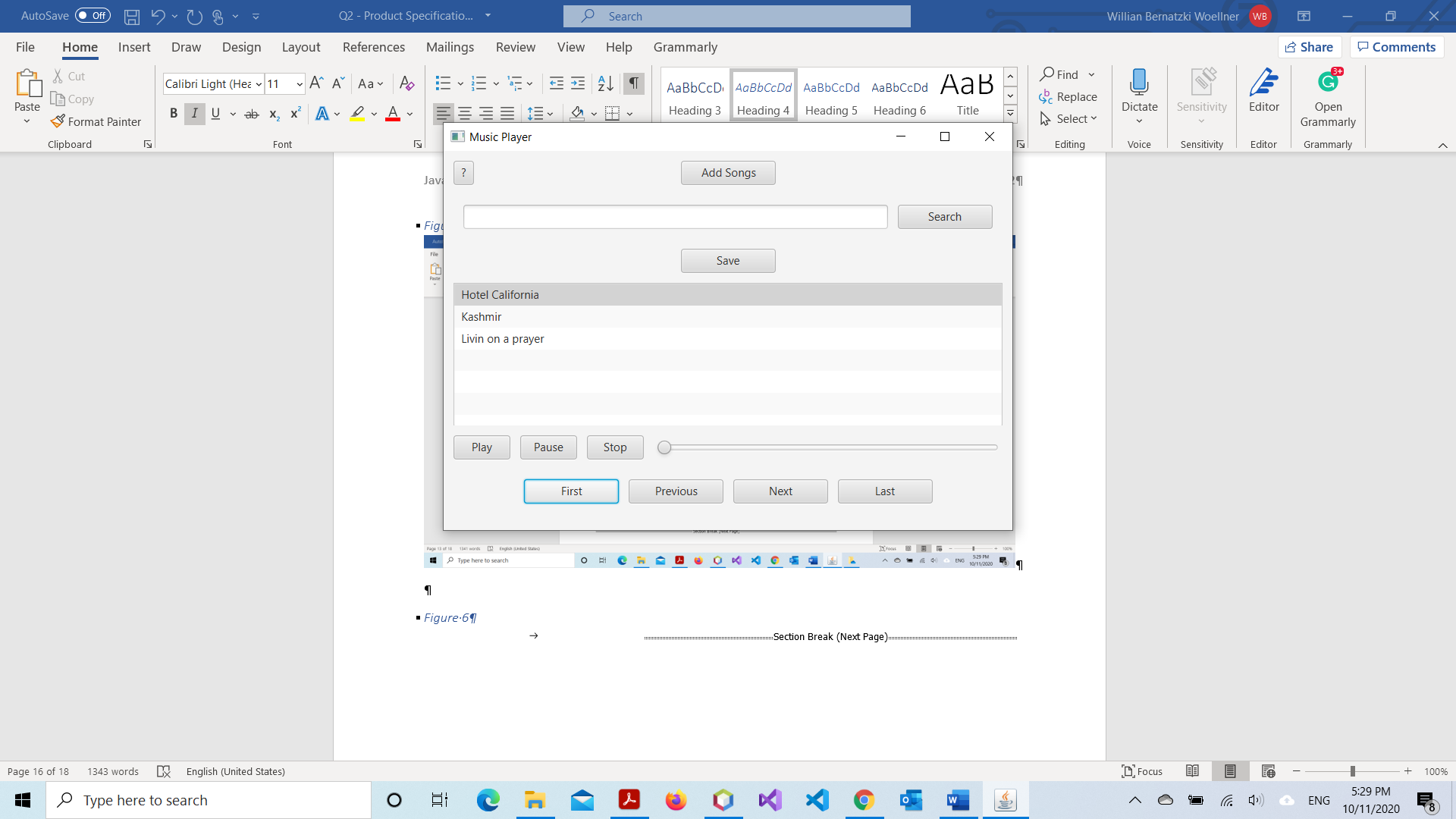
#### Figure 4



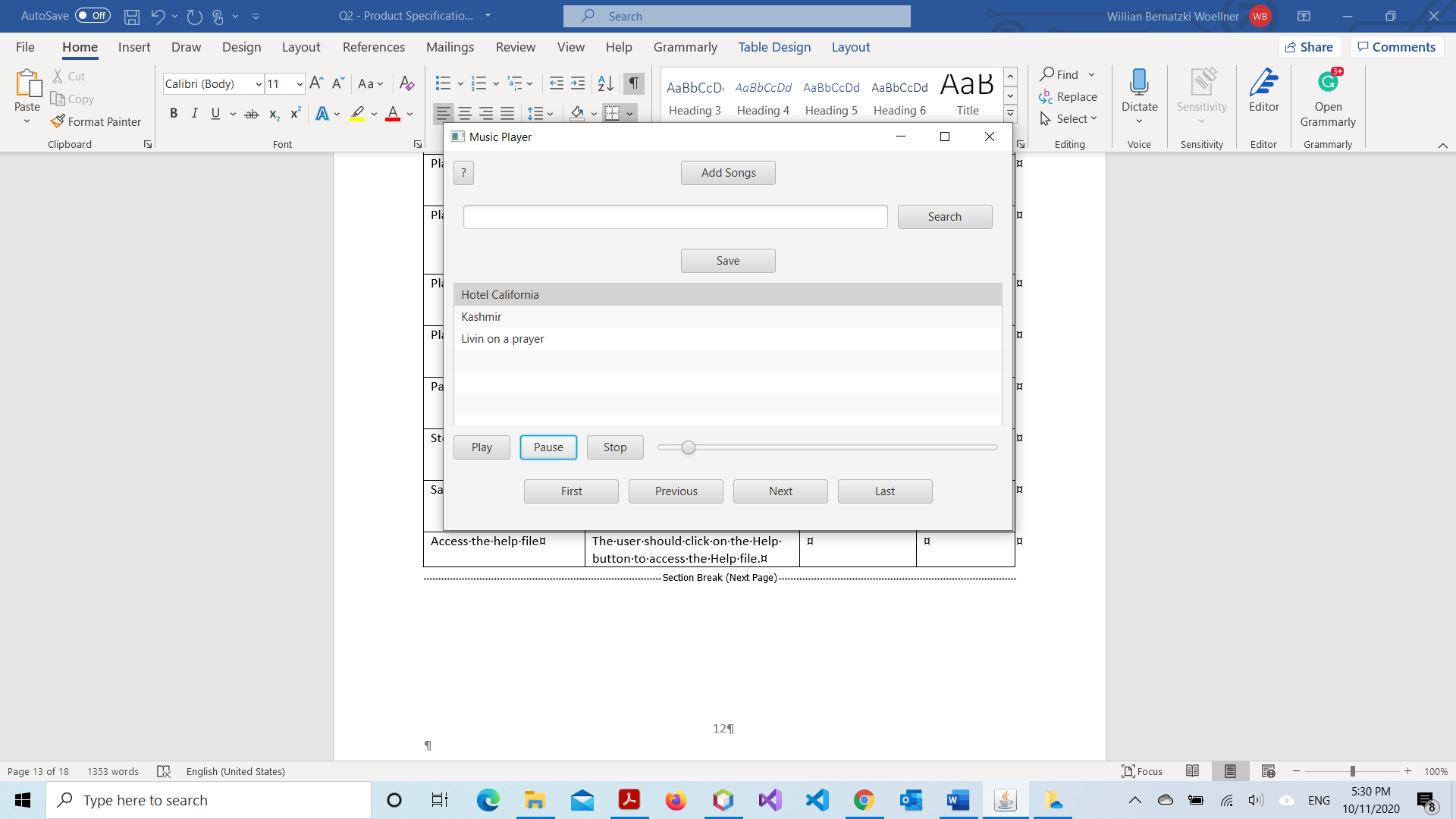
#### Figure 5



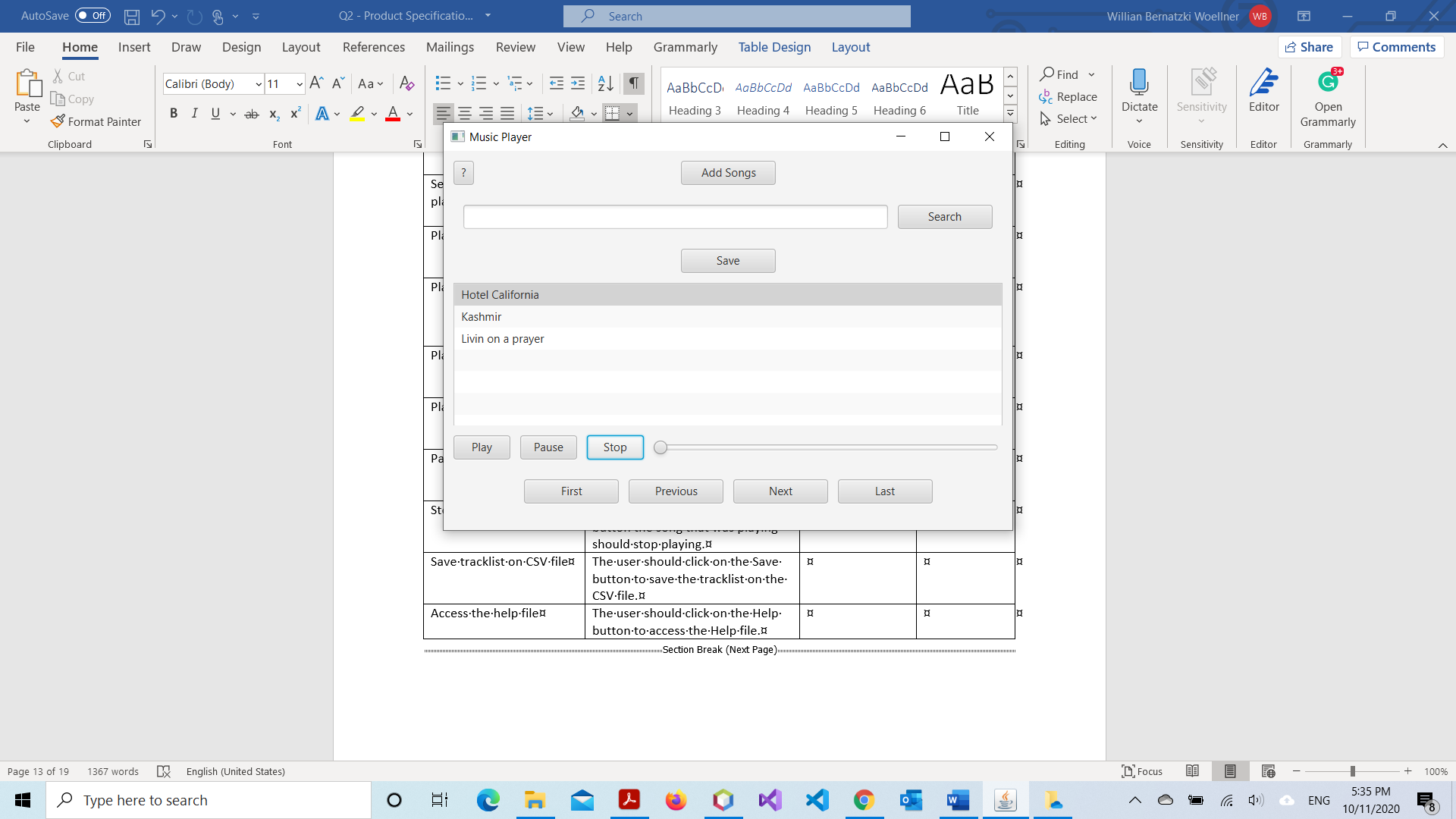
#### Figure 6



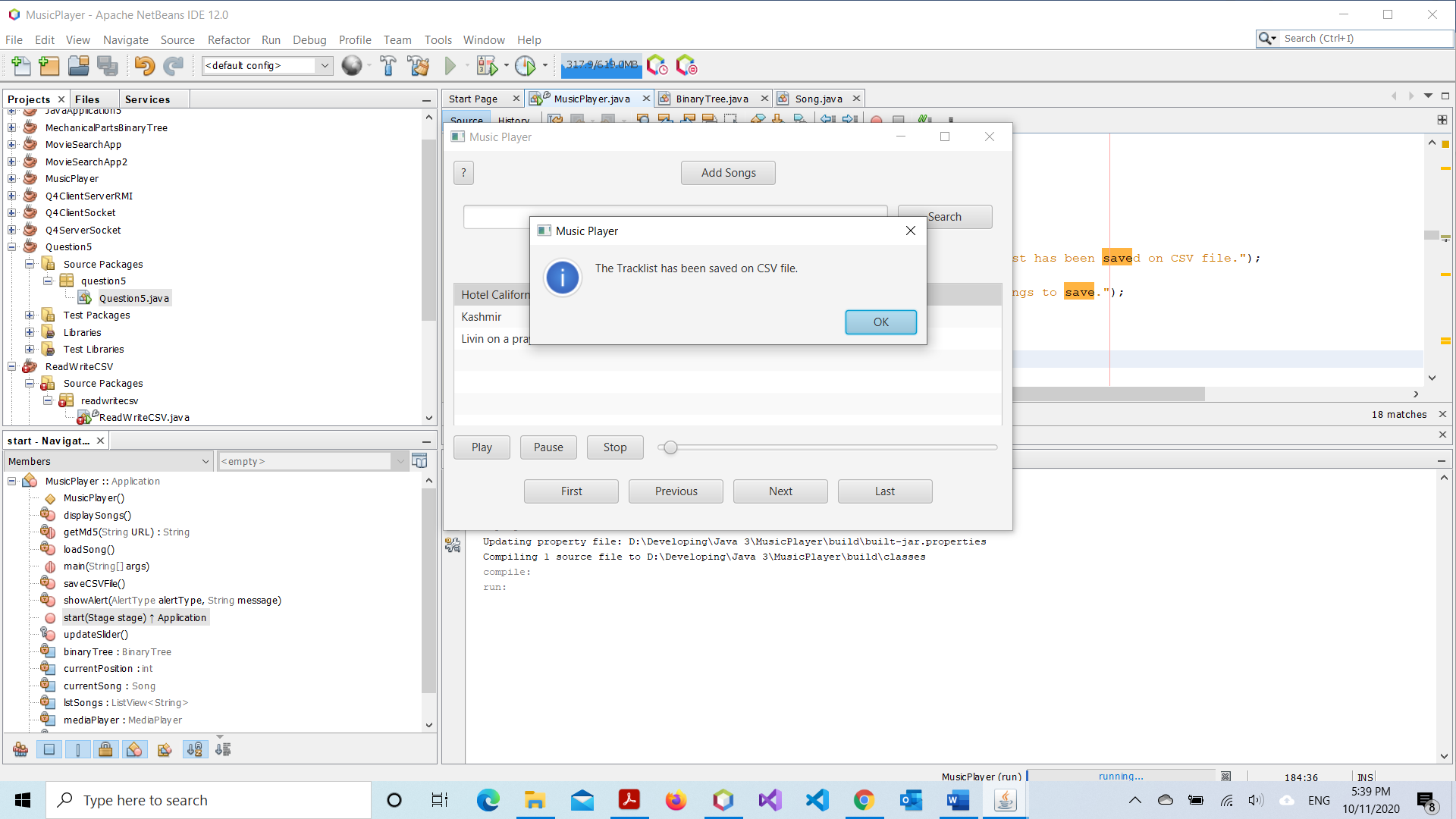
#### Figure 7



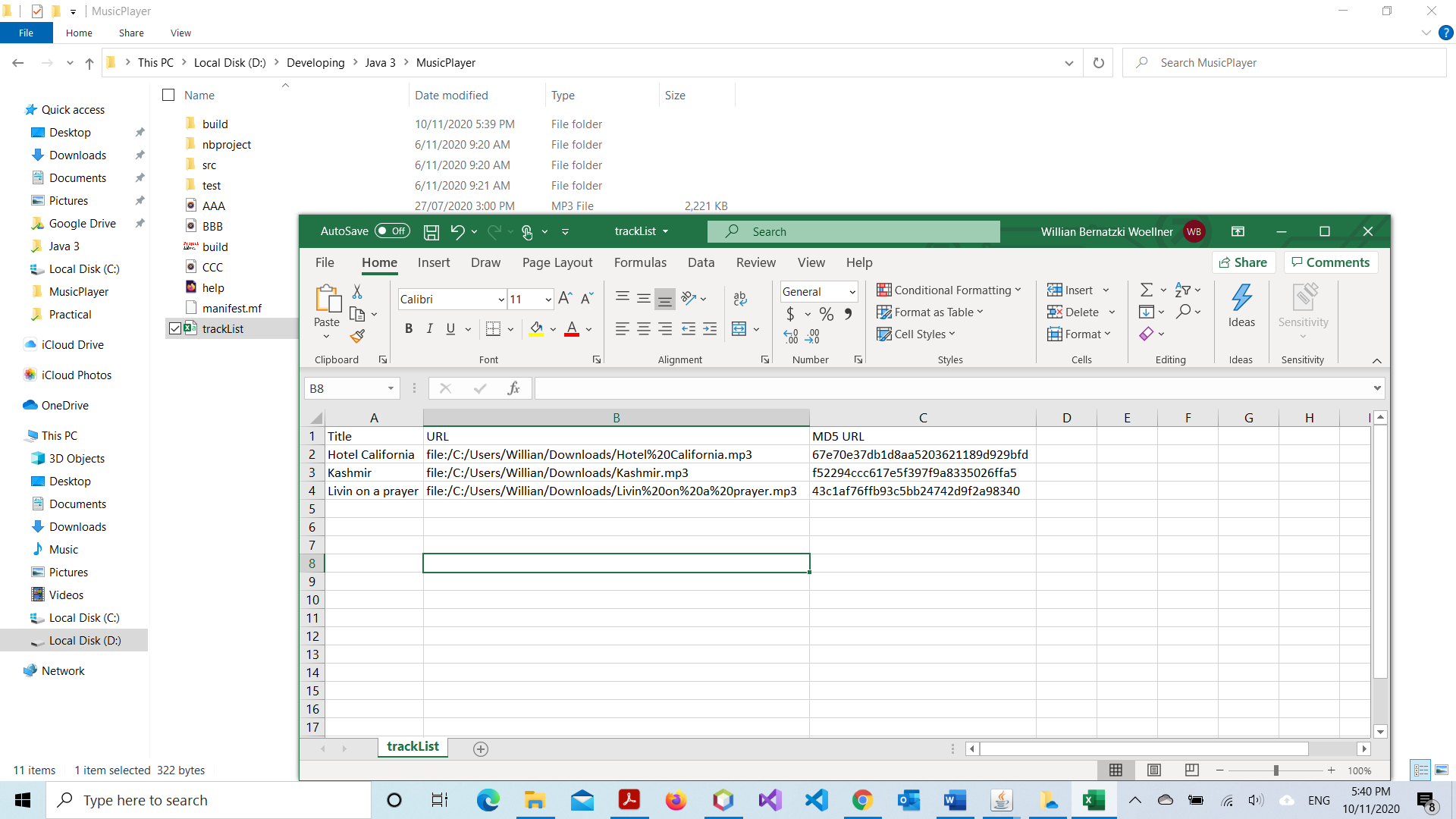
#### Figure 8



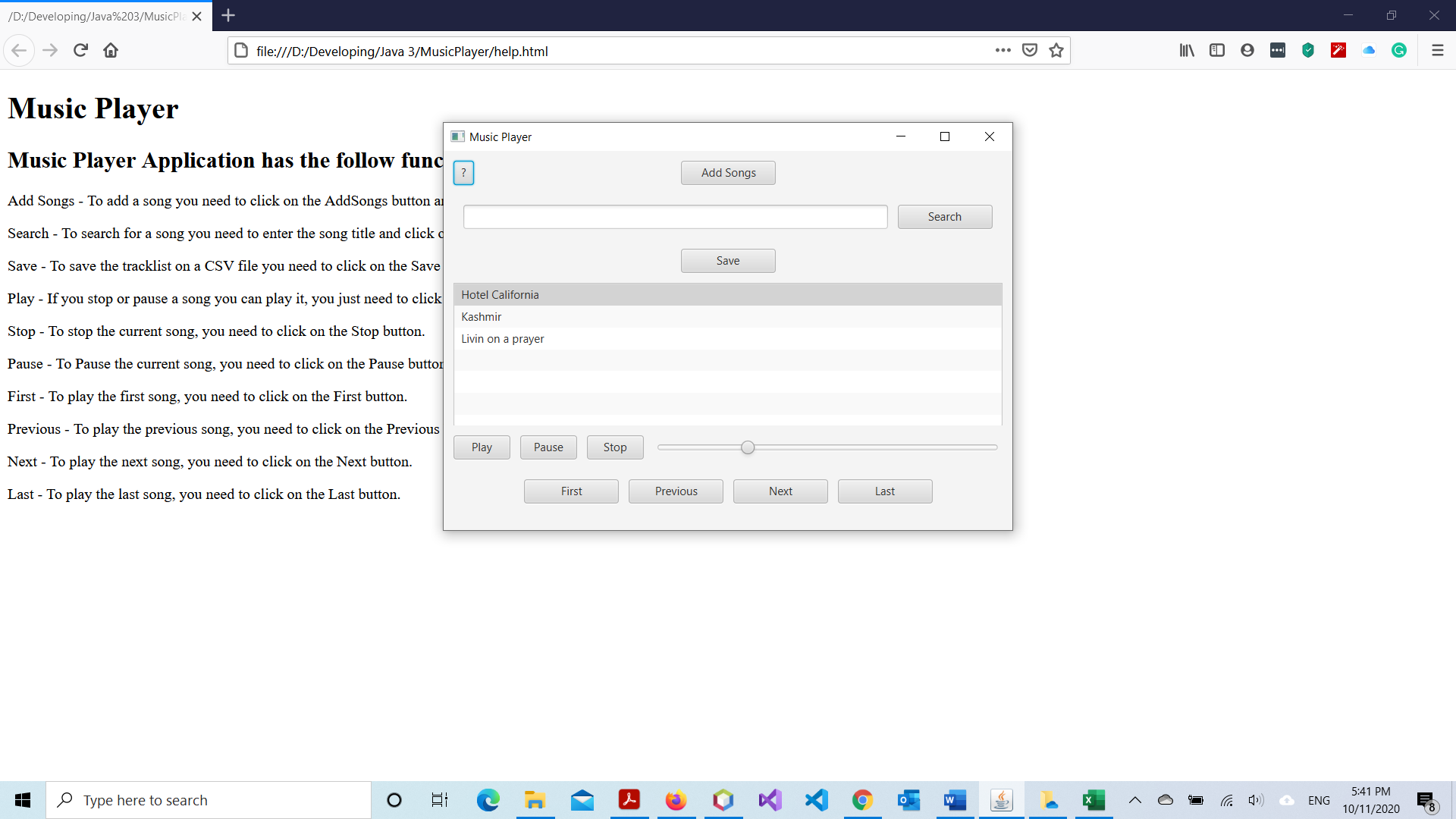
#### Figure 9



#### Figure 10



#### Figure 11



# Product Design Specification Approval

The undersigned acknowledge they have reviewed the Music Player 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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Role: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Title: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Role: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# References

Guru 99. (2020, September 18). *Agile Methodology: What is Agile Software Development Model?* . Retrieved from Guru 99: https://www.guru99.com/agile-scrum-extreme-testing.html

Mallawaarachchi, V. (2017, September 4). *10 Common Software Architectural Patterns in a nutshell*. Retrieved from Towards Data Science: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013