

Software Requirements Specification

for

Media Player, Release 1.0

Version 1.0 approved

Prepared by Chelsea Davis, Jess Albrecht, and James Felts

Team Ctrl-Alt-Delete

December 8, 2015

Table of Contents

Table of Contents

Revision History

1. Introduction

- 1.1 Purpose
- 1.2 Document Conventions
- 1.3 Intended Audience and Reading Suggestions
- 1.4 Project Scope
- 1.5 References

2. Overall Description

- 2.1 Product Perspective
- 2.2 Product Features
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design and Implementation Constraints
- 2.6 User Documentation
- 2.7 Assumptions and Dependencies

3. System Features

- 3.1 Finding media files
- 3.2 Getting media tags
- 3.3 inserting media information into database
- 3.4 Searching database by Artist/Genre

4. External Interface Requirements

- 4.1 User Interfaces
- 4.2 Hardware Interfaces
- 4.3 Software Interfaces
- 4.4 Communications Interfaces

5. Other Nonfunctional Requirements

- 5.1 Performance Requirements
- 5.2 Safety Requirements
- 5.3 Security Requirements
- 5.4 Software Quality Attributes

6. Other Requirements

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: Issues List

Revision History

Name	Date	Reason For Changes	Version
CD/JA/JF	11/19/15	Initial draft	1.0 draft 1
CD/JA	11/23/15	Changes to initial draft for turn-in	1.0 draft 2
JA/JF	12/8/15	updated to include final info	1.0 Release

1. Introduction

1.1 Purpose

This SRS describes the software functional and nonfunctional requirements for release 1.0 of Team Ctrl-Alt-Delete's Media Player project. This document is intended to be used by the members of the project team that will implement and verify the correct functioning of the system. Unless otherwise noted, all requirements specified here are high priority and committed for release 1.0.

1.2 Intended Audience and Reading Suggestions

This document is intended for developers, testers, and users so they know what features we have planned, have completed, and how to use them. Suggested reading order is simply beginning to end, as the beginning of the document is general and covers high-level information, while the end of the document covers the more specific details of the program.

1.3 Project Scope

Our software is intended for minimalist media consumers who want a basic media playback experience that won't soak up memory but provide more advanced features such as playlists and media searching.

1.4 References

Team Coding Standards -

<https://docs.google.com/document/d/1FxQWAX4gOGwdqSOjnU6yMg9hnKdIAwxmcmMx0MXdcJ4/edit?ts=56057ee4>

2. Overall Description

2.1 Product Perspective

We have been tasked by Tom Capaul, for our software engineering class project, with creating a new media player that finds local media files, organizes them in a database and allows playback from playlists as well as searching and sorting of files by artist and genre.

2.2 Product Features

The Media player will scan through the MediaPlayer.exe directory, gather & display info from usable media files, and allow the user to play their files. It will also allow the user to search through the files by artist and genre, and includes various controls such as song looping, volume control, and a time slider to allow advancing through a given song.

2.3 User Classes and Characteristics

Expected users include students, teachers, graders, and average computer media consumers with previous media playing experience. There will be no special privileges or security levels. Our interface is designed to be familiar to the average media player with familiar interfaces such as play, pause buttons, timeline and volume sliders and listing available media to be played which reacts to mouse input.

2.4 Operating Environment

Software will run in Windows 7 or greater, and .NET 4.5.1 or newer.

2.5 Design and Implementation Constraints

Use of Sqlite is required for storing media file information such as location, artist, genre, etc. Security considerations include preventing Sql injection when searching for Artist/Genre. The media player must play MP3's and AVI. Try to use as little libraries as possible to reduce overall size. For our coding standards please view our attached coding standards PDF (1.4 References).

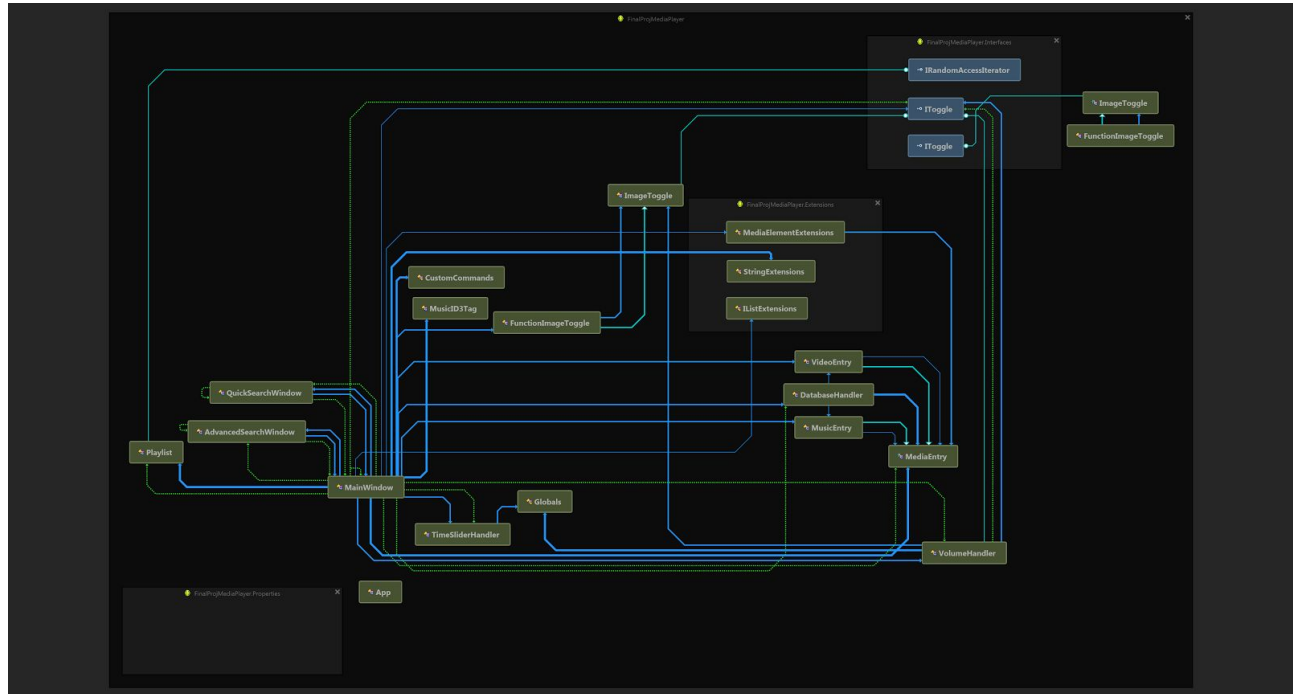
2.6 User Documentation

A ReadMe file will be included with help and contact information as well as brief descriptions of navigation and use.

2.7 Assumptions and Dependencies

This application uses and requires Visual C++ 2013 Update 2 for x64 (required for sqlite, System.data.sqlite dll also included), as well as Windows .NET 4.5.1 and the Windows API code pack(available on newget, visual studio will try to install when opening sln).

3. System Features



<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

Our system features include:

- Finding media files in MediaPlayer.exe location
- Storing found files and file information in sqlite database
- Play, pause, fast forward, rewind to navigate music file
- Volume adjustment and muting
- Playlist reading and executing.

3.1 Finding Media Files

3.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>
High priority to find media files in the music and videos folder in order to navigate the media files.

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

No user action required.

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: .mp3's and .avi's in the MediaPlayer.exe directory

3.2 Getting Tag Info From Media Files

3.2.1 Description and Priority

Medium priority to gather information from media files (necessary for searching, but not necessarily for displaying or playing media). Goes through file urls in given list and gathers info from tags, including the file's title, artist, genre, and length.

3.2.2 Stimulus/Response Sequences

No user action required other than having files available to gather information from. (Feature will still complete even without files to scan from)

3.2.3 Functional Requirements

REQ-1: Program successfully found directory containing media files & gathered their locations.

REQ-2: Files found are either .mp3 or .avi file

Even if there are no files to gather info from, the UI will still open.

3.3 Putting tag info into a database

3.3.1 Description and Priority

Medium priority to take the gathered information and putting it into a SQL database.

3.3.2 Stimulus/Response Sequences

No user action required other than having files available to gather information from.
(Feature will still complete even without files to scan from)

3.3.3 Functional Requirements

REQ-1: List of tags is not empty

3.4 Searching database by Artist/Genre

3.4.1 Description and Priority

Medium Priority to search database by Artist/Genre

3.1.2 Stimulus/Response Sequences

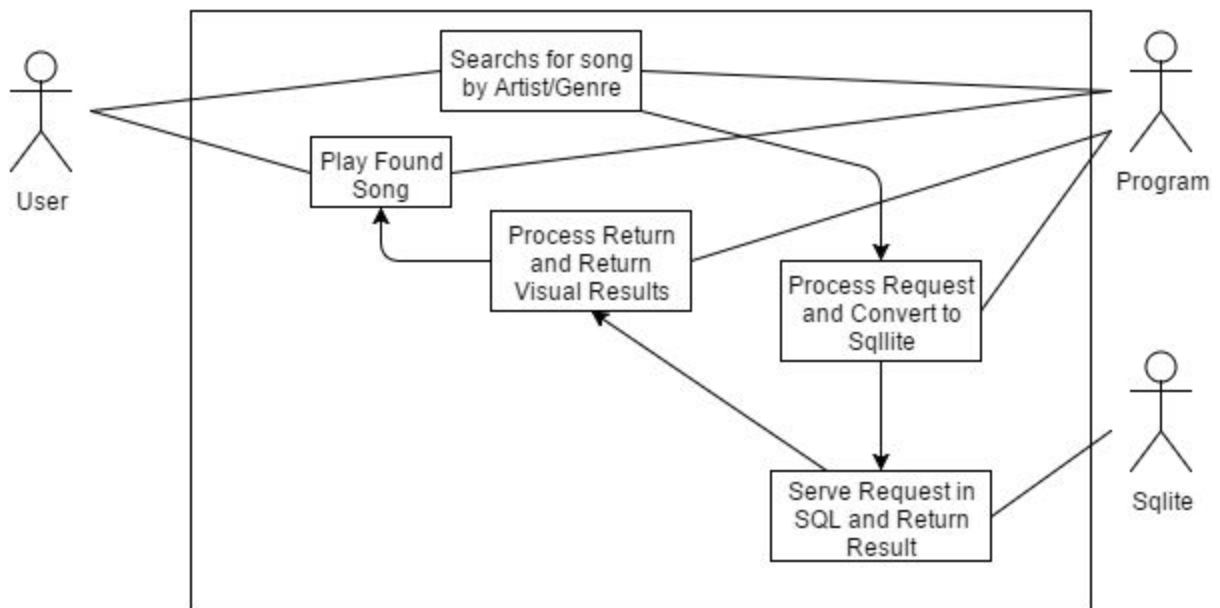
User must enter text into the quick search bar or pull up the advanced search window to search for desired Artist/Genre. Program will take text entered and query database for information and return a list of results.

3.1.3 Functional Requirements

REQ-1: Must have songs/video in the database, if none exist then display none

REQ-2: Files must have Artist/Genre information available, else display none

Find Songs



Use Case Description: Search by Artist/Genre

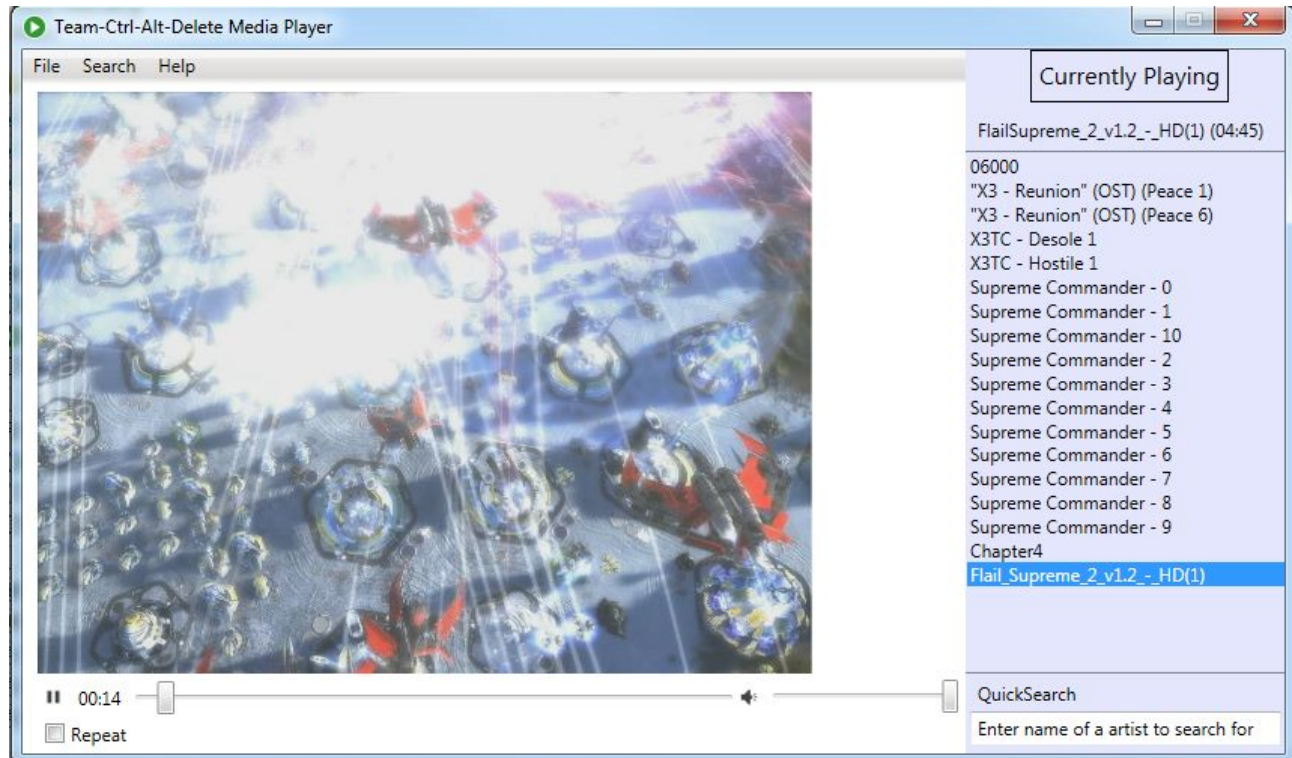
- Description: Search the database of songs by Artist/Genre
- Preconditions:
 - Must have songs/video in the database
 - Files must have Artist/Genre information available
- Post condition: player receives a list of songs with the Artist/Genre searched
- Trigger: typing into the quick search box or into the advanced search window
- Main Scenario:
 1. user types an Artist/Genre into the search box
 2. program converts text into query
 3. the sqlite database is queried by the program and returns results
 4. program creates a list and presents it to the user
 5. user clicks on desired song
- Extension
 - 1a: User types an Artist/Genre that doesn't exist
 - 4a: nothing is presented in return
 - 5a: User clicks on "default playlist" to display all available media

4. External Interface Requirements

4.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

Main User Interface



4.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

4.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

The program will connect to a sqlite database, uses the Windows API Code Pack.

4.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

Performance requirements would be as seamless interaction as possible, music must playback at correct speed and must react to input almost instantaneously.

5.2 Safety Requirements

May strain eye sight from over use, eat lots of carrots and take breaks from time to time.

5.3 Security Requirements

Because the program uses a sqlite database, the program will be made resistant to sql injection.

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>

Possible problems with running on windows 10 via an Access Violation Exception caused by Windows Api. Installer may not work when an Anti Virus is running.