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/1FxQWAX4gOGwdqSOjnU6yMg9hnKdlAwxmcmMx0MXdcJ4/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 thought 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, 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).

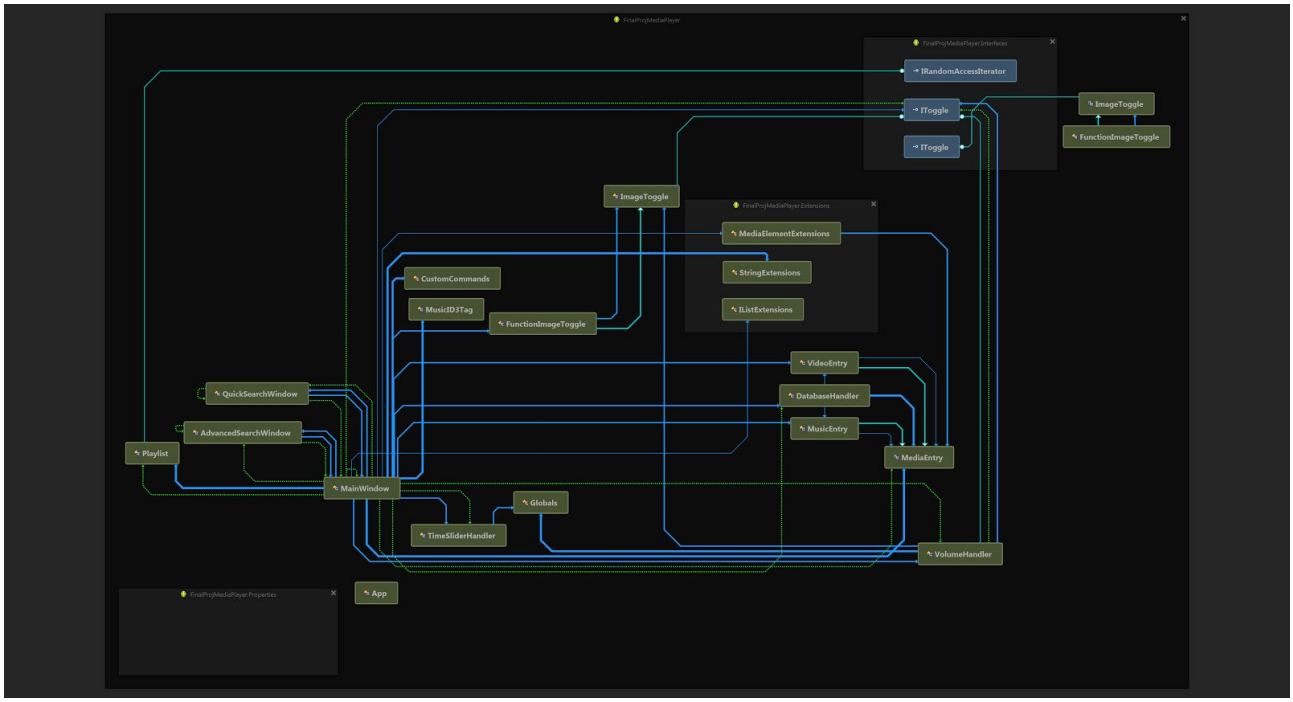
2.6 User Documentation

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 nuget, visual studio will try to install when opening sln).

3. System Features



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

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

No user action required.

3.1.3 Functional Requirements

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

3.4.2 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.

3.4.3 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

3.5 Playlists

3.5.1 Description and Priority

Medium Priority create playlists

3.5.2 Stimulus/Response Sequences

User selects the media and presses enter

3.5.3 Functional Requirement

REQ-1: Must be able to create playlists

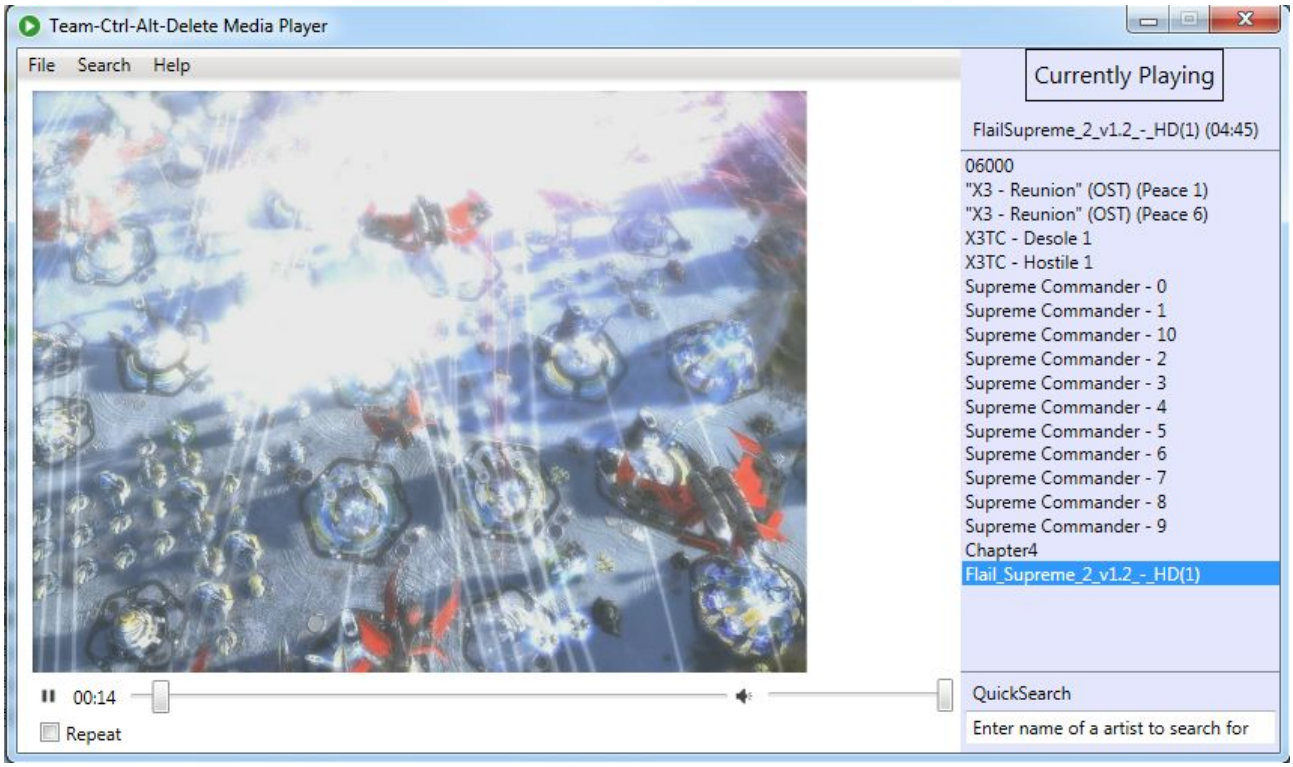
REQ-2: Must be able to save playlists

REQ-3: Must be able to load playlists

4. External Interface Requirements

4.1 User Interfaces

Main User Interface



4.2 Hardware Interfaces

Interfaces with the sound devices available in windows and uses them to produce the audio.

4.3 Software Interfaces

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

4.4 Communications Interfaces

When communicating between the database and the program we use SQLiteCommand.Parameters in order to prevent SQL injection from occurring.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

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

Must prevent SQL injection in user input.

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

Running different types of media will be important to users and developers. Very easily maintainable and expandable for new media navigation and organization.

6. Other Requirements

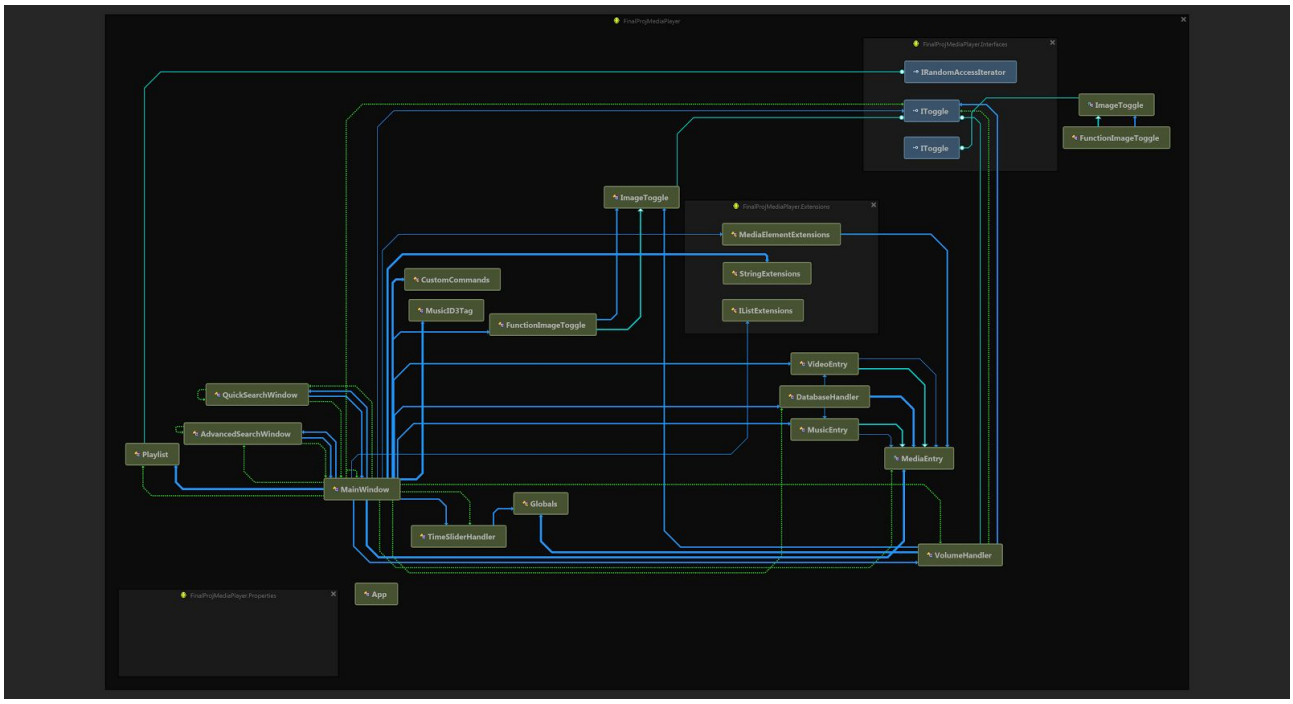
none

Appendix A: Glossary

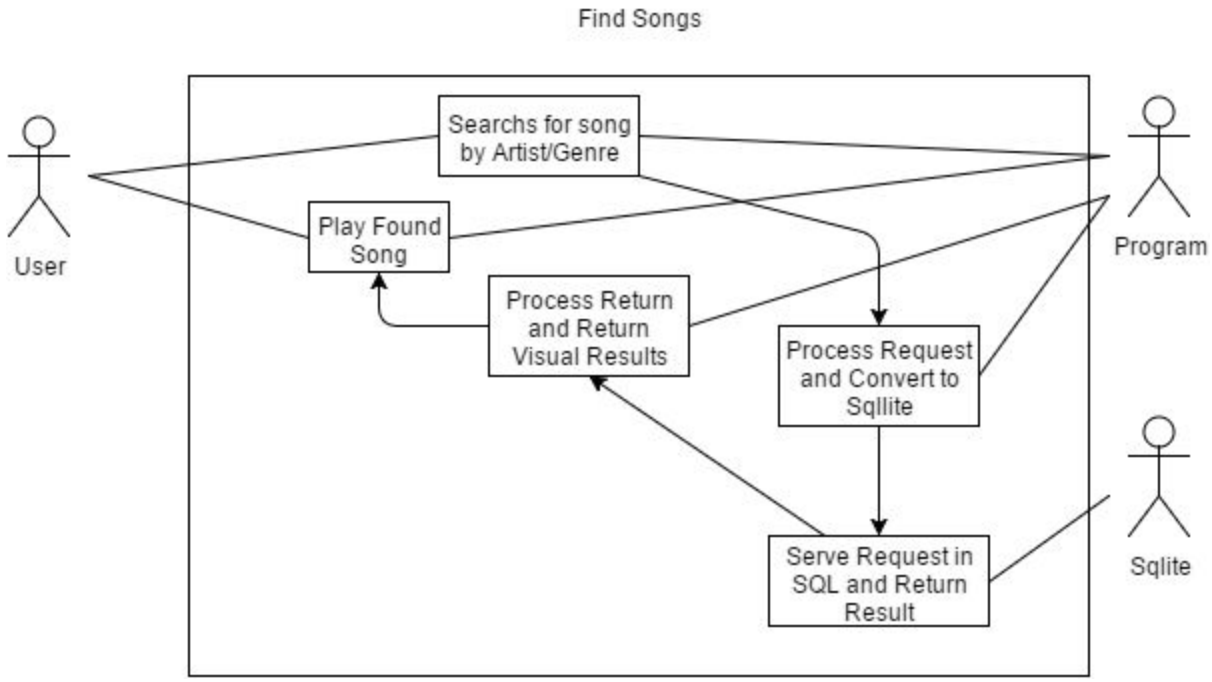
None

Appendix B: Analysis Models

Class Diagram



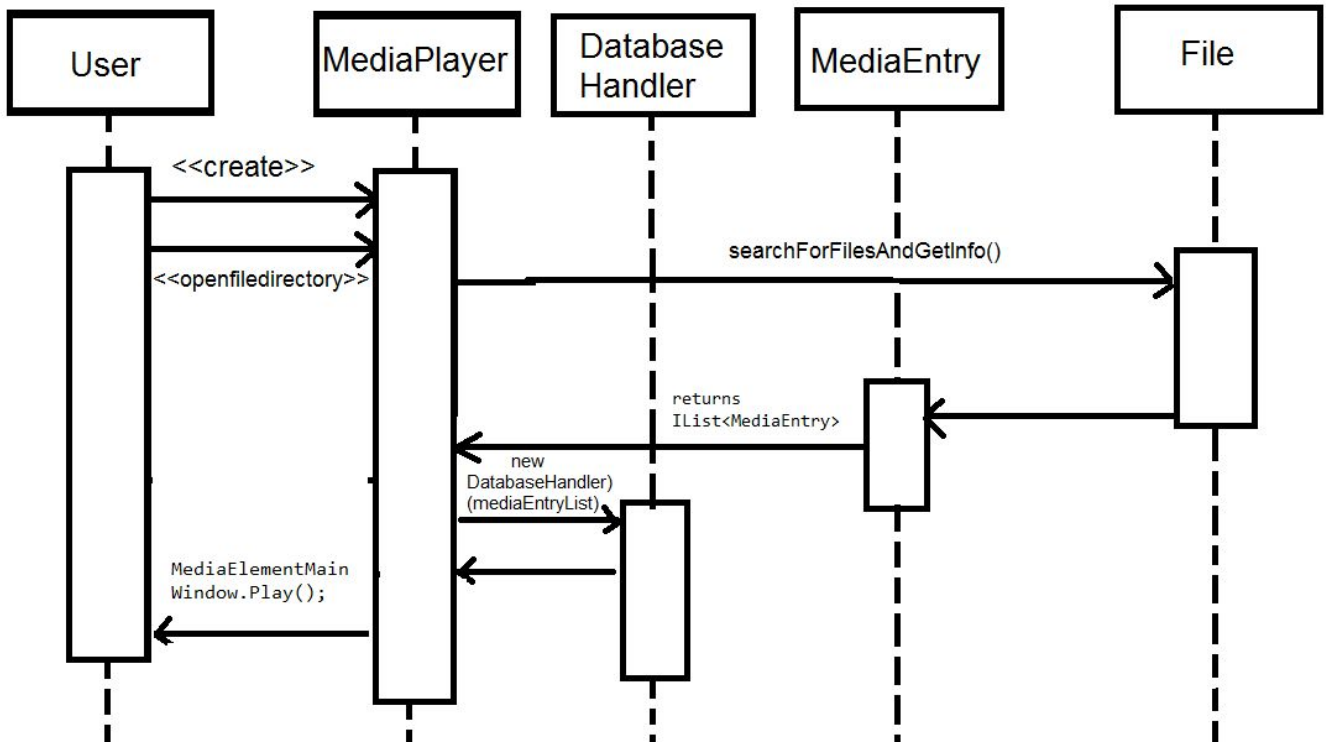
Use Case Diagram



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

Sequence Diagram



Appendix C: Issues List

Possible problems with running on windows 10/8.1 via an Access Violation Exception caused by Windows Api. Installer is currently not working with anti virus. Unable to pull Genre from Avi files. No support for alternative directories.