Software Requirements Specification for

myAudiofiler

Version 1.0

Group 12

Vasanth Banumurthy	178001351	vb331@scarletmail.rutgers.edu
Samantha Ames	182004892	sfa50@scarletmail.rutgers.edu
Henry Chau	194003133	hc869@scarletmail.rutgers.edu

<u>Instructor</u>
Juan Zhai

<u>Course</u>
SOFTWARE ENGINEERING 01:198:431

<u>Lab Section</u>
01

<u>TA's</u>
Zhixing Zhang
Zhenting Wang

02/17/2023

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 Introduction

myAudiofiler is a vinyl record management web application that allows users to catalog and search through their collections. The application is developed using HTML, CSS, and JavaScript paired with Node.js, Express.js, and MongoDB for the backend.

We built this app to serve not only record stores but also music fans who want to organize their collections.

1.1 Document Purpose

This SRS document serves as a myAudiofiler version 1.0 guide for anyone who considers themselves a stakeholder. It details the user requirements our app aims to fulfil, the reason why the app exists. The SRS is a general overview of the entire system and how all the requirements are connected to offer the best possible experience for the end user. Our aim is to simplify the complex tools used to create this project. Hence, you will find that the backend implementation is not too specific. However, you will also find that leaving certain things open ended gives us more room to experiment with later versions.

1.2 Product Scope

The goal of the project was to use modern tools and create a modern solution for a timeless problem. myAudiofiler was created to be easy to use. This means whether you're a veteran collector or a newbie, you'll feel right at home. Now you can benefit from the nostalgia of physical media and the convenience of modern technology.

Above all, we also wanted to leave room for improvements and updates for later versions. Some ideas we have…

- use API to preview songs in album
- add multiple users
- personalize UI
- create wish list
- see other users collections

1.3 Intended Audience and Document Overview

This document is readily available to fellow developers who wish to contribute to the project. We are always open to new ideas. Moreover, we hope that demystifying how this app was made will help testers, including the professor and TA's, offer valuable feedback.

For readers, we recommend starting with the Section 2.1 product overview and 2.2 product functionality. This will help clarify any confusion with the diagrams in section 3.

We have included several videos in section 2 that go over the technical implementation tools. You can find a preview in 1.4 below.

1.4 Definitions, Acronyms and Abbreviations

SRS: Software Requirements Specification, is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill the needs of all business/users.

HTML: HyperText Markup Language, markup language for web page creation, some of its use cases are Web development, Internet navigation and Web documentation.

CSS: Cascading Style Sheets, CSS is a language for specifying how documents are presented to users.

JavaScript: JavaScript is a scripting or programming language that allows programmers to implement complex features on web pages. It is commonly use to create more dynamic interactions when developing web pages, applications and servers.

Node js: Node.js runtime environment for building fast and scalable server-side and networking applications. It runs on the V8 JavaScript runtime engine, and it uses event-driven, non-blocking I/O architecture

Express js: Express js is a Node js framework that is designed for building APIs, web applications and cross platform mobile applications.

MongoDB: MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents.

1.5 Document Conventions

In general this document follows the IEEE formatting requirements. It uses the Arial font and italics for comments. Document text is single spaced and maintains “1” margins.

In addition, certain liberties were taken to make the document more reader friendly. The app name is always written in blue. The version number can be found in red. **Bolding** is used to emphasize and highlighting is used to distinguish.

1.6 References and Acknowledgments

HTML:
https://developer.mozilla.org/en-US/docs/Web/HTML

CSS:
https://developer.mozilla.org/en-US/docs/Web/CSS

JavaScript:
https://developer.mozilla.org/en-US/docs/Web/JavaScript

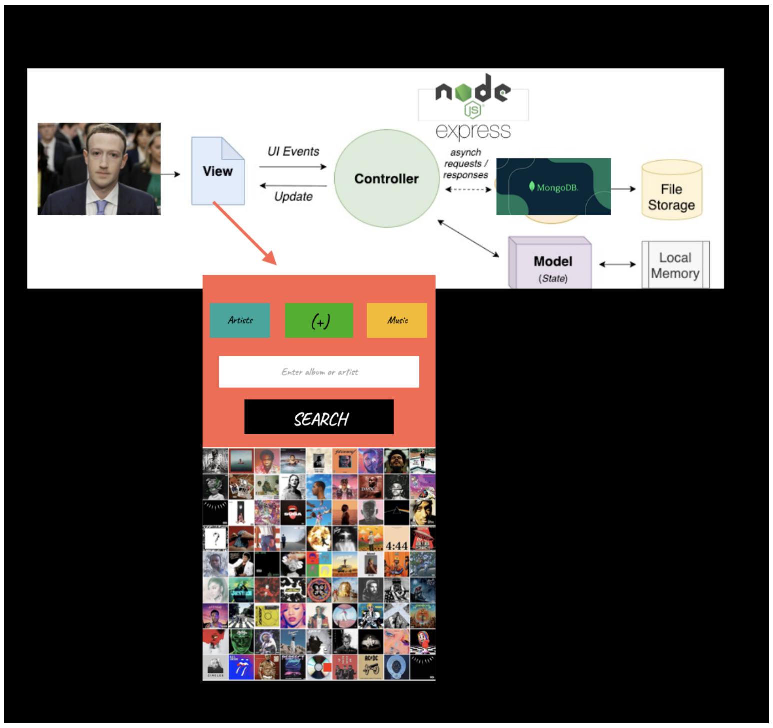
Node and Express:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

MongoDB:
https://www.mongodb.com/docs/

2 Overall Description

2.1 Product Overview

In an age where things are increasingly digital, many people still prefer the feel of physical music. They see album covers as art. They live for the extra content the artists include with the record. They love getting lost in the record store. The act of putting on a record becomes a ritual, an escape from the digital world. Yet, it’s not easy to keep track of all the books scattered throughout the house. myAudiofiler makes it easy to catalog records and even offers intuitive search features. It ensures that wherever you go, you can always showcase your collection!



2.2 Product Functionality

1. Add an artist
2. Add an album (*Title, Artist, Release Date, Description, Rating*)
3. Search (*Title, Artist, Release Date*)
4. View (*Artist and Album*)
5. Home (*Recently added*)

Design and Implementation Constraints

No maintenance is required. The application will support a single user per device.

The main limitation is knowledge of languages, frameworks, and environments used. To combat this, we have included a list of videos that might help.

HTML & CSS:
https://www.youtube.com/watch?v=G3e-cpL7ofc

JavaScript:
https://www.youtube.com/watch?v=PkZNo7MFNFg

Node and Express:
https://www.youtube.com/watch?v=Oe421EPjeBE

MongoDB:
https://www.youtube.com/watch?v=ofme2o29ngU

COMET is used for software design and the UML modeling language.

COMET
https://www.informit.com/articles/article.aspx?p=349038&seqNum=7

UML
https://www.uml.org/

2.3 Assumptions and Dependencies

The main assumption is that the user has enough records to add to the collection to test all the features. For instance, the search and view features would be limited if the user only has a few records.

Another assumption is that the user is able to locate all pertinent information regarding the record such as the date published.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The home page features recently added albums. It displays the covers. Clicking on an album brings you to the album view with details. The album view allows you to view the artist and edit details. You can even remove the album from the collection.

The home page also allows the user to search for albums based on the title, artist, or publishing date.

Three tabs will be located in the header.
The Artist tab will bring you to a list of artists.
The Albums tab will bring you to a list of albums.
The (+) tab will let you add an artist or album.

3.1.2 Hardware Interfaces

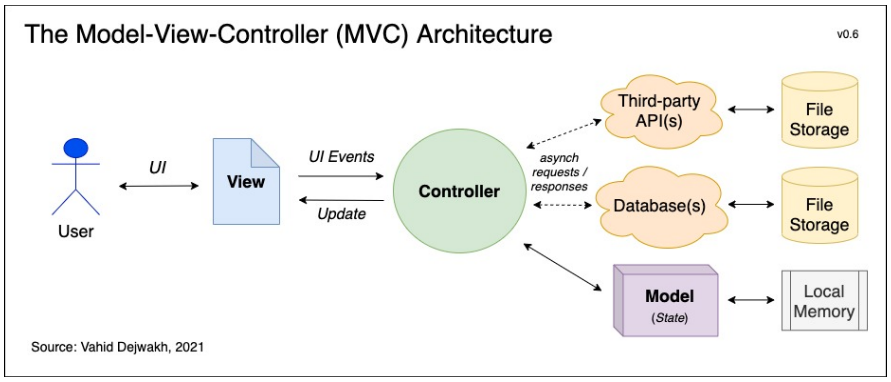
The user will interact with the screen.

On a desktop, users can use a mouse and keyboard.
On mobile, they can use touch.

The application will use the MVC pattern described below.

3.1.3 Software Interfaces

The Model View Controller (MVC) design pattern is used.

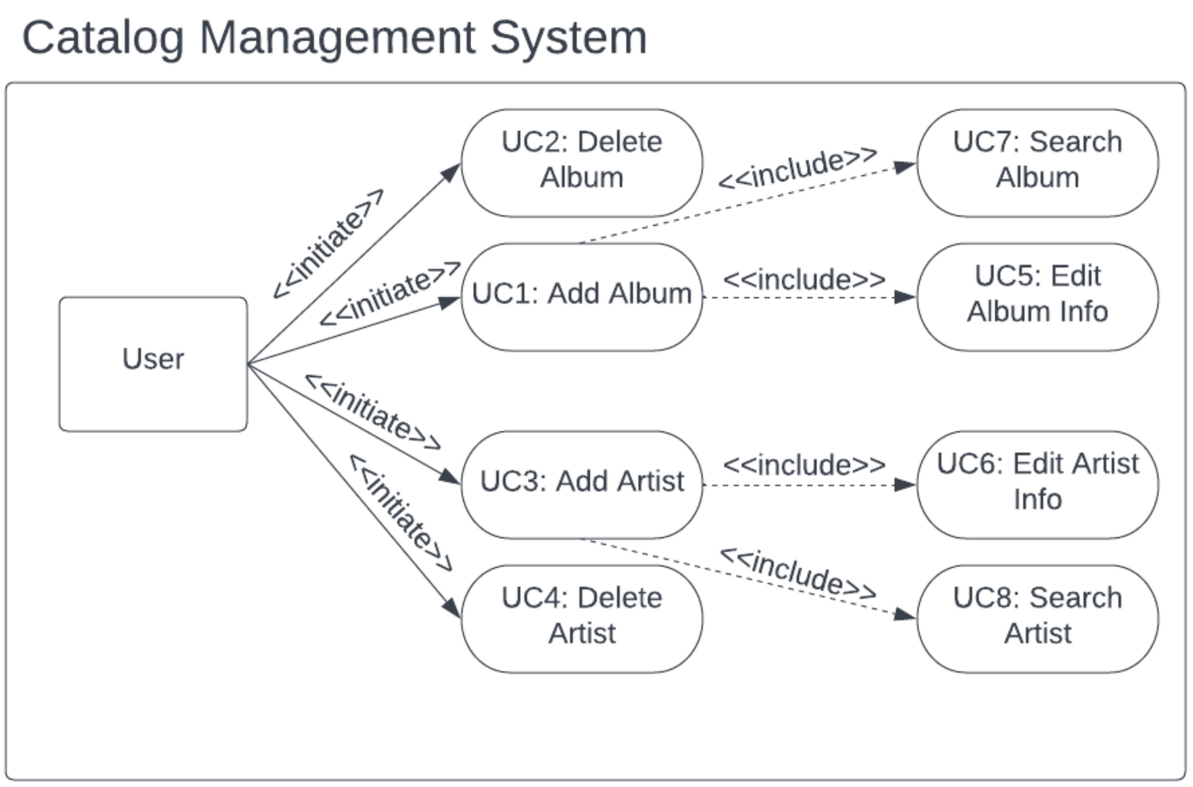


Our goal with the software was to optimize it for a range of platforms. As a responsive web application, myAudiofiler also works as intended on mobile devices.

3.2 Functional Requirements

Identifier	Priority	Requirement
REQ-1	5	The system shall allow for users to add or remove albums from their catalog.
REQ-2	5	The system shall allow for users to add or remove artists from their catalog.
REQ-3	4	The system shall maintain a grid-list catalog of music album covers in alphabetical order, by precedence of artist then year.
REQ-4	4	The system shall maintain a list of artists corresponding to the albums in a user's collection.
REQ-5	4	The system shall allow for the modification of album information, i.e., artist name, album title, album artwork, user rating, etc.
REQ-6	3	The system shall allow the user to view album information, i.e., artist name, album title, album cover, user rating, etc., upon clicking album artwork.
REQ-7	3	The system shall allow the user to search for an album based on one of the following parameters: artist name, album title, year, etc.
REQ-8	2	The system shall allow the user to view albums created by the artist upon clicking on the artist's name from the artist list.
REQ-9	2	The system shall maintain a list of recently added albums on the user's homepage.
REQ-10	1	The system shall notify the user when the addition of a duplicate album is attempted.

3.3 Use Case Model



In Figure 3.3, the User is the actor that initiates the program by adding a new album or artist, or deleting an existing album or artist. The subsequent use cases that follow the addition of an album are searching for an existing album in the user's catalog or editing the information for a selected album. The subsequent use cases that follow the addition of an artist are searching for an existing artist in the user's catalog or editing the information for a selected artist.

3.3.1 Use Case #1 UC1: Add Album

UC1:	**Add Album**
Author:	Samantha Ames
Purpose:	To allow a user to add an album and its corresponding information to their personal catalog.
Requirements Traceability:	UC5, UC7
Priority:	5
Preconditions:	n/a
Post-conditions:	The added album populates in the “recently added” portion of the user’s homepage, the user’s general catalog, and the page of the corresponding artist.
Actors:	User
Extends:	n/a
Flow of Events:	1. **User** opens myAudiofiler and selects the menu item “Add Album” 2. **User** is prompted to input album information 3. **User** enters information and selects “Add Album” 4. The new album is cataloged for the user
Includes:	n/a
Notes/Issues:	The information must be valid, i.e., “Year” must not be a future date.

3.3.2 Use Case #2 UC2: Delete Album

UC2:	**Delete Album**
Author:	Samantha Ames
Purpose:	To allow the user to delete an existing album from their catalog.
Requirements Traceability:	UC1
Priority:	5

Preconditions:	The album must have been previously added by the user and present in the general catalog.
Post-conditions:	The album must no longer be available for viewing in any part of the catalog, and no longer associated with an artist.
Actors:	User
Extends:	UC1
Flow of Events:	1. **User** view the grid-list catalog of albums 2. **User** selects an album 3. **User** selects “Delete Album” 4. **User** is prompted to confirm deletion and selects “Yes” 5. Album is no longer available to view in catalog and is no longer associated with an artist
Includes:	UC1
Notes/Issues:	The album must be present in the user’s catalog.

3.3.3 Use Case #3 UC3: Add Artist

UC3:	**Add Artist**
Author:	Samantha Ames
Purpose:	To allow a user to add an artist and their corresponding information to the user’s personal catalog.
Requirements Traceability:	UC6, UC8
Priority:	5
Preconditions:	n/a
Post-conditions:	The added artist will populate the user’s artists list.
Actors:	User
Extends:	n/a
Flow of Events:	1. **User** opens myAudiofiler and selects the menu item “Add Artist” 2. **User** is prompted to input artist information 3. **User** enters information and selects “Add Artist” 4. The new artist is cataloged for the user

Includes:	n/a
Notes/Issues:	n/a

3.3.4 Use Case #4 UC4: Delete Artist

UC4:	**Delete Artist**
Author:	Samantha Ames
Purpose:	To allow the user to delete an existing artist from their catalog.
Requirements Traceability:	UC3
Priority:	5
Preconditions:	The artist must have been previously added by the user and present in the artist list.
Post-conditions:	The artist must no longer be available for viewing in any part of the catalog or artist list. Any existing album associated with the artist must also be removed.
Actors:	User
Extends:	UC3
Flow of Events:	1. **User** view the artist list 2. **User** selects an artist 3. **User** selects “Delete Artist” 4. **User** is prompted to confirm deletion and selects “Yes” 5. Artist is no longer available to view in catalog or artists list, and any associated album is also removed
Includes:	UC3
Notes/Issues:	The artist must be present in the user’s artists list and any corresponding albums must also be subsequently removed.

3.3.5 Use Case #5 UC5: Edit Album Info

UC5:	**Edit Album Info**

Author:	Samantha Ames
Purpose:	To allow a user to edit the information pertaining to a selected album.
Requirements Traceability:	UC1
Priority:	4
Preconditions:	The album must have been previously added to the catalog by the user.
Post-conditions:	Once updated, the album information refreshes with the new details.
Actors:	User
Extends:	UC1
Flow of Events:	1. **User** view the grid-list catalog of albums 2. **User** selects an album 3. **User** selects “Edit Album” 4. **User** inputs new information and selects “Update” 5. Album information repopulates with new details
Includes:	UC1
Notes/Issues:	Information must be valid, i.e., “Year” must not be a future date.

3.3.6 Use Case #6 UC6: Edit Artist Info

UC6:	**Edit Artist Info**
Author:	Samantha Ames
Purpose:	To allow a user to edit the information pertaining to a selected artist.
Requirements Traceability:	UC3
Priority:	4
Preconditions:	The artist must have been previously added to the artists list by the user.
Post-conditions:	Once updated, the artist information refreshes with the new details.

Actors:	User
Extends:	UC3
Flow of Events:	1. **User** views the artists list 2. **User** selects an artist 3. **User** selects “Edit Artist” 4. **User** inputs new information and selects “Update” 5. Artist information repopulates with new details
Includes:	UC3
Notes/Issues:	n/a

3.3.7 Use Case #7 UC7: Search Album

UC7:	**Search Album**
Author:	Samantha Ames
Purpose:	To allow user to search for an album within their personal catalog, based on parameters such as album title, artist name, etc.
Requirements Traceability:	UC1
Priority:	3
Preconditions:	A successful search requires that the album be present in the user’s catalog and its information match the query.
Post-conditions:	For an album to return to the user upon search, it must be present in the catalog and its information must match the user's query; otherwise, empty return.
Actors:	User
Extends:	UC1
Flow of Events:	1. **User** opens myAudiofiler and selects the menu item “Search Album” 2. **User** is prompted to input album information 3. **User** enters information and selects “Search” 4. If an album exists in the catalog with matching information to the user’s query, an album is returned; otherwise, empty return
Includes:	UC1

Notes/Issues:	Possible that query could be misspelled

3.3.8 Use Case #8 UC8: Search Artist

UC8:	**Search Artist**
Author:	Samantha Ames
Purpose:	To allow user to search for an artist within their artists list based on name.
Requirements Traceability:	UC3
Priority:	3
Preconditions:	A successful search requires that the artist be present in the user's artists list and their information match the query.
Post-conditions:	For an artist to return to the user upon search, it must be present in the artists list and its information must match the user's query; otherwise, empty return.
Actors:	User
Extends:	UC3
Flow of Events:	1. **User** opens myAudiofiler and selects the menu item "Search Artist" 2. **User** is prompted to input artist information 3. **User** enters information and selects "Search" 4. If an artist exists in the artists list with matching information to the user's query, an artist is returned; otherwise, empty return
Includes:	UC1
Notes/Issues:	Possible that query could be misspelled

4 Other Non-functional Requirements

4.1 Performance Requirements

TODO: Provide performance requirements based on the information you collected from the client/professor. For example, you can say "P1. The secondary heater will be engaged if the desired temperature is not reached within 10 seconds">

P1. Performing any task of the program should not take more than 10s considering the library is reasonably size.

4.2 Safety and Security Requirements

TODO:

- *Provide safety/security requirements based on your interview with the client - again you may need to be somewhat creative here. At the least, you should have some security for the mobile connection.*

Incase of any data corruption and data loss on the application, we will use data duplication to keep the same data in different locations to ensure consistency and back up/ disaster recovery use case.

4.3 Software Quality Attributes

TODO: Use subsections (e.g., 4.3.1 Reliability, 4.3.2 Adaptability, etc…) provide requirements related to the different software quality attributes. Base the information you include in these subsections on the material you have learned in the class. Make sure, that you do not just write "This software shall be maintainable…" Indicate how you plan to achieve it, & etc…Do not forget to include such attributes as the design for change (e.g. adapting for different sensors and heating/AC units, etc.). Please note that you need to include ***at least*** *2 quality attributes. You can Google for examples that may pertain to your system.>*

Performance: The speed of searching, adding and deleting albums should be reasonably fast by writing optimal algorithms querying for the database.

Usability: This application has great usability with simple operations and control which are easy to understand. User interface is also aesthetically pleasing and user friendly.

Compatibility: This is a web based application so it is compatible with any device with internet access inside a web browser.

Testability: This application is relatively easy to test with only a handful of important functions, any edits to the album should be reflected on the application and also the database. Good testability leads easy implementation of a new feature in the future.

5 Other Requirements

n/a

Appendix A – Data Dictionary

Column	**Data Type**	**Description**	**Requirements**
album_title	string	title for album	
album_year	int	album release year	Year must not be a future date
artist_name	string	name of artist	
album_art	JPEG, PNG	thumbnail for album	

Appendix B - Group Log

Project correspondence took place primarily via Zoom, with the first few meetings covering different possible ideas for the project from each member. After assessing the level of experience in different languages and skill sets, we concluded that a library management system with specific personalization preferences would be the most conducive for our group. In completing the project specification proposal, each group member completed a different portion of the document, all whilst corresponding together on Zoom for specifications, diagrams, etc. We also keep consistent communication for sharing resources via email and a group chat.