Ada's Firewall SRD

Software Requirements Document for MyMusicList

Last Updated: May 1, 2020

MyMusicList

A music ratings application for recording your experiences with music.

MyMusicList was developed by:

Ada’sFirewall

Ada’s Firewall Team Members:
Eric Cortes-Aguilera
William Higdon
Quinn Tjin-a-soe
Fernando Villarreal

This document was written and edited by:
Fernando Villarreal (Primary Contributor)
Eric Cortes-Aguilera
William Higdon
Quinn Tjin-a-soe

University of North Carolina at Greensboro

Table of Contents

Introduction

Purpose

The purpose of this document is to describe, in detail, the software MyMusicList and its features. It also describes the functional, technical, and non-functional requirements of the software.

The purpose of MyMusicList is to provide its users a way of recording and managing a record of their experiences with music. MyMusicList will let users search for and rate music.

Document Conventions

This document is divided into five major sections:

- Introduction (this section).
- Description.
- Functional Requirements.
- Technical Requirements.
- Non-Functional Requirements.

Important terms that precede their definitions or explanations are **bolded**.

Intended Audience

The intended audience, or users, of MyMusicList are people who wish to record their experiences with music. This audience includes people who enjoy listening to music regularly and wish to record their experiences with music.

Definitions

In this section, a list of terms that are used throughout this document are defined. The term that is being defined will be **bolded** like so.

Ada's Firewall is the name of the organization that developed MyMusicList and collectively wrote this document. The team members of Ada's Firewall are: Eric Cortes-Aguilera, William Higdon, Quinn Tjin-a-soe, and Fernando Villarreal.

Administrators refers to any person that has access to MyMusicList features or functionality that is not accessible to the Intended Audience. Software developers and engineers that work on the development and/or maintenance of MyMusicList are also considered Administrators.

AlbumObject: *See MusicObject*.

API stands for Application Programming Interface. An API is an interface for a software that allows third party software developers to use the functionality of that software. In the context of this document, the term "API" may be used to refer to the external API used for MyMusicList.

Application or **App** are alternate terms for software. These terms may be used interchangeably throughout this document. These terms may also be used to refer to MyMusicList, in certain contexts.

ArtistObject: *See MusicObject*.

IDE stands for Integrated Development Environment. An IDE is a desktop application used by software developers to write and run code.

Endpoint is another name for functions or subroutines, in the context of computer programming.

GUI stands for Graphical User Interface. A GUI lets a user interact with a software through a visual representation with buttons, textboxes, links, etc.

Java is a popular object-oriented coding language. MyMusicList was primarily written in Java.

JSON is a file format that stores data objects in key-value pairs and lists. JSON files or objects consist of text that can be easily read by humans and parsed by machines.

MyMusicList is the name of the software that this document is about. MyMusicList may also be referred to as "the app" in certain contexts.

MusicObject or **MusicObjects** are terms that refer to artists, albums, and tracks/songs. In the context of this document, these things are music objects. As such, they may also be referred to as ArtistObject, AlbumObject, and TrackObject respectively.

OS stands for Operating System. For example: Windows 10 is an operating system.

RecordObject or **RecordObjects** are terms used to refer to the implementation of records from the Database such as users, ratings, and plan-to-listen records. Each of these record types have their own subclass which inherits from RecordObject.

String is a type of data that contains text, in the context of computer programming.

TrackObject: *See MusicObject*.

Users refers to any person that uses MyMusicList. Users also refers to the Intended Audience, as previously described. The term Users does not refer to Administrators despite the fact that they could also be considered users of the application. Whenever this term is used it should be interpreted to refer to all non-Administrator users.

We, in the context of this document, only and always refers to team members of Ada's Firewall.

JavaFX refers to a software platform which enables the use of graphical user interfaces to be implemented into the project.

Java FXML refers to a file type which is similar to the known HTML computer language which enables to code the appropriate design for the application which is later then used by the software platform JavaFX.

Project Scope

The purpose and scope of this project is to provide a potential user an application where they can make a list of all of the music they've listened to or plan to listen to in the future and give them a rating. This project will utilize a datastore with text files and connect to the Spotify API. Above all, we hope to provide the user with a good application that will allow them to keep track of all the music they've listened to and rate it.

Technical Challenges

This section describes the technical challenges encountered throughout the development of MyMusicList.

Music API Related Challenges

The nature of MyMusicList demands that the app must have a way of accessing up-to-date information on music. The idea of having either or both users and administrators manually create and store music information for the app's database was out of the question. Instead we decided to use the Spotify API to access music information whenever it is necessary, although this introduces a reliance on having a consistent internet connection to use the app.

Deciding to use the Spotify API introduced a number of challenges. Some of those challenges included learning how to use the API, and installing and learning additional software. When testing the API, exhaustive tests had to be performed to understand the general structure of the JSON Objects returned by the API's endpoints. The process of performing these tests and studying the documentation for the Spotify API would help form a foundation for the Spotify API Translator class.

User Interface Challenges

The usage of a GUI was required to develop MyMusicList and thus the approach to fulfill this requirement was through the use of JavaFX as well as Java fxml files. This presented a problem due to the group's lack of knowledge regarding the proper usage and operation of such software platforms. During the entire process of development many challenges were faced while incorporating new desired features to the application. One example of those such challenges was facing the predicament of passing information between the controllers to establish proper communication between the view pages. A solution was developed buth such solution contained minor errors which needed modification to properly incorporate into the application. This desired function of communication between controllers presented the problem of information successfully being able to be passed but immediately being erased as the view page was initialized. This soon was later able to be solved which allowed the ability for the application to properly function. The use of tables in the user interface presented problems regarding its appropriate use, this too was also solved and allowed the application to work accordingly.

References

This section outlines a list of references, resources, and tools used to help create MyMusicList.

Language and IDE

MyMusicList was primarily written in Java, specifically Java 8. Parts of the app were written using JavaFX FXML. Our team used a couple of different versions of the Netbeans IDE as our primary coding environment.

Music API Related References

Spotify for Developers Web API and its Documentation:
MyMusicList uses the Spotify API for accessing music information such as artists, tracks, and albums. Documentation for its API can be found at this URL: https://developer.spotify.com/documentation/web-api

cURL:
cURL is a free command line tool used for transferring data with URLs. cURL is used to access information from the Spotify API. More information about cURL can be found at this URL: https://curl.haxx.se/

Description

Product Features

MyMusicList lets its users:

- Search for music.
- Rate music.
- Maintain a list of the music they have rated.

User Characteristics

Users of MyMusicList may have any or all of the following characteristics:

- Enjoy listening to music.
- Wish to maintain a record of the music they have listened to.
- Wish to maintain a record of the music they have rated.

Operating Environment

MyMusicList is a desktop application for computers with a Windows OS. The computer should have a persistent internet connection for MyMusicList to run with all its essential features.

Design and Implementation Constraints

At the highest level, the main constraints in the design and implementation of MyMusicList were that the application was to be written in Java and that it follows MVC architecture standards.

Several of the classes that handle important functionality such as Music API requests and Database requests take a tool-based approach, meaning that there is one class in these domains that are meant to be used as a tool by other developers to make requests. The other classes act as helpers and perform the more complex operations behind the scenes. For example, the group of Music API classes have one class, the adapter, that is able to fulfill all the requests that a developer would want to make such as searching for and loading music objects. A developer can easily learn how to use the adapter without knowing how the translator class works. This approach makes it easier for other team members to use these classes as it eliminates the need to learn how to use several classes in one given domain.

The nature of the requests to the Music API and the Database required the need for custom objects for both music items and records from the database. Requests to the Music API return MusicObjects and lists of MusicObjects. Database requests return lists of or individual RecordObjects including users, ratings, and plan-to-listen records; all custom created.

MyMusicList's use of an external Music API also demands the need for a consistent internet connection on the device the app is used on. As previously mentioned, cURL is used to make these external requests.

Assumptions and Dependencies

Use of MyMusicList assumes the user wants to keep track of all the music they listen to and be able to rate it. As for dependencies, it relies on the Spotify API, cURL 7.68.0, a Java JSON package, and JavaFX classes.

Functional Requirements

This section lists and describes the primary and secondary functions of MyMusicList. The primary functions are features and functionality that are essential to the application. The secondary functions are extra features that are not essential for the main functionality of the application.

Primary Requirements

The following is a list of the main features and functionality of MyMusicList.

Main Features

User Accounts and Profiles: Users can create an account for maintaining their record of the music they have listened to or rated.

User Signup and Login: New users of the app will be able to sign up for a new account and log in to their account from any computer that has MyMusicList installed.

Update User Information: Current users will be able to update their information whenever they see fit. This includes their username, password, email, and their actual music list and individual ratings.

Search for Artists and Music:Users can search for artists and music using a standard search bar and receive a list of music items related to the search.

Rate Artists and Music: Users can add any of the artists and/or the music they search for to their music list and rate it.

Music API Functionality

This section describes the main functions of the Music API that my MyMusicList uses.

Search: The search function performs a search for music objects on the Music API. The search mainly requires a keyword which tells it what to look for. It may also require a string called “type” which tells it whether to look for artists, albums, and/or tracks. The search also accepts a number called “limit” which tells it how many music objects of each type to return.

Load MusicObjects by ID: There are three functions that load ArtistObjects, AlbumObjects, or TrackObjects by their ID. The functions accept an ID and use it to load the corresponding MusicObject from the Music API. The ID of the object is dependent on the Music API being used.

Load AlbumObject and its tracks: This function accepts an existing AlbumObject and reloads it with its tracks. AlbumObjects loaded through other means are not loaded with their associated TrackObjects. If the tracks of an AlbumObject are needed, then this function must be used to load them.

Load Albums of an Artist: This function accepts either an ArtistObject or an artist ID to load the albums of the artist. It also accepts a “limit” number for how many AlbumObjects to load.

Database Functionality

This section describes each of our database classes and their properties.

DatabaseCreate: This class creates records in our datastore. It contains methods for creating a new record in the database, a new user record, a new rating record, and a new plan to listen record.

DatabaseRead: This class reads and returns records using a scanner. It contains methods for reading basic records, users, ratings, and records for items to listen to.

DatabaseUpdate: This class contains all the methods for updating a file in the datastore. It contains methods for updating a user’s password, email, username, rating, and their plan to listen list. It also has a helper method for updating a file.

DatabaseDelete: This class is for deleting records. It contains methods for deleting records in the plan to listen, user rating, and user info files. It also contains a method for displaying the logic used in deleting the files.

DatabaseException: This is a class that inherits from another class called the Exception class. It throws exception messages for database operations.

DatabaseInterface: This class holds all the methods for the database classes. It also creates files for user info, user login, user rating, and plan to listen.

Secondary Requirements

MyMusicList does not have any implemented secondary features. All of the application’s features in its 1.0 version are primary. Regardless, here is a list of some features that could be added as extras for possible future updates or versions.

- **More Music Lists**: A feature MyMusicList could benefit from is the addition of more MusicLists for individual users. Either by letting users create and maintain as many custom lists as they want or by defining and supporting more specialized lists for everyone.

- **Display Images and Art**: MyMusicList could have support for displaying images such as album art. The addition of this feature would require an overhaul to the look of the GUI but it could make MyMusicList more visually appealing.

Technical Requirements

This section describes all of the technical requirements of MyMusicList, from the different types of operating systems and software being utilised, to the GUI for user interface.

Operating Systems / Compatibility

MyMusicList can run on any modern desktop operating system with Java 8 installed. Testing could not be done on Apple or Linux OS to ensure compatibility although applications written in Java should be platform independent.

Interface Requirements

The user, hardware, software, and communications interface requirements are described below.

User Interface

MyMusicList requires the use of Java fxml files which are utilized by the software platform JavaFX. To properly operate the MyMusicList application the use of both of these items is an absolute necessity, failure to do so results in the inability to either launch and or use the application whatsoever. MyMusicList also requires all the Java fxml files to be located under the Views file in the application with the known name of the file to be saved in all of the controllers which will access the file along with all the needed methods or variables for the fxml file saved under such controller(s).

Hardware Interface

MyMusicList is a desktop application that accepts input from a standard mouse and keyword.

Software Interface

MyMusicList uses the following additional software:

Spotify Web API: MyMusicList relies on the Spotify Web API to get music information.

cURL 7.68.0 or later: cURL is a command-line tool used to transfer data with URLs. The Spotify API is accessed by executing cURL commands in the Command Prompt.

Communications Interface

MyMusicList communicates with the Spotify Web API for requests of music information.

This app supports all types of web browsers or internet browsing software.

Nonfunctional Requirements

This section describes all of the nonfunctional requirements for MyMusicList. This includes all of the software quality attributes, cost and delivery date, performance requirements, and safety recovery requirements.

Performance Requirements

MyMusicList is expected to have little to no delays upon opening the application, as well as, in between requesting queries through the Spotify API and displaying the search results. Because MyMusicList is a desktop application that does not connect to a server for database functionality, users should have no issues in saving records of their ratings.

Safety Recovery Requirements

In the event that a user has forgotten their login information, the information along with the email address that they have provided upon creating their account on MyMusicList will allow them to recover their account information and reset their password. Note: Password reset through email was not implemented for the 1.0 version of MyMusicList.

Policy Requirements

In accordance with the UNCG Academic Honor Code, MyMusicList and the required documents were of our own and met the standards of the Final Project Requirements.

Software Quality Attributes

The following is a list of the quality attributes of the MyMusicList software.

Availability

MyMusicList will be available for download in the following Git repository: https://github.com/Ada-s-Firewall/Code_Repository. Any music and artists that can be found on Spotify will be available to add to a user's list and be rated.

Correctness

MyMusicList will perform all of its functions correctly and without any errors.

The Music API of MyMusicList will operate correctly and return the appropriate music objects with the correct information.

Maintainability

MyMusicList will require regular or occasional maintenance to ensure continuous functionality of its Music API operations. If changes are made to how Spotify handles external requests or how it organizes JSON objects, then those changes will require modifications to the SpotifyAPITranslator class.

Reusability

The Music API and MusicObject classes may be reused for other projects and applications that are to feature similar functionality to the Music API functions of MyMusicList. All that would be required in addition to those classes is a JSON package and for cURL 7.68.0 or later to be installed on the device.
(This refers to how parts of the code we wrote can be reused if at all possible)

Portability

MyMusicList is available for use on any Windows PC. However, testing was not done on Mac, Linux, or any other OS to ensure compatibility.

Process Requirements

Methodology Used

The methodology that we used to create this program is the Agile development method. A large emphasis was placed on prototyping and writing code early on versus prior organization and planning.

Time Constraints

There were several time constraints we had to acknowledge when we were in development. Due to unexpected time constraints including the COVID-19 pandemic in early 2020 and some implementation issues, we weren't able to implement everything we originally planned for MyMusicList.

Cost and Delivery Date

MyMusicList was developed as an open-source application, primarily for fulfilling a project for a Software Engineering class. As such, MyMusicList will be available to download for free at our GitHub repository: https://github.com/Ada-s-Firewall/Code_Repository.

The 1.0 version of MyMusicList is expected to be released on May 1, 2020.