CISC 699 Final Applied Project

Software Requirements Specification

Title: MusicOn, a music player emulator

Professor: Abrar Qureshi

Author: Prafulla Chandra Munugoti

Term: Summer 2023

Date: 06/04/2023

➢ Problem Statement

Throughout human history, audio has played a vital role in our everyday lives. We are constantly exposed to different forms of audio in all aspects of our lives. One such form of audio that is particularly important is music. Music is a powerful tool that can be used to express our feelings and emotions. It can also provide a much-needed break from the hardships of life. For many people, music is an important way to relieve stress. It can also act as a form of therapy, helping us to relax and feel calm. On the other hand, music can also be a source of joy and happiness, lifting our spirits and boosting our morale. In addition to its emotional and psychological benefits, music can also have a positive impact on our physical health. Studies have shown that listening to music can lower blood pressure, reduce heart rate, and improve sleep quality. Music can also help to improve our cognitive function and memory.

Overall, music is a versatile and powerful tool that can be used to improve our lives in many ways. It is a form of communication, a source of entertainment, and a way to promote physical and mental health. Owing to many advantages of listening to music for humans from every walk of life during various situations they are in. MusicOn is one such application that would allow a seamless, robust, freeware customizable music player emulator. It allows users to listen to the audio and immerse in their own world by reducing their day-to-day worries and hurdles. This emulator is to help them immerse by enabling them to play music and digital audio files, creating a playlist by using a backend database to store, add, modify, and remove information on the existing playlists. The MusicOn emulator application's graphical user interface opens with various functionalities like playing a song, creating, and displaying a playlist, pausing, and resuming a long song, and change the song, play the previous or next song using a user-friendly command line user interface with each option available to pick as needed. In this project we are bringing not just music, we are adding two more categories of sound forms and some more functionalities to an existing category of Music. It includes providing a music mix based on recently played songs, and recommendations based on artist, genre, and album.
In the two new categories, we are MUSICON is not just software that runs only mp3 or audio files, it would allow an addition to pick AM/FM stations based on a city, and frequency after picking the location. Users can select any station based on category (sports/news/traffic). It would allow users to select their favorite podcasts from the available database, select by podcasters, and select by category (investment, education, meditation, sleep, etc.).

➢ System Requirements

The required system functions for the MusicOn a music player emulator product includes,

Music:

- Add song to songs DB.
- Delete song from songs DB.
- Show songs from songs DB.
- Search songs by artist/Title/Year/Type from songs DB.
- Play a Song stored in the songs DB.
- Create a playlist to playlists DB.
- Show all playlists in playlists DB.
- Add songs to an existing playlist.
- Play songs in selected playlists in 1). Normal Mode 2). Shuffle Mode 3). Loop Mode 4. Single song Loop Mode
- Provide music mixes to users based on their past usage of the application to play songs.
- Provide music recommendations to users based on the artist.
- Provide music recommendations to users based on genre.
- Provide music recommendations to users based on the album.
- Add support to run just a single song in loop mode.

AM/FM:

- Users can select the FM/AM based on location.
- Users can select the FM/AM based on frequency after picking the location.
- Users can select any station based on category (sports/news/traffic)

Podcasts:

- Users can select their favorite podcasts from the available database.
- Users can select podcasters.
- Users can select by category (investment, education, meditation, sleep, etc.).

As a music player emulator, the interaction between the user and the service will be done through the Command Line Interface options.

➢ Functional Requirements

Requirement	ID	Name	Description
FR1 Song Functions	FR1.1	Addition	Functionality to add songs to the existing songs database
	FR1.2	Deletion	Functionality to delete songs from the songs database
	FR1.3	Display	Functionality to display songs in the existing songs database
	FR1.4	Search by Title	Functionality to search for the available songs in songs database
	FR1.5	Search by Artist	Functionality to search for available songs in songs database by artist name
	FR1.6	Search by Album	Functionality to search for available songs in songs database by album name
	FR1.7.1	Play Song Normal Mode	Functionality to play a song in normal mode displaying lyrics
	FR1.7.2	Play Song Shuffle Mode	Functionality to play a song in shuffle mode displaying lyrics
	FR1.7.3	Play Song Loop Mode	Functionality to play a song in loop mode displaying lyrics
	FR1.8	Sort Songs alphabetically	Functionality to sort the songs in alphabetical order
	FR1.7.4	Play Single Song Loop Mode	Functionality to play a select song in an infinite loop with displaying lyrics.

FR2 Playlist Functions	FR2.1	Creation	Functionality to create a new playlist for users.
	FR2.2	Deletion	Functionality to delete an existing playlist from playlists database
	FR2.3	Display	Functionality to show all existing playlist from playlists database
	FR2.4	Song Addition to Playlist	Functionality to add songs to existing playlist from playlists database
	FR2.5	Song Deletion from Playlist	Functionality to delete songs to existing playlist from playlists database
	FR2.6	Search Playlist by Title	Functionality to search for the existing playlist by title from existing playlists database.
	FR2.7.1	Play Songs Normal Mode	Functionality to play all songs from selected playlist in normal mode displaying lyrics
	FR2.7.2	Play Songs Shuffle Mode	Functionality to play all songs from selected playlist in shuffle mode displaying lyrics
	FR2.7.3	Play Songs Loop Mode	Functionality to play all songs from selected playlist in loop mode displaying lyrics

FR3 AM/FM Functions	F3.1	Creation	Functionality to add new FM stations to a city.
	FR3.2	Deletion	Functionality to delete an existing FM station from FM stations database
	FR3.3	Display	Functionality to show all existing FM stations from stations database
	FR3.4	Search station by frequency	Functionality to search for the existing stations by frequency from existing stations database.
	FR3.5	Search station by category	Functionality to search for the existing stations by category from existing stations database.
	FR3.6	Play an FM station	Functionality to play selected FM from existing stations database.
	FR3.7	Skip to next FM station	Functionality to play next frequency FM from existing stations database.
	FR3.8	Skip to previous FM station	Functionality to play previous frequency FM from existing stations database.

FR4 Podcast Functions	F4.1	Creation	Functionality to add new podcasts to Database.
	FR4.2	Deletion	Functionality to delete an existing podcast from FM podcasts database
	FR4.3	Display	Functionality to show all existing podcasts from podcasts database
	FR4.4	Search podcasts by podcaster	Functionality to search for the podcasts by podcaster from existing podcasts database.
	FR4.5	Search podcasts by category	Functionality to search for the podcasts by category from existing podcasts database.
	FR4.6	Play podcast	Functionality to play selected podcasts from existing podcasts database.
	FR4.7	Skip to next podcast	Functionality to play the next podcast from the existing podcast database.
	FR4.8	Skip to previous podcast	Functionality to play previous frequency FM from existing podcast database.

FR5 Recommend Functions	F5.1	Creation of Music Mixes	Functionality to create temporary music mixes from Database.
	FR5.2	Next song by artist	Functionality to recommend the next song by an artist available in the songs database.
	FR5.3	Next song by genre	Functionality to recommend the next song by genre of the song in the songs database.
	FR5.4	Next song by album	Functionality to recommend the next song in album available in the songs database.

➢ Non-Functional Requirements

Requirement	Req.ID	Name	Description
NFR1 Performance	NFR1.1	Robustness	System is robust to deal and act upon common error scenarios like unavailable metadata, unsupported file types, data missing, compile issues.
	NFR1.2	Quickness	System is quick enough to play music and response time to be as minimal as possible to any of the user action without any time kill for loading, creating, playing buffering to have a pleasant experience.
	NFR1.3	Quick Recovery	In case of failures, it should be able to fail or recover quickly.
NFR2 Quality	NFR2.1	Memory Management	System should ensure that there is no memory leak for any of its functions.
	NFR2.2	Compatibility	The system is compatible with any technology, platform, and software that it comes in contact with in terms of API.
	NFR2.3	Error Handling	System should not cause or trigger any events that creates an unrecoverable state.
	NFR2.4	Exception handling	System should be capable of handling any exceptions that occur in the way of usage.
NFR3 Reliability	NFR3.1	Reliable	System needs to be reliable on its performance. example: play functionality should take same time for shuffle, or loop or do another operation.
NFR4 Consistency	NFR4.1	Consistent behavior	System should have a consistent behavior on its functionalities and should not behave differently to different users for same function.
NFR5 Security	NFR5.1	Safety	System needs to strongly secure user data and prevent them from theft or data leak and not lose confidential information
NFR6 Availability	NFR6.1	Available	System should be available for multiple users at the same time and should support the feature and not crash.
NFR7 Usability	NFR7.1	Usage	System is easy to operate and friendly to users. User interaction should be as error prone as possible.
NFR8 Scalability	NFR8.1	Scalable	System must be easy to scale depends on the traffic volume which helps in cost of resource allocation.
NFR9 Maintainability	NFR9.1	Maintenance	system need to be easy to maintain, not so complicated to add new features. System is breaking due to new code changes; it should be a quicky recovery back to using previous versions.

➢ Use cases

1.1 Library

Use Case ID	001
Use Case Name	Library
Actors	Prafulla Chandra Munugoti (developer)
Description	This use case is for library level usage, the library is a set of all songs that are liked/added, a super set of all play lists. The player can add, delete, and search by song/artist from the library.
Preconditions	Python 3 installed Necessary Python packages installed
Postconditions	NA
Normal Flow	• Start the tool • Login as a user • Add a song to the library • Display songs in the library • Add 10+ songs into the list • Display songs in the library • Search a song using song title • Play the searched song • Search songs by artist name • Play the searched song • Delete a song from the library • Delete all songs from the library
Alternative Flows	• Search a song that does not exist • Search an artist that does not exist
Exception Flows	• Delete a song that does not exist

1.2 Playlist

Use Case ID	002

Use Case Name	Playlist
Actors	Prafulla Chandra Munugoti (developer)
Description	This use case will be mainly for the play list that can be created, deleted, and store the songs and could be played in 3 different modes.
Preconditions	Python 3 installed Necessary Python packages installed
Postconditions	NA
Normal Flow	• Start the tool • Login as user • Create a play list • Add one song to the play list • Play the song from the list • Add 10+ music into the list • Play the list in normal mode • Play the list in shuffle mode • Play the list in loop mode • Delete a song from the play list • Delete the play list
Alternative Flows	• Play the song from outside the list
Exception Flows	• Play an empty list • Add one song to a list that does not create • Delete a song that does not exist

1.3 FM

Use Case ID	003
Use Case Name	FM station
Actors	Prafulla Chandra Munugoti (developer)
Description	This use case is for FM library level usage, the FM library is a set of all stations that are added, a superset of all stations in a location by frequency, and topic. The player can add, delete, and search by frequency/topic from the FM stations library.

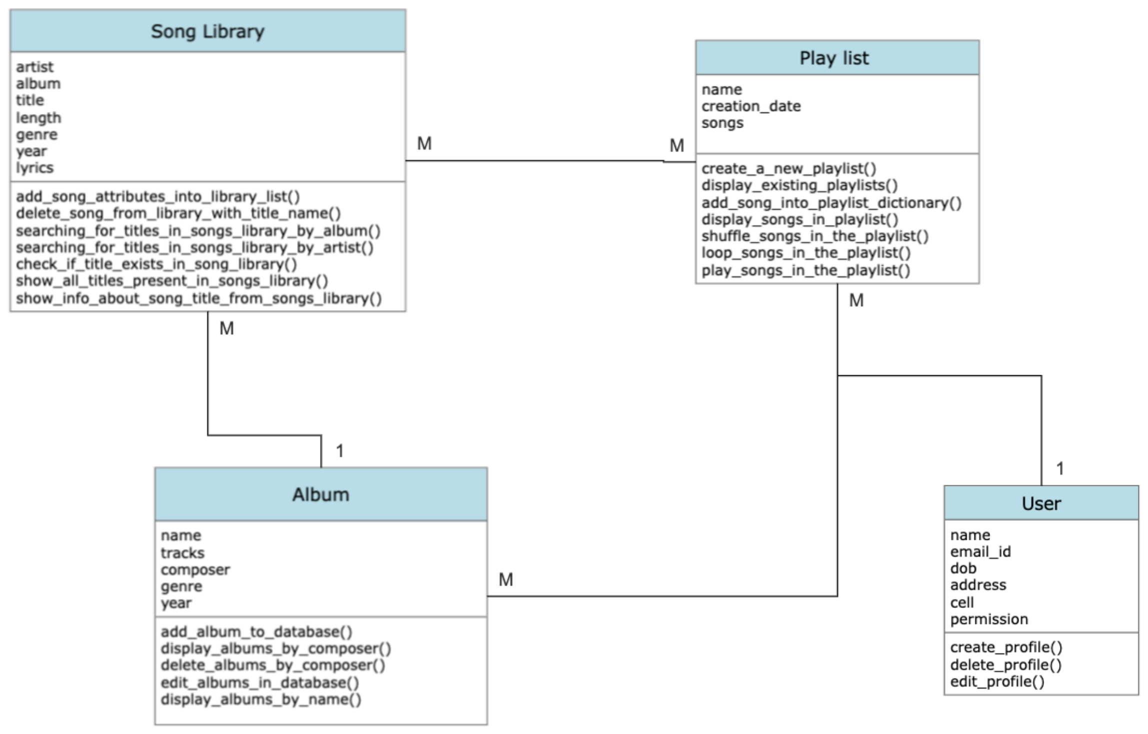
Preconditions	Python 3 installed Necessary Python packages installed
Postconditions	NA
Normal Flow	• Start the tool. • Login as a user • Add a station to the station's library. • Display stations in the station's library • Search a station using station frequency. • Play the searched station. • Search stations by topics • Play the searched station. • Delete a station from the library. • Delete all stations from the library.
Alternative Flows	• Search a station that does not exist. • Search a topic station that does not exist
Exception Flows	• Delete a topic station that does not exist. • Delete a frequency station that does not exist.

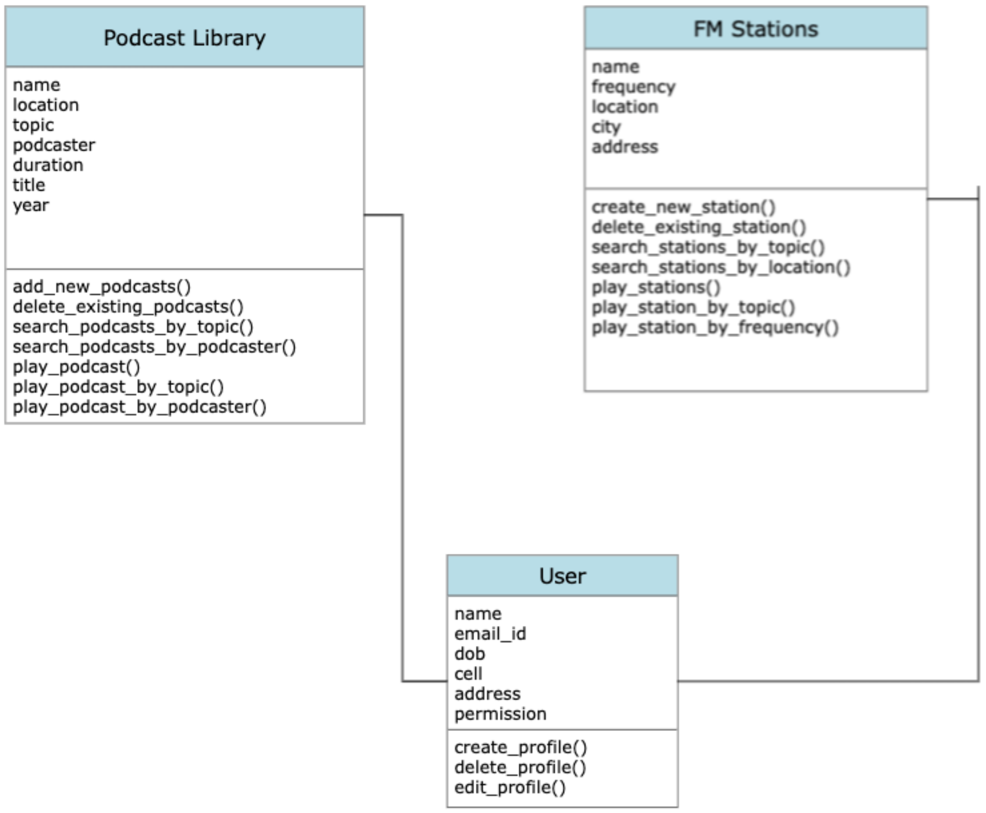
1.3 Podcasts

Use Case ID	004
Use Case Name	Podcasts
Actors	Prafulla Chandra Munugoti (developer)
Description	This use case is for podcast selections from a set of all podcasts that are added, a superset of all stations in a location by podcaster, and a topic. The player can add, delete, and search by podcaster/topics from the podcasters database.
Preconditions	Python 3 installed Necessary Python packages installed
Postconditions	NA
Normal Flow	• Start the tool. • Login as a user • Add a podcast to the podcast's library. • Display stations in the podcast's library • Search a podcast using podcast names.

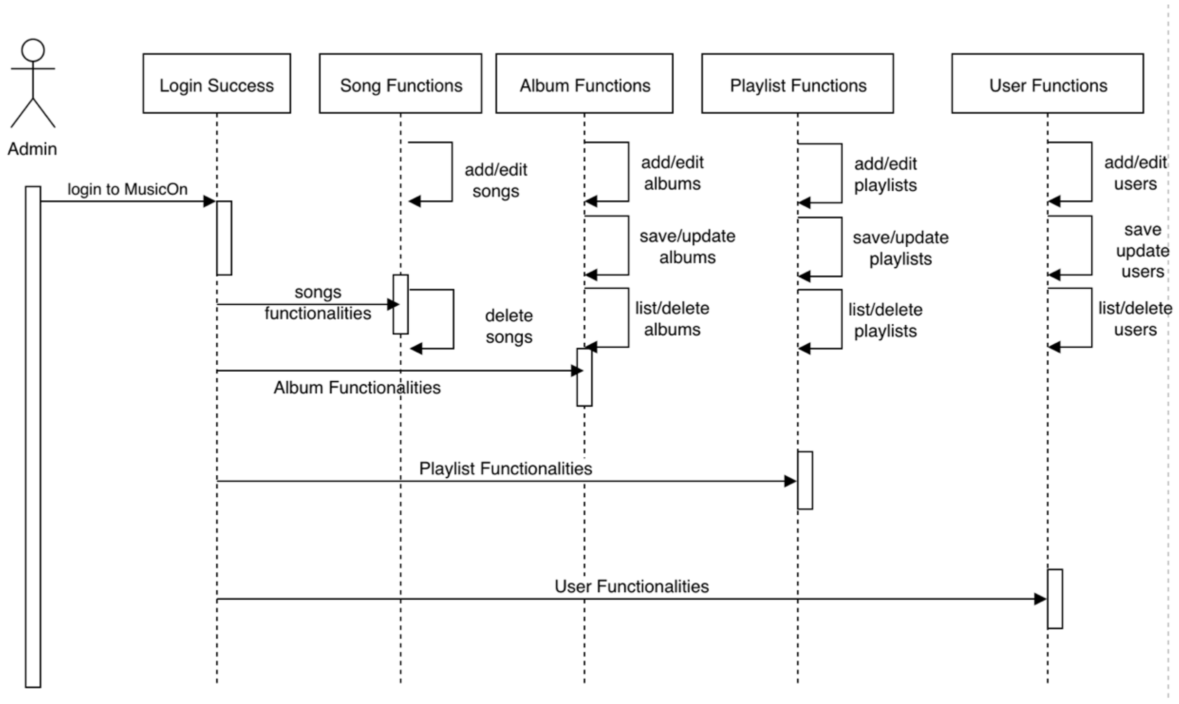
	• Play the searched podcast. • Search podcasts by topics. • Play the searched podcast. • Delete a podcast from the library. • Delete all podcasts from the library.
Alternative Flows	• Search a podcast that does not exist. • Search a topic podcast that does not exist
Exception Flows	• Delete a topic podcast that does not exist. • Delete a podcast that does not exist.

➢ Class Diagram

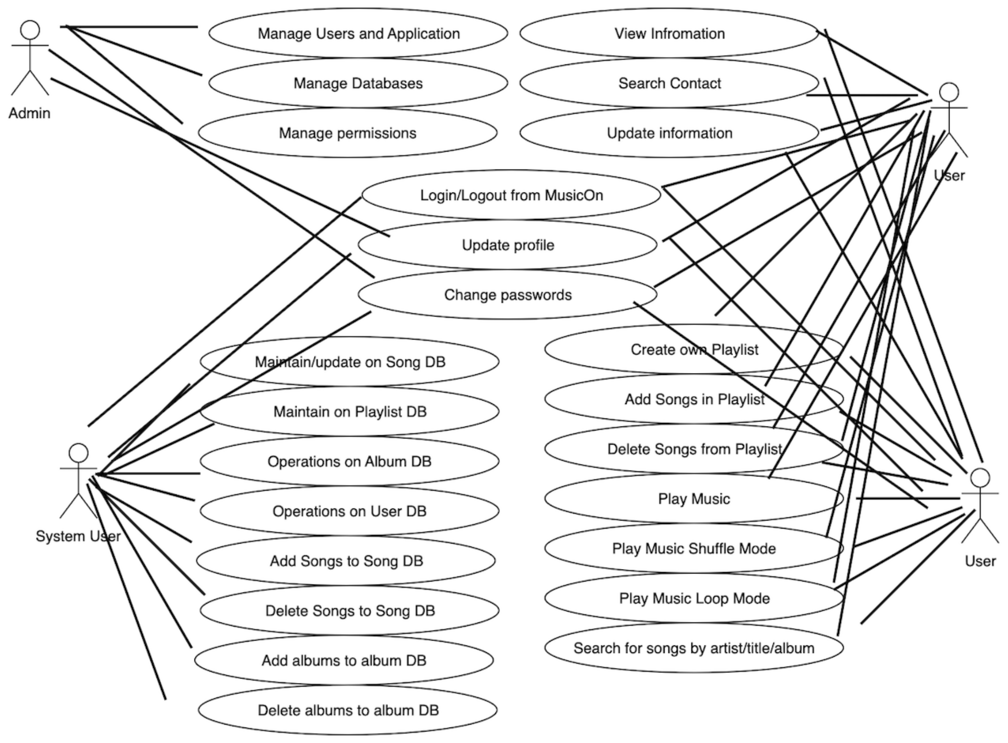

Song Library
artist
album
title
length
genre
year
lyrics
add_song_attributes_into_library_list()
delete_song_from_library_with_title_name()
searching_for_titles_in_songs_library_by_album()
searching_for_titles_in_songs_library_by_artist()
check_if_title_exists_in_song_library()
show_all_titles_present_in_songs_library()
show_info_about_song_title_from_songs_library()
Play list
name
creation_date
songs
create_a_new_playlist()
display_existing_playlists()
add_song_into_playlist_dictionary()
display_songs_in_playlist()
shuffle_songs_in_the_playlist()
loop_songs_in_the_playlist()
play_songs_in_the_playlist()
Album
name
tracks
composer
genre
year
add_album_to_database()
display_albums_by_composer()
delete_albums_by_composer()
edit_albums_in_database()
display_albums_by_name()
User
name
email_id
dob
address
cell
permission
create_profile()
delete_profile()
edit_profile()
M
M
M
1
M
1
M

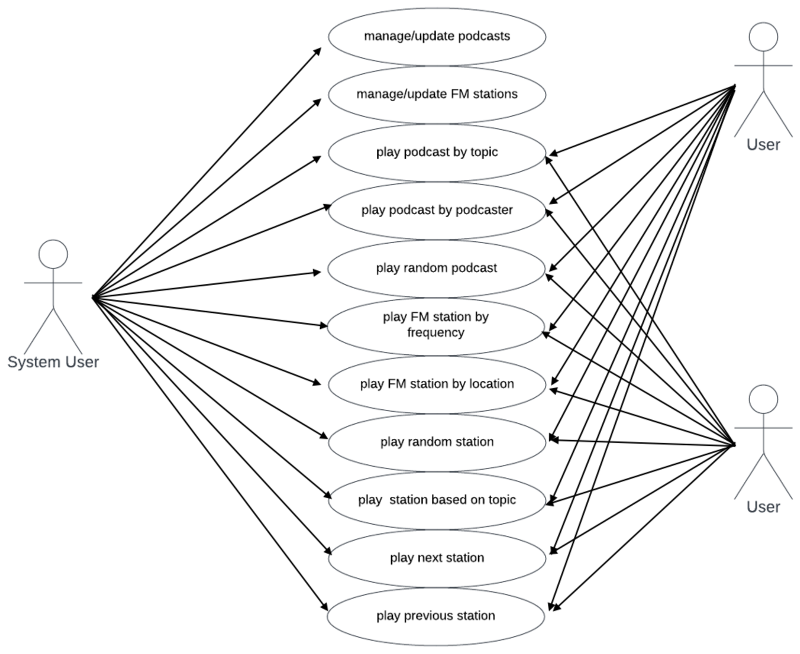


➢ Sequence Diagram:



➢ Use Case Diagram:


Admin
Manage Users and Application
Manage Databases
Manage permissions
View Infromation
Search Contact
Update information
User
Login/Logout from MusicOn
Update profile
Change passwords
Maintain/update on Song DB
Maintain on Playlist DB
Operations on Album DB
Operations on User DB
Add Songs to Song DB
Delete Songs to Song DB
Add albums to album DB
Delete albums to album DB
System User
Create own Playlist
Add Songs in Playlist
Delete Songs from Playlist
Play Music
Play Music Shuffle Mode
Play Music Loop Mode
Search for songs by artist/title/album
User


manage/update podcasts
manage/update FM stations
play podcast by topic
play podcast by podcaster
play random podcast
play FM station by frequency
play FM station by location
play random station
play station based on topic
play next station
play previous station
System User
User
User