Fontys University Of Applied Sciences

MusicMaven

Web Application

User Requirements Specification (URS)

*Software Engineering*

**Author:**

Martin Hristev

*Version: 0.1*

Eindhoven

**2 0 2 3**

[**1. Agreements with client 3**](#_gnt5y06kfry2)

[**2. Functional requirements 3**](#_12mnk9bq6ebt)

[2.1 Web application 3](#_nh1ukd12r3t7)

[2.2 Desktop Application 4](#_55z3dh5lips0)

[2.3 MoSCoW 5](#_d8avwcf8rj60)

[**3. Use cases 5**](#_st5z5xrwafkv)

[3.1 UC-01 5](#_10mwxsn3vv24)

[3.2 UC-02 6](#_68q466ydhbm8)

[3.3 UC-03 6](#_4nqe2oaazj18)

[3.4 UC-04 7](#_i8feoutoiaqd)

[3.5 UC-05 8](#_qykpui35ac4w)

[3.6 UC-06 8](#_haj4kveus11p)

[3.7 UC-07 9](#_85qpmliyq36l)

[3.8 UC-08 9](#_qxftfurwtgnd)

[3.9 UC-09 10](#_q3aoq8uq3gea)

[3.9 UC-10 10](#_sgzl48sz7k64)

[3.11 UC-11 11](#_get0d2osl411)

[**4. GUI 11**](#_77k35xqeukya)

[4.1 Login 11](#_6xrl474yvkwo)

[4.2 Music Section 12](#_auzvhihzytet)

[4.3 Artist - More Info 12](#_3qrka01jhun5)

[4.4 Album - More Info 13](#_gjtil9lhlrf1)

[4.5 Create An Album 13](#_8pvgkpr2xa80)

[4.6 Users Section 14](#_ew21jove7234)

[4.7 Users Section 14](#_gnvzla6qjdfd)

[**5. Website wireframe 15**](#_z5lla4h9mr09)

[5.1 Home 15](#_a49czeglnpdi)

[5.2 Top 100 16](#_xgjsray2wtsd)

[5.3 All Reviews 17](#_a4tu90maq6hx)

[5.4 Single Review Page 18](#_lbmg8sy3tqgd)

[5.5 Single Album Page 19](#_i3z350bnewn8)

[5.6 Single User Page 20](#_op4p5lul4fld)

# 1. Agreements with client

* The MusicMaven platform will be developed in accordance with the specifications outlined in the project plan and URS document.
* Throughout the development process, the development team will provide regular updates on project progress and milestones, ensuring that the client is kept informed of all developments. Any feedback or suggestions from the client will be incorporated into the project, ensuring that the final product meets all requirements and specifications.
* The development team will focus on completing the basic requirements of the platform, which include the ability for users to write reviews about artists, songs, and albums. Feedback from tutors will be incorporated into the project at regular intervals.
* If there is sufficient time and resources available, the development team will also integrate the Spotify API to allow users to generate a random playlist based on their input.
* Furthermore, MusicMaven will have a desktop application that will be used mainly for administration purposes. This application will enable the client to manage different users and data in the database.

# 2. Functional requirements

## 2.1 Web application

**FR-01**: *User Login*

*Constraint*: User can log in using email and password.

**FR-02**: *User Registration*

*Constraint*: User can create a new account by providing a valid email address, username, and password.

**FR-03**: *Personalized Review System*

*Constraint*: Users can rate on albums, songs, and artists from 1 to 10 and share their thoughts.

**FR-04:** *Follow Other Users*

Description: Users can follow other users and stay updated with their latest reviews and recommendations.

**FR-05**: *Join The Conversation*

Description: User can like reviews to engage with the community.

**FR-06**: *Discover New Music*

Description: Users can explore a list of popular reviews and recommendations from the community.

Constraint: It is available even without logged in user.

**FR-07**: *Spotify API Integration*

Description: User can generate a random playlist based on their input using the Spotify API.

Constraint: The user should have a valid Spotify account.

**FR-08:** *Search for Artist, Song, Album, or Other Users*

Description: Users can search for specific artists, songs, albums, or other users by entering keywords in a search bar.

## 2.2 Desktop Application

**FR-09**: *Create Users, Albums, Artists and Songs*

Administrators can create different users, albums, artists, and songs to the database via the desktop application.

**FR-10**: *Delete Users, Albums, Artists and Songs*

Administrators can delete users, albums, artists, and songs to the database via the desktop application.

**FR-11:** *Administration Login*

The desktop application will only be accessible to administrators who have valid login credentials.

## 2.3 MoSCoW

| **Must** | **Should** | **Could** | **Won’t** |
| --- | --- | --- | --- |
| User Login (**FR-01**) | Follow Other Users (**FR-04**) | Spotify API Integration (**FR-07**) | Allow music streaming |
| User Registration (**FR-02**) | Join The Conversation (**FR-05**) | Account confirmation after sign up | Have a feature for data analysis |
| Personalized Review System (**FR-03**) | Search for Artist, Song, Album, or Other Users (**FR-08**) | Forget password feature | Have a reporting system |
| Discover New Music (**FR-06**) |  |  |  |
| Create Users, Albums, Artists and Songs (**FR-09**) |  |  |  |
| Delete Users, Albums, Artists and Songs (**FR-10**) |  |  |  |
| Administration Login (**FR-11**) |  |  |  |

# 3. Use cases

## 3.1 UC-01

| **ID**: UC-01  **Functional Requirement**: FR-01 (*User Login)*  **Actor**: User  **Description**: User Authentication  **Trigger**: Client wants to log in to MusicMaven platform  **Pre-conditions**:   * The actor has a registered account on the MusicMaven platform   **Post-conditions**:   * The actor is successfully authenticated and gains access to their account   **Main Success Scenario**:   1. The actor enters their registered email and password. 2. The system validates the actor’s credentials and logs in the actor.   **Extensions**:   * 3a. If the actor enters an invalid email or password, the system displays an error message and prompts the actor to re-enter their credentials. |
| --- |

## 3.2 UC-02

| **ID**: UC-02  **Functional Requirement**: FR-02 (*User Registration*)  **Actor**: User  **Description**: Creating a new account on the MusicMaven platform.  **Trigger**: Client wants to create a new account on the MusicMaven platform.  **Pre-conditions**:   * Client has internet access. * Client is not already logged in to the platform.   **Post-conditions**:   * Client has successfully created a new account on the platform.   **Main Success Scenario**:   1. Client enters their email, username and password and confirms. 2. The platform verifies that the email address is valid and not already in use and creates a new account. 3. Client is redirected to the login page.   **Extensions**:  2a. If the email address is not valid, the platform displays an error message and prompts the client to enter a valid email address.  2b. If the email address is already in use, the platform displays an error message and prompts the client to enter a different email address.  2c. f the password does not cover the basic rules for strong password, the platform displays an error message and prompts the client to enter a valid one. |
| --- |

## 3.3 UC-03

| **ID**: UC-03  **Functional Requirement**: FR-03 (*Personalized Review System*)  **Actor**: User  **Description:** Actor can rate and share their thoughts on albums, songs, and artists.  **Trigger:** The actor wants to rate and share their thoughts on an album, song, or artist.  **Pre-Condition**: The actor must be authenticated and logged in to the MusicMaven platform.  **Post-Condition**: The actor’s rating and review are stored in the MusicMaven database.  **Main Success Scenario**:   1. The actor navigates to the page of the album, song, or artist they want to review and starts writing a review. 2. The system displays a review form. 3. The actor enters their rating and writes their review in the form and submits. 4. The system stores the actor’s rating and review in the database.   **Extensions**:  2a. The actor is not logged in.   * The system redirects the actor to the login page. * The actor logs in with their credentials. * The system redirects the user back to the album, song, or artist page. |
| --- |

## 3.4 UC-04

| **ID**: UC-04  **Functional Requirement**: FR-04 (*Follow other users*)  **Actor**: User  **Description**: The actor wants to follow other users to stay updated with their latest reviews and recommendations.  **Trigger**: The actor clicks on the "Follow" button next to another user's profile.  **Pre-Condition**: The actor must be logged in and have an active account.  **Post-Condition**: The actor is now following the selected user.  **Main Success Scenario**:   1. The user navigates to the profile of the user they want to follow and “follow” them. 2. The system adds the selected user to the list of users the client is following.   **Extensions**:  1a. If the user does not have an active account, they will be redirected to the registration page.  1b. If the selected user has already been followed, “Unfollow” button will appear. |
| --- |

## 

## 3.5 UC-05

| **ID**: UC-05  **Functional Requirement**: FR-05 (*Join The Conversation)*  **Actor**: User  **Description**: Actor wants to join the conversation by liking reviews.  **Trigger**: The actor wants to engage with the community by liking a review.  **Pre-conditions**: The actor is logged in to the platform.  **Post-conditions**: The actor has successfully liked a review, and the review has been updated.  **Main Success Scenario**:   1. The actor navigates to the review they want to interact with and likes it. 2. The system updates the review with the actor's "like".   **Extensions**:  2a. If the actor has already liked the review, the "like" button will lead to removing the like. |
| --- |

## 3.6 UC-06

| **ID**: UC-06  **Functional Requirement**: FR-06 (*Discover New Music*)  **Actor**: User  **Description**: Actor can discover new music through the platform without logging in.  **Trigger**: Actor wants to discover new music.  **Pre-Condition**: Actor has access to the MusicMaven platform.  **Post-Condition**: Actor can view popular reviews and recommendations even without logging in.  **Main Success Scenario**:   1. Actor opens the MusicMaven platform on their web browser. and starts reading reviews   **Extensions**:  None. |
| --- |

## 3.7 UC-07

| **ID**: UC-07  **Functional Requirement**: FR-07 (*Spotify API Integration*)  **Actor**: User  **Description**: Actor can take to generate a random playlist using the Spotify API.  **Trigger**: Actor selects the "Generate Playlist" option from the platform.  **Pre-condition**: Actor is logged in to the platform and has provided the required input for generating a playlist.  **Post-condition**: A random playlist is generated.  **Main Success Scenario**:   1. Actor provides input for the playlist, such as genre, mood, or artist. 2. System sends the input to the Spotify API.   **Extensions**:  1a. If the actor does not provide the required input, the platform displays an error message.  2a. If the connection to the Spotify API fails, the platform displays an error message and prompts the user to try again later. |
| --- |

## 3.8 UC-08

| **ID**: UC-08  **Functional Requirement**: FR-08 (*Search for Artist, Song, Album, or Other Users)*  **Actor**: User  **Description**: The actor can search for specific song, artist, album or user.  **Trigger**: Actor enters keywords into the search bar and clicks the search button.  **Pre-condition**: The actor is logged in to their account.  **Post-condition**: The actor is directed to the page corresponding to the selected search result.  **Main Success Scenario**:   1. The actor clicks on the search bar and enters keywords. 2. The system displays a list of results matching the entered keywords. 3. The actor selects the desired result and is directed to the corresponding page.   **Extensions**:  2a. If no results are found, the system displays a message indicating that there are no matching results. |
| --- |

## 3.9 UC-09

| **ID**: UC-09  **Functional Requirement**: FR-09 (*Create Users, Albums, Artists and Songs)*  **Actor**: Administrator  **Trigger**: The actor launches the MusicMaven desktop application.  **Pre-Condition**: The actor has valid login credentials to access the desktop application.  **Post-Condition**: The actor has successfully created a user or new music content to the database.  **Main Success Scenario**:   1. The actor launches the MusicMaven desktop application. 2. The system prompts the actor to provide login credentials. 3. The actor enters their valid login credentials. 4. The system validates the credentials and grants access to the dashboard. 5. The actor can manage creates new users or music content. 6. The system updates the database.   **Extensions**:  2a. If the login credentials are invalid, the application displays an error message. |
| --- |

## 3.9 UC-10

| **ID**: UC-10  **Functional Requirement**: FR-10 (*Delete Users, Albums, Artists and Songs)*  **Actor**: Administrator  **Trigger**: The actor launches the MusicMaven desktop application.  **Pre-Condition**: The actor has valid login credentials to access the desktop application.  **Post-Condition**: The actor has successfully deleted a user or new music content to the database.  **Main Success Scenario**:   1. The actor launches the MusicMaven desktop application. 2. The system prompts the actor to provide login credentials. 3. The actor enters their valid login credentials. 4. The system validates the credentials and grants access to the dashboard. 5. The actor can manage delete users or music content. 6. The system updates the database.   **Extensions**:  2a. If the login credentials are invalid, the application displays an error message. |
| --- |

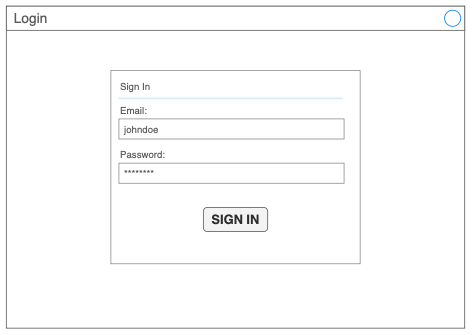
## 3.11 UC-11

| **ID**: UC-11  **Actor**: Administrator  **Functional Requirement**: FR-11 (*Administration Login*)  **Description**: Authentication for administrators who want to access the desktop application.  **Trigger**: The actor wants to access the desktop application.  **Pre-condition**: The actor has valid login credentials.  **Post-condition**: The actor is granted access to the desktop application if their credentials are valid.  **Main Success Scenario**:   1. The actor opens the desktop application. 2. The system displays the login screen. 3. The actor enters credentials. 4. The system verifies the actor's credentials and grants access to the actor.   **Extensions**:  4a. If the credentials are not recognized, the application displays an error message and prompts the actor to try again. |
| --- |

# 4. GUI

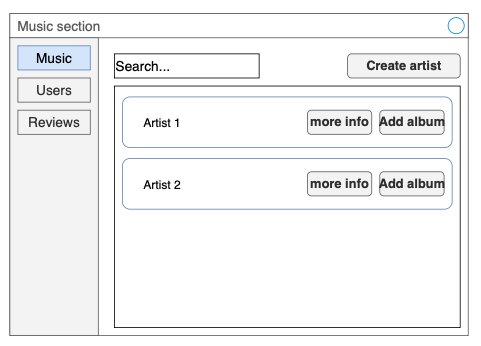
In this section, the graphical user interface of the desktop app will be presented.

## 4.1 Login

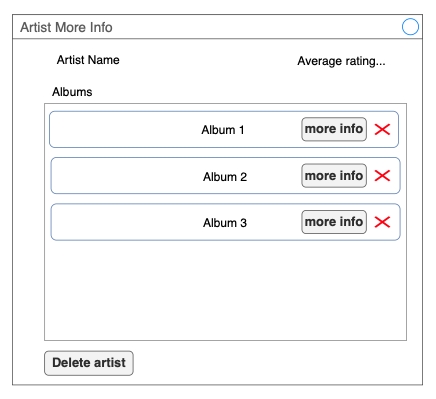
User authenticates with email and password.

## 4.2 Music Section

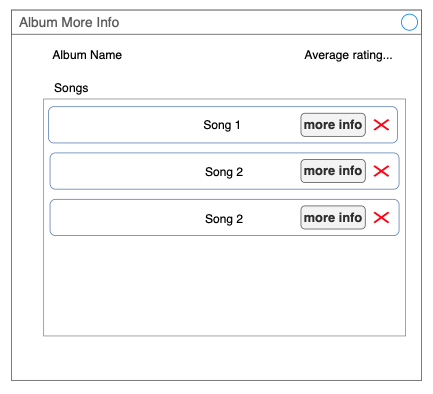
The "Music" section in the side menu enables the administrator to manage and explore the music database. It provides options to search for specific artists, create new ones, add new albums to an existing artist, and retrieve more detailed information about a selected artist.



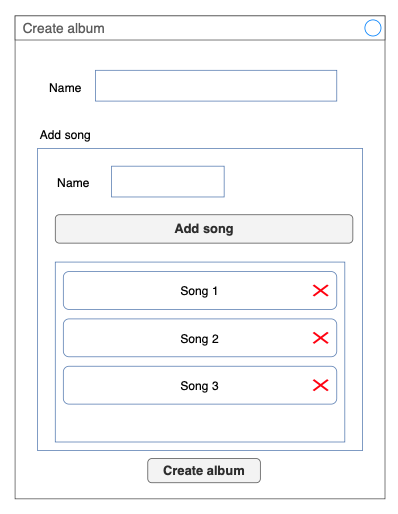
## 4.3 Artist - More Info

Clicking "More Info" in the Music section reveals a comprehensive form with details about the selected artist, including a list of their albums and average user ratings.

## 4.4 Album - More Info

Clicking "More Info" in "Artist - More Info" section displays a comprehensive form with additional album details, including all songs and average user rating.

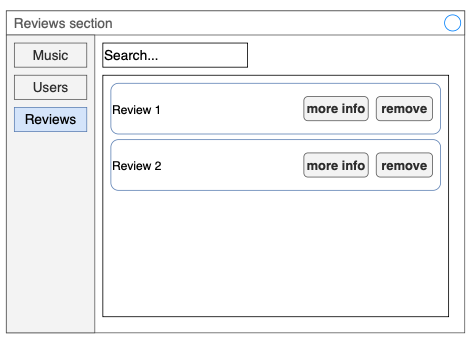
## 4.5 Create An Album

Once the administrator clicks on the "Add Album" button in the "Music" section, a pop-up window will appear, allowing the administrator to input all the necessary information about the album. From this window, the administrator can start adding songs to the album.

## 4.6 Users Section

"Users" section in the side menu allows administration to search for specific users, retrieve more information, or ban them.

## 4.7 Users Section

The "Reviews" section in the side menu allows the administration to search for specific user reviews, retrieve more information, or remove them if necessary.

# 5. Website wireframe

## 5.1 Home

The homepage features top artists and latest reviews for user convenience.

## 

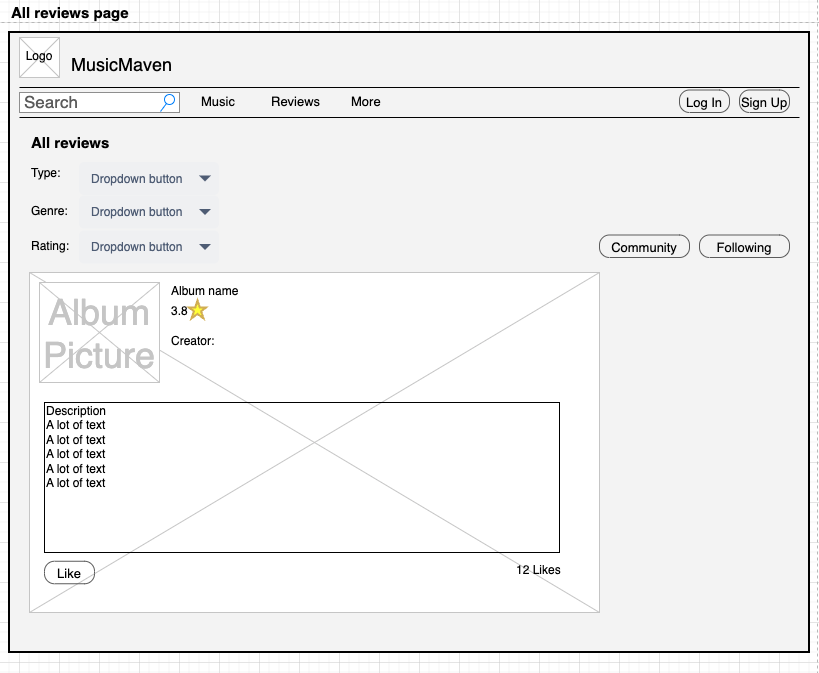
## 5.2 Top 100

The Top 100 page presents users with a curated list of the most popular music content. It provides a convenient way to discover the most sought-after songs, albums, or artists.

## 

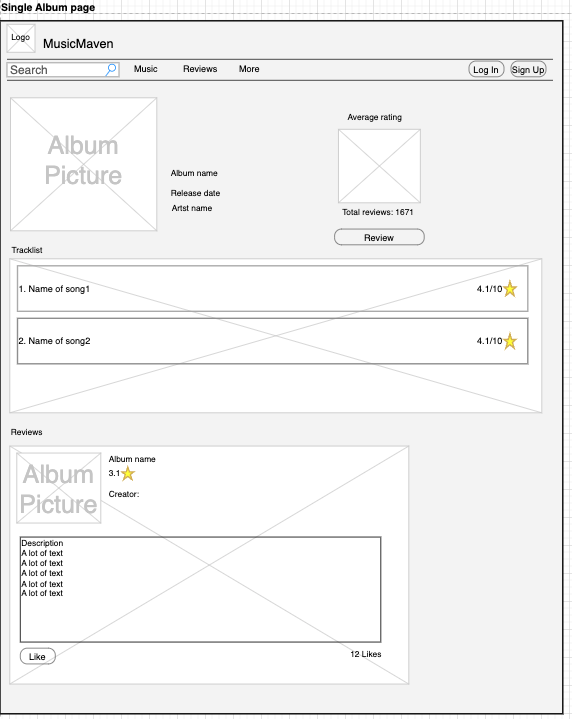
## 5.3 All Reviews

The "All Reviews" page displays a comprehensive list of all user-submitted reviews, providing a convenient way for users to browse through and gain insights into the community's feedback. The page also offers filtering options, allowing users to sort reviews based on criteria such as rating, date, or relevance.



## 5.4 Single Album Page

The Single View Album page displays a detailed view of a specific album, providing users with in-depth information about the album's content and features. The page typically includes information about the artist, the album's release date, track listing, and cover art. Users can also access reviews, ratings, and other related content from this page.



## 5.5 Single User Page

The Single User page provides a comprehensive view of a specific user's activity and profile within a music platform.