*School of Health and Society*

*Department Design and Computer Science*

*Kristianstad University*

*Course: DA122B*

*Program: Bachelor Programme in Software Development*

*Authors: Fatih Sylr, Federica Communi, Felicity Thompson, Viktor Nikolov*

<To Be Advised>

Table of Contents

[1. Introduction 4](#_Toc479157145)

[2. Requirements 5](#_Toc479157146)

[2.1 Use Case Overview 6](#_Toc479157147)

[2.1.1 Actors 6](#_Toc479157148)

[2.1.1.1 Free Trial User 6](#_Toc479157149)

[2.1.1.2 Account Holder User 6](#_Toc479157150)

[2.1.1.3 Data Entry Clerk 7](#_Toc479157151)

[2.1.1.4 Administrator 7](#_Toc479157152)

[2.1.2 Use Cases 7](#_Toc479157153)

[2.1.2.1 Listen to music 7](#_Toc479157154)

[2.1.2.2 Create playlists 7](#_Toc479157155)

[2.1.2.3 Modify user details 7](#_Toc479157156)

[2.1.2.4 Log in 7](#_Toc479157157)

[2.1.2.5 Look for music/songs 7](#_Toc479157158)

[2.1.2.6 Add or remove contacts 7](#_Toc479157159)

[2.1.2.7 Send messages 7](#_Toc479157160)

[2.1.2.8 Rate songs 7](#_Toc479157161)

[2.1.2.9 Giving feedback 8](#_Toc479157162)

[2.1.2.10 Modify albums 8](#_Toc479157163)

[2.1.2.11 Upload music 8](#_Toc479157164)

[2.1.2.12 Create/remove artists 8](#_Toc479157165)

[2.1.2.13 Create/remove data entry clerks 8](#_Toc479157166)

[3. Design and Implementation 9](#_Toc479157167)

[3.1 Classes 9](#_Toc479157168)

[3.1.1 <Name of Class> 9](#_Toc479157169)

[3.1.2 <Name of Class> 9](#_Toc479157170)

[3.2 Class Interactions and Use Case mappings 9](#_Toc479157171)

[3.2.1 <Name of Use Case> 9](#_Toc479157172)

[3.3 Database 9](#_Toc479157173)

[4. Test Results 10](#_Toc479157174)

[5. Summary and Conclusion 11](#_Toc479157175)

[5.1 Weekly Progress 11](#_Toc479157176)

[5.1.1 Week 1 11](#_Toc479157177)

[5.1.2 Week 2 11](#_Toc479157178)

[5.1.3 Week 3 11](#_Toc479157179)

[5.1.4 Week 4 11](#_Toc479157180)

[5.1.5 Week 5 11](#_Toc479157181)

[5.1.6 Week 6 11](#_Toc479157182)

[5.1.7 Week 7 11](#_Toc479157183)

[5.2 Difficulties and challenges 11](#_Toc479157184)

[5.2.1 <Name of Challenge/Difficulty 1> 11](#_Toc479157185)

[5.3 Correctness of time estimates 12](#_Toc479157186)

[5.4 Priority decisions 12](#_Toc479157187)

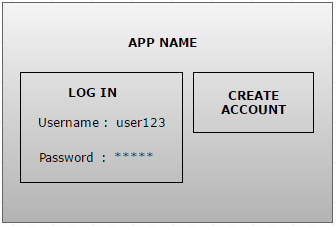
[5.5 Conclusion 12](#_Toc479157188)

[6 References 13](#_Toc479157189)

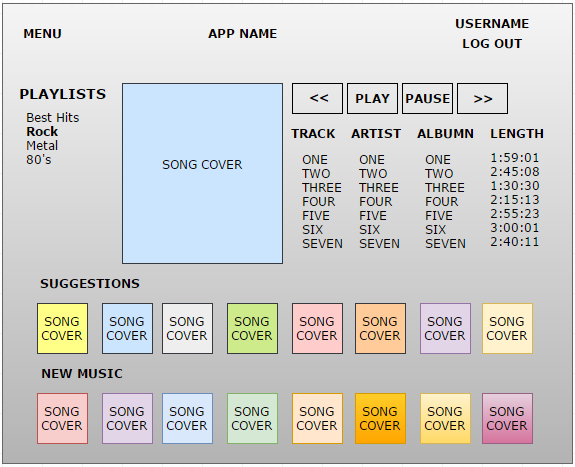
# Introduction

“insert-app-name-here” is a windows based application that allows instant streaming access to music of any genre. This app allows for instant access without the need for large storage or illegal downloads. The developers take care of the storage and piracy issues by gaining legal access to all music and offering an infinite remote library of music to the user. Subscriptions are offered to paying customers as well as trial users, who are offered restricted functionality. Users can create and manage their own playlists as well as browsing and listening to friend’s playlists.

Music is so readily available for illegal download and we the developers see the importance of giving what’s due back to the musicians and production companies. Additionally, obtaining music unorthodoxly can require quite large storage devices. Our aim is to provide everyone access to the music that they prefer, while still supporting musicians and production companies.



**Figure 1 – Mock up design for the main menu**



**Figure 2 – Mock up design for use case – 2.1.2.1**

# Requirements

<Instructions: The table below shall contain the requirements for your application. They shall be numbered from 1..N. Each requirement has a short name and a longer description. Hour estimates for tasks and their priority shall be put into the ProjectLibre Gantt chart project file. In this case, the priority values are between 0-1000 where 0 is lowest priority, 500 for medium priority (default), 1000 for highest priority. >

Table 1 below lists all the requirements for this project.

|  |  |
| --- | --- |
| **Req.**  **No** | **Req. Name: Description** |
| 1 | **Log in and Sign in scene:** a scene for users that are not logged in. It displays a log in and sign in windows with appropriate text fields for necessary user information (see **figure** **1**). |
| 2 | **Welcome scene**: the homepage for the logged in user. It shows content and album cover of the song currently playing, a list of new albums and a list of suggestions based on the user’s preferred genre. It contains a list of the user’s playlist and (optionally) of user’s contacts. It could optionally contain a list of the contacts recently played albums (see **figure 2**). |
| 3 | **Main menu**: a drop-down menu with options of logging out, accessing settings, help, program manual and quitting. |
| 4 | **Playing music**: a simple audio player. |
| 5 | **Creating/Modifying playlists**: the user should be able to create completely private, contact-only-private or public playlists with the songs available on the app. |
| 6 | **Sharing playlists**: users should be able to share private playlists with their contacts so all parts can access them and modify them. |
| 7 | **Adding/removing contacts**: users should be able to look for other users by username and add them as contacts to see what music they listen to and their contact-only-private playlists. |
| 8 | **Search function**: possibility to look for a specific song, artist, album, genre or user through a search bar. |
| 9 | **Send message**: possibility to send private message to contacts. |
| 10 | **Song rating:** users should be able to rate songs through a five-star rating function. |
| 11 | **Feedback**: users should be able to give feedback to songs in the form of comments. A message window with the suggestion to give feedback would pop up automatically each time the user rates a song. |
| 12 | **Statistics page**: top chart page of songs and artists, listed by rating or by number of times they are included in users’ playlists. |
| 13 | **Personalization**: users can personalize their profile by adding a profile picture and writing a short description of themselves. |
| 14 | **Facade pattern**: different types of users have different access to the program’s functionalities depending on their level of authorization (administrator, data entry clerk, account holder user or free trial user). |
| 15 | **Database**: storage of songs, genres and user information in database tables. |
| 16 | **GUI**: a graphical user interface with a brand-specific style and format. |
| 17 | **Adding/removing artists and songs**: the administrator should be able to add and remove artists and songs from the database. |
| 18 | **Paying scene**: page the supports payment of monthly fee for registered and free trial users. |
| 19 | **Single song page**: a page for every song that contains information such as its artist, album, year of publication, length, album cover, average rating and a list of users’ comments. |
| 20 | **Album page**: a page for every single album that contains information such as album cover, list of songs included and their average rating, year of publication, artist. |
| 21 | **Artist page**: page with information about the artist such as short biography, components, genre, albums published and years of activity. |
| 22 | **Instant messaging** (OPTIONAL): support of instant messaging between contacts. |

## Use Case Overview

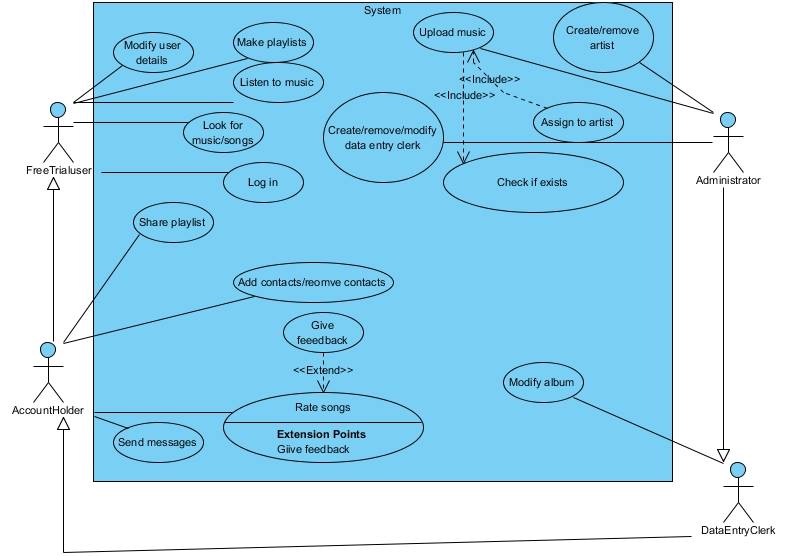


Figure 3 – Use Case Diagram

## Actors

### Free Trial User

User on 30-days free trial of the app. This user has limited access to the app’s functionalities and can only listen to music and create playlists without being able to interact with other contacts.

### Account Holder User

User that has paid for his/her monthly subscription fee. He/she can access all free trial user functionalities plus adding contacts, sharing playlists, sending messages, and giving feedback.

### 2.1.1.3 Data Entry Clerk

Company employee that can access all account holder’s functionalities plus the authorization to modify the music and image databases of the application.

### 2.1.1.4 Administrator

The administrator has full access to the application’s functionalities. He/she can upload new music, update the databases and create/remove data entry clerk accounts.

## Use Cases

### 2.1.2.1 Listen to music

After logging into the system all users can choose a song from their playlist(s) or from the application’s suggestions and play it with an audio player. The audio player includes a play/pause, stop and volume regulation option, and optionally a repeat and shuffle option for playlists *(see requirement 4 in Table 1)*.

### 2.1.2.2 Create playlists

All users can create a new playlist by clicking on a button on the GUI. Possible functions include (re)naming the playlist, adding songs, removing songs, and sharing the playlist with a contact (only for account holders - *see requirement 5 and 6 from Table 1)*. Playlists can be private (visible only to the user who created them), contact-only private (visible also to the user’s contacts) and public (visible to everyone). Adding a song to a playlist would influence its popularity on the application and eventually increase the number of times it is suggested on the main page, or its inclusion in top charts.

### 2.1.2.3 Modify user details

All users can personalize their profile by adding a profile picture and a short description of themselves, which are public and visible to every user inside the application *(see requirement 13 from Table 1)*.

### 2.1.2.4 Log in

All users can log in on the log in page by providing their username and password *(see requirement 1 from Table 1)*.

### 2.1.2.5 Look for music/songs

All users can look for songs, artists, genres, albums, and other contacts through a search bar. The search function will return the results that start with or contain the letters indicated by the user on the search bar *(see requirement 8 from Table 1)*.

### 2.1.2.6 Add or remove contacts

Account holding users can add another user as a personal contact. This would allow them to share private playlists, see each other’s contact-only playlists and send each other messages *(see requirement 7 from Table 1)*.

### 2.1.2.7 Send messages

Account holding users can send (optionally instant) messages to their contacts *(see requirement 9 and 22 from Table 1)*.

### 2.1.2.8 Rate songs

Account holding users can rate a song through a five-star rating. The rating would influence the popularity of the song on the application and eventually increase the number of times it is suggested on the main page, or its inclusion in top charts *(see requirement 10 from Table 1)*.

### 2.1.2.9 Giving feedback

Giving feedback would be an automatic suggestion from the application to the users who provide ratings. It would be in form of a comment that is visible *(see requirement 11 from Table 1)*.

### 2.1.2.10 Modify albums

Data entry clerks can contribute to the database organization by adding existing songs to albums and linking albums to their corresponding artists.

### 2.1.2.11 Upload music

The administrator can upload new songs after checking if they already exist in the database. He/she can then link them to the corresponding artists.

### 2.1.2.12 Create/remove artists

The administrator can organize the database by creating new artist and album pages *(see requirement 15 from Table 1)*.

### 2.1.2.13 Create/remove data entry clerks

The administrator can create and remove data entry clerk accounts so they can be in charge of the system organization. Free trial users are automatically created when a user registers on the application on the sign in page, and they are automatically upgraded to account holding user once they have paid their subscription fee *(see requirement 14 from Table 1)*.

# Design and Implementation

< Instructions: Describe your design in this chapter. >

## Classes

<Description of classes. List one class per sub chapter and add some class diagrams to illustrate relations (inheritance and/or associations) between the main classes. The UML does not need to be extremely detailed, but the most important attributes and methods shall be shown.>

### 3.1.1 <Name of Class>

< Description of this class, including UML. >

### 3.1.2 <Name of Class>

…

## Class Interactions and Use Case mappings

<Instructions: the sub chapters here shall correspond to the use cases in chapter 2, and each use case shall contain a UML **sequence diagram** of the classes that are involved in that use case, and how they interact to implement the use case, including method calls. >

### 3.2.1 <Name of Use Case>

<A sequence diagram of the classes involved in this use case, and how they interact. You may write some explaining text here, and/or you may use notes in the diagram itself.>

## Database

<Show your database design with ER diagram(s). >

# Test Results

Table 2 below contains the current status of implemented and tested requirements.

<Instructions: This table shall map 1-1 to the table in Chapter 2. The test result for each requirement shall be one of the following: NOT IMPLEMENTED, PASSED or FAILED.>

Table 1 - Test Results

|  |  |  |
| --- | --- | --- |
| **Req.**  **No** | **Req. Name** | **Test Result** |
| 1 | <Requirement 1 name> | <NOT IMPLEMENTED/PASSED/FAILED> |
| 2 | <Requirement 2 name> | … |
| .. | .. | .. |

# Summary and Conclusion

This chapter contains a summary and conclusion of the work that was carried out in this project as well as reflections and thoughts about working methods and challenges.

## Weekly Progress

Below is a short summary of what was done each week.

<Instructions: Describe what you did this week. You can see it as a developer’s weekly diary. Try to answer the following questions: What did you do this week? Did you meet any challenges? What was difficult? Did you get stuck with something? What went well and what went bad? What have you learned during this week?>

### 5.1.1 Week 1

Week 1 the group made a final decision to make a music streaming app. It was an easy choice, as all members were in favor. There was much discussion on whether a mobile platform app be created using Android Studio or whether to make a Java FX desktop app. We took into consideration the man hours that would be expended learning Android Studio to make a basic app and decided that those man hours could instead be utilized to expand functionality. Java FX was the final decision as the entire team has experience with the framework.

### 5.1.2 Week 2

TODO

### 5.1.3 Week 3

TODO.

### 5.1.4 Week 4

TODO.

### 5.1.5 Week 5

TODO.

### 5.1.6 Week 6

TODO.

### 5.1.7 Week 7

TODO.

## Difficulties and challenges

Below is a list of notable challenges that came up during this project and that took a long time to solve.

### 5.2.1 <Name of Challenge/Difficulty 1>

<Instructions: List the most difficult tasks in this project and describe why they were difficult. Did you learn something, e.g. how to handle very difficult programming problems?>

## Correctness of time estimates

<Instructions: Look back on your time estimates and discuss your results. How accurate were they? What have you learned about time estimates and how can you get better in next project?>

## Priority decisions

<Instructions: Look back on your feature priority settings. Did you prioritize the right features? Did you succeed to deliver the highest prioritized features? Have you learned anything about setting priorities>

## Conclusion

<Instructions: Look back on the whole project. Here you can write a bit more freely about your thoughts on this project. What was your overall experience? How was the teamwork? What did you learn? Can you list some points that you will do better in next project? Other thoughts. >

# References

<Instructions: In this chapter, you shall list references to external sources, books, web sites, etc. In this document, we use the Vancouver Referencing System [1], also called the author-number system. >

[1] Wikipedia: Vancouver System. (Available: <http://en.wikipedia.org/wiki/Vancouver\_system>)