HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and Communications Technology

A picture containing logo

Description automatically generated

Software Requirement Specification

**JELLIMIX MUSIC PLAYER PROJECT**

Subject: *Management of Software Project*

GROUP 08

|  |  |  |  |
| --- | --- | --- | --- |
| Nguyễn Thị Hồng Anh | 20176679 | Nguyễn Thị Minh Châu | 20184328 |
| Trần Thái Dương | 20176732 | Đào Cao Duy | 20176738 |
| Nguyễn Việt Hoàng | 20176762 | Nguyễn Trí Hùng | 20176773 |
| Trần Phi Hùng | 20176774 | Phạm Vũ Minh | 20176818 |
| Nguyễn Mạnh Phúc | 20176825 | Lê Thanh Sơn | 20176860 |

*Hanoi, October 2021*

Table of contents

Table of contents 1

1 Introduction 2

1.1 Objective 2

1.2 Scope 2

1.3 Glossary 2

1.4 References 2

2 Overall Description 3

2.1 Actors 3

2.2 Use case diagrams 3

2.3 Business processes 3

3 Detailed Requirements 4

3.1 Use case specification for “Use case 1” 5

3.2 Use case specification for “Use case 2” 6

4 Supplementary specification 8

4.1 Functionality 8

4.2 Usability 8

4.3 Reliability 8

4.4 Performance 8

4.5 Supportability 8

4.6 Other requirements 8

# Introduction

## Objective

The purpose of this document is to present a detailed description of JELLIMIX MUSIC Website. It will explain the purposes and features of the system, the system interfaces, as well as what the system will do, the contrains under which it must operate and how the system will react to external components.

This document is intended for both the stake holders and developers of the website

## Scope

The JelliMix Music website is built for a wide range of users who want to listen to music online. The system allows users to enjoy music anywhere, anytime, with low to no cost for high quality features.

In details, when the user enters JelliMix, he can see various music genres, top 10 hottest songs, and the list of most frequently listened songs in the past days. User can immediately select any of them to start listening to. While the song is being played, user can interact with the players with options of pause, repeat, forward, backward, or turn on shuffling for random suggested songs. These suggestions come from the most recent data about user’s listened song and his favorite tracks, genres, and artists. They can also add any song to the current playing queue.

User can search for a specific track by providing some clues. The returned result is the list of matched songs, and if the user chooses to see information about one of them, details about it, including name, lyrics, artists, singers, composed day, albums included, length, listen times and reviews about the song. If user wants to search for the album, JelliMix replies with a list of matched albums, and information about the album are name, release day and list of songs in the album. Besides, users can also look up for artist or band. The data returned for this request are the artist’s or band’s name, day started to compose or perform, and the lists of songs belonging to them.

Users can freely listen to music on JellyMix, or create an account to use more features of the website. When having become the website’s member, user can create a playlist of song for convenience when listening to his favorites. Also, the 20 most recently listened songs are also recorded so that he can easily find them later.

JelliMix also allows members to create or join a community that has the same interest as them. Users can find friends by name or ID and follow them, or add them to friend list to chat together in the future. After that, they can see activities of their friends or users that they are following. They can also create a group and add member to better communicate and share their interests.

## Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| JelliMix | Name of the system |
| User | The person who uses the website |
| Member | The person who registers for an account in the website |
| Shuffle | The action of randomly changing order of songs in a playlist or playing queue |
| Playing queue | A list of songs to be played, which is created by an user at a specific moment. This will not be associated to any account and will be forgotten/deleted after the user leaves the website |
| Playlist | A saved list of songs, created by a member |
| Follow | The action of registering to get notification about one user |
| Friend | The member that connects to another member and has his activity status shown to the other. Friend can also chat together |
| Add friend | The action of making a member a friend and add them to friend list |
| Friend list | The list of added friend, associated with each member |
| Group | A set of members that connect together and share same interest. Group member can chat together. |
| Account | Created by a member to save his information, as well as to use futher functionalites of the system |

## References

*<Listing the referenced material used in this documents, including the one related to the project>*

# Overall Description

< *Describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in section 3, and makes them easier to understand*. *In a sense, this section tells the requirements in plain English for the consumption of the customer. Section 3 will contain a specification written for the developers*>

## Actors

## Use case diagrams

*\*\*\* Insert our lovely usecase diagram here*

## Business processes

*<If the software has business processes, please use activity diagrams to visualize them. Please note that they are different from the activity diagrams for the use case flow of events>*

# Detailed Requirements

***<Please provide use case specifications here. You can refer the below template>***

<*This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:*

1. *Specific requirements should be stated with all the characteristics of a good SRS*
2. *correct*
3. *unambiguous*
4. *complete*
5. *consistent*
6. *ranked for importance and/or stability*
7. *verifiable*
8. *modifiable*
9. *traceable*
10. *Specific requirements should be cross-referenced to earlier documents that relate*
11. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
12. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*

*Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*

*Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:*

*The system shall… A required, must have feature*

*The system should… A desired feature, but may be deferred til later*

*The system may… An optional, nice-to-have feature that may never make it to implementation.*

*Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*

*Avoid examples, this is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address”. The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables*>

## Use case specification for “Use case 1”

**Use Case “Name of use case”**

1. **Use case code**

UC00X

1. **Brief Description**

This use case describes the interaction between <actor(s)> and <name\_of\_the\_system> when <actor(s)> wish(es) to ...

1. **Actors**
   1. **Name of Actor 1**
2. **Preconditions**
3. **Basic Flow of Events**
4. The actor(s)
5. …

i. The software displays … (see Table T)

…

1. **Alternative flows**

Table N-Alternative flows of events for UC Place order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Location** | **Condition** | **Action** | **Resume location** |
|  | At Step S | If … | * Action 1 | Resumes at Step Q |
|  | At Step O | If … | * Action 2 | Use case ends |

1. **Activity diagrams**
2. **Input data**

Table A-Input data of …

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Mandatory** | **Valid condition** | **Example** |
|  |  |  |  |  |  |

1. **Output data**

Table B-Output data of …

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Display format** | **Example** |
|  |  |  |  |  |

1. **Postconditions**

## Use case specification for “Use case 2”

**Use Case “Name of use case”**

1. **Use case code**

UC00X

1. **Brief Description**

This use case describes the interaction between <actor(s)> and <name\_of\_the\_system> when <actor(s)> wish(es) to ...

1. **Actors**
   1. **Name of Actor 1**
2. **Preconditions**
3. **Basic Flow of Events**
4. The actor(s)
5. …

i. The software displays … (see Table T)

…

1. **Alternative flows**

Table N-Alternative flows of events for UC Place order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Location** | **Condition** | **Action** | **Resume location** |
|  | At Step S | If … | * Action 1 | Resumes at Step Q |
|  | At Step O | If … | * Action 2 | Use case ends |

1. **Activity diagrams**
2. **Input data**

Table A-Input data of …

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Mandatory** | **Valid condition** | **Example** |
|  |  |  |  |  |  |

1. **Output data**

Table B-Output data of …

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Data fields** | **Description** | **Display format** | **Example** |
|  |  |  |  |  |

1. **Postconditions**

# Supplementary specification

*<Presenting other requirements if necessary, including non-functional requirements such as performance, reliability, usability, and supportability; or other technical requirements such as database system, used technology…>*

## Functionality

*<Functional requirements that are general to many use cases>*

## Usability

<*Requirements that relate to, or affect, the usability of the system. Examples include ease-of-use requirements or training requirements that specify how readily the system can be used by its actors*>

## Reliability

*<Any requirements concerning the reliability of the system. Quantitative measures such as mean time between failure or defects per thousand lines of code should be stated>*

## Performance

*<The performance characteristics of the system. Include specific response times. Reference related use cases by name>*

## Supportability

*<Any requirements that will enhance the supportability or maintainability of the system being built>*

## Other requirements

*<Descriptions of other requirements are located here>*