List of Figures

List of table

1. Preface

이 장에서는 독자층, 보고서가 다루는 범위 보고서의 목적, 문서의 구조에 대하여 설명한다.

1.1. 독자층

이 문서는 10개의 섹션으로 구성되어 있으며 각 세션마다 서브섹션으로 구성되어있다. 3조가 이 문서의 메인 독자이며 소프트웨어 공학 담당 교수님, 조교님이 메인 독자가 될 수 있다.

1.2. 범위

이 문서는 기분에 맞는 음악을 추천하는 “Music Pet”을 구현하기 위해 사용되는 디자인의 정의로써 소프트웨어 엔지니어링이나 소프트웨어 품질 엔지니어링에서 사용된다.

1.3. 목적

이 문서는 “Music Pet”의 디자인 측면에 대해 기술하는 것이 목표이다. 이 문서는 “Music Pet”을 직접 개발하는 데 필요한 소프트웨어 아키텍처와 소프트웨어 디자인 결정 요소가 기술되어있다. 또한 “Music Pet”을 여러 관점에서 보기 위한 아키텍처에 대한 전체적 overview가 설명되어있다. 그리고 이 문서에서는 SRS에서 언급되었던 모듈들의 디자인과 구조를 구체화할 것이다. Use case와 sequential diagram과 activity diagram, class diagram에 대해서도 설명한다.

1.4. 문서 구조

1. Preface: 이 장에서는 독자층, 문서의 범위, 목적, 문서의 구조에 대해 설명한다.

2. Introduction: 이 장에서는 sds에서 사용되는 다이어그램을 설명하고 디자인을 만들기 위해 사용한 툴을 소개한다. 또한 참고자료에 대해서도 설명한다.

3. System Architecture – Overall: 시스템 아키텍처를 context diagram, sequence diagram, use case diagram으로 설명한다.

4. System Architecture – frontend: 시스템의 프론트엔드를 기술한다.

5. System Architecture – backend: 시스템의 백엔드를 기술한다.

6. Testing Plan: 테스트 계획에 대해 설명한다.

7. Development Plan: 프론트엔드와 백엔드 개발에 필요한 툴을 설명하고 제약사항과 가정과 의존성에 대해 기술한다.

8. Supporting Information: 문서의 변경사항을 기록한다.

2. Introduction

“Music Pet”은 사용자의 기분과 날씨를 파악해 노래를 추천해주거나 음악 재생 상황을 설정하여 그에 따른 노래를 들려주는 프로그램이다. 이를 통해 사용자의 감정을 추스리게 도와준다.

2.1. 목표

이번 장에서는 Software Design Specification에 사용되는 다이어그램을 설명하고 디자인 단계에서 필요한 툴들을 설명할 것이다.

2.2. 다이어그램

2.2.1. UML

통합 모델링 언어로써 모델을 만드는 표준언어이다. 즉 UML은 소프트웨어 시스템을 구성하는 구성요소를 지정하고 시각화하는 기능을 제공한다. UML은 구조 다이어그램 7가지, 행위 다이어그램 7가지로 총 14가지의 다이어그램이 있다. 구조 다이어그램은 시스템의 개념, 관계의 측면에서 요소들을 나타내고 요소들의 정적인 면을 나타낸다. 행위 다이어그램은 각 요소들 간의 변화나 흐름, 주고받는 데이터의 동작을 보여준다.

2.2.2. Use case diagram

Use case diagram은 유저가 시스템과 어떻게 상호작용하는지 기술한다. Use case diagram은 각각의 use case들을 타원 형태의 node안에 정보들을 명시하여 구별한다. 그리고 각각의 use case들은 스틱 형태로 표현된 사용자들을 선으로 연결하여 유저와 시스템간의 상호작용을 나타낸다. 상호작용이 이루어지는 방향을 기술하기 위해 화살표로 유저와 시스템을 연결하기도 한다. Use case diagram은 시스템의 기능적 요구사항을 시각화하고 소비자와 개발자가 가지고 있는 상이한 이해관계들을 조율하는데 사용되곤 한다.

2.2.3. Sequence diagram

Sequence diagram은 액터와 시스템의 객체들 간의 상호 작용과 객체들 간의 상호 작용을 모델링하기 위해 주로 사용한다. Sequence diagram은 유스케이스 인스턴스의 상호작용 순서를 나타낸다.

2.2.4. Class diagram

Class diagram은 시스템의 클래스들과 그들 간의 연관을 보여주는 객체지향 시스템 모델을 개발할 때 사용된다. Class diagram을 표현하는 방법은 먼저 실세계를 관찰하고 필수 객체들을 식별하고 이들을 클래스로 표현하는 것이다. 그 후 클래스들 사이에 선을 연결하여 연관의 존재를 표시할 수 있다.

2.2.5. Context diagram

Context diagram은 시스템의 경계를 정의하고 외부 객체가 시스템과 어떻게 상호작용하는지 나타낸 다이어그램이다. 그리고 전체적인 시스템 간의 관계를 간단히 나타냄으로써 쉽게 이해할 수 있도록 기술한 것이 특징이다.

2.3. Applied tools

2.3.1. Microsoft Powerpoint

UML 다이어그램 만드는데 사용된다.

2.4. 프로젝트 범위

이 프로젝트는 코로나19로 힘들어하는 사람들을 위해 음악을 통한 기분전환으로 우울감 및 외로움 극복하고 간단하고 편안하게 음악 감상을 할 수 있도록 도와주기 위해 만들어졌다. “Music Pet”은 사용자의 기분에 따라 기분을 증폭시키거나 완화하는 음악을 추천해줌으로써 감정을 추스릴 수 있도록 하고 어플의 개인적인 활용을 위해 재생 상황 설정하고 개인 맞춤 플레이리스트를 생성할 수 있도록 디자인되었다.

2.5. References

Ian Sommerville, 소프트웨어 공학

Team 16, 2021 fall. Software Design Specification, SKKU

https://www.nextree.co.kr/p6753/

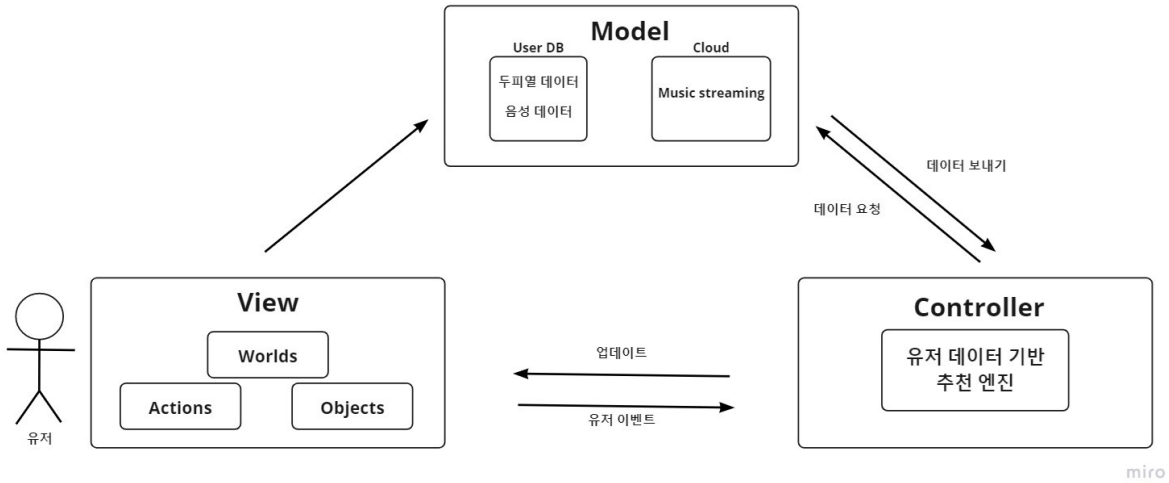
3. System Architecture - Overall

3.1. Objectives

이 장에서는 시스템 아키텍처, context diagram, sequence diagram 및 use case diagram의 전반적인 양상에 대해 나타내고 있다.

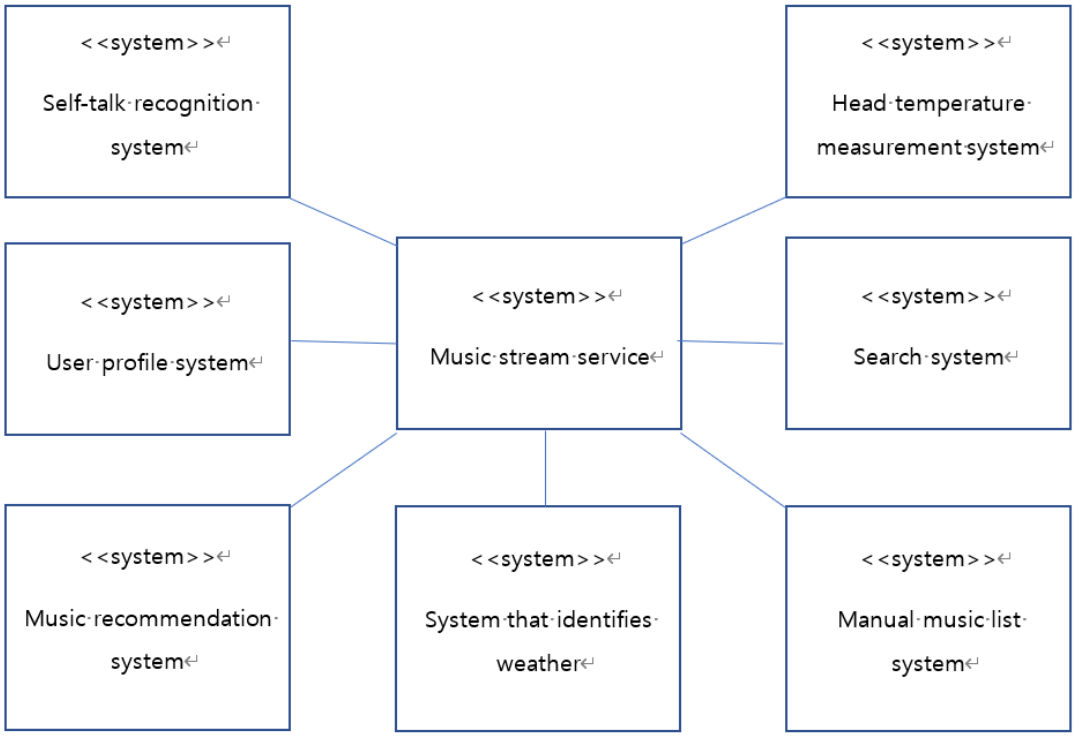
3.2. System Organization

이 서비스는 유저 데이터 기반 추천 엔진을 기반으로 사용자가 필요한 음악을 제공한다. 사용자는 앱(Worlds)을 통해 음악 스트리밍 서비스에 접근할 수 있으며, 본인이 실제적으로 원하는 음악 리스트를 생성할 수 있다. 모델은 사용자의 동작들(Actions)로부터 생성된 데이터들을 저장한다. 컨트롤러는 기기(Objects)로부터 전달받은 두피열 데이터와, 음성 데이터를 모델에게 요청하고, 유저 데이터 기반 추천 엔진을 활용하여 사용자에게 맞춤형 음악 서비스를 제공한다.



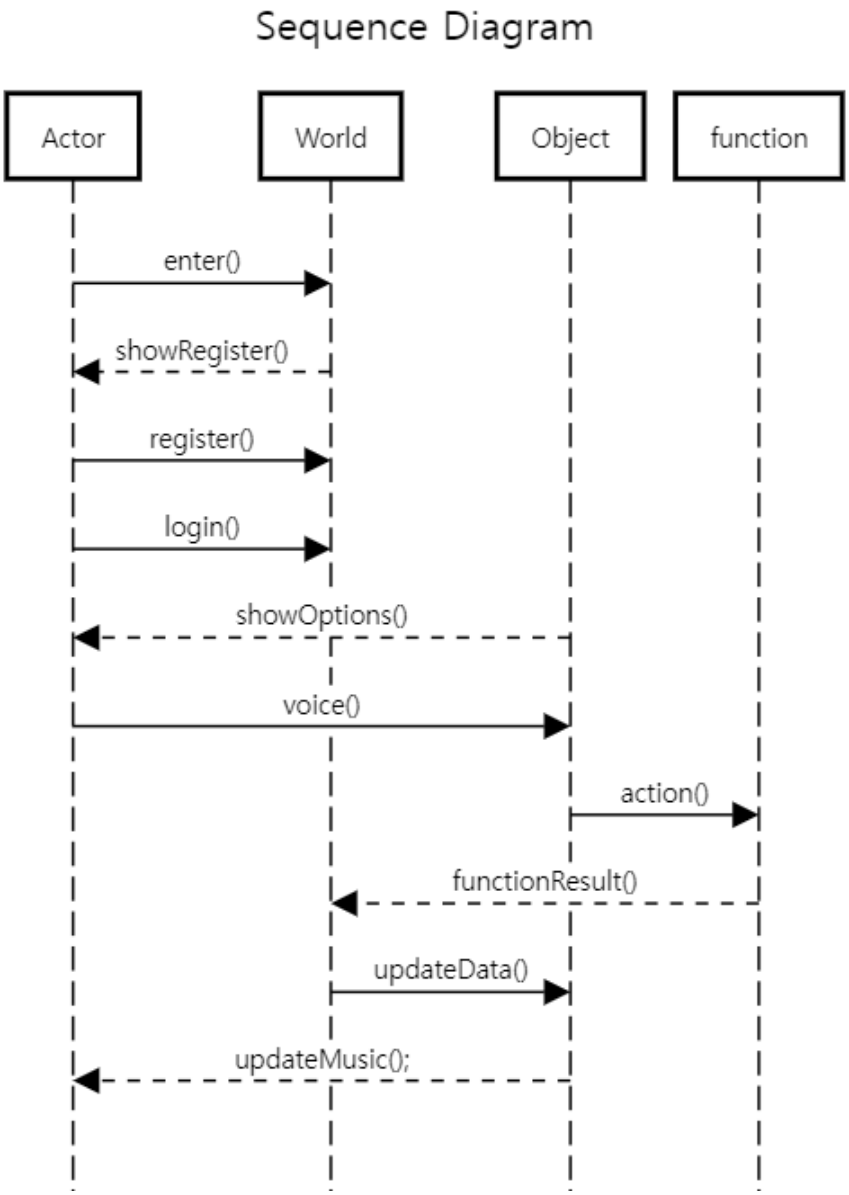
[Figure 1] Overall system architecture

3.2.1. Context Diagram



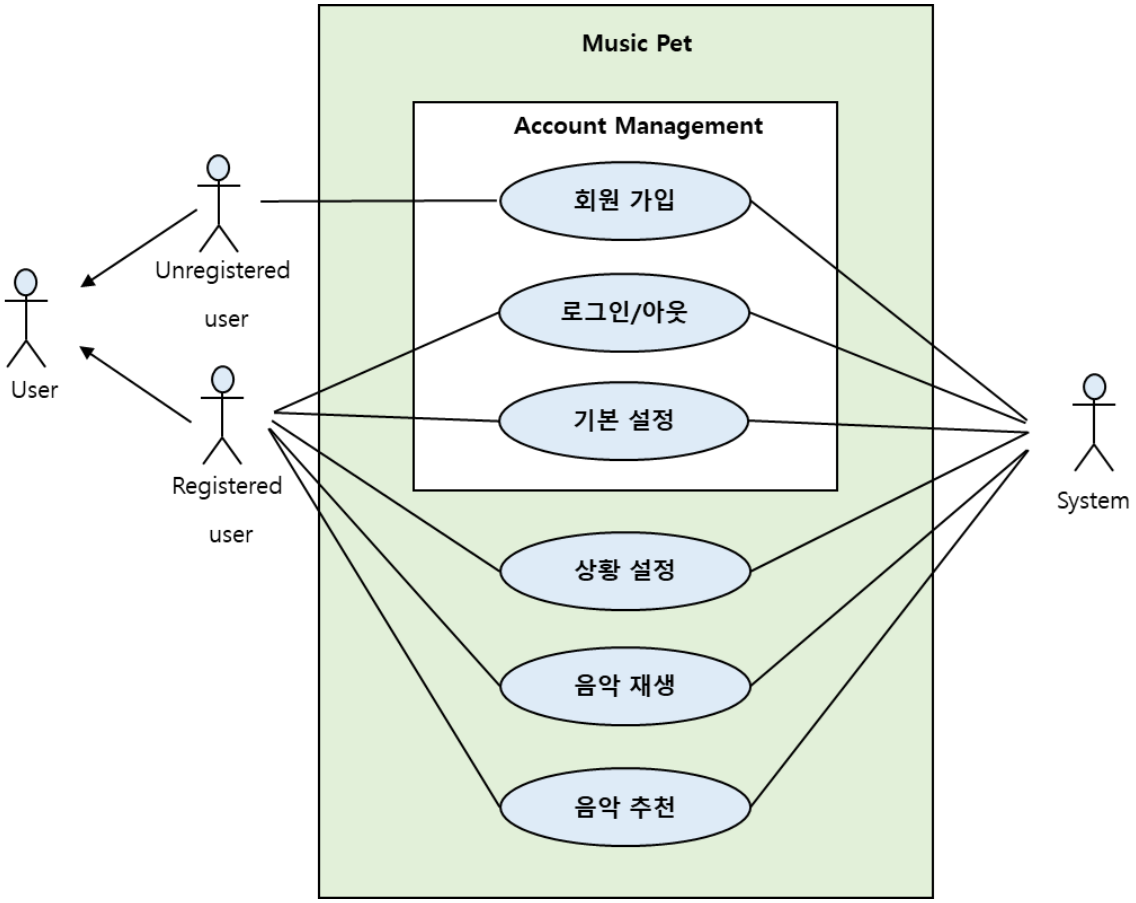
[Figure 2] Overall Context Diagram

3.2.2. Sequence Diagram



[Figure 3] Overall Sequence Diagram

3.2.3. Use Case Diagram



[Figure 4] Use Case Diagram

4. System Architecture – Frontend

4.1. Objectives

이 장에서는 프론트엔드 시스템을 구성하는 함수와 구조, 속성을 설명하고, 각 요소들 간의 관계를 다이어그램을 통해 표현한다.

4.2. Subcomponents

4.2.1. 회원가입

회원가입 클래스는 가입되지 않은 사용자를 관리한다. 회원 가입을 진행하여 사용자의 정보를 수집하고, 서버에 저장한다.

4.2.1.1. Attributes

다음은 로그인 및 회원가입 객체가 갖는 속성이다.

- User Info: 회원 가입 시 작성되는 사용자의 정보이다.

다음은 사용자 정보 객체가 갖는 속성이다.

- User ID: 사용자가 로그인 시 사용할 ID이다.

- Password: 사용자가 로그인 시 사용한 비밀번호이다.

- Email Address: 사용자의 이메일 주소이다.

- Birth date: 사용자의 생년월일이다.

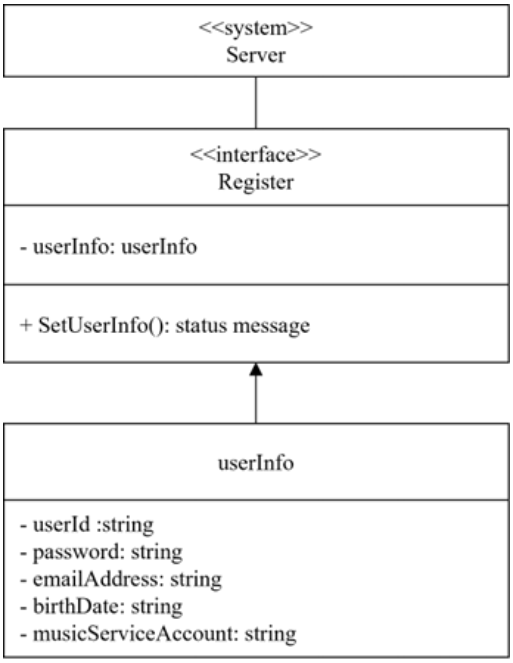
- Music Service Account: 서비스와 연결한 음원 스트리밍 서비스의 계정이다.

4.2.1.2. Methods

다음은 회원 가입 객체가 가지는 메소드이다.

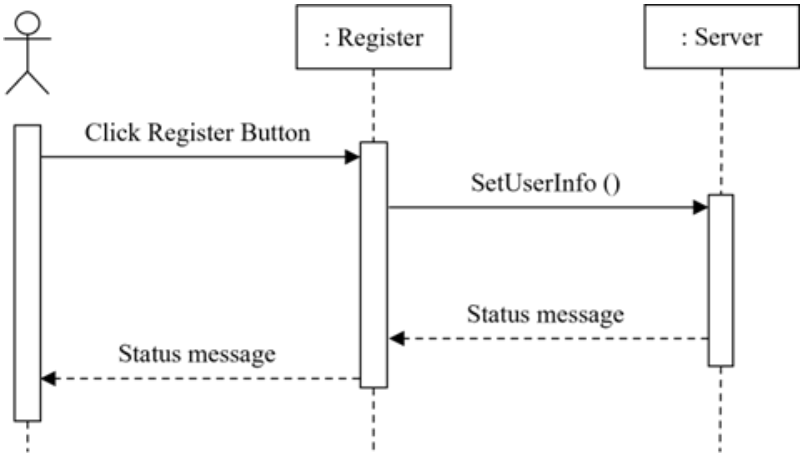
- SetUserInfo ()

4.2.1.3. Class Diagram



[Figure 5] Class Diagram - 회원가입

4.2.1.4. Sequence Diagram



[Figure 6] Sequence Diagram - 회원가입

4.2.2. 로그인

로그인 클래스는 전체 사용자의 로그인 세션을 관리한다. 사용자는 ID와 비밀번호 입력을 통해시스템에 접속할 수 있다. 자동 로그인을 허용한 사용자는 최초 로그인 이후 해당 클래스를 거치지 않는다.

4.2.2.1. Attributes

다음은 로그인 객체가 갖는 속성이다.

- User ID: 로그인 시 사용되는 사용자의 ID이다.

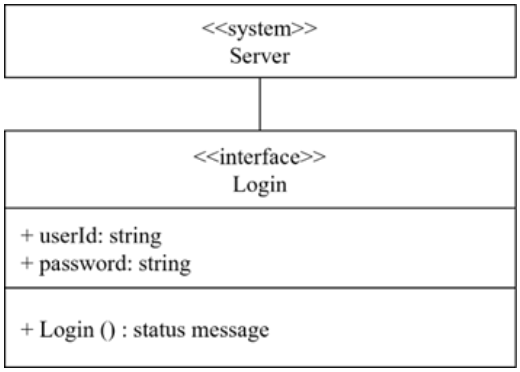
- Password: 로그인 시 사용되는 사용자의 비밀번호이다.

4.2.2.2. Methods

다음은 로그인 객체가 가지는 메소드이다.

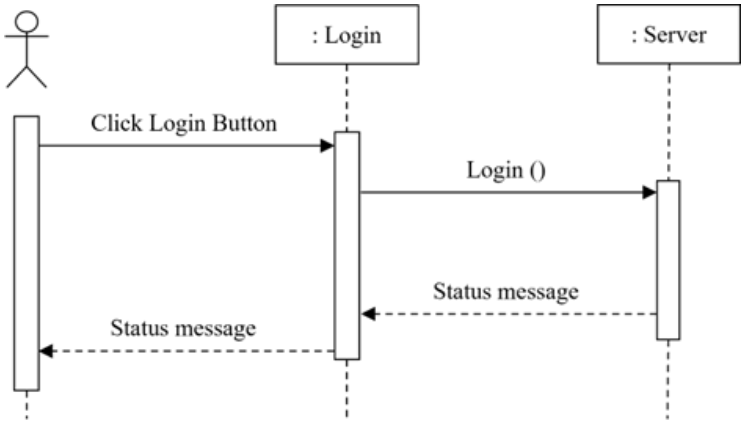
- Login()

4.2.2.3. Class Diagram



[Figure 7] Class Diagram - 로그인

4.2.2.4. Sequence Diagram



[Figure 8] Sequence Diagram - 로그인

4.2.3. 기본 설정

기본 설정 클래스는 사용자의 스피커를 연결 및 관리한다. 현재 연결된 스피커를 확인과 새로운 스피커의 연결이 가능하며, 스피커의 작동 여부를 검사한다.

4.2.3.1 Attributes

다음은 기본 설정 객체의 속성이다.

- Speaker: 핸드폰과 블루투스로 연결된 스피커이다.

- Speaker List: 현재 연결된 스피커 목록이다.

다음은 스피커 객체가 갖는 속성이다.

- Speaker Name: 스피커의 이름을 나타낸다.

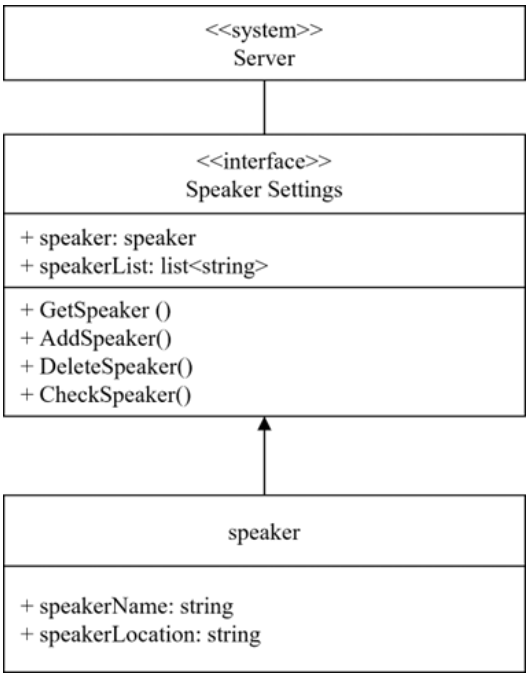
- Speaker Location: 스피커의 위치를 나타낸다.

4.2.3.2. Methods

다음은 기본 설정 객체의 Methods이다.

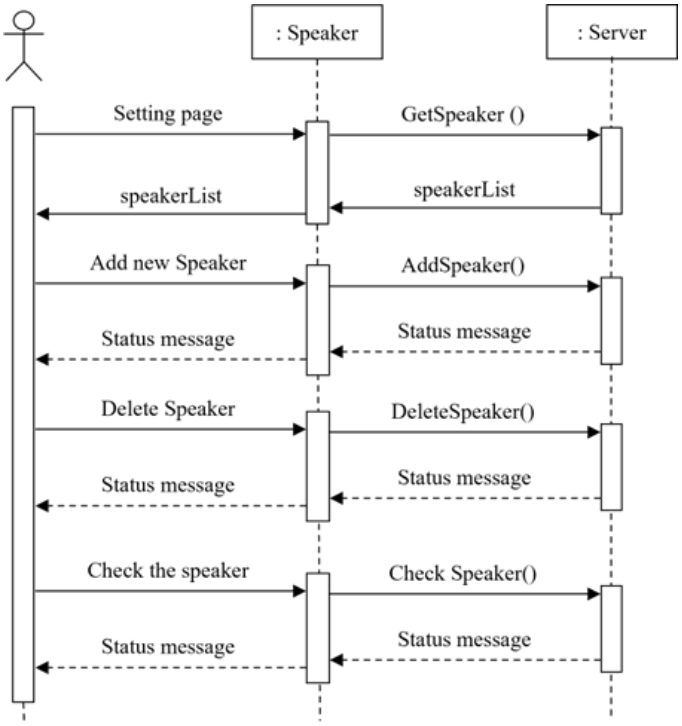
- GetSpeaker ()
- AddSpeaker ()
- DeleteSpeaker ()
- CheckSpeaker ()

4.2.3.3. Class Diagram



[Figure 9] Class Diagram - 기본 설정

4.2.3.4. Sequence Diagram



[Figure 10] Sequence Diagram - 기본 설정

4.2.4. 상황 설정

상황 설정 클래스는 사용자의 음악 재생 상황 설정을 관리한다. 사용자가 직접 시간, 장소 그리고 음성 인식 결과를 입력하여 상황을 설정할 수 있으며 각 상황에 맞는 플레이 리스트를 지정할 수 있다.

4.2.4.1 Attributes

다음은 상황 설정의 속성이다.

- Condition: 사용자가 설정한 상황이다.

- Situation List: 사용자가 설정한 상황 리스트이다.

- Playlist: 사용자가 상황에 따라 설정할 플레이리스트이다.

다음은 Condition 객체가 갖는 속성이다.

- Time: 사용자가 설정한 시간

- Location: 사용자가 설정한 장소

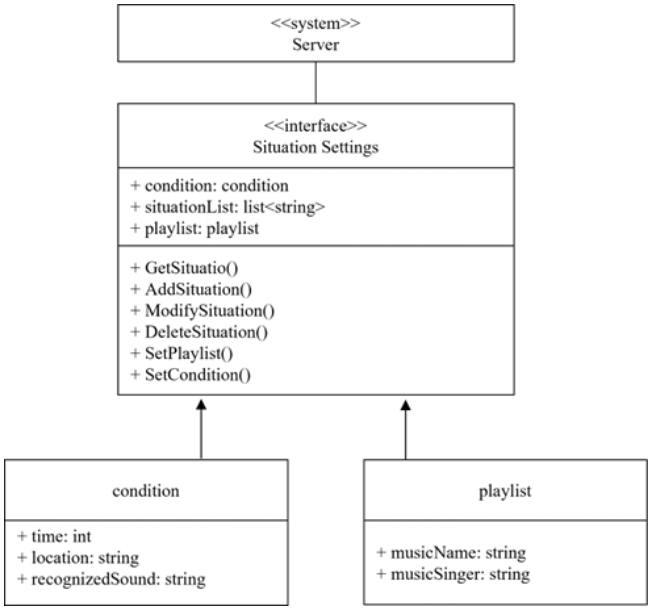
- Recognized Sound: 사용자가 설정한 음성 인식 결과

4.2.4.2. Methods

다음은 상황 설정 객체가 가지는 메소드이다.

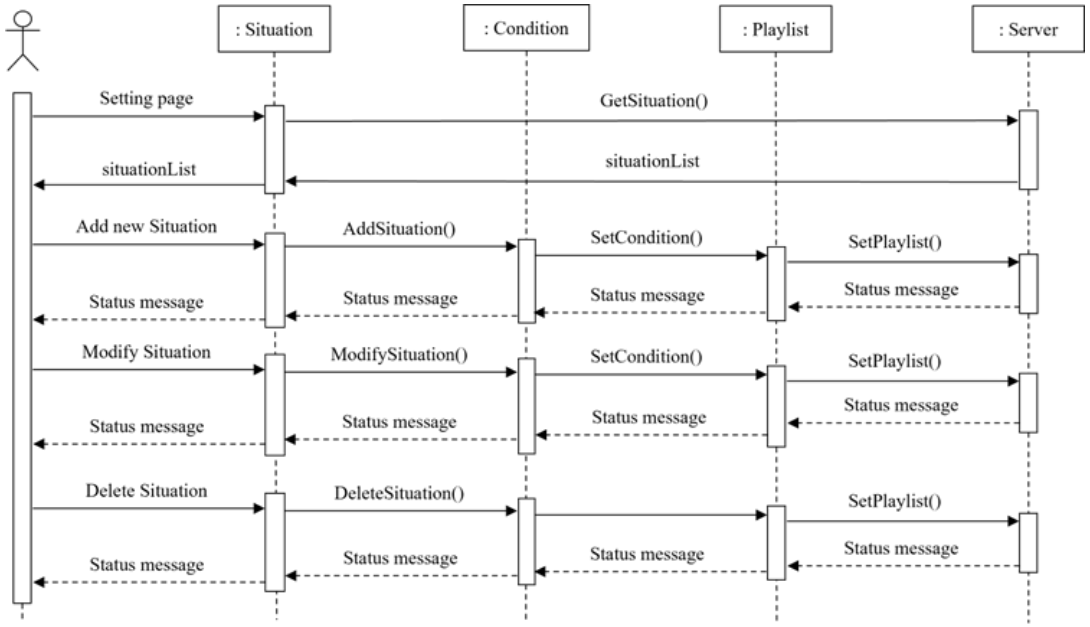
- GetSituation ()
- AddSituation ()
- ModifySituation ()
- DeleteSituation ()
- SetCondition ()
- SetPlaylist ()

4.2.4.3. Class Diagram



[Figure 11] Class Diagram - 상황 설정

4.2.4.4. Sequence Diagram



[Figure 12] Sequence Diagram - 상황설정

4.2.5. 음악 재생

음악 재생 클래스는 상황 설정에 알맞게 음악을 재생한다. 현재 상황이 조건에 부합하는지 확인한 후 설정된 플레이리스트를 재생한다. 사용자는 음악을 재생, 정지 또는 넘길 수 있다.

4.2.5.1 Attributes

다음은 음악 재생의 속성이다.

- Situation: 사용자가 설정한 상황(조건과 플레이리스트)이다.

- Music: 사용자가 상황에 따라 재생될 음악에 대한 데이터이다.

다음은 Situation 객체가 갖는 속성이다.

- Time: 사용자가 설정한 시간

- Location: 사용자가 설정한 장소

- Recognized Sound: 사용자가 설정한 음성 인식 결과

4.2.5.2. Methods

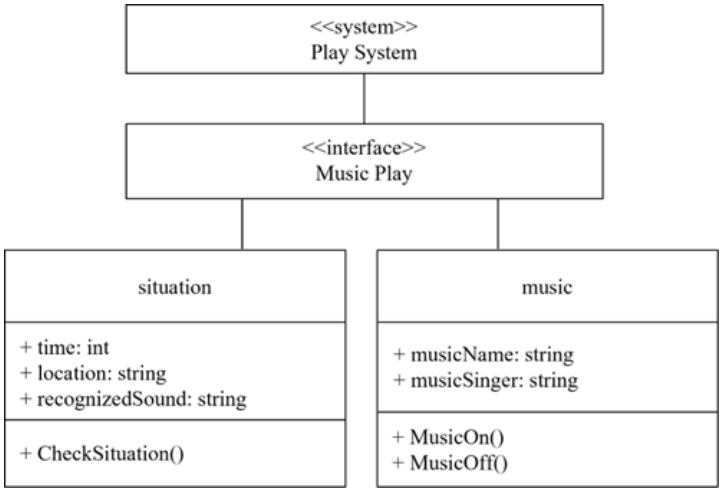
다음은 음악 재생 객체가 가지는 메소드이다.

- CheckSituation()

- MusicOn()

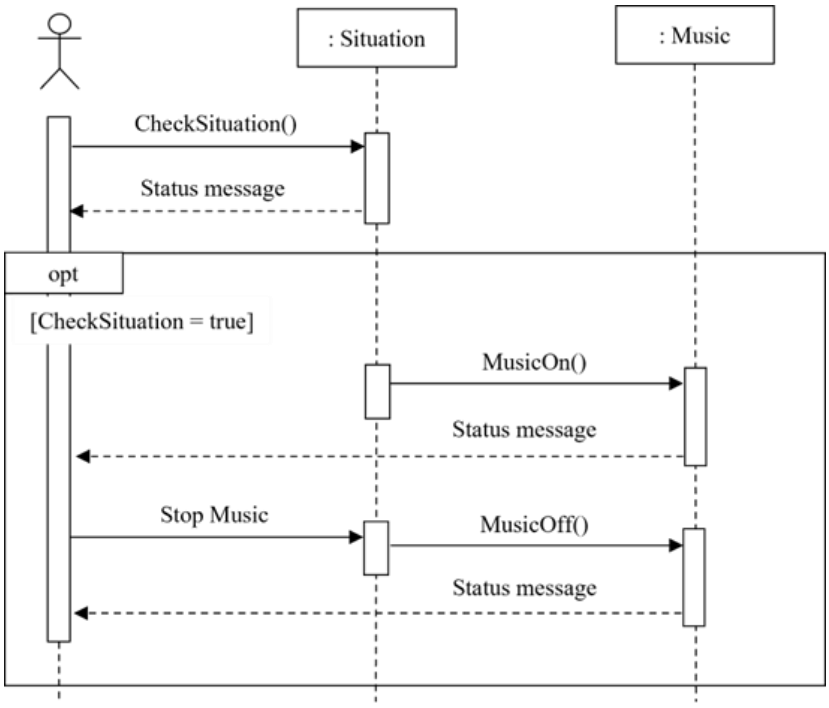
- MusicOff()

4.2.5.3. Class Diagram



[Figure 13] Class Diagram - 음악재생

4.2.5.4. Sequence Diagram



[Figure 14] Sequence Diagram - 음악재생

4.2.6. 음악 추천

음악 추천 클래스는 사용자별 맞춤 추천 시스템을 관리한다. 사용자는 음악 추천 페이지에서 현재 기분과 기분에 맞는 두가지 테마의 음악을 추천 받을 수 있다. 추천 받은 음악은 바로 재생하거나 플레이리스트에 추가할 수 있다.

4.2.6.1. Attributes

다음은 음악 추천 객체가 갖는 속성이다.

- User Feeling: 데이터를 통해 추정된 사용자의 현재 기분이다.

- Music: 사용자의 취향에 맞게 추천할 음악 데이터이다.

다음은 음악 객체가 갖는 속성이다.

- Positive Music: 현재 기분을 유지시킬 목적으로 추천된 음악이다.

- Negative Music: 현재 기분을 반전시킬 목적으로 추천된 음악이다.

4.2.6.2. Methods

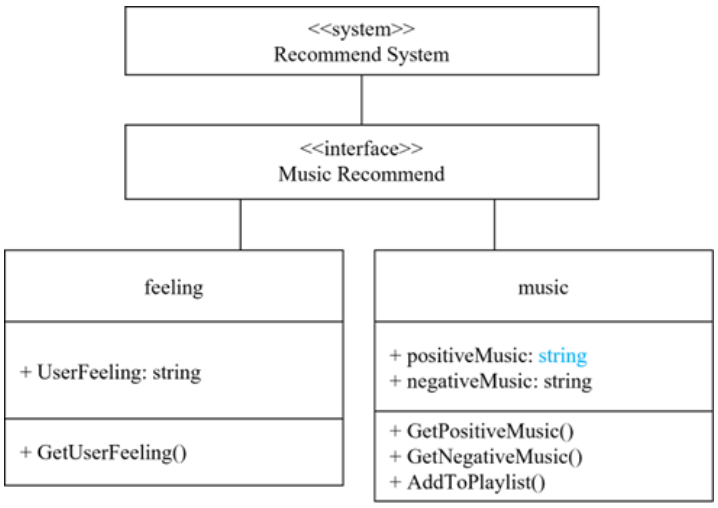
다음은 음악 추천 객체가 가지는 메소드이다.

- GetUserFeeling ()

- GetPositiveMusic ()

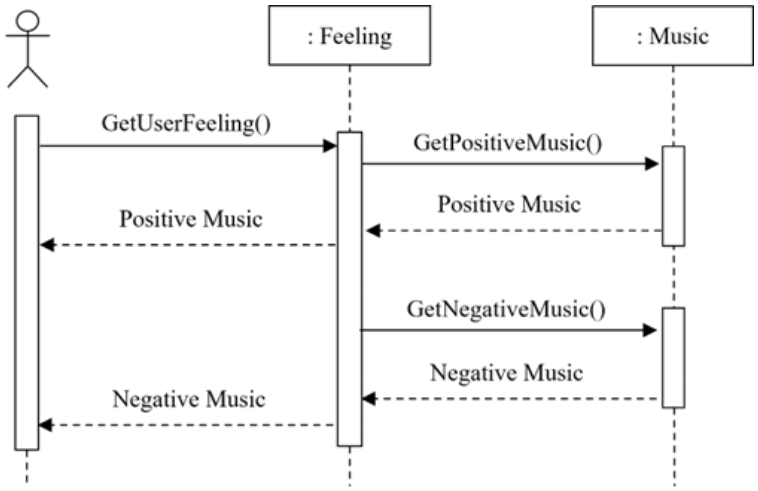
- GetNegativeMusic ()

4.2.6.3. Class Diagram



[Figure 15] Class Diagram - 음악 추천

4.2.6.4. Sequence Diagram



[Figure 16] Sequence Diagram - 음악 추천

4.2.7. 플레이리스트 관리

플레이리스트 관리 클래스는 플레이리스트 시스템을 관리한다. 사용자는 플레이리스트 관리 페이지에서 플레이리스트를 생성하거나 삭제할 수 있고, 이미 존재하는 플레이리스트를 수정할 수 있다.

4.2.7.1. Attributes

플레이리스트 객체의 속성은 다음과 같다.

-Playlist Name: 플레이리스트의 이름이다.

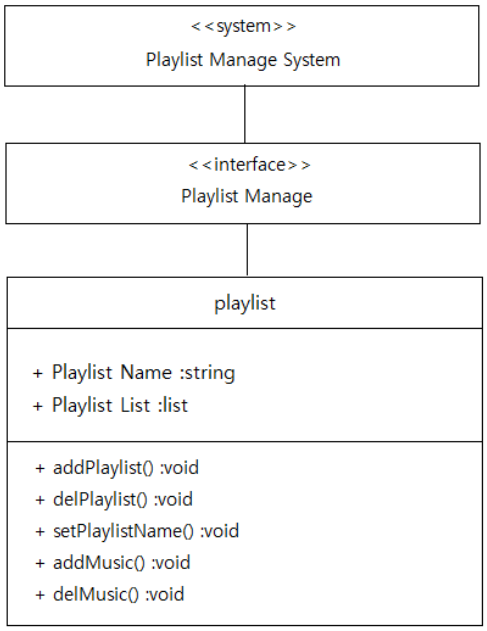
-Playlist List: 플레이리스트들과 그 안의 음악들을 저장하고 있다.

4.2.7.2. Methods

플레이리스트 인터페이스가 가진 메소드들은 다음과 같다.

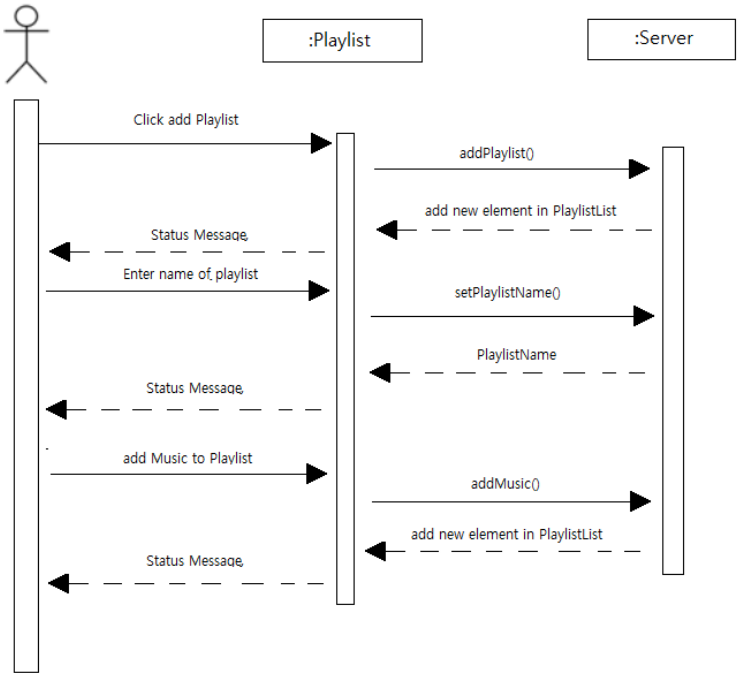
- addPlaylist()
- delPlaylist()
- setPlaylistName()
- addMusic()
- delMusic()

4.2.7.3. Class Diagram



[Figure 17] Class Diagram - 플레이리스트 관리

4.2.7.4. Sequence Diagram



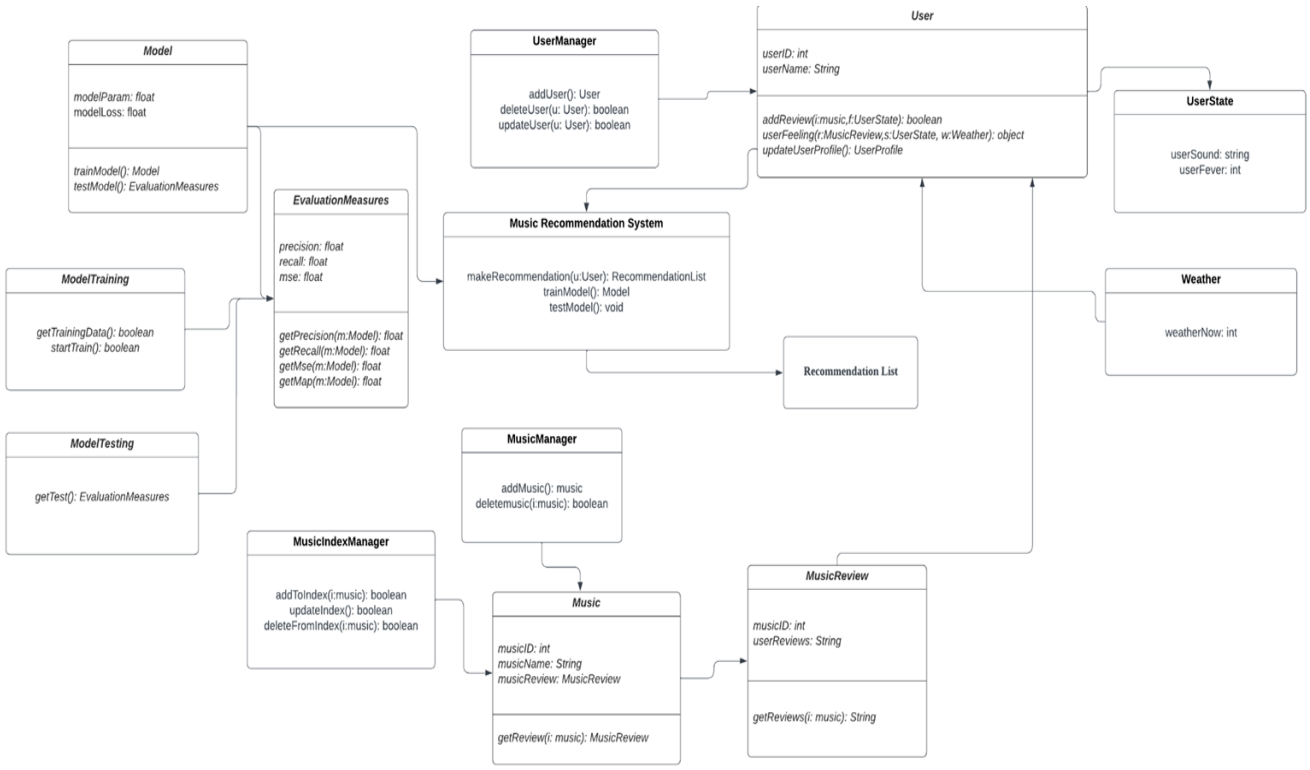
[Figure 18] Sequence Diagram - 플레이리스트 관리

5. System Architecture - Backend

5.1. Objectives

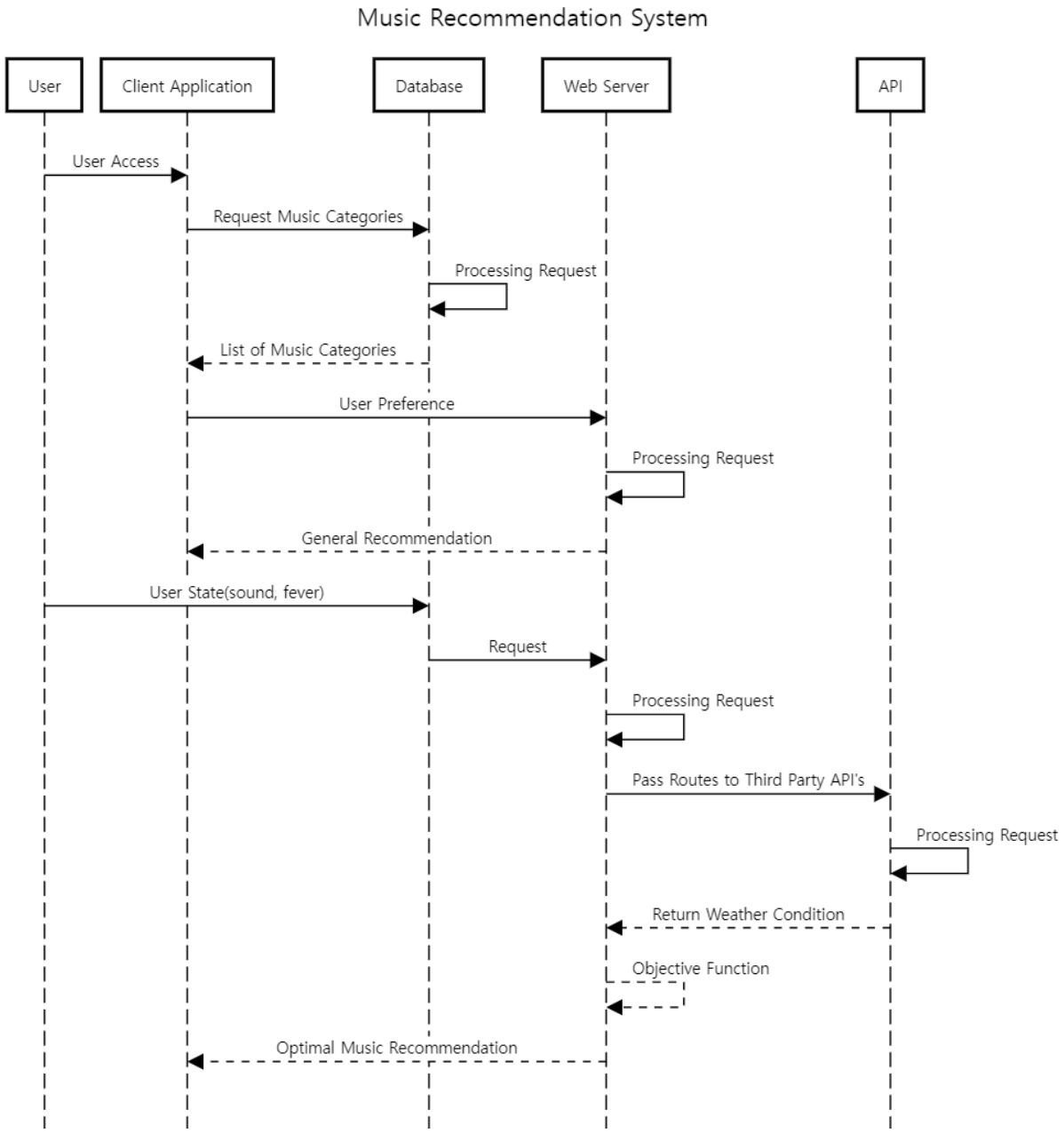
이 장에서는 DB를 포함한 백엔드 시스템의 구조를 설명한다.

5.2. Class Diagram



[Figure 19] Class Diagram

5.3. Sequence Diagram



[Figure 20] Sequence Diagram

6. Testing plan

6.1. Objectives

Testing의 종류에는 development testing, release testing, user testing이 있다. 이 테스트는 잠재적인 에러와 버그를 감지하여 안정적인 배포와 프로그램의 완결성을 도모한다.

6.2. Testing Policy

6.2.1. Development Testing

Development Testing은 버그를 미연에 방지하여 버그로 인한 비용을 줄여준다. 소프트웨어 개발주기 전단계에 걸쳐 테스트가 진행되어 코드의 품질을 높이고 새로운 기능 개발에 필요한 시간을 줄여주는 데 중요한 역할을 한다. 아직 테스팅이 충분히 이루어지지 않은 초기 단계이기 때문에 구성요소들 간의 충돌이 발생할 수 있다. 따라서 정적 코드 분석, 데이터 플로우 분석, 피어 코드 리뷰, 단위 테스팅 등 여러 테스트가 이뤄져야 한다.

6.2.1.1. Performance Testing

Performance Testing은 특정 task가 주어진 상황에서 얼만큼 잘 반응하고 안정적으로 처리하는지 탐색하는 과정이다.

유저의 음성 데이터나 두피열 데이터를 기반 음악 추천 서비스인 'Music Pet'은 사용자 취향에 잘 맞는 음악 추천이 성능을 판가름할 기준이 된다. 사용자의 기분이나 상황에 적합한 음악을 추천하기 위해서는 유저의 취향을 드러낼 수 있는 정보들과 음악의 상관성을 객관적으로 판단할 수 있도록 방대한 데이터가 요구된다. 기존에 가지고 있는 빅데이터들을 빠르게 처리하고 효율적으로 가공할 수 있는 정교한 알고리즘으로 음악 추천 성능을 높일 수 있다.

6.2.1.2. Reliability Testing

Reliability는 development testing에서 중요한 한 축을 다루고 있다. 오작동으로 인해 음성 데이터, 두피열 데이터, 사용자 맞춤리스트가 제거되는 현상이 일어나지 않도록 방지하는 것이 reliability testing의 역할이다.

시스템이 제대로 작동하려면 우선 시스템을 구성하고 있는 unit이 제대로 작동하는지 unit developing testing을 진행해야 하고 이들이 정상적으로 결합되어 있어야 한다. 거기에 추가로 기능이 들어갈 때마다 반복적으로 development testing이 진행되어야 한다.

6.2.1.3. Security Testing

사용자 음성 데이터, 두피열 데이터를 기반으로 한 음악 추천 솔루션을 제공하는데 있어 외부에 개인에 민감한 정보가 유출되지 않도록 관리하는 것은 매우 중요하다. 사용자의 개인정보를 보호하기 위한 강력한 보안 기능이 요구된다.

Security Testing은 데이터를 보호하고 정보를 저장, 처리, 배포하는데 있어서 사용될 보안 메커니즘의 결함을 밝히기 위한 프로세스이다. Security testing는 시스템에 가해질 악의적인 공격으로부터 대비하여 시스템의 잠재적 취약성을 보완하고 사용자 개인정보의 유출을 방지한다. 보안 취약성은 데이터의 유출, 손실, 기업 이미지에 대한 손실로 이어질 수 있는 만큼 많은 노력이 요구된다.

6.2.2. Release Testing

Release Testing는 소프트웨어 릴리즈 후보가 사용자에게 준비되었다는 확신을 팀에 제공하는 코딩 관행 및 테스트 전략을 말한다. Release Testing는 소프트웨어 릴리즈에서 오류와 버그를 찾아 제거하여 사용자에게 릴리즈할 수 있도록 하는 것을 목표로 한다.

Release Testing는 소프트웨어 릴리즈 후보를 만들면서 시작한다. 릴리즈 후보는 이 릴리즈에 대해 선택한 모든 새 소프트웨어 기능과 버그 수정을 포함하는 코드 기준의 스냅샷을 나타낸다. 이 소프트웨어 생산 준비 완료된 상태로 패키지화 되어 있다. 사용자가 사용할 수 있도록 테스트하고자 하는 것은 이 소프트웨어 릴리즈 후보이다.

다른 테스트 접근법과 마찬가지로 Release Testing는 통제된 환경에서 소프트웨어 릴리즈 후보를 실행하는 시스템을 파괴하는 것을 목표로 한다. 개발팀은 이 한 릴리즈 후보를 개선할 수 있도록 노력해 실제 사용자에게 전달되기 전에 문제가 수정할 수 있도록 할 것이다.

6.2.3. User Testing

사용자 테스트는 실제 사용자가 어떻게 시스템을 이용하는지 확인할 수 있는 중요한 테스트이다. 사전에 노출되지 않은 사용자들을 대상으로 테스트를 진행할 것이다.

6.2.4. Testing Case

테스트 케이스는 가장 기본적인 기능들인 음악 재생과 음악 추천, 플레이리스트 관리를 포함하고 있으며 상황 설정, 스피커 장치 추가와 같은 부가 기능을 포함할 예정이다. 또한 세 번 이상의 테스트를 진행한 후 사용자들에게서 피드백을 받아 반영할 것이다.

7. Development Plan

7.1. Objectives

이 장에서는 애플리케이션 개발을 위해 필요한 기술과 환경에 대해 설명한다.

7.2. Frontend Environment

7.2.1. Adobe Illustrator

[Figure 21] Adobe Illustrator

Adobe Illustrator는 벡터 드로잉 프로그램으로 쉬운 그래픽 작업이 가능하다. 성공적인 브랜딩과 시각적으로 편안한 서비스의 제공을 위해 아이콘과 로고 작업이 필요하므로, Adobe Illustrator를 이용해 Music Pet에서 사용될 아이콘 및 로고를 제작한다.

7.2.2. Android Studio

[Figure 22] Android Studio

Android Studio는 안드로이드 및 안드로이드 전용 애플리케이션을 제작하기 위한 공식 통합 개발 환경이다. 다양한 프로그래밍 언어를 지원하고 IntelliJ IDEA 인터페이스를 갖춘 지능형 코드 편집기를 특징으로 하여 개발자들이 더 쉽고 정확하게 코드를 만들 수 있도록 한다. 프론트엔드 개발을 위해 XML 레이아웃을 제공하여 가시적으로 응용프로그램의 구조를 구축할 수 있다. 해당 툴을 통해 안드로이드 버전 Music Pet의 UI 레이아웃과 구조를 만든다.

7.2.3. Xcode

[Figure 23] Xcode

Xcode는 iOS 애플리케이션을 제작하기 위한 개발 툴 모음으로, iOS를 위한 프로그래밍 언어인 Swift를 사용한다. Xcode도 Android Studio처럼 코드 구현 및 사용자 인터페이스 디자인을 쉽게 할 수 있다. 해당 툴을 통해 iOS 버전 Music Pet의 UI와 구조를 만든다.

7.3. Backend Environment

백엔드 환경의 전반적인 환경적 요소들은 다음과 같다.

7.3.1. MySQL

[Figure 24] MySQL

MySQL은 인기 있고 사용하기 편한 데이터베이스이다. “Music Pet” 프로젝트를 진행하는데 있어 MySQL이 줄 수 있는 이점은, 첫째 보안 수준이 높은 데이터베이스라는 것이다. MySQL은 설치 시 암호 보안 수준을 설정할 수 있고 루트 사용자의 암호를 정의하며 익명계정을 제거하고 모든 사용자가 접근할 수 있는 기본 테스트 데이터베이스를 제거하여 보안을 향상하는데 도움이 되는 스크립트와 함께 설치되기 때문에 보안을 중요시하는 “Music Pet” 프로젝트를 진행하는데 있어 이점이 있다. 둘째, MySQL은 속도가 빠른 데이터베이스이다. 빠른 속도는 개발자와 사용자 모두에게 장점이 있다. 마지막으로 MySQL은 복제 기능을 지원하여 안정성, 가용성, 내결함성을 향상할 수 있다.

7.3.2. Firebase

[Figure 25] Firebase

파이어베이스는 클라우드 스토리지, 실시간 데이터베이스, 머신 러닝 키트 등 다양한 기능을 제공해 모바일과 웹 애플리케이션 개발을 지원한다. “Music Pet”은 파이어베이스의 실시간 데이터베이스, 머신 러닝 키트를 활용하여 음악 추천 기능을 개발한다.

7.3.3. Nodejs

[Figure 26] nodejs

확장 가능한 네트워크 애플리케이션(서버)를 개발하는 데 사용되는 소프트웨어 플랫폼이다. 자바스크립트를 문자 언어로 사용하며 내장 HTTP 서버 라이브러리를 포함한다. 이를 통해 웹 서버가 Apache와 같은 별도의 소프트웨어 없이 작동할 수 있게 하여 웹 서버의 동작을 보다 효과적으로 제어할 수 있다.

7.4. Constraints

이 소프트웨어는 문서에서 언급한 내용을 토대로 구현되어진다. 전반적인 틀은 유지하되, 세부적인 요소는 개발자가 선호하는 방식으로 유연하게 진행한다.

- 시스템 비용 및 유지보수 비용을 고려한다.
- 전반적인 시스템 성능 개선을 모색하는 방향으로 결정한다.
- 향후 유지보수를 고려하고 소스코드를 작성할 때 충분한 주석을 추가한다.
- 보다 사용자 친화적이고 편리한 방향으로 개발한다.
- 오픈 소스를 가능한 많이 활용한다.

- 서버 과부하 없이 기능을 구현한다.
- 시스템 리소스 낭비를 방지하기 위해 소스 코드를 최적화한다.
- 미래의 시스템 확장성 및 가용성을 고려한다.
- Android 6.0(API 23) 이상, IOS 13.0 이상의 버전을 대상으로 개발한다.
- Windows 10 환경에서 Android Studio를 통해 개발한다.
- IOS 환경에서 swift를 통해 개발한다.

7.5. Assumptions and Dependencies

이 문서의 모든 시스템은 Android 장치와 IOS를 기반으로 설계 및 구현됩니다. 모든 콘텐츠는 최소 API 23 이상을 갖춘 안드로이드 운영 체제와 IOS 13.0을 기반으로 하며 다른 운영체제나 버전에는 적용되지 않을 수 있다.

8. Supporting Information

8.1. Software Design Specification

이 소프트웨어 디자인 명세서는 IEEE Recommendation(IEEE Recommended Practice for Software Design Description, IEEE-Std-1016)에 따라 작성되었다.

8.2. Document History

[Table 1] Document History

날짜	버전	설명	글쓴이
2022/05/06	0.1	명세서 작성 시작	전체
2022/05/11	1.0	1,2장 작성	임형선
2022/05/11	1.1	3장 작성	송홍빈
2022/05/11	1.2	4장 작성	김지현, 장준

2022/05/12	1.3	5장 작성	송홍빈
2022/05/12	1.4	6장 작성	임형선, 장준
2022/05/12	1.5	7장 작성	송홍빈, 김지현
2022/05/13	2.0	리뷰	전체