SONGSTERR

-Design Specification-

2016310936 우승민

Contents

1. Preface 5

1.1 Objective 5

1.2 Readership 5

1.3 Documents Contents 6

A. Introduction 6

B. System Architecture 6

C. Protocol Design 6

D. Database Design 6

E. Testing Plan 6

F. Development Plan 6

G. Index 6

2. Introduction 7

2.1 Objectives 7

2.2 Applied Diagram 7

3. System Architecture – Overall 10

3.1 Objectives 10

3.2 Fronted Architecture 10

3.4 Search System 11

3.5 Favorite / History System 11

3.6 Popular System 12

4. System Architecture - Fronted 13

4.1 objectives 13

4.2. Subcomponents 13

5. System Architecture - backend 20

5.1 objectives 20

5.2. Overall Architecture 20
5.3. Subcomponents 20
6. Protocol Design 23
6.1 Objectives 23
6.2. REST API 23
6.3. JSON 24
6.4. Details 24
A. Authentication 24
B. User 25
C. Score 27
7. Database Desing 28
7.1 Objectives 28
7.2 E-R Diagram 28
7.3 Relational Schema 30
7.4 SQL DDL 31
8. Testing Plan 33
8.1. Objectives 33
8.2. Testing policy 33
A. Development Testing 33
B. Release Testing 33
C. User Testing 33
D. Testing Case 34
9. Development Plan 35
9.1. Objectives 35
9.2. Frontend Environment 35
9.3. Backend Environment 36

10. Index 37

10.1 Figures 37

10.2 Diagrams 37

1. Preface

1.1 Objective

본 Preface에서는 이 문서의 예상 독자를 설정하며, 문서의 전반적인 내용의 세부사항들에 대한 간략한 개요를 제공한다.

1.2 Readership

본 문서는 엔지니어를 비롯한 다양한 독자층에게 읽힐 것을 산정한다.

1.3 Documents Contents

A. Introduction

본 문서에 사용된 다이어그램과 표현 도구들에 대해 설명하고, system의 목표 및 전반적인 개발범위에 대해 서술한다.

B. System Architecture

System과 system을 이루는 각 sub-system의 구조와 세분화된 기능에 대해 전반적으로 설명한다.

C. Protocol Design

Frontend system 와 backend system 간의 상호작용을 이루는 인터페이스와 해당 인터페이스에 사용된 기술에 대해 설명한다.

D. Database Design

Requirements specification에 작성된 database 요구 사항을 바탕으로, 각 data의 관계를 E-R diagram을 통해 표현하고, Relational Schema, SQL DDL을 작성한다.

E. Testing Plan

본 system의 개발 이후 진행될 test의 계획에 대해 설명한다.

F. Development Plan

본 system을 개발하는데 필요한 도구, 라이브러리 등의 기술과 개발 계획에 대해 설명한다.

G. Index

본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

2. Introduction

2.1 Objectives

본 문서에 사용된 다이어그램과 표현 도구들에 대해 설명하고, system의 목표 및 전반적인 개발범위에 대해 서술한다.

2.2 Applied Diagram

A. UML Diagram

UML Diagarm 은 전반적인 모델, 시스템의 관계들을 나타내기 위해 사용되는 개발 모델링 언어이다. System 의 전반적인 Workflow 를 시각화해준다.

다양한 기호와 정의를 포괄하고 있어 개발자와 사용자간의 커뮤니케이션 수단을 제공하는 데 효율적으로 활용할 수 있다. Package Diagram, Class Diagram, E-R Diagram 로 구성된다.

B. Package Diagram

Package Diagram 은 구조 다이어그램으로 system 내 요소들의 배열과 구성을 보여주며 sub-system, package 간의 상호관계를 표현해준다.

C. Class Diagram

Class Diagram 은 object classes 와 classes 간의 상호관계를 보여주는데 사용된다. 각 objects 가 사용하는 variables, methods, links 들로 구성된다.

D. Sequence Diagram

Sequence Diagram 은 사용자와 objects 간의 상호작용을 나타낸다. 특정 function 이나 동작에 따라 발생하는 프로세스의 순서를 제시하여 동작이 이루어지는 과정을 볼 수 있다.

E. E-R Diagram

E-R diagram 은 각 entity 가 갖고 있는 속성과 관계에 대해 나타낸다. Database 를 설계하는데 사용되며, Relational Schema, SQL DDL 을 작성하는데 기반이 된다.

2.3 Applied Tool

A. Draw.io

Draw.io는 온라인 모델링 툴로 많은 기본 템플릿과 도형을 제공한다. 대부분의 다이어그램을 제작하는데 사용되었다.

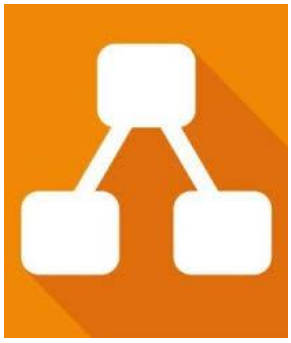

Figure 1. Draw.io Logo

B. Power Point

Power Point는 주로 presentation 자료를 제작하는데 사용되지만, 내장된 도형 기능이 다양하여 diagram을 제작하는 것 또한 유용하다.

Figure 2. Power Point Logo

2.4 Project Scope

본 시스템은 취미생활을 새롭게 배우기 어려운 상황에서 기타를 비롯한 몇 악기를 쉽게 배울 수 있는 환경을 제공하여 새로운 취미생활을 갖게 해주는 것이 주 목표이다. 본 시스템의 핵심 기능은 노래를 재생하는 동시에 악보의 위치를 움직이며 사용자가 따라하기 쉽게 해주는 것이다. 이 기능을 중심으로 추가적인 편의성을 제공하도록 설계하였다.

먼저 Frontend System 은 사용자와의 상호작용을 담당하며, Backend System 은 Frontend System 에서 전달받은 데이터 요청을 수행한다. 만약 사용자가 Fronted 에서 원하는 노래의 제목을 검색하면 Backend 에서 전달받아 system 의 database 에 있는 악보들과 비교하여 충족하는 악보들을 Fronted 에 넘기고 사용자에게 보여줄 수 있게 해준다.

3. System Architecture – Overall

3.1 Objectives

시스템의 architecture의 전반적인 구조에 대해 서술한다. 시스템의 전체적인 구조 및 sub-system의 구성 및 관계에 대해 설명한다.

3.2 Fronted Architecture

사용자가 직접 보고 사용하는 영역으로 Sign in, Search, Favorite/History/Popular, Play(추가기능 포함)로 크게 4개의 시스템으로 전체적인 시스템이 구성된다. User가 Fronted에 요청한 내용을 Backend(Server)로 넘겨 필요한 data를 Database에서 넘겨 받습니다.

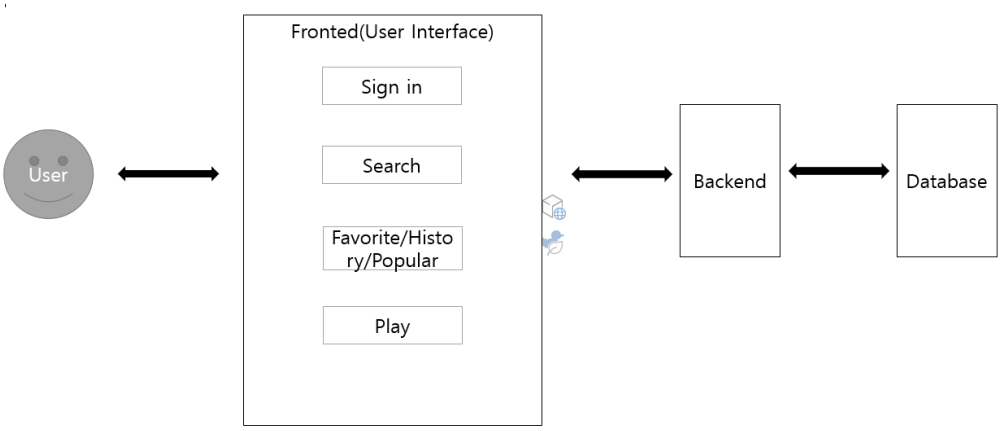

Diagram 1. Fronted Architecture

3.3 Backend

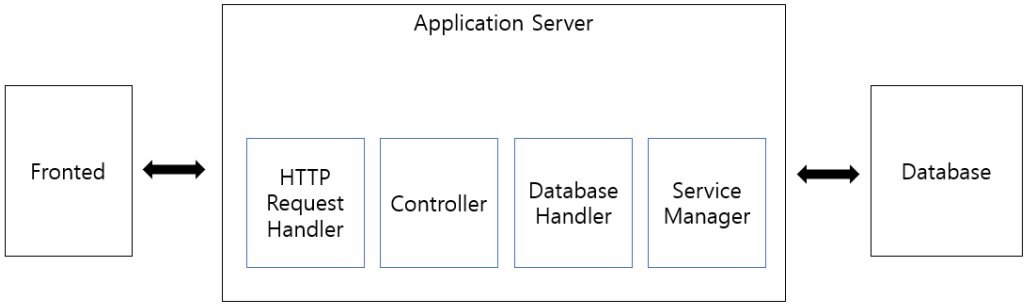

Diagram 2. Backend Architecture

3.4 Search System

사용자가 원하는 곡의 제목이나 음악가의 이름을 입력하여 악보를 찾는 system이다. User가 server에 keyword를 입력하면 server에서는 Database에 해당 keyword가 포함된 악보가 있는지 확인하고 있으면 user에게 목록을 보여주고 없으면 빈 화면을 보여준다.

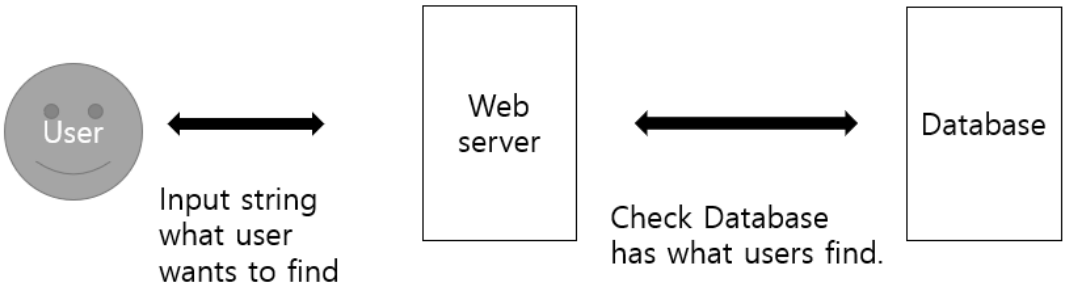

Diagram 3. Search System architecture

3.5 Favorite / History System

사용자가 search system을 통해 찾은 악보의 데이터를 database에 저장하여 따로 목록을 만드는 system이다. Favorite의 경우는 사용자가 표시를 한 악보를 따로 모아두고 History는 최근 본 목록을 FIFO 방식으로 저장한다. 해당 system은 모바일 디바이스의 cache에도 악보를 저장하여 오프라인 상에서도 볼 수 있게 만들었다.

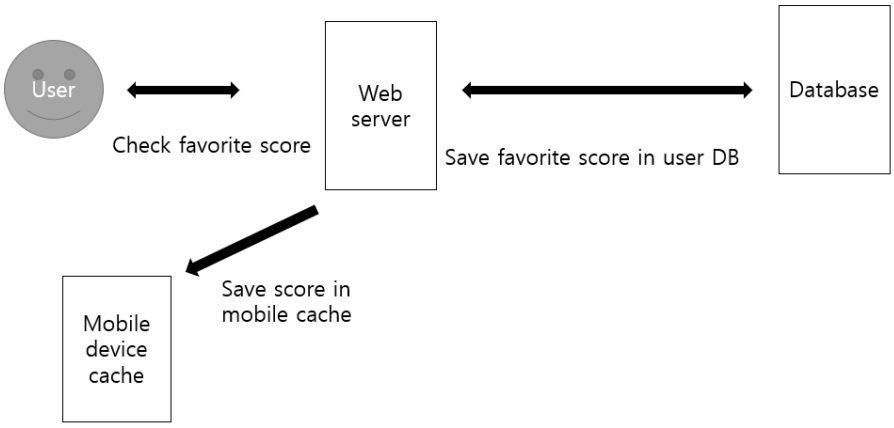

Diagram 4. Favorite System architecture

3.6 Popular System

Search system을 통해서 user들이 본 악보들의 정보를 database에서 가장 많이 본 악보를 순위별로 보여주는 System이다. 특별히 찾는 곡이 없는 user들이 인기있는 음악이 무엇인지 보고 연주해 볼 수 있게 해준다.

4. System Architecture - Fronted

4.1 objectives

전체 시스템의 architecture 중 사용자와 상호작용을 하는 Fronted system의 구조와 각 컴포넌트의 구성 및 관계에 대해 설명한다.

4.2. Subcomponents

A. Search

1. Class Diagram

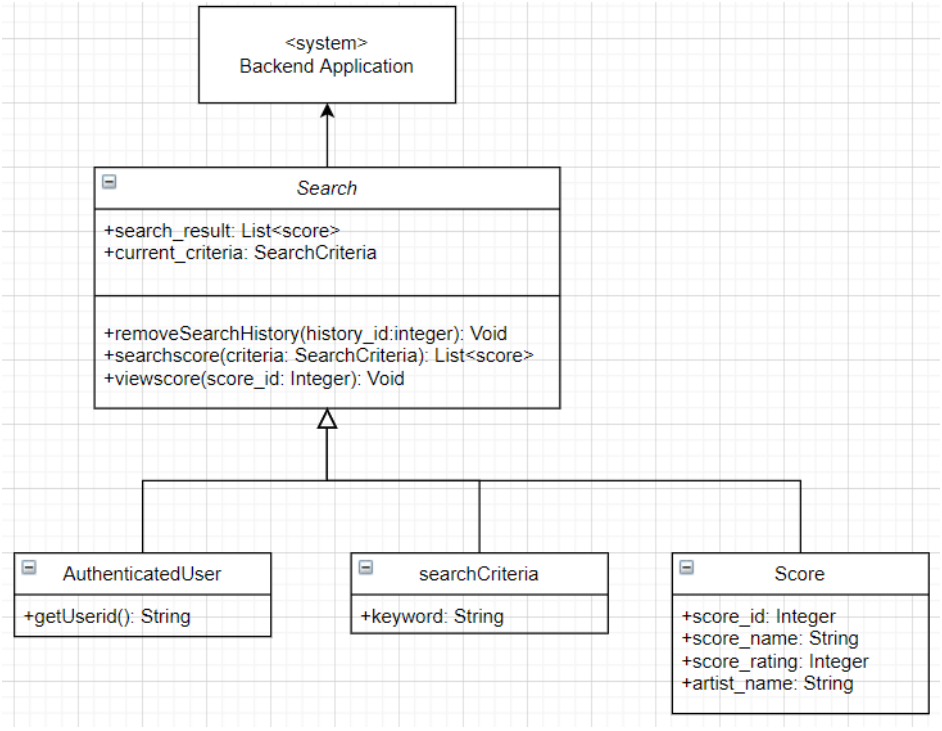

Diagram 5. Search Class Diagram

1) Search– 객체

A. attributes

+ search_result: 검색 조건에 따른 검색 결과 목록

+ current_criteria: 현재 검색 조건

B. methods

+ removeSearchHistory(history_id: Integer): 검색 기록 삭제

+ searchScore(criteria: SearchCriteria): 악보를 검색

+ viewscore(score_id: Integer): 특정 악보를 선택하여 본다. (Score 컴포넌트로 이동)

AuthenticatedUser – 인증 유저 객체

A. attributes

+ getUserid: 유저의 id를 입력 받는다.

3) SearchCriteria

A. attributes

+ keyword: 검색 조건으로 쓰인 문자열

4) Score – 악보 객체

A. attributes

+ score_id : 악보의 id

+ score_name : 악보의 이름

+ score_rating : 악보의 난이도

+ artist_name : 작곡가(가수)의 이름

B. History page

1. Class Diagram

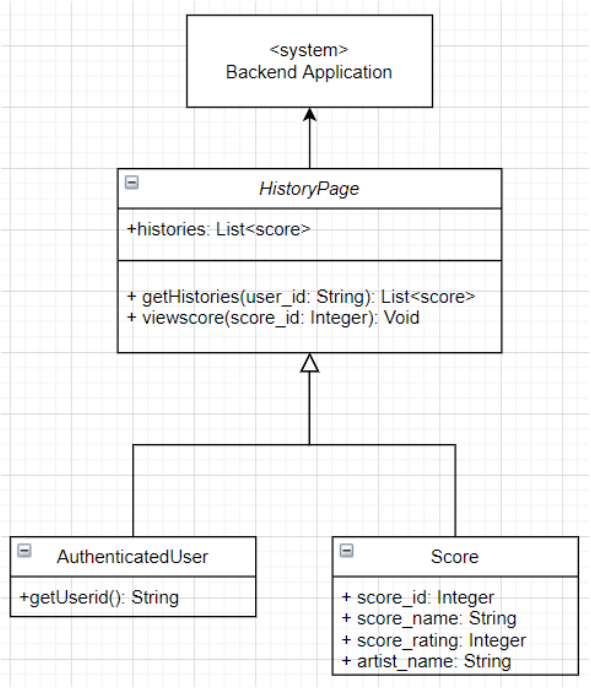

Diagram 6. History Class Diagram

1) HistoryPage – 객체

A. attributes

+ Histories: history page내에 로드되어 있는 악보 목록

B. methods

+ getscores(criteria: Criteria): 사용자가 등록한 악보 목록을 backend server에서 가져온다.

+ viewsocre(score_id: Integer): 특정 악보를 선택하여 본다. (Score 컴포넌트로 이동)

2) AuthenticatedUser – 인증 유저 객체

A. attributes

+ getUserid: 유저의 id를 입력 받는다.

3) Score – 악보 객체

A. attributes

+ score_id : 악보의 id

+ score_name : 악보의 이름

+ score_rating : 악보의 난이도

+ artist_name : 작곡가(가수)의 이름

2. Sequence Diagram

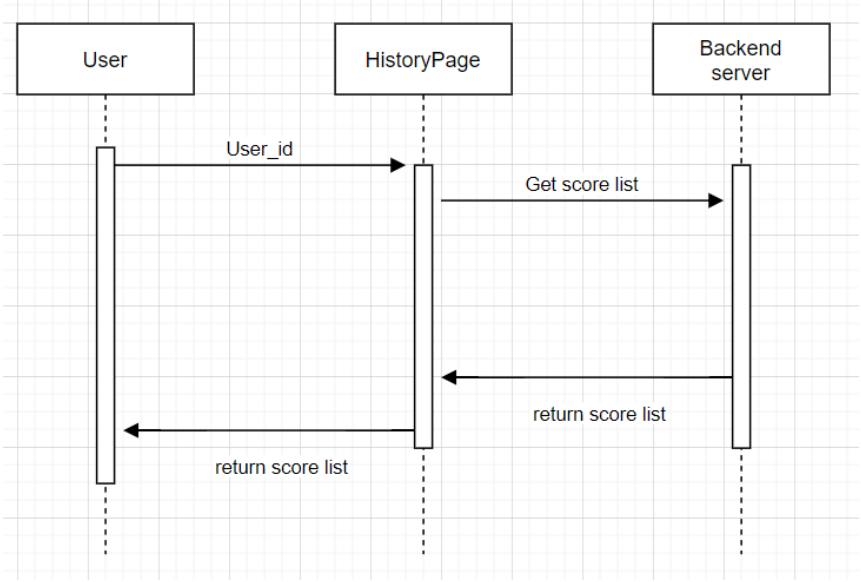

Diagram 7. History Sequence Diagram

C. Favorite page

1. Class Diagram

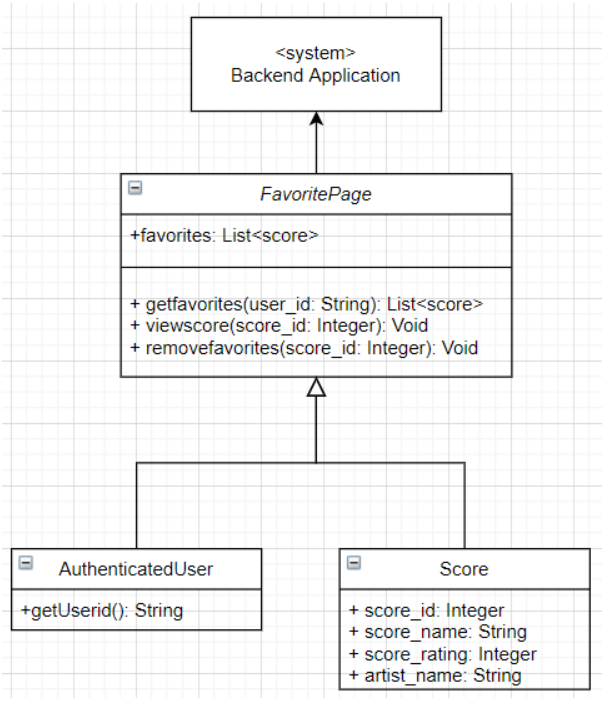

Diagram 8. Favorite Class Diagram

1) FavoritePage – 객체

A. attributes

+ favorites: favorite page내에 로드되어 있는 악보 목록

B. methods

+ getscores(criteria: Criteria): 사용자가 등록한 악보 목록을 backend server에서 가져온다.

+ viewsocre(score_id: Integer): 특정 악보를 선택하여 본다. (Score 컴포넌트로 이동)

2) AuthenticatedUser – 인증 유저 객체

A. attributes

+ getUserid: 유저의 id를 입력 받는다.

3) Score – 악보 객체

A. attributes

+ score_id : 악보의 id

+ score_name : 악보의 이름

+ score_rating : 악보의 난이도

+ artist_name : 작곡가(가수)의 이름

2. Sequence Diagram

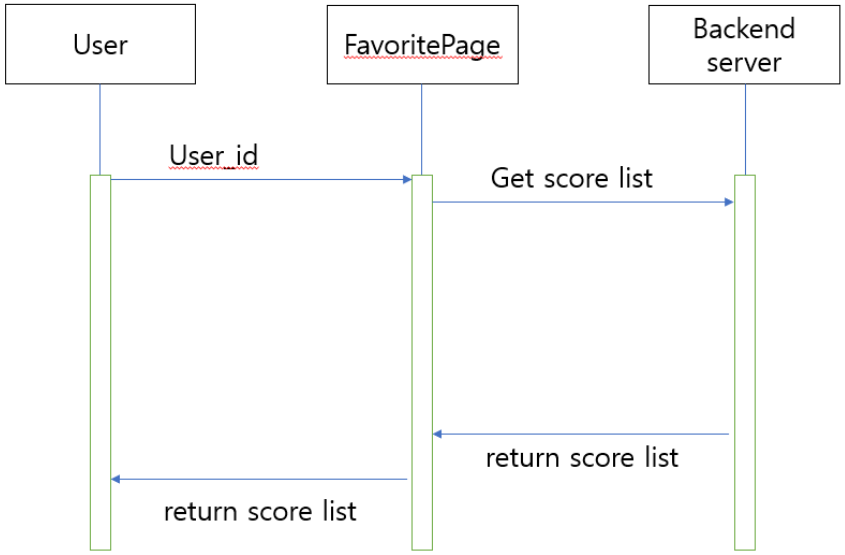

Diagram 9. Favorite Sequence Diagram

D. Popular page

1. Class Diagram

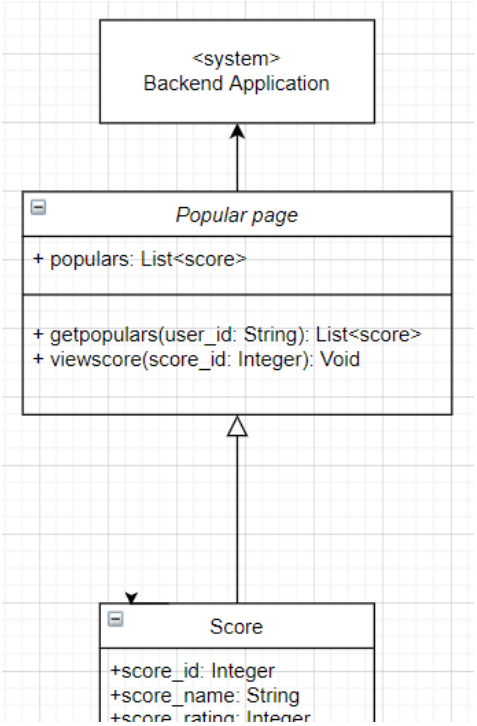

Diagram 10. Popular Class Diagram

1) Popularpage – 객체

A. attributes

+ populars: popular page내에 로드되어 있는 악보 목록

B. methods

+ getscores(criteria: Criteria): 사용자가 등록한 악보 목록을 backend server에서 가져온다.

+ viewsocre(score_id: Integer): 특정 악보를 선택하여 본다. (Score 컴포넌트로 이동)

2) Score – 악보 객체

A. attributes

+ score_id : 악보의 id

+ score_name : 악보의 이름

+ score_rating : 악보의 난이도

+ artist_name : 작곡가(가수)의 이름

5. System Architecture - backend

5.1 objectives

전체 시스템의 architecture 중 사용자와 상호작용을 하는 Fronted system을 제외한 backend system 및 sub-system에 대해 설명한다.

5.2. Overall Architecture

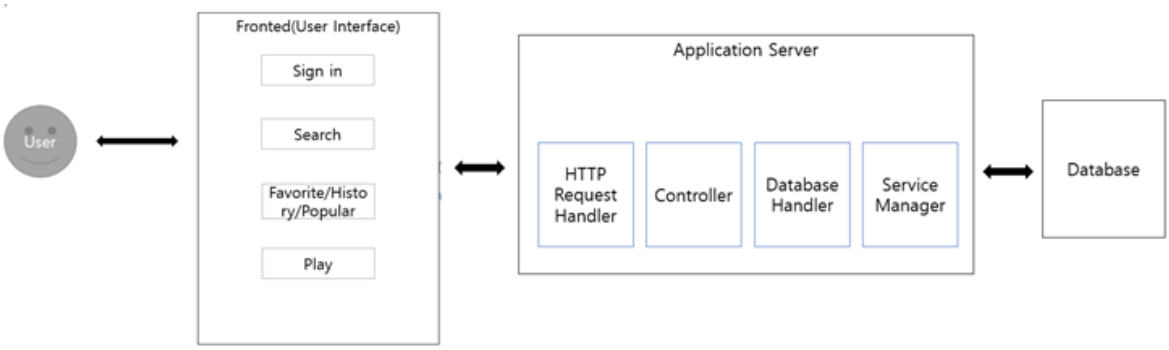

Diagram 11. System Overall Architecture

5.3. Subcomponents

A. Application Server

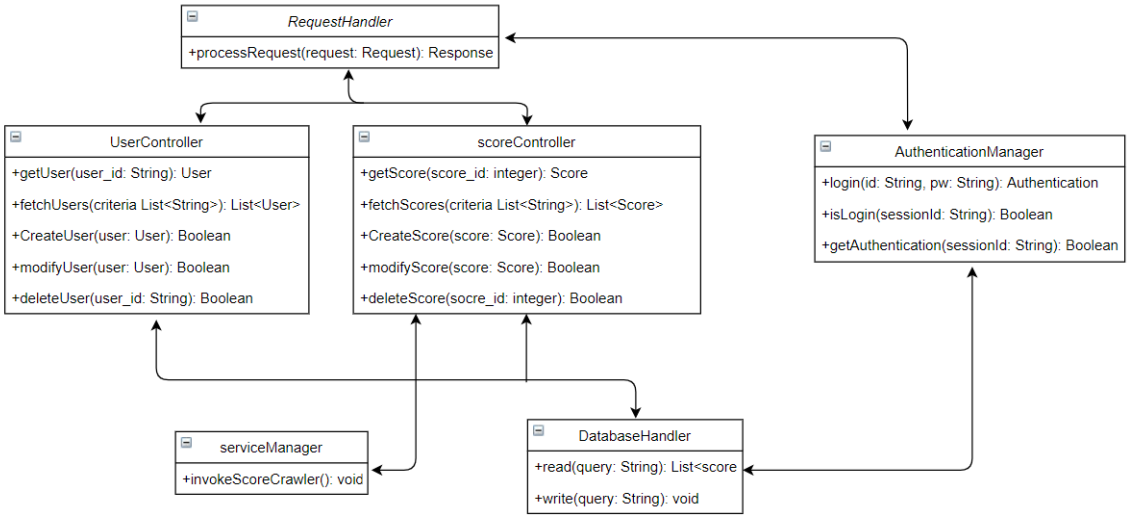

Diagram 12. Application Server Class Diagram

1) Request Handler: Frontend 의 요청 처리

A. attributes: 없음

B. processRequest(request: Request): controller 에 요청을 나누어 처리하고 반환한다.

2) Controllers

A. attributes: 없음

B. getEntity(entity_id): entity 를 가져오는 method

C. fetchEntities(criteria: List<String>) 검색 조건에 맞는 entity list 를 가져오는 method

D. createEntity(entity: Entity): entity 를 생성하는 method

E. modifyEntity(entity: Entity): entity 를 수정하는 method

F. deleteEntity(entity_id): entity 를 삭제하는 method

3) Service Manager

A. invokeScoreCrawler(): Score crawler 를 실행하는 method

4) Authentication Manager

A. login(id: String, pw:String): ID/PW 로 login 하고 인증 토큰을 발급하는 method

B. isLogin(token: String): 인증이 필요할 때 사용자의 로그인 토큰을 확인하는 method

C. getAuthentication(sessionId; String): 로그인 토큰을 바탕으로 사용자의 정보를 불러오는 method

5) Database Handler

A. read(queryL: String): database 조회 query string 을 database 에 질의한다.

B. write (queryL: String): database 입력 query string 을 database 에 질의한다.

B. Score Crawling System

1) class diagram

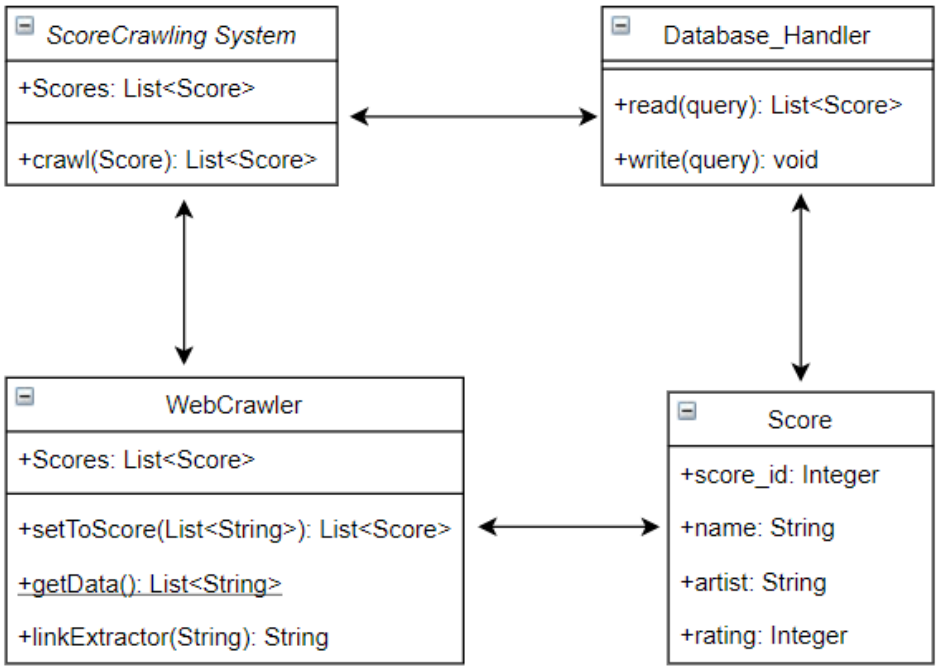

Diagram 13. Score Crawling System Class Diagram

2) sequence diagram

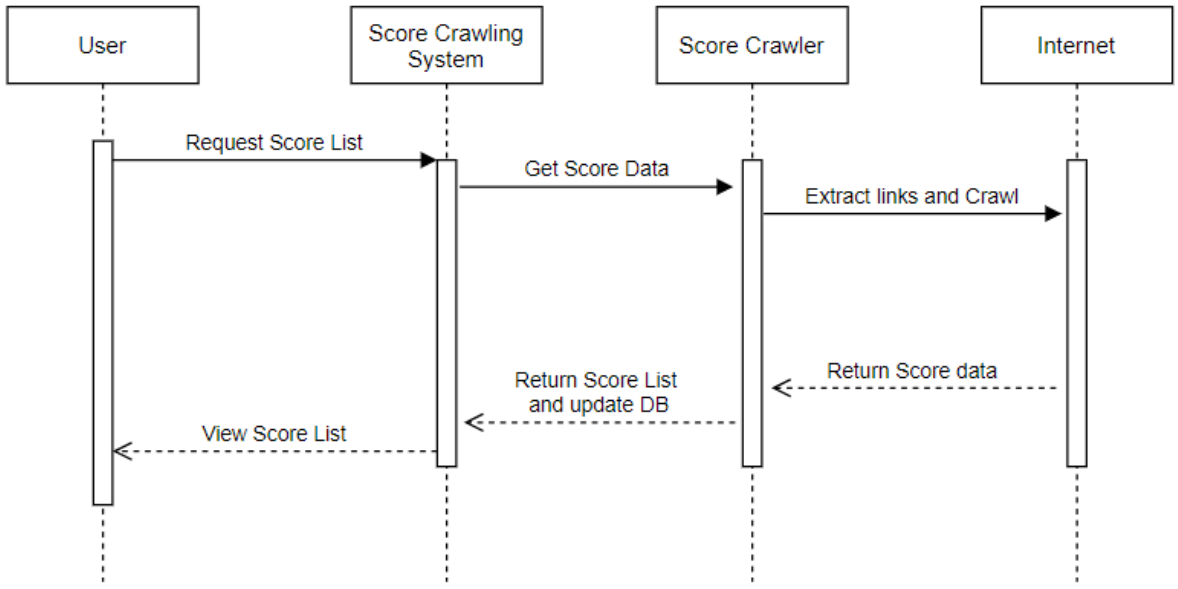

Diagram 14. Score Crawling System Sequence Diagram

6. Protocol Design

6.1 Objectives

Frontend system과 Backend system의 상호작용에 이용되는 protocol의 구조에 대해 설명하고, 각 인터페이스에 대해 기술한다.

6.2. REST API

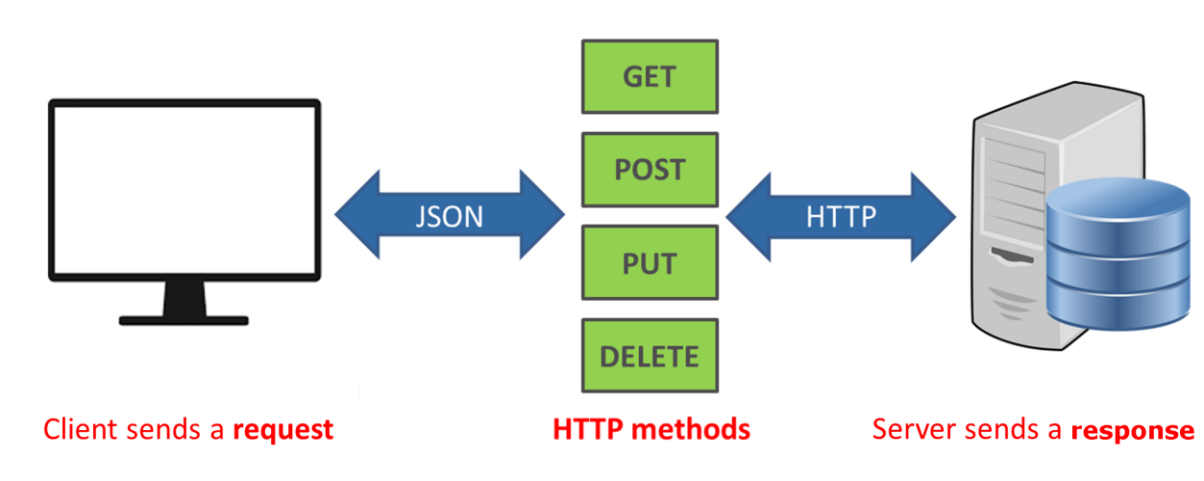

Figure 3. REST API

본 시스템은 Frontend 와 Backend 사이의 통신으로 HTTP 를 이용하고, 형식은 REST API 형식을 사용한다. REST API 란 'Representational State Transfer'로 서버에 저장된 자원들을 이름으로 구분하여 해당 자원의 상태를 주고받는 API 형식이다.

REST API 는 HTTP URI, HTTP methods, JSON 3 가지를 사용한다. HTTP URI 은 서버가 보관하는 데이터를 나타낸다. HTTP methods 는 서버의 자원에 접근하여 상태를 조작하기 위해 요청하는 행위로 GET, POST, PUT, DELETE 4 가지가 있다. 마지막으로 JSON 은 client 의 요청에 대한 서버의 응답 형식으로 사용된다.

REST API 를 사용하면, server 와 client 사이의 의존성이 줄고, 본 시스템을 포함하여 다른 시스템에서의 자원 또한 쉽게 이용할 수 있다.

6.3. JSON

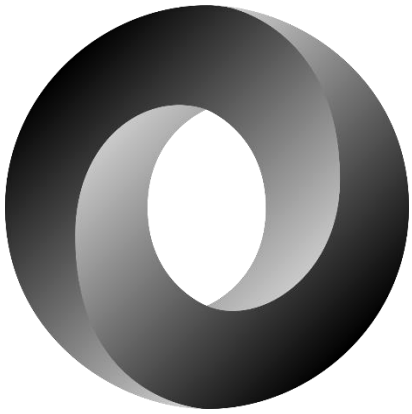

Figure 4. JSON

JSON(JavaScript Object Notation)은 "속성-값 쌍" 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML(AJAX가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수 값을 표현하는 데 적합하다.

6.4. Details

A. Authentication

a. Sign in

- Request

Method	POST	
URI	/authentication/signin	
Request Body	Id	사용자 ID
	Password	사용자 password

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (ID/PW 불일치)	
Success response	success	true
Failure response	success	False
	message	Reason of failure (없는 ID 혹은 ID와 PW 불일치)

B. User

a. Get

- Request

Method	GET	
URI	/user/:id	
Parameters	-	-
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (ID에 해당하는 user가 없음)	
Success response	user	User object
Failure response	-	-

b. Add/ Remove Favorite

- Request

Method	POST/ADD || post/DELETE	
URI	/user/:id/favorites	
Parameters	Score_id	악보 ID
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	-	
Success response	-	-
Failure response	-	-

c. Add history

- Request

Method	POST/ADD	
URI	/user/:id/histories	
Parameters	Score_id	악보 ID
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	404 NOT Found (검색 조건에 해당하는 score가 없음)	
Success response	scores	Score List
Failure response	-	-

C. Score

a. Search

- Request

Method	GET	
URI	/scores	
Parameters	Score_id	악보 ID
	Score_name	악보 이름
	Score_rating	악보 난이도
	Artist_name	음악가 이름

- Response

Success Code	200 OK	
Failure Code	404 NOT Found (검색 조건에 해당하는 score가 없음)	
Success response	scores	Score List
Failure response	-	-

b. Popular

- Request

Method	POST/ADD	
URI	/scores/popular	
Parameters	Score_id	악보 ID

- Response

Success Code	200 OK	
Failure Code	404 NOT Found (검색 조건에 해당하는 score가 없음)	
Success response	scores	Score List
Failure response	-	-

7. Database Desing

7.1 Objectives

세부적인 데이터베이스의 설계에 대해 설명한다. E-R Diagram과 Relational Schema로 Entity간의 관계를 도식화하고, SQL DDL로 value의 데이터 타입을 명시한다.

7.2 E-R Diagram

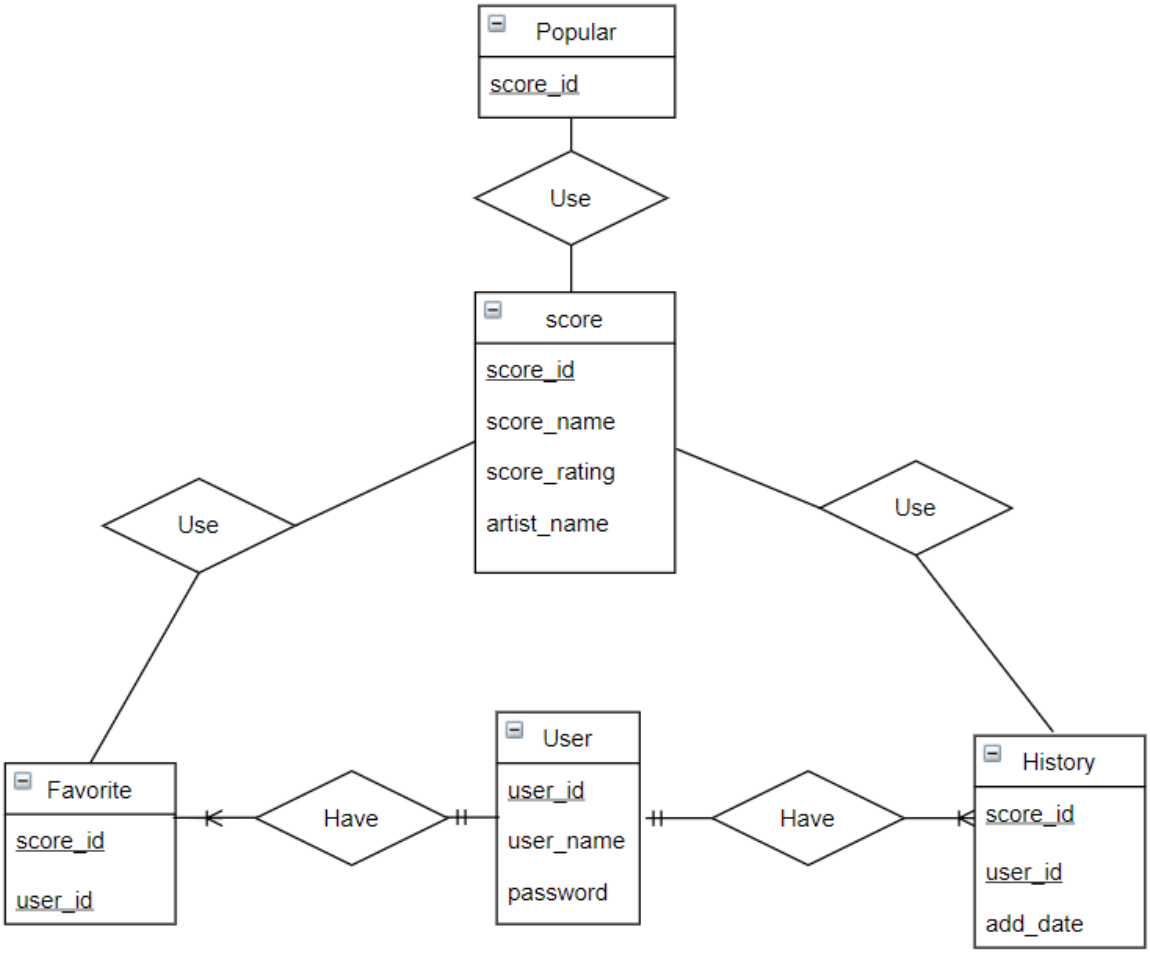

Diagram 15. Overall E-R Diagram

본 시스템에는 User, Score, Favorite, History, Popular 총 5개의 Entity가 존재한다. 각각의 Entity는 네모 박스의 형태로 표현되고 해당 Entity의 Key에 해당하는 Attribute들은 아래 박스에 표현된다. Primary Key는 밑줄로 표시한다. Entity 관계는 줄과 마름모를 사용하여 표현한다. 특정 Entity가 다른 Entity와 복수의 관계를 가질 수 있을 때는 해당 Entity 쪽으로 삼지창 모양의 선을 세 개 그어 표현한다.

A. User

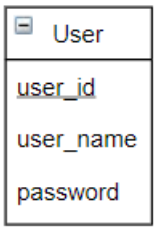

Diagram 16. User Entity

User Entity는 사용자의 정보를 표현한다. user_id 속성이 primary key이며, 이름, 패스워드 정보를 가지고 있다

B. Score

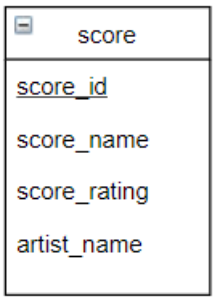

Diagram 17. Score Entity

Score Entity는 악보에 대한 정보를 가지고 있으며, primary key 는 score_id 이다. 악보 제목, 난이도, 음악가 이름 정보를 가지고 있다.

C. Favorite

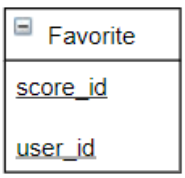

Diagram 18. Favorite Entity

Favorite Entity는 user가 favorite으로 등록한 악보에 대한 정보를 가지고 있으며, primary key 는 score_id, user_id 두개 모두이다.

d. History

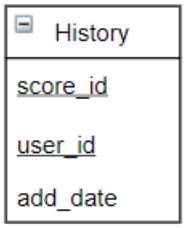

Diagram 19. Histoy Entity

History Entity는 user가 열람한 악보에 대한 정보를 가지고 있으며, primary key 는 score_id, user_id 두개이다. 기록이 누적되면 이전 악보들의 정보가 삭제되기 때문에 열람한 시간인 add_date 정보가 존재한다.

e. Popular

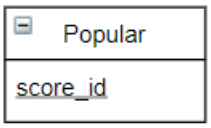

Diagram 20. Popular Entity

Popular Entity는 user들이 열람한 악보 중 누적 열람횟수가 높은 악보들에 대한 정보를 가지고 있으며, primary key 는 score_id이다.

7.3 Relational Schema

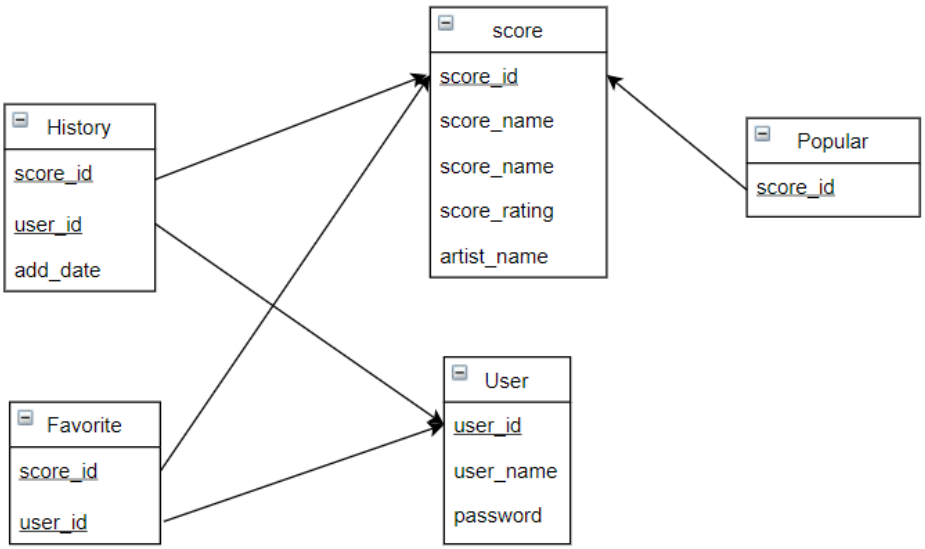

Diagram 21. Relational Schema

7.4 SQL DDL

A. User

```
CREATE TABLE User
{
   user_id INT NOT NULL,
   user_name VARCHAR(10) NOT NULL,
   password PAWWORD NOT NULL,
   PRIMARY KEY (user_id)
}
```

B. Score

```
CREATE TABLE Score
{
   score_id INT NOT NULL,
   score_name VARCHAR(10) NOT NULL,
   score_rating INT NOT NULL,
   artist_name VARCHAR(10) NOT NULL,
   PRIMARY KEY (score_id)
}
```

C. Favorite

```
CREATE TABLE Favorite
{
   score_id INT NOT NULL,
   user_id INT NOT NULL,
   PRIMARY KEY (score_id, user_id),
   FOREIGN KEY (score_id) REFERENCES Score(score_id),
   FOREIGN KEY (user_id) REFERENCES User(user_id)
}
```

d. History

```
CREATE History History
{
   score_id INT NOT NULL,
   user_id INT NOT NULL,
   add_date DATE NOT NULL,
   PRIMARY KEY (score_id, user_id),
   FOREIGN KEY (score_id) REFERENCES Score(score_id),
   FOREIGN KEY (user_id) REFERENCES User(user_id)
}
```

e. Popular

```
CREATE TABLE Popular
{
   score_id INT NOT NULL,
   PRIMARY KEY (score_id),
   FOREIGN KEY (score_id) REFERENCES Score(score_id)
}
```

8. Testing Plan

8.1. Objectives

본 시스템의 Testing 계획에 대해 서술한다.

8.2. Testing policy

A. Development Testing

system의 전체적인 Synchronization 및 오류 발견과 예방에 초점을 맞추어 개발 과정에서 일어날 수 있는 오류들을 방지하는 것을 목표로 한다. 이를 위해 코드 검토, Data flow 및 사용되어지는 Metrics에 대한 검토와, Peer Code Review등을 계획한다. Reliability, Security, Performance과 같은 Non-functional requirements에 초점을 맞추어 Testing을 진행한다.

1) Reliability

Reliability는 전체적인 System의 신뢰성으로 어플리케이션에서 제공되어지는 악보가 늘 최신으로 Up date 되어야 한다. 동시에 악보가 저작권에 문제가 없도록 하여야 한다. 악보들의 저작권에 대한 검토와, 실시간으로 최신 악보들이 Crawling 되어지고 있는지에 대한 검토가 요구된다.

2) Security

본 시스템은 높은 보안수준은 요구되지 않으나, 고객들이 가입시작성한 개인정보의 보안에 신경을 써야 한다. DB에 대한 접근 권한과, 고객의 ID/PW에 대한 암호화에 대한 검토가 요구된다.

B. Release Testing

System의 배포는 점진적인 방향으로 진행되어야 한다. 하지만 개발 및 Testing 기간이 짧은 것을 감안하여 초기 배포 버전에 대한 Testing만을 진행하는 것을 목표로 한다.

C. User Testing

실질적인 System사용에 있어 사용자의 입장에서 일어날 수 있는 오류들을 검토하여야 한다. 개발 이후 사용자의 입장에서 어플리케이션 사용시 일어날 수 있는 시나리오들을 가정하고, 각 시나리오들의 진행과정에서 오류가 없는지 검토하는 방식으로 이루어진다.

D. Testing Case

1) Unit testing

내가 favorite 표시를 한 악보들이 favorite page에 저장되었는지 확인

Favorite 표시를 취소하였을 때 favorite page에서 삭제되었는지 확인

오프라인 상에서도 악보를 볼 수 있는지 확인

2) Component testing

Search를 통해 찾은 악보들 중에서 본 악보들이 history page에 자동으로 저장되는지 확인

그 중 favorite 표시한 악보들이 favorite page에도 저장되었는지 확인

3) System testing

Search를 통해 찾은 악보들 중 본 악보의 재생이 제대로 실행되는지 확인

재생하면서 추가기능들 모두 잘 작동하는지 확인

그와 동시에 history page에 자동으로 저장되었는지 확인

오프라인 상에서 History page에서 본 악보를 보았을 때 제대로 재생이 되는지 확인

마찬가지로 추가기능들 모두 잘 작동하는지 확인

4) Acceptance testing

Customer가 우선 개별의 아이디와 비밀번호로 자신만의 database에 접속이 되는지 확인

System testing과 동일한 방법을 수행함

Customer들이 충분히 이용할 수 있을 정도로 악보를 보유하는지 확인

9. Development Plan

9.1. Objectives

실재 개발 단계에서 사용할 기술 및 환경에 대해 기술한다.

9.2. Frontend Environment

A. Vue.js

Figure 5. Vue.js

Vue.js는 웹 개발을 단순화하고 정리하기 위해 개발된 대중적인 자바스크립트 프론트엔드 프레임 워크이다. 싱글 파일 컴포넌트 기법으로 하나의 컴포넌트에 필요한 HTML 템플릿, 자바스크립트, CSS 스타일시트 등을 하나의 파일에 모두 작성할 수 있어서 코드 집적도와 유지보수성을 높일 수 있으며, MVC 패턴을 코드를 정리할 수 있게 도와준다. 다른 자바스크립트 라이브러리를 사용하는 웹 애플리케이션 프로젝트에 Vue.js를 도입하기 쉽다.

9.3. Backend Environment

A. Java

Figure 6. Java

Java는 객체 지향 프로그래밍 언어로서, JSP(Java Server Page)와 Servlet 등의 기술을 이용해 다양한 웹 애플리케이션 서버를 구현하는 데 사용되고 있다.

B. Node.js

Figure 7. Node,js

Node.js는 확장성 있는 네트워크 애플리케이션 개발에 사용되는 소프트웨어 플랫폼이다. 작성 언어로 자바스크립트를 활용하며 Non-blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다.

10. Index

10.1 Figures

Figure 1. Draw.io Logo 8

Figure 2. Power Point Logo 8

Figure 3. REST API 23

Figure 4. JSON 24

Figure 5. Vue.js 35

Figure 6. Java 36

Figure 7. Node.js 36

10.2 Diagrams

Diagram 1. Fronted Architecture 10

Diagram 2. Backend Architecture 10

Diagram 3. Search System architecture 11

Diagram 4. Favorite System architecture 11

Diagram 5. Search Class Diagram 13

Diagram 6. History Class Diagram 15

Diagram 7. History Sequence Diagram 16

Diagram 8. Favorite Class Diagram 17

Diagram 9. Favorite Sequence Diagram 18

Diagram 10. Popular Class Diagram 18

Diagram 11. System Overall Architecture 20

Diagram 12. Application Server Class Diagram 20

Diagram 13. Score Crawling System Class Diagram 22

Diagram 14. Score Crawling System Sequence Diagram 22

Diagram 15. Overall E-R Diagram 28
Diagram 16. User Entity 29
Diagram 17. Score Entity 29
Diagram 18. Favorite Entity 29
Diagram 19. Histoy Entity 30
Diagram 20. Popular Entity 30
Diagram 21. Relational Schema 30