포팅 메뉴얼

1. 개발 환경 및 기술 스택

- Backend
    - JAVA 17 (17.0.9 2023-10-17 LTS)
    - Spring boot 3.2.3
    - OAUTH 2.0
    - Spring Security 6.2.1
    - Spring Data JPA
    - Mysql 8.0.35
    - Redis 5.0.7
- Frontend
    - Node.js 20.10.0
    - npm 10.2.3
    - react 18.2.0
    - vite 5.1.6

- Recommend
  - python 3.8
  - numpy 1.19.5
  - pandas 1.1.5
- Infra
  - Ubuntu 20.04.6 LTS
  - Nginx 1.18.0
  - Jenkins 2.444
  - docker 25.0.3
- IDE
  - IntelliJ Ultimate
  - Vscode
  - PyCharm

2. 설정 파일 및 환경 변수 정보

1. **BackEnd**
   - application.yml (로컬용)

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver  # DB 드라이
    url: jdbc:mysql://localhost:3306/perfectfit?useUnicode=
    username : ssafy
    password : ssafy

  jpa:
    open-in-view: false
    defer-datasource-initialization: true
    generate-ddl: true
    hibernate:
      ddl-auto: create-drop
```

```
    properties:
      hibernate:
        format_sql: false                    # 하이버네이트가
        use_sql_comments: true
        show_sql: true                      # 하이버네이트가 실
        jdbc:
          batch_size: 100                   # insert/update
        default_batch_fetch_size: 100
  #      ddl-auto: create-drop                # ddl 자동 작
  # data_테이블명.sql 관련 실행 setting
  sql:
    init:
      mode: always
      data-locations:
        - 'classpath:/data_artist.sql'
        - 'classpath:/data_genre.sql'
        - 'classpath:/data_song.sql'
        - 'classpath:/data_member.sql'
        - 'classpath:/data_my_list.sql'
        - 'classpath:/data_song_count.sql'
        - 'classpath:/data_song_history.sql'

  # NoSQL setting
  data:
    # Redis setting
    redis:
      host: localhost
      port: 6379

# jwt setting
jwt:
  secret-key:
    access: 612834t213b497d23d94h2374693471269477h9qf87weo
    refresh: 612834t213b497d23d94h2374693471269477h9qf87we

  expired-min:
    access: 60   # 액세스 토큰 만료제한시간 60분 (1시간) (60)
    refresh: 10080  # 리프레쉬 토큰 만료제한시간 10080분 (7일) (
```

```
oauth:
  kakao:
    client-id: df089f203219a3b6f8b59089d4381d02
    client-secret: w3dHiDuDOiluFi2vjWpzSo1BbBhTnvpu
    redirect-uri: http://localhost:5173/member/loading/kak
    scope:
      - profile_nickname
      - profile_image
      - account_email

  naver:
    client-id: tzXmova1GZfX43ujVeQU
    client-secret: 3BfuMlsrSV
    redirect_uri: http://localhost:5173/member/loading/nav
    scope:
      - nickname
      - email
      - profile_image

youtube:
  api:
    key: AIzaSyDqnJkym6OA2vJAasKGP8gjsP4sOwMZfxo
```

- application.yml (배포용)

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver  # DB 드라이
    url: jdbc:mysql://j10c205.p.ssafy.io:3306/perfectfit?u
    username : root
    password : c205c205


  jpa:
    open-in-view: false
    defer-datasource-initialization: true
    generate-ddl: true
```

```
    hibernate:
      ddl-auto: create-drop
    properties:
      hibernate:
        format_sql: false                    # 하이버네이트가
        use_sql_comments: true
        show_sql: true                       # 하이버네이트가 실
        jdbc:
          batch_size: 100                    # insert/update
        default_batch_fetch_size: 100
  #      ddl-auto: create-drop                 # ddl 자동 작
  # data_테이블명.sql 관련 실행 setting
  sql:
    init:
      mode: always
      data-locations:
        - 'classpath:/data_artist.sql'
        - 'classpath:/data_genre.sql'
        - 'classpath:/data_song.sql'
        - 'classpath:/data_member.sql'
        - 'classpath:/data_my_list.sql'
        - 'classpath:/data_song_count.sql'
        - 'classpath:/data_song_history.sql'

  # NoSQL setting
  data:
    # Redis setting
    redis:
      host: j10c205.p.ssafy.io
      port: 6379

# jwt setting
jwt:
  secret-key:
    access: 612834t213b497d23d94h2374693471269477h9qf87weo
    refresh: 612834t213b497d23d94h2374693471269477h9qf87we

  expired-min:
```

```
    access: 60   # 액세스 토큰 만료제한시간 60분 (1시간) (60)
    refresh: 10080  # 리프레쉬 토큰 만료제한시간 10080분 (7일) (1

oauth:
  kakao:
    client-id: df089f203219a3b6f8b59089d4381d02
    client-secret: w3dHiDuDOiluFi2vjWpzSo1BbBhTnvpu
        redirect-uri: https://j10c205.p.ssafy.io/member/lo
    scope:
      - profile_nickname
      - profile_image
      - account_email

  naver:
    client-id: tzXmova1GZfX43ujVeQU
    client-secret: 3BfuMlsrSV
    redirect_uri: http://localhost:5173/member/loading/nav
    scope:
      - nickname
      - email
      - profile_image

youtube:
  api:
    key: AIzaSyDqnJkym6OA2vJAasKGP8gjsP4sOwMZfxo
```

2. **FrontEnd**

- .env

```
VITE_BASE_URL = https://j10c205.p.ssafy.io/
VITE_REGION = ap-northeast-2
VITE_BUCKET = perfectfitssafy
VITE_ACCESS_KEY_ID = AKIAW3MECNPMTQYOQY6F
VITE_SECRET_ACCESS_KEY_ID = zZ0hTjnFHtpBpxPZnm1tO65CVFePLj
VITE_MR_URL = https://perfectfitssafy.s3.ap-northeast-2.am
```

- package.json

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite --host 0.0.0.0 --port 5173",  // 이부분이 H
    "build": "tsc && vite build",
    "lint": "eslint . --ext ts,tsx --report-unused-disable
    "preview": "vite preview"
  },
```

3. Recommend
    - secrets.json

```
{
  "DB" : {
    "user" : "root",
    "password" : "c205c205",
    "host" : "j10c205.p.ssafy.io",
    "port" : 3306,
    "database" : "perfectfit"
  }
}
```

4. AI
    - settings.py

```
DATABASES = {
    "default": {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'perfectfitAI',
        'USER': 'root',
        'PASSWORD': 'c205c205',
```

```
        'HOST': 'j10c205.p.ssafy.io',
        'PORT': '3306'
    }
}
```

3. 빌드 및 배포

1) Docker 설치

```
# 1. Docker 엔진의 공식 버전을 설치하기 전 충돌하는 패키지를 제거해야 한다
for pkg in docker.io docker-doc docker-compose docker-compose

# 2. Docker apit 저장소 설정
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" |
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# 3. Docker 패키지 설치
# 최신 버전 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io do

# 4. hello-world 이미지를 실행하여 Docker 엔진 설치가 성공했는지 확인
sudo docker run hello-world
```

2) Docker compose 설치

```
# 설치
sudo apt update
sudo apt install docker.io
sudo apt install docker-compose

# 설치 확인
docker --version
docker-compose --version
```

3) Docker로 젠킨스 컨테이너 설치

```
# docker container에 마운트할 볼륨 디렉토리 생성
cd /home/ubuntu && mkdir jenkins-data

# 외부에서 접속할 포트를 오픈하고 상태 확인
# 8080은 주로 쓰이는 포트라 다른 포트로 실행
sudo ufw allow 9001
sudo ufw reload
sudo ufw status

# docker 명령어로 jenkins container 생성 및 구동
sudo docker run -d -p 9001:8080 -v /home/ubuntu/jenkins-data:
sudo docker run -d \
-p 9001:8080 \
-p 50000:50000 \
-v /home/ubuntu/jenkins:/var/jenkins_home \
-v /usr/bin/docker:/usr/bin/docker \
-v /var/run/docker.sock:/var/run/docker.sock \
-u root \
--name jenkins \
jenkins/jenkins:2.444

# 초기 설정에 필요함 비밀번호 확인
sudo docker logs jenkins
```

4) docker compose를 통한 실행

- Jenkins를 통해 docker-compose.yml파일의 구성을 실행시킨다.

docker-compose.yml

```
version: '3.8'

services:
  backend:
    image: kimhyeokil/back:latest
    container_name: back
    build:
      context: ./BackEnd
      dockerfile: Dockerfile
    ports:
      - "9002:8080"
    networks:
      - jenkins-network


# Docker volume 정의
networks:
  jenkins-network:
    external: true
```

BE Jenkinsfile

```
pipeline {
    agent any
    environment {
```

```
        REPO = "s10-ai-speech-sub2/S10P22C205"
        DOCKERHUB_REGISTRY = "kimhyeokil/back"
        DOCKERHUB_CREDENTIALS = credentials('Docker-hub')
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage('Setup Environment') {
            steps {
                dir("${env.WORKSPACE}/BackEnd"){
                    script {
                        sh "ls -al"
                        sh "chmod +x ./gradlew"
                    }
                }
            }
        }
        stage("Build") {
            steps {
                script {
                    sh "docker build -t ${DOCKERHUB_REGISTRY}
                }
            }
        }
        stage("Login") {
            steps {
                 sh "echo \${DOCKERHUB_CREDENTIALS_PSW} | doc
            }
        }
        stage("Tag and Push") {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        sh "docker push ${DOCKERHUB_REGISTRY}
                    }
```

```
                }
            }
        }
        stage('Prune old images'){
            steps{
                script{
                    sh "docker ps"
                }
            }
        }
        stage('Pull') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        sh "docker stop back || true"  // Ign
                        sh "docker rm back || true"    // Ign
                        sh "docker rmi ${DOCKERHUB_REGISTRY}|
                        sh "docker pull ${DOCKERHUB_REGISTRY}
                    }
                }
            }
        }
        stage('Up') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        try {
                            sh "docker-compose -f ${env.WORKS

                        } catch(Exception e) {
                            sh "docker restart back || true"
                        }
                    }
                }
            }
        }
```

```
    }
}
```

BE Dockerfile

```
# 빌드 스테이지
FROM amazoncorretto:17.0.7-alpine AS builder
USER root
WORKDIR /back
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
# gradlew 실행 권한 부여
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

# 실행 스테이지
FROM openjdk:17
WORKDIR /back
COPY --from=builder /back/build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
VOLUME /tmp
```

FE Jenkinsfile

```
pipeline {
    agent any
    environment {
        REPO = "s10-ai-speech-sub2/S10P22C205"
        DOCKERHUB_REGISTRY = "kimhyeokil/front"
        DOCKERHUB_CREDENTIALS = credentials('Docker-hub')

    }
```

```
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage("Build") {
            steps {
                script {
                    sh "ls -al"

                    sh "docker build -t ${DOCKERHUB_REGISTRY}
                }
            }
        }

        stage("Login") {
            steps {
                sh "echo \${DOCKERHUB_CREDENTIALS_PSW} | dock

            }
        }
        stage("Tag and Push") {
            steps {
                script {
                    withCredentials([usernamePassword(credent
                        sh "docker push ${DOCKERHUB_REGISTRY}
                    }
                }
            }
        }

        stage('Prune old images') {
            steps {
                script {
                    sh "docker ps"
                    // sh "docker system prune --filter until
```

```
                }
            }
        }
        stage('Pull') {
            steps {
                script {
                    withCredentials([usernamePassword(credent
                        sh "docker stop front || true"
                        sh "docker rm front || true"
                        sh "docker rmi ${DOCKERHUB_REGISTRY}|
                        sh "docker pull ${DOCKERHUB_REGISTRY}
                    }
                }
            }
        }

        stage('Up') {
            steps {
                script {
                    withCredentials([usernamePassword(credent
                        try{
                            sh "docker run -d --name front -p
                            ${DOCKERHUB_REGISTRY}"

                            // sh "docker-compose -f ${env.WO

                        } catch (Exception e){
                            sh "docker restart front || true"
                        }
                    }
                }
            }
        }
    }
}
```

FE Dockerfile

```
# 1단계: 빌드 단계
FROM node:latest as build

# 작업 디렉토리 설정
WORKDIR /app

# 의존성 설치
COPY package*.json ./
RUN npm install --force

# 앱 소스 코드 복사
COPY . .

# 포트 노출
EXPOSE 5173

# 컨테이너 시작 시 실행할 명령
CMD ["npm", "run", "dev"]
```

Recommend Jenkinsfile

```
pipeline {
    agent any
    environment {
        REPO = "s10-ai-speech-sub2/S10P22C205"
        DOCKERHUB_REGISTRY = "kimhyeokil/recommend"
        DOCKERHUB_CREDENTIALS = credentials('Docker-hub')
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage("Build") {
            steps {
```

```
                script {
                    sh "docker build -t ${DOCKERHUB_REGISTRY}
                }
            }
        }
        stage("Login") {
            steps {
                 sh "echo \${DOCKERHUB_CREDENTIALS_PSW} | doc
            }
        }
        stage("Tag and Push") {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        sh "docker push ${DOCKERHUB_REGISTRY}
                    }
                }
            }
        }
        stage('Prune old images'){
            steps{
                script{
                    sh "docker ps"
                }
            }
        }
        stage('Pull') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        sh "docker stop recommend || true"  /
                        sh "docker rm recommend || true"    /
                        sh "docker rmi ${DOCKERHUB_REGISTRY}|
                        sh "docker pull ${DOCKERHUB_REGISTRY}
                    }
                }
            }
        }
```

```
        stage('Up') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        try {
                            // sh "docker-compose -f ${env.WO
                            sh "docker run -d --name recommen
                            ${DOCKERHUB_REGISTRY}"


                        } catch(Exception e) {
                            sh "docker restart recommend || t
                        }
                    }
                }
            }
        }


    }
}
```

Recommend Dockerfile

```
# 기본 이미지 설정
FROM tiangolo/uvicorn-gunicorn-fastapi:python3.8

# 작업 디렉토리 설정
WORKDIR /app

# 호스트의 모든 파일을 컨테이너의 /app 디렉토리로 복사
COPY . /app

# 필요한 패키지 설치
RUN pip install --no-cache-dir -r requirements.txt
```

```
# FastAPI 애플리케이션을 실행
CMD ["uvicorn", "recommend.main:app", "--host", "0.0.0.0", "-
```

AI Jenkinsfile

```
pipeline {
    agent any
    environment {
        REPO = "s10-ai-speech-sub2/S10P22C205"
        DOCKERHUB_REGISTRY = "kimhyeokil/ai"
        DOCKERHUB_CREDENTIALS = credentials('Docker-hub')
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage("Build") {
            steps {
                script {
                    sh "docker build -t ${DOCKERHUB_REGISTRY}
                }
            }
        }
        stage("Login") {
            steps {
                 sh "echo \${DOCKERHUB_CREDENTIALS_PSW} | doc
            }
        }
        stage("Tag and Push") {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        sh "docker push ${DOCKERHUB_REGISTRY}
                    }
                }
```

```
            }
        }
        stage('Prune old images'){
            steps{
                script{
                    sh "docker ps"
                }
            }
        }
        stage('Pull') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        sh "docker stop ai || true"  // Ignor
                        sh "docker rm ai || true"    // Ignor
                        sh "docker rmi ${DOCKERHUB_REGISTRY}|
                        sh "docker pull ${DOCKERHUB_REGISTRY}
                    }
                }
            }
        }
        stage('Up') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswo
                        try {
                            // sh "docker-compose -f ${env.WO
                            sh "docker run -d --name ai -p 90
                            ${DOCKERHUB_REGISTRY}"


                        } catch(Exception e) {
                            sh "docker restart ai || true"  /
                        }
                    }
                }
            }
        }
```

```
    }
}
```

AI Dockerfile

```
# 베이스 이미지 설정
FROM python:3.10

# 작업 디렉토리 설정
WORKDIR /app

# 필요한 파일 복사
COPY requirements.txt /app/

# 의존성 설치
RUN pip install --no-cache-dir -r requirements.txt
RUN apt update
RUN apt install ffmpeg -y

# 소스 코드 복사
COPY . /app/

# 컨테이너 실행시 실행할 명령어
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

5) NGINX 설정

default

```
##
# You should look at the following URL's in order to grasp a
# of Nginx configuration files in order to fully unleash the
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials
```

```
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from si
# leave it as reference inside of sites-available where it wi
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provi
# applications, such as Drupal or Wordpress. These applicatio
# available underneath a path with that package name, such as
#
# Please see /usr/share/doc/nginx-doc/examples/ for more deta
##

# Default server configuration
#
#server {
#       listen 80 default_server;
#       listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configura
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

#       root /var/www/html;

        # Add index.php to the list if you are using PHP
```

```
#       index index.html index.htm index.nginx-debian.html;

#       server_name _;


#       location / {
#               proxy_pass http://j10c205.p.ssafy.io:9003;
#               proxy_set_header Host $host;
#               proxy_set_header X-Real-IP $remote_addr;

#       }
#       location /api/v1 {
#               proxy_pass http://j10c205.p.ssafy.io:9002;
#               proxy_set_header Host $host;
#               proxy_set_header X-Real-IP $remote_addr;

#       }
#       location /decommendations {
#               proxy_pass http://j10c205.p.ssafy.io:9005;
#                proxy_set_header Host $host;
#                proxy_set_header X-Real-IP $remote_addr;
#       }
#       location /ai {
#                proxy_pass http://j10c205.p.ssafy.io:9006;
#                proxy_set_header Host $host;
#                proxy_set_header X-Real-IP $remote_addr;
#        }


        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.4-fpm.soc
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
```

```
        #}

        # deny access to .htaccess files, if Apache's documen
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
#}
server {
        listen 9004 ssl;
        server_name j10c205.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/j10c205.p.ssafy
        ssl_certificate_key /etc/letsencrypt/live/j10c205.p.s

        location / {
                proxy_pass http://localhost:9001;

        }

}


# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available
# to sites-enabled/ to enable it.
#
#server {
#       listen 80;
#       listen [::]:80;
#
#       server_name example.com;
#
#       root /var/www/example.com;
#       index index.html;
#
#       location / {
```

```
#                try_files $uri $uri/ =404;
#       }
#}

server {

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configura
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        #root /var/www/html;

        # Add index.php to the list if you are using PHP
        #index index.html index.htm index.nginx-debian.html;
    server_name j10c205.p.ssafy.io; # managed by Certbot



        location / {
                proxy_pass http://localhost:9003;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
        }
```

```
        location /api/v1 {
                proxy_pass http://localhost:9002;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;

        }

        location /recommendations {
                proxy_pass http://localhost:9005;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
        }

        location /ai {
                proxy_pass http://localhost:9006;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.4-fpm.soc
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's documen
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
```

```
    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j10c205.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/j10c205.p.ssafy
    include /etc/letsencrypt/options-ssl-nginx.conf; # manage
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

}
server {
    if ($host = j10c205.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name j10c205.p.ssafy.io;
    return 404; # managed by Certbot


}
```

4. 외부 서비스 및 정보

- 카카오 소셜 로그인 정보

개인정보

항목 이름	ID	상태	
닉네임	profile_nickname	● 필수 동의	설정
프로필 사진	profile_image	● 필수 동의	설정
카카오계정(이메일)	account_email	● 필수 동의 [수집]	설정

- 네이버 소셜 로그인 정보

5. DB dump 설명

테이블들은 JPA로 자동으로 DB와 매핑하여 Spring boot 서버 내에서 자동으로 생성되게끔 설정하였습니다.

해당 db dump 데이터는 perfectfit\BackEnd\src\main\resources 폴더내에 저장되어 있습니다.

- data_member.sql (회원 데이터)
- data_artist,sql (가수 데이터)
- data_genre.sql (장르 데이터)
- data_my_list.sql (찜 목록 데이터)
- data_song_count.sql (노래당 불러진 횟수 데이터)
- data_song_history.sql (누가 어떤 노래를 불렀는 지에 대한 히스토리 데이터)
- data_song.sql (노래 데이터)

- data_lyrics.sql(가사 데이터)