포팅 메뉴얼

1. 빌드 및 배포

기술 스택

FastApi

Skill	Version
`Python`	`3.9`
`FastAPI`	`0.1110.1`
`PyTorch`	`2.2.1`
`Uvicorn`	`0.29.0`
`PyOpenGL`	`3.1.7`
`Animated Drawings`	`0.0.1`
`libosmesa6`	
`libglib2.0-0`	
`libgl1`	
`libosmesa6`	
`freeglut3`	
`libglfw3`	

TorchServe(Animated Drwaings)

Skill	Version
`Python`	`3.8.13`
`Torch`	`2.0.0`
`mmcv-full`	`1.7.0`
`mmdet`	`2.27.0`
`Torchserve`	`0.10.0`
`mmpose`	`0.29.0`
`Torchvision`	`0.15.1`
`miniconda3`	`3.12.2`

Backend

Skill	Version
`Java`	`17`
`Gradle`	`8.5`
`Spring boot`	`3.2.3`
`Hibernate`	`6.4.1`
`MySQL`	`8.0.29`
`Redis`	`7.2.4`
`OAuth`	`3.2.3`
`Spring Security`	`6.2.2`

`QueryDsl`	`5.0.0`
`JWT`	`0.11.5`
`Swagger`	`2.0.2`
`Mockito`	`5.11.0`
`JUnit`	`5.9.1`
`Spring Cloud AWS`	`2.2.6`
`Firebase`	`9.2.0`

Frontend

Skill	Version
`HTML5`	
`CSS`	
`TypeScript`	`5.4.5`
`Node.js`	`20.11.0`
`NPM`	`10.2.4`
`React`	`10.2.3`
`React Query`	`5.29.2`
`Firebase`	`10.11.1`
`Sass`	`1.77.1`
`Vite`	`5.2.0`
`Axios`	`1.6.8`

IDE

Skill	Version
`Visual Studio Code`	`1.85.1`
`IntelliJ IDEA`	`2023.3.2`
`Android Studio`	`Hedgehog 2023.1.1`
`Xcode`	`15.3`
`Pycharm`	`2024.1`

Infra & Deploy

Skill	Version
`Linux`	`5.15.0-1056-aws`
`Ubuntu`	`20.04`
`Docker`	`26.0.0`
`Docker-compose`	`v2.25.0`
`Portainer`	`2.16.2`
`Nginx`	`1.18.0`
`Jenkins`	`2.440.2`
`Capacitor`	`6.0.0`
`Android SDK`	`34`
`Swift`	`5`
`Prometheus`	`2.52.0`
`Grafana`	`10.4.3`
`Node_Exporter`	`1.8.0`

포트 포워딩

```
80 - HTTP
443 -  HTTPS
9090 - Jenkins
3306 : 3306 - MySQL
6379 : 6379 - Redis
5173 : 80 - React (blue)
5174 : 80 - React (green)
8090 : 8080 - Spring boot (blue)
8091 : 8080 - Spring boot (green)
8000:8000 - FAST API (blue)
8001:8000 - FAST API (green)
8080:8080, 8081:8081 - Torchserve
9000:9000 - Portainer
```

설정 파일

lemon.conf(Nginx)

```
server {
  listen 80; #80포트로 받을 때
  server_name 도메인 주소;
  return 301 https://도메인 주소$request_uri;

}

server {
  listen 80;
  server_name 젠킨스 도메인 주소;

  location / {
    proxy_pass http://localhost:9090;
  }
}

server {
  listen 443 ssl http2;
  server_name 도메인 주소;
  include /etc/nginx/conf.d/service-url-front.inc;
  include /etc/nginx/conf.d/service-url-back.inc;
  include /etc/nginx/conf.d/service-url-ai.inc;

  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/도메인 주소/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/도메인 주소/privkey.pem;


  location / {
    proxy_pass $service_url_front;
  }

  location /api { # location 이후 특정 url을 처리하는 방법을 정의
    proxy_pass $service_url_back; # Request에 대해 어디로 리다이렉트하는지
    proxy_redirect off;
    charset utf-8;
```

```
    proxy_read_timeout 300;
    proxy_http_version 1.1;
    proxy_set_header Connection "upgrade";
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-NginX-Proxy true;
  }

  location /ai { # location 이후 특정 url을 처리하는 방법을 정의
    proxy_pass $service_url_ai; # Request에 대해 어디로 리다이렉트하는지
    proxy_redirect off;
    charset utf-8;

    proxy_read_timeout 300;
    proxy_http_version 1.1;
    proxy_set_header Connection "upgrade";
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-NginX-Proxy true;
  }

}
```

deploy.sh

```
# 실행 중인 도커 컴포즈 확인
EXIST_A=$(sudo docker-compose -p lemon-blue -f docker-compose.blue.yml ps | grep Up)

if [ -z "${EXIST_A}" ] # -z는 문자열 길이가 0이면 true. A가 실행 중이지 않다는 의미.
then
        # Green이 실행 중인 경우
        START_CONTAINER=blue
        TERMINATE_CONTAINER=green
        START_PORT_BACK=8090
        START_PORT_FRONT=5173
        START_PORT_AI=8000
        TERMINATE_PORT_BACK=8091
        TERMINATE_PORT_FRONT=5174
        TERMINATE_PORT_AI=8001
else
        # Blue가 실행 중인 경우
        START_CONTAINER=green
        TERMINATE_CONTAINER=blue
        START_PORT_BACK=8091
        START_PORT_FRONT=5174
        START_PORT_AI=8001
        TERMINATE_PORT_BACK=8090
        TERMINATE_PORT_FRONT=5173
        TERMINATE_PORT_AI=8000
fi
```

```
echo "lemon-${START_CONTAINER} up"

# 실행해야하는 컨테이너 docker-compose로 실행. -p는 docker-compose 프로젝트에 이름을 부여
# -f는 docker-compose파일 경로를 지정
sudo docker-compose -p lemon-${START_CONTAINER} -f docker-compose.${START_CONTAINER}.yml up -d

for cnt in {1..10} # 10번 실행
do
        echo "check server start.."

        # 스프링부트에 등록했던 actuator로 실행되었는지 확인
        UP=$(curl -s http://127.0.0.1:${START_PORT_BACK}/api/actuator/health | grep 'UP')
        if [ -z "${UP}" ] # 실행되었다면 break
        then
                echo "server not start.."
        else
                break
        fi

        echo "wait 10 seconds" # 10 초간 대기
        sleep 10
done

if [ $cnt -eq 10 ] # 10번동안 실행이 안되었으면 배포 실패, 강제 종료
then
        echo "deployment failed."
        exit 1
fi

echo "server start!"
echo "change nginx server port"

# sed 명령어를 이용해서 아까 지정해줬던 service-url.inc의 url값을 변경해줍니다.
# sed -i "s/기존문자열/변경할문자열" 파일경로 입니다.
# 종료되는 포트를 새로 시작되는 포트로 값을 변경해줍니다.
echo ">>> 전환할 Port: $START_PORT_BACK"
echo ">>> 전환할 Port: $START_PORT_FRONT"
echo ">>> 전환할 Port: $START_PORT_AI"
echo ">>> Port 전환"
# 아래 줄은 echo를 통해서 나온 결과를 | 파이프라인을 통해서 service-url.inc에 덮어쓸 수 있습니다.
echo "set \$service_url_back  http://127.0.0.1:${START_PORT_BACK};" | sudo tee /etc/nginx/conf.d
echo "set \$service_url_front  http://127.0.0.1:${START_PORT_FRONT};" | sudo tee /etc/nginx/conf
echo "set \$service_url_ai  http://127.0.0.1:${START_PORT_AI};" | sudo tee /etc/nginx/conf.d/ser

# 새로운 포트로 스프링부트가 구동 되고, nginx의 포트를 변경해주었다면, nginx 재시작해줍니다.
echo "nginx reload.."
sudo service nginx reload

# 기존에 실행 중이었던 docker-compose는 종료시켜줍니다.
echo "lemon-${TERMINATE_CONTAINER} down"
sudo docker-compose -p lemon-${TERMINATE_CONTAINER} -f docker-compose.${TERMINATE_CONTAINER}.yml
echo "success deployment"
```

docker-compose-blue.yml

```yaml
version: "3.8"
services:
  backend:
    build:
      context: ./back
      dockerfile: Dockerfile.prod
    restart: always
    ports:
      - 8090:8080
  frontend:
    build:
      context: ./front
      dockerfile: Dockerfile.prod
    restart: always
    ports:
      - 5173:80
  ai:
    build:
      context: ./ai-server
      dockerfile: Dockerfile.prod
    restart: always
    ports:
      - 8000:8000
    extra_hosts:
      - "host.docker.internal:host-gateway"
```

docker-compose-green.yml

```yaml
version: "3.8"
services:
  backend:
    build:
      context: ./back
      dockerfile: Dockerfile.prod
    restart: always
    ports:
      - 8091:8080
  frontend:
    build:
      context: ./front
      dockerfile: Dockerfile.prod
    restart: always
    ports:
      - 5174:80
  ai:
    build:
      context: ./ai-server
      dockerfile: Dockerfile.prod
    restart: always
    ports:
      - 8001:8000
    extra_hosts:
      - "host.docker.internal:host-gateway"
```

service-url-front.inc

```
set $service_url_front  http://127.0.0.1:5173;
```

service-url-back.inc

```
set $service_url_back  http://127.0.0.1:8090;
```

service-url-ai.inc

```
set $service_url_ai  http://127.0.0.1:8000;
```

환경 변수

Jenkins String Parameter

```
KAKAO_CLIENT_ID=
KAKAO_CLIENT_SECRET=
JWT_SECRET_KEY=
LINE_CLIENT_ID=
LINE_CLIENT_SECRET=
BUCKET_NAME=
S3_ACCESS_KEY=
S3_SECRET_KEY=
MYSQL_HOST=www.lettermon.com
MYSQL_PROD_PORT=3306
MYSQL_USER=root
MYSQL_PASSWORD=B103lemon!
REDIS_HOST=www.lettermon.com
REDIS_PROD_PORT=6379
AI_URL=http://lettermon.com
DIR_NAME=prod
BASE_URL=http://lettermon.com/api
SHARE_URL=http://lettermon.com
```

.env.production

```
VITE_BASE_URL=https://도메인 주소/api
VITE_KAKAO_REDIRECT_URI="https://도메인 주소/kakao/callback"
VITE_LINE_CLIENT_ID=
VITE_LINE_REDIRECT_URI=https://도메인 주소/line/callback
VITE_LINE_STATE=
VITE_APP_GA_TRACKING_ID=
VITE_KAKAO_JAVASCRIPT_KEY=
VITE_KAKAO_TEMPLATE_ID=
VITE_REFRESH_TOKEN=/user/token
VITE_FCM_API_KEY=
VITE_FCM_AUTHDOMAIN=
VITE_FCM_PROJECTID=
VITE_FCM_STORAGEBUCKET=
VITE_FCM_MESSAGINGSENDERID=
VITE_FCM_APPID=
VITE_FCM_MEASUREMENTID=
VITE_FCM_VAPID_KEY=
```

.env.prod

```
S3_BUCKET=버킷명
S3_PATH=디렉토리명
CREDENTIALS_ACCESS_KEY=
CREDENTIALS_SECRET_KEY=
```

특이사항

Jenkins Execute shell

```
sed -i "s/\${BUCKET_NAME}/${BUCKET_NAME}/" "./back/src/main/resources/application.yml"
sed -i "s/\${S3_ACCESS_KEY}/${S3_ACCESS_KEY}/" "./back/src/main/resources/application.yml"
sed -i "s#\${S3_SECRET_KEY}#${S3_SECRET_KEY}#" "./back/src/main/resources/application.yml"
sed -i "s/\${KAKAO_CLIENT_ID}/${KAKAO_CLIENT_ID}/" "./back/src/main/resources/application-secret
sed -i "s/\${KAKAO_CLIENT_SECRET}/${KAKAO_CLIENT_SECRET}/" "./back/src/main/resources/applicatio
sed -i "s#\${BASE_URL}#${BASE_URL}#" "./back/src/main/resources/application-secret.yml"
sed -i "s#\${SHARE_URL}#${SHARE_URL}#" "./back/src/main/resources/application.yml"
sed -i "s/\${JWT_SECRET_KEY}/${JWT_SECRET_KEY}/" "./back/src/main/resources/application-secret.y
sed -i "s/\${LINE_CLIENT_ID}/${LINE_CLIENT_ID}/" "./back/src/main/resources/application-secret.y
sed -i "s/\${LINE_CLIENT_SECRET}/${LINE_CLIENT_SECRET}/" "./back/src/main/resources/application-
sed -i "s/\${MYSQL_HOST}/${MYSQL_HOST}/" "./back/src/main/resources/application-prod.yml"
sed -i "s/\${MYSQL_PROD_PORT}/${MYSQL_PROD_PORT}/" "./back/src/main/resources/application-prod.y
sed -i "s/\${MYSQL_USER}/${MYSQL_USER}/" "./back/src/main/resources/application-prod.yml"
sed -i "s/\${MYSQL_PASSWORD}/${MYSQL_PASSWORD}/" "./back/src/main/resources/application-prod.yml
sed -i "s/\${REDIS_HOST}/${REDIS_HOST}/" "./back/src/main/resources/application-prod.yml"
sed -i "s/\${REDIS_PROD_PORT}/${REDIS_PROD_PORT}/" "./back/src/main/resources/application-prod.y
sed -i "s#\${AI_URL}#${AI_URL}#" "./back/src/main/resources/application.yml"
sed -i "s/\${DIR_NAME}/${DIR_NAME}/" "./back/src/main/resources/application-prod.yml"

sudo cp /home/ubuntu/env/.env.development /var/lib/jenkins/workspace/prod-ci-cd/front
sudo cp /home/ubuntu/env/.env /var/lib/jenkins/workspace/prod-ci-cd/front
sudo cp /home/ubuntu/env/.env.production /var/lib/jenkins/workspace/prod-ci-cd/front
sudo cp /home/ubuntu/env/ai/.env.prod /var/lib/jenkins/workspace/prod-ci-cd/ai-server


echo 'jenkins build started...'
pwd
cd back
chmod +x gradlew
./gradlew clean
./gradlew bootJar
cd ..
./deploy.sh
```

DB 접속 정보

- MySQL 호스트 : www.lettermon.com:3306
  - MySQL 유저 : root
  - MySQL 패스워드 : B103lemon!
- Redis 호스트 : www.lettermon.com:6379
  - Redis 패스워드 : B103lemon!

배포 순서

1. `ufw allow`로 필요한 포트번호의 방화벽 허용
2. **Docker 설치**
3. **Docker-compose 설치**
4. **MySQL 설치 및 Redis 설치**, 포트와 유저, 패스워드 설정하여 배포
5. **Certbot 설치**
6. `certbot certonly --nginx -d <도메인 주소>` 로 https 인증
7. **Nginx 설치**
8. `/etc/nginx/sites-available/lemon.conf`에 리버스 프록시 작성
9. `/etc/nginx/conf.d` 경로에 `service-url-front.inc`, `service-url-back.inc`, `service-url-ai.inc` 작성
10. **Jenkins 설치**
11. Jenkins에서 새로운 Item 추가, Gitlab 연동 및 웹훅 설정 (String parameter 및 Execute Shell 추가)
12. `/var/lib/jenkins/workspace/Item이름` 경로에 `docker-compose.blue.yml`, `docker-compose.green.yml`, `deploy.sh` 작성
13. `/home/사용자 계정ID/env` 경로에 `.env.production`추가
14. `/home/사용자 계정ID/env/ai` 경로에 `.env.prod`추가
15. Jenkins에서 파라미터와 함께 build
16. `/var/lib/jenkins/workspace/Item이름/ai-server/torchserve` 폴더로 이동하여
    - `docker build -t docker_torchserve .`
    - `docker run -d --name docker_torchserve -p 8080:8080 -p 8081:8081 docker_torchserve`
    - 위의 명령어로 컨테이너 실행 후, `curl http://localhost:8080/ping` 입력하여 healthy 를 반환받으면 성공
    - 빌드 과정 중 오류 발생시 우분투 환경에 따라 문제가 발생할 수 있으므로, 아래의 링크의 issue를 읽고 필요한 패키지를 설치 및 osmesa를 활성화 해볼 것.
      - https://github.com/facebookresearch/AnimatedDrawings/issues/99#issue-1669192538
      - https://github.com/facebookresearch/AnimatedDrawings/issues/99#issuecomment-1711021198

2. 외부 서비스 정보

카카오 로그인

- https://developers.kakao.com/docs/latest/ko/getting-started/quick-start 문서를 참고하여 카카오 앱 등록, ID와 키 발급
- https://developers.kakao.com/docs/latest/ko/kakaologin/prerequisite 문서를 참고하여 카카오 로그인 활성화 설정, 간편가입 사용 설정, Redirect URI 등록

라인 로그인

- https://developers.line.biz/en/docs/liff/getting-started/ 문서를 참고하여 라인 앱 등록, ID와 키 발급 및 Callback URL 설정

3. DB 덤프파일

schema.sql

```
-- 데이터베이스 생성
DROP DATABASE IF EXISTS lemon;
CREATE DATABASE lemon DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE lemon;

-- 사용자 정보 테이블
CREATE TABLE users (
```

```
    users_id INT AUTO_INCREMENT NOT NULL COMMENT '사용자 아이디',
    nickname VARCHAR(50) NOT NULL COMMENT '사용자 닉네임',
    nickname_tag VARCHAR(50) NOT NULL COMMENT '사용자 닉네임 태그',
    provider VARCHAR(50) COMMENT '로그인 소셜 종류',
    role VARCHAR(50) COMMENT '관리자 여부 확인',
    provider_id VARCHAR(50) COMMENT '소셜 id',
    is_language BOOLEAN COMMENT '사용 언어',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    notification_token VARCHAR(255) COMMENT '파이어베이스 토큰',
    PRIMARY KEY (users_id)
) COMMENT '회원 정보';

-- 알림 정보 테이블
CREATE TABLE notification(
    notification_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '알림 아이디',
    friend_name VARCHAR(100) NOT NULL COMMENT '작성자 닉네임',
    type INT COMMENT '알림 타입',
    receiver_id INT NOT NULL COMMENT '받는 사람 아이디',
    is_check BOOLEAN NULL COMMENT '확인 여부',  -- 'NUL NULL'을 'NULL'로 수정
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    friend_tag VARCHAR(100) COMMENT '친구 닉네임 태그',
    sketchbook_name VARCHAR(100) COMMENT '스케치북 이름',
    sketchbook_tag VARCHAR(100) COMMENT '스케치북 태크',
    sketchbook_uuid VARCHAR(100) COMMENT '스케치북 uuid',
    PRIMARY KEY(notification_id),                -- 'PRIMANY KEY'를 'PRIMARY KEY'로 수정
    FOREIGN KEY(receiver_id) REFERENCES users(users_id)  -- 'users(users_id)'를 'users(user_id)'
) COMMENT '알림 정보';

-- 그룹 정보 테이블
CREATE TABLE groups_info (
    groups_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '그룹 아이디',
    group_name VARCHAR(255) NOT NULL COMMENT '그룹 이름',
    owner_id INT COMMENT '그룹 소유자 id 외래키',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (groups_id),
    FOREIGN KEY (owner_id) REFERENCES users(users_id) ON DELETE CASCADE ON UPDATE CASCADE
) COMMENT '그룹 정보';

-- 친구 정보 테이블
CREATE TABLE friends (
    friends_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '친구 아이디',
    friend_id INT COMMENT '친구 id 외래키',
    users_id INT COMMENT '사용자 id 외래키',
    groups_id BIGINT COMMENT '그룹 id 외래키',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (friends_id),
    FOREIGN KEY (friend_id) REFERENCES users(users_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (users_id) REFERENCES users(users_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (groups_id) REFERENCES groups_info(groups_id) ON DELETE CASCADE ON UPDATE CASCAD
) COMMENT '친구 정보';
```

```sql
-- 캐릭터 정보 테이블
CREATE TABLE characters (
    characters_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '캐릭터 아이디',
    nickname VARCHAR(50) COMMENT '캐릭터 닉네임',
    main_character BOOLEAN COMMENT '메인 캐릭터 여부',
    image_url TEXT COMMENT '이미지 url',
    users_id INT COMMENT '유저 아이디',
    FOREIGN KEY (users_id) REFERENCES users(users_id) ON DELETE CASCADE ON UPDATE CASCADE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (characters_id)
) COMMENT '캐릭터 정보';

-- 모션 정보 테이블
CREATE TABLE motion (
    motion_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '모션 아이디',
    name VARCHAR(50) COMMENT '모션 이름',
    gif_url VARCHAR(100) COMMENT 'gif s3 url',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (motion_id)
) COMMENT '모션 정보';

-- 캐릭터 모션 정보 테이블
CREATE TABLE character_motion (
    character_motion_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '캐릭터 모션 아이디',
    image_url TEXT COMMENT '이미지 url',
    characters_id BIGINT COMMENT '캐릭터 id 외래키',
    motion_id BIGINT COMMENT '모션 id 외래키',
    FOREIGN KEY (characters_id) REFERENCES characters(characters_id) ON DELETE CASCADE ON UPDATE
    FOREIGN KEY (motion_id) REFERENCES motion(motion_id) ON DELETE CASCADE ON UPDATE CASCADE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (character_motion_id)
) COMMENT '캐릭터 모션 정보';

-- 스케치북 정보 테이블
CREATE TABLE sketchbook (
    sketchbook_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '스케치북 아이디',
    is_public BOOLEAN COMMENT '공개여부',
    write_possible BOOLEAN COMMENT '작성여부',
    share_link TEXT COMMENT '공유 링크',
    name VARCHAR(50) COMMENT '스케치북 이름',
    sketchbook_tag VARCHAR(50) COMMENT '스케치북 태그',
    sketchbook_uuid VARCHAR(50) COMMENT '스케치북 uuid',
    is_represent BOOLEAN COMMENT '대표여부',
    users_id INT COMMENT '유저 id, 외래키',
    FOREIGN KEY (users_id) REFERENCES users(users_id) ON DELETE CASCADE ON UPDATE CASCADE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (sketchbook_id)
) COMMENT '스케치북 정보';
```

```
-- 스케치북 캐릭터 모션 정보 테이블
CREATE TABLE sketchbook_character_motion (
    sketchbookcharactermotion_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '스케치북 캐릭터 모션 아이디'
    sketchbook_id BIGINT COMMENT '스케치북 id, 외래키',
    character_motion_id BIGINT COMMENT '캐릭터모션 id, 외래키',
    FOREIGN KEY (sketchbook_id) REFERENCES sketchbook(sketchbook_id) ON DELETE CASCADE ON UPDATE
    FOREIGN KEY (character_motion_id) REFERENCES character_motion(character_motion_id) ON DELETE
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (sketchbookcharactermotion_id)
) COMMENT '스태키북 캐릭터 모션 정보(다대다 테이블)';

CREATE TABLE favorite_sketchbook (
  favoritesketchbook_id BIGINT AUTO_INCREMENT PRIMARY KEY,
  user_id INTEGER NOT NULL,
  sketchbook_id BIGINT NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
  CONSTRAINT `fk_favorite_sketchbook_user`
    FOREIGN KEY (`user_id`)
    REFERENCES `users` (`users_id`)
    ON DELETE CASCADE,
  CONSTRAINT `fk_favorite_sketchbook_sketchbook`
    FOREIGN KEY (`sketchbook_id`)
    REFERENCES `sketchbook` (`sketchbook_id`)
    ON DELETE CASCADE
);

-- 편지 정보 테이블
CREATE TABLE letter (
    letter_id BIGINT AUTO_INCREMENT NOT NULL COMMENT '편지 아이디',
    sender_id INTEGER COMMENT '보낸 사람 정보',
    receiver_id INTEGER COMMENT '받은 사람 정보',
    FOREIGN KEY (sender_id) REFERENCES users(users_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (receiver_id) REFERENCES users(users_id) ON DELETE CASCADE ON UPDATE CASCADE,
    content TEXT COMMENT '편지 내용',
    sketchbook_character_motion_id BIGINT COMMENT '스케치북캐릭터모션 id, 외래키',
    FOREIGN KEY (sketchbook_character_motion_id) REFERENCES sketchbook_character_motion(sketchbo
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP COMMENT '생성 시간',
    modified_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '수정시간'
    is_deleted BOOLEAN COMMENT '삭제 여부',
    PRIMARY KEY (letter_id)
) COMMENT '편지 정보';
```

data.sql

```
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'jesse_dance', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/jes
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'jumping_jacks', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/j
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'hello', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/hello.gif
INSERT INTO motion (is_deleted, created_at, name, gif_url)
```

```
VALUES (false, now(), 'wow', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/wow.gif');
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'stand', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/stand.gi
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'hi', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/hi.gif');
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'happy', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/happy.gi
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'excited', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/excite
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'super', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/super.gi
INSERT INTO motion (is_deleted, created_at, name, gif_url)
VALUES (false, now(), 'dab_dance', 'https://letter-monster.s3.ap-northeast-2.amazonaws.com/dab_
```