AUDISAY

AUDISAY ~시각 장애인을 위한 AI 접근성 도서 뷰어 서비스~

포팅 메뉴얼

목차

I. 개요

1. 프로젝트 개요 : AI 기반 접근성 도서 뷰어 서비스 "AUDISAY"

1-1. 문제 및 배경

현재 시각장애인을 위한 대체자료는 전체 도서의 약 1.06%에 불과하며, 기존 도서를 DAISY 형식으로 변환하는 작업은 대부분 수작업으로 진행되어 매우 더딥니다. 또한, 시중 전자책 서비스는 접근성 모드 부재, 작은 글씨와 복잡한 인터페이스, 이미지에 대한 대체 텍스트 부족 등으로 시각장애인들에게 적합하지 않은 환경을 제공하고 있습니다.

1-2. 목표

본 프로젝트는 AI 기술을 활용하여 시각장애인을 위한 대체자료를 자동으로 제작하고, 이를 기반으로 접근성 중심의 독서 앱을 개발하는 것을 목표로 합니다.

1-3. 프로젝트 소개

"AUDISAY"는 시각장애인을 위한 AI 기반 접근성 도서 서비스입니다. 이 서비스는 시각장애인이 쉽게 접근할 수 있는 도서 자료를 자동으로 제작하고, 사용자 친화적인 인터페이스를 제공하여 독서의 불편함을 해소하고자 합니다. AI 기술을 통해 이미지 및 텍스트의 대체 자료를 효율적으로 생성하고, 사용자가 편리하게 접근할 수 있는 디지털 독서 환경을 제공합니다.

2. 프로젝트 사용 도구

- 이슈 관리 : JIRA
- 형상 관리 : Gitlab
- 커뮤니케이션 : Notion, Mattermost
- 디자인 : Figma
- UCC : 모바비, 타입캐스트
- CI/CD : Jenkins

3. 개발환경

3-1. Frontend

React-Native	0.76.0

React-Native React	18.3.1
Node.js	>=18
TypeScript	5.0.4
Android Build Tools	35.0.0
Min SDK	24
Compile SDK	35
Target SDK	34
NDK	26.1.10909125
Visual Studio Code	1.92.2

3-2. Backend

Java	Liberica JDK 21.0.5+11 2024-10-15 LTS (BellSoft)
Spring Boot	3.3.5
MySQL	8.0.32
MongoDB	8.0.3
Redis	7.4.1
ElasticSearch	8.15.3
Logstash	8.15.3
kibana	8.15.3
JPA/QueryDSL	5.0.0
Swagger	2.2.0
FastAPI	0.115.4
Django	5.1.2
YOLO	DocLayout-YOLO
IntelliJ IDEA	IDE IntelliJ IDEA 2024.1.4

3-3. Infra

AWS EC2	Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1063-aws x86_64)
Docker	27.2.0
Jenkins	2.475
Nginx	1.27.2

4. 외부 서비스

- Amazon S3(Standard): application-prod.properties, base.py 에 해당 내용 있음
  (과금이 발생할 수 있는 키입니다. 취급 주의)
- OpenAI API : base.py 에 해당 내용 있음
  (과금이 발생할 수 있는 키 - 취급 주의)
- NAVER AI CLOVA : base.py 에 해당 내용 있음
  (과금이 발생할 수 있는 키 - 취급 주의)
- Azure AI Vision API : base.py 에 해당 내용 있음
  (과금이 발생할 수 있는 키 - 취급 주의)

5. Gitgnore 처리한 핵심 키들

- Spring : application- secret.properties, .env
- (₩src₩main₩resources, 또는 classPath 에 위치)
- Django : .env (최상단 위치)

II. 빌드

1. 환경변수 형태

1-1. Spring application-prod.properties

```
server.port=${BACKEND_PORT}
server.servlet.context-path=/${CONTEXT_PATH}
# MYSQL
spring.datasource.username=${MYSQL_ROOT_USERNAME}
spring.datasource.url=jdbc:${DB_TYPE}://${MYSQL_BINDING_HOST}:${MYSQL_SERVER_POR
T}/${MYSQL_SCHEMA_NAME}${MYSQL_OPTIONS}
spring.datasource.password=${MYSQL_ROOT_PASSWORD}
spring.datasource.driver-class-name=${MYSQL_DRIVER_CLASS_NAME}
# JPA setting
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=${DB_DIALECT}
# AWS S3
```

```
cloud.aws.region.static=${AWS_REGION}
cloud.aws.s3.bucket=${AWS_S3_BUCKET}
cloud.aws.credentials.access-key=${AWS_ACCESS_KEY}
cloud.aws.credentials.secret-key=${AWS_SECRET_KEY}
cloud.aws.stack.auto=false
cloud.aws.s3.cover.prefix.url=${AWS_S3_COVER_PREFIX_URL}
# swagger path
springdoc.swagger-ui.path=/swagger
paths-to-match=/${CONTEXT_PATH}/**
# current server
current_server_url=${CURRENT_SERVER_URL}
prod_server_url=${PROD_SERVER_URL}
# mongo db
spring.data.mongodb.uri=${MONGO_DB_URI}
# elastic
spring.elasticsearch.uris=${ELASTIC_HOST_PORT}
spring.elasticsearch.username=${ELASTIC_USERNAME}
spring.elasticsearch.password=${ELASTIC_PASSWORD}
spring.elasticsearch.min.score=${ELASTIC_MIN_SCORE}
# redis
spring.data.redis.host=${REDIS_HOST}
spring.data.redis.port=${REDIS_SERVER_PORT}
spring.data.redis.password=${REDIS_PASSWORD}
```

1-2. Django base.py

```
# OCR 설정
NAVER_OCR_INVOKE_URL = env('NAVER_OCR_INVOKE_URL')
NAVER_OCR_SECRET_KEY = env('NAVER_OCR_SECRET_KEY')

# AWS S3 설정
AWS_ACCESS_KEY = env('AWS_ACCESS_KEY')
AWS_SECRET_KEY = env('AWS_SECRET_KEY')
AWS_S3_BUCKET = env('AWS_S3_BUCKET')
AWS_REGION = env('AWS_REGION')
```

```
# Azure 설정
#Azure AI Vision API
AZURE_VISION_ENDPOINT = env('AZURE_VISION_ENDPOINT')
AZURE_VISION_KEY = env('AZURE_VISION_KEY')
AZURE_VISION_REGION = env('AZURE_VISION_REGION')

# OpenAI 설정
OPENAI_AUTH = env('OPENAI_AUTH')
```

2. 빌드 및 배포 설정

2-1. CICD - Docker-compose.yml

```
version: "3.9"

services:
  # 서버 컨테이너
  nginx:
    image: ${DOCKER_IMAGE}:${DOCKER_TAG_NGINX}-latest
    container_name: ${DOCKER_TAG_NGINX}
    ports:
      - "80:80"
      - "443:443"
    environment:
      - TZ=Asia/Seoul
      - SERVER_NAME=${SERVER_NAME}
      - CONTEXT_PATH=${CONTEXT_PATH}
      - BACKEND_PORT=${BACKEND_PORT}
      - DJANGO_PORT=${DJANGO_PORT}
      - DOCKER_TAG_SPRING=${DOCKER_TAG_SPRING}
      - DOCKER_TAG_DJANGO=${DOCKER_TAG_DJANGO}
    volumes:
      - /home/ubuntu/data/certbot/conf:/etc/letsencrypt
      - /home/ubuntu/data/certbot/www:/var/www/certbot
    networks:
      - backend-network
```

```
    depends_on:
      - springboot
      - django

  certbot:
    image: certbot/certbot
    container_name: certbot
    volumes:
      - /home/ubuntu/data/certbot/conf:/etc/letsencrypt
      - /home/ubuntu/data/certbot/www:/var/www/certbot
    depends_on:
      - nginx
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew --webroot -w
/var/www/certbot; sleep 60d & wait $!; done;'"

  springboot:
    image: ${DOCKER_IMAGE}:${DOCKER_TAG_SPRING}-latest
    container_name: ${DOCKER_TAG_SPRING}
    environment:
      - TZ=Asia/Seoul
      - BACKEND_PORT=${BACKEND_PORT}
      - CONTEXT_PATH=${CONTEXT_PATH}
      - MYSQL_ROOT_USERNAME=${MYSQL_ROOT_USERNAME}
      - DB_TYPE=${DB_TYPE}
      - SERVER_NAME=${SERVER_NAME}
      - MYSQL_SERVER_PORT=${MYSQL_SERVER_PORT}
      - MYSQL_SCHEMA_NAME=${MYSQL_SCHEMA_NAME}
      - MYSQL_OPTIONS=${MYSQL_OPTIONS}
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - MYSQL_DRIVER_CLASS_NAME=${MYSQL_DRIVER_CLASS_NAME}
      - DB_DIALECT=${DB_DIALECT}
      - AWS_REGION=${AWS_REGION}
      - AWS_S3_BUCKET=${AWS_S3_BUCKET}
      - AWS_ACCESS_KEY=${AWS_ACCESS_KEY}
      - AWS_SECRET_KEY=${AWS_SECRET_KEY}
```

```
    - AWS_S3_COVER_PREFIX_URL=${AWS_S3_COVER_PREFIX_URL}
    - CURRENT_SERVER_URL=${CURRENT_SERVER_URL}
    - PROD_SERVER_URL=${PROD_SERVER_URL}
    - MYSQL_BINDING_HOST=${MYSQL_BINDING_HOST}
    - MONGO_DB_URI=${MONGO_DB_URI}
    - ELASTIC_PASSWORD=${ELASTIC_PASSWORD}
    - ELASTIC_USERNAME=${ELASTIC_USERNAME}
    - ELASTIC_HOST_PORT=${ELASTIC_HOST_PORT}
    - REDIS_HOST=${REDIS_HOST}
    - REDIS_SERVER_PORT=${REDIS_SERVER_PORT}
    - REDIS_PASSWORD=${REDIS_PASSWORD}
    - ELASTIC_MIN_SCORE=${ELASTIC_MIN_SCORE}
  expose:
    - ${BACKEND_PORT}
  networks:
    - backend-network
  depends_on:
    - mysql
    - redis
    - mongodb
    - kafka

django:
  image: ${DOCKER_IMAGE}:${DOCKER_TAG_DJANGO}-latest
  container_name: ${DOCKER_TAG_DJANGO}
  environment:
    - TZ=Asia/Seoul
    - DJANGO_SETTINGS_MODULE=config.settings.prod
    - DJANGO_SECRET_KEY=${DJANGO_SECRET_KEY}
    - MYSQL_SCHEMA_NAME=${MYSQL_SCHEMA_NAME}
    - MYSQL_ROOT_USERNAME=${MYSQL_ROOT_USERNAME}
    - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
    - MYSQL_BINDING_HOST=${MYSQL_BINDING_HOST}
    - MYSQL_SERVER_PORT=${MYSQL_SERVER_PORT}
    - NAVER_OCR_INVOKE_URL=${NAVER_OCR_INVOKE_URL}
```

```
    - NAVER_OCR_SECRET_KEY=${NAVER_OCR_SECRET_KEY}
    - AWS_REGION=${AWS_REGION}
    - AWS_S3_BUCKET=${AWS_S3_BUCKET}
    - AWS_ACCESS_KEY=${AWS_ACCESS_KEY}
    - AWS_SECRET_KEY=${AWS_SECRET_KEY}
    - AZURE_VISION_ENDPOINT=${AZURE_VISION_ENDPOINT}
    - AZURE_VISION_KEY=${AZURE_VISION_KEY}
    - AZURE_VISION_REGION=${AZURE_VISION_REGION}
    - OPENAI_AUTH=${OPENAI_AUTH}
    - FASTAPI_URL=${FASTAPI_URL}
    - MONGO_DB_URI=${MONGO_DB_URI}
    - DAPHNE_WORKERS=4
    - REDIS_HOST=${REDIS_HOST}
    - REDIS_SERVER_PORT=${REDIS_SERVER_PORT}
    - REDIS_PASSWORD=${REDIS_PASSWORD}
    - SERVER_NAME=${SERVER_NAME}
    - REDIS_BINDING_PORT=${REDIS_BINDING_PORT}
  expose:
    - ${DJANGO_PORT}
  networks:
    - backend-network
  depends_on:
    - mysql
    - kafka

# db 컨테이너
mysql:
  image: mysql:8.0.32
  container_name: mysql-con
  environment:
    - TZ=Asia/Seoul
    - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
  volumes:
    - mysql-vol:/var/lib/mysql
  ports:
```

```
      - "${MYSQL_BINDING_PORT}:3306"
    networks:
      - backend-network

  redis:
    image: redis:latest
    container_name: my-redis
    environment:
      - TZ=Asia/Seoul
    volumes:
      - redis-vol:/data
      - ${REDIS_DEFAULT_CONFIG_FILE}:/usr/local/etc/redis/redis.conf
    ports:
      - "${REDIS_BINDING_PORT}:6379"
    command: redis-server /usr/local/etc/redis/redis.conf
    networks:
      - backend-network

  mongodb:
    image: mongo:latest
    container_name: mongodb-con
    volumes:
      - mongo-vol:/data/db
    environment:
      - TZ=Asia/Seoul
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_INITDB_ROOT_USERNAME}
      -
MONGO_INITDB_ROOT_PASSWORD=${MONGO_INITDB_ROOT_PASSWORD}
    ports:
      - "${MONGO_BINDING_PORT}:27017"
    networks:
      - backend-network

  # Elastic search
  elasticsearch:
```

```
  image: docker.elastic.co/elasticsearch/elasticsearch:8.15.3
  container_name: elasticsearch
  ports:
    - "${ELASTIC_BINDING_PORT}:9200"
  environment:
    - node.name=elasticsearch
    - discovery.type=single-node
    - xpack.security.enabled=true
    - xpack.security.http.ssl.enabled=false
    - ELASTIC_PASSWORD=${ELASTIC_PASSWORD}
    - ES_JAVA_OPTS=-Xms512m -Xmx512m
  networks:
    - backend-network
  ulimits:
    memlock:
      soft: -1
      hard: -1
  volumes:
    - es-vol:/usr/share/elasticsearch/data

kibana:
  image: docker.elastic.co/kibana/kibana:8.15.3
  container_name: kibana
  ports:
    - "${KIBANA_BINDING_PORT}:5601"
  environment:
    - ELASTICSEARCH_HOSTS=${ELASTICSEARCH_HOSTS}
    - ELASTICSEARCH_SERVICEACCOUNTTOKEN=${KIBANA_SERVICE_TOKEN}
  networks:
    - backend-network
  depends_on:
    - elasticsearch

logstash:
  image: docker.elastic.co/logstash/logstash:8.15.3
```

```
    container_name: logstash
    environment:
      - ELASTICSEARCH_HOSTS=${ELASTICSEARCH_HOSTS}
      - ELASTIC_PASSWORD=${ELASTIC_PASSWORD}
      - ELASTIC_USERNAME=${ELASTIC_USERNAME}
      - MYSQL_ROOT_USERNAME=${MYSQL_ROOT_USERNAME}
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - DB_TYPE=${DB_TYPE}
      - MYSQL_DRIVER_CLASS_NAME=${MYSQL_DRIVER_CLASS_NAME}
      - MYSQL_BINDING_HOST=${MYSQL_BINDING_HOST}
      - MYSQL_SERVER_PORT=${MYSQL_SERVER_PORT}
      - MYSQL_SCHEMA_NAME=${MYSQL_SCHEMA_NAME}
      - LS_JAVA_OPTS=-Xmx256m -Xms256m
      - LOGSTASH_OPTS="--xpack.monitoring.enabled=false"
    ports:
      - "${LOGSTASH_BINDING_PORT}:9600"
    networks:
      - backend-network
    depends_on:
      - elasticsearch
    volumes:
      -
/home/ubuntu/logstash/logstash.conf:/usr/share/logstash/pipeline/logstash.conf #
logstash pipeline
      -
/home/ubuntu/logstash/.logstash_jdbc_last_run:/usr/share/logstash/.logstash_jdbc
_last_run

# 전역 설정
volumes:
  mysql-vol:
    external: true
  redis-vol:
    external: true
  mongo-vol:
```

```
    external: true
  es-vol:
    external: true

networks:
  backend-network:
    name: backend-network
    driver: bridge
```

2-2. CICD - Dockerfile

- **Dockerfile (Nginx)**

```
FROM nginx:alpine
LABEL authors="LEE JIHYE"
COPY nginx.conf.template /etc/nginx/nginx.conf.template
EXPOSE 80
CMD ["/bin/sh", "-c", "envsubst '$DOCKER_TAG_SPRING $BACKEND_PORT
$SERVER_NAME $CONTEXT_PATH $DOCKER_TAG_DJANGO $DJANGO_PORT' <
/etc/nginx/nginx.conf.template > /etc/nginx/nginx.conf && nginx -g 'daemon
off;'"]
```

- **Dockerfile (Spring)**

```
FROM openjdk:21-jdk-slim
WORKDIR /app
ARG JAR_FILE_PATH=./build/libs/audisay-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE_PATH} deploy.jar
ENTRYPOINT ["java", "-jar", "/app/deploy.jar"]
```

- **Dockerfile (Django)**

```
FROM python:3.12
# Install Poppler
RUN apt-get update && apt-get install -y poppler-utils
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
```

```
EXPOSE 8000
# ASGI 설정
CMD ["daphne", "-b", "0.0.0.0", "-p", "8000", "config.asgi:application"]
```

2-3. CICD - Nginx.comf.template

```
events {
    worker_connections 1024;
}

http  {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    sendfile on; # 로컬에 저장된 파일 전송

    gzip on;
    gzip_comp_level 5;
    gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;

    server {
        listen 80;
        server_name ${SERVER_NAME};

        # Let's Encrypt 인증서 발급을 위한 설정
        # Let's Encrypt 가 도메인 소유권을 확인하는 데 사용
        location /.well-known/acme-challenge/ {
            allow all;
            root /var/www/certbot;
        }
        # HTTP 를 HTTPS 로 리다이렉트
        location / {
            return 301 https://$host$request_uri;
        }
    }
```

```
    server {
        listen 443 ssl;
        server_name ${SERVER_NAME};
        server_tokens off;

        # 인증서 체인
        ssl_certificate /etc/letsencrypt/live/${SERVER_NAME}/fullchain.pem;
        # 개인키
        ssl_certificate_key /etc/letsencrypt/live/${SERVER_NAME}/privkey.pem;
        # SSL 추가 설정
        include /etc/letsencrypt/options-ssl-nginx.conf;
        # DH 파라미터
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # 보안 헤더
        # HSTS(HTTP Strict Transport Security) 설정
        add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;
        add_header X-Content-Type-Options nosniff;
        add_header X-Frame-Options DENY;
        add_header X-XSS-Protection "1; mode=block";
        # add_header Content-Security-Policy "default-src 'self'; script-src 'self'
'unsafe-inline'";
        add_header Referrer-Policy "no-referrer-when-downgrade";

        # Spring Boot 외부 연결
        location /${CONTEXT_PATH}/ {
            proxy_pass http://${DOCKER_TAG_SPRING}:${BACKEND_PORT};
            proxy_set_header Host $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;
        }
```

```
        # Django 외부 연결
        location /${CONTEXT_PATH}/registration/ {
            # 대용량 파일 전송 설정 / 타임아웃 설정
            client_max_body_size 2G;
            proxy_read_timeout 300;
            proxy_connect_timeout 300;
            proxy_send_timeout 300;

            proxy_pass http://${DOCKER_TAG_DJANGO}:${DJANGO_PORT};
            proxy_set_header Host $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;
        }

        location / {
            root /usr/share/nginx/html;
            index index.html index.htm;
            try_files $uri $uri/ /index.html;
        }
    }
```

2-4. CICD - Jenkins Pipeline

```
pipeline {
    agent any
    environment {
        DOCKERHUB_CREDENTIALS_ID = 'dockerhub-jenkins'
        DOCKERHUB_CREDENTIALS = credentials('dockerhub-jenkins')
        DOCKER_IMAGE = 'jihye9807/audisay'

        DOCKER_TAG_NGINX = "nginx"
        DOCKER_TAG_SPRING = "springboot"
        DOCKER_TAG_DJANGO = "django"
        springDockerImage = ''
        nginxDockerImage = ''
        djangoDockerImage = ''
```

```
    }
    tools {
        jdk 'JDK 21'
    }

    stages {
        stage('GitLab-Clone') {
            steps {
                git branch: 'develop', credentialsId: 'gitlab_account', url:
'https://lab.ssafy.com/s11-final/S11P31D208'
            }
        }
        stage('Build') {
            parallel {
                stage('Spring-Build') {
                    steps {
                        echo "Spring build"
                        dir("./BE/Spring/audisay") {
                            sh "chmod +x ./gradlew"
                            sh "./gradlew clean build -x test --stacktrace"
                        }
                    }
                }
                stage('Django-Build') {
                    steps {
                        echo "Django build"
                        dir("./BE/Django") {
                            sh "pip install -r requirements.txt"
                            sh "python manage.py collectstatic --noinput --
settings=config.settings.prod"
                        }
                    }
                }
            }
        }
```

```
        stage('Build-Docker-Images') {
            parallel {
                stage('Spring-Docker-Build') {
                    steps {
                        echo "Spring Docker Build"
                        dir('./BE/Spring/audisay') {
                            script {
                                springDockerImage =
docker.build("${DOCKER_IMAGE}:${DOCKER_TAG_SPRING}-latest")
                            }
                        }
                    }
                }
                stage('Django-Docker-Build') {
                    steps {
                        echo "Django Docker Build"
                        dir('./BE/Django') {
                            script {
                                djangoDockerImage =
docker.build("${DOCKER_IMAGE}:${DOCKER_TAG_DJANGO}-latest")
                            }
                        }
                    }
                }
                stage('Nginx-Docker-Build') {
                    steps {
                        echo "Nginx Docker Build"
                        script {
                            nginxDockerImage =
docker.build("${DOCKER_IMAGE}:${DOCKER_TAG_NGINX}-latest")
                        }
                    }
                }
            }
        }
```

```
        stage('Push-Docker-Images') {
            steps {
                script {
                    docker.withRegistry("", env.DOCKERHUB_CREDENTIALS_ID) {
                        springDockerImage.push()
                        djangoDockerImage.push()
                        nginxDockerImage.push()
                        springDockerImage.push("${DOCKER_TAG_SPRING}-
${env.BUILD_NUMBER}")
                        djangoDockerImage.push("${DOCKER_TAG_DJANGO}-
${env.BUILD_NUMBER}")
                        nginxDockerImage.push("${DOCKER_TAG_NGINX}-
${env.BUILD_NUMBER}")
                    }
                }
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploy stage'

                withCredentials([file(credentialsId: 'docker-env-file', variable:
'DOCKER_ENV_FILE')]) {
                    sh '''
                        # Secret File 을 .env 로 복사
                        # .env 파일의 권한을 600 으로 설정
                        cp ${DOCKER_ENV_FILE} .env
                        chmod 600 .env
                        # 컨테이너 재시작, 기존 데이터 유지
                        docker-compose up -d
                    '''
                }
            }
        }
        stage('Cleanup') {
```

```
            steps {
                echo 'cleanup docker image'
                // 아래 스크립트의 리스트에 env.DOCKER_TAG_DJANGO 추가할 것
                script {
                    sh "docker image prune -a -f"
                    [env.DOCKER_TAG_SPRING, env.DOCKER_TAG_NGINX,
env.DOCKER_TAG_DJANGO].each { service ->
                        sh "docker rmi ${DOCKER_IMAGE}:${service}-
${env.BUILD_NUMBER} || true"
                    }
                }
            }
        }
    }
    post {
        always {
            echo 'I complete CI/CD'
        }
        success{
            echo 'I success CI/CD'
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout:
true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae",
returnStdout: true).trim()
                mattermostSend (color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
${Author_ID}(${Author_Name})₩n(<${env.BUILD_URL}|Details>)",
                endpoint:
'https://meeting.ssafy.com/hooks/wei9cccmrjf7pqj3dj6nqjxwua',
                channel: 'cicd'
                )
            }
        }
        failure{
```

```
            echo 'I fail CI/CD'
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout:
true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae",
returnStdout: true).trim()
                mattermostSend (color: 'danger',
                message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
${Author_ID}(${Author_Name})₩n(<${env.BUILD_URL}|Details>)",
                endpoint:
'https://meeting.ssafy.com/hooks/wei9cccmrjf7pqj3dj6nqjxwua',
                channel: 'cicd'
                )
            }
        }
    }
}
```

2-5. 접속 정보 등 프로젝트(ERD) 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

- env