포팅 메뉴얼

속성	
유형	배포 및 운영

Front-end

Stack

- axios 1.3.4
- lottie-react 2.4.0
- react 17.0.2
- react-dom 17.0.2
- react-router-dom 6.8.2
- recoil 0.7.7
- rsuite 5.28.3
- styled-components 5.3.8
- typescript 4.9.5
- react-speech-recognition 3.10.0
- react-lazy-load-image-component 1.5.6

Api

- Kakao Map Api
- Kakao Login Api

Tools

- Figma

Library

Name	Description
Kakao	카카오 소셜 로그인
KakaoMap	카카오 지도 라이브러리
Lottie	리액트 애니메이션 라이브러리

Name	Description
Rsuite	리액트 css 프레임워크
react-speech-recognition	리액트 음성 인식 라이브러리
react-lazy-load-image-component	리액트 지연 로딩 라이브러리

Back-end

- Data analistic

- jamo 0.4.1
- Mecab-ko-for-Google-Colab
- krwordrank 1.0.3
- gensim 4.0.0
- scikit-learn 1.2.2

- Django

- Django 3.2
- Python 3.9

▼ requirement.txt

```
asgiref==3.6.0
cffi==1.15.1
cryptography==39.0.2
django-cors-headers==3.14.0
django-environ==0.10.0
djangorestframework==3.14.0
djongo==1.3.6
dnspython==2.3.0
joblib==1.2.0
numpy==1.24.2
pandas==1.5.3
pycparser==2.21
PyJWT==2.6.0
pymongo==4.3.3
python-dateutil==2.8.2
python-dotenv==1.0.0
pytz==2022.7.1
scikit-learn==1.2.2
scipy==1.10.1
six==1.16.0
```

```
sqlparse==0.2.4
threadpoolctl==3.1.0
```

- Spring Boot

IntelliJ IDEA 2022.3.1 (Ultimate Edition)

Spring Boot 2.7.9

JAVA 11

▼ build.gradle

```
implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'
implementation 'org.springframework.boot:spring-boot-starter-web'
compileOnly 'org.projectlombok:lombok'
developmentOnly 'org.springframework.boot:spring-boot-devtools'
annotationProcessor 'org.projectlombok:lombok'
testImplementation 'org.springframework.boot:spring-boot-starter-test'
implementation 'org.springframework.boot:spring-boot-starter-webflux'

implementation group: 'org.json', name: 'json', version: '20230227'

// security 관련 의존성
implementation 'org.springframework.boot:spring-boot-starter-security'

//jwt 관련 의존성
implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
implementation 'io.jsonwebtoken:jjwt-impl:0.11.5'
implementation 'io.jsonwebtoken:jjwt-jackson:0.11.5'
```

AWS 환경 설정

Java 11 설치

```
sudo apt-get update
sudo apt-get install openjdk-11-jdk
# Check java version
java -version
```

Python 설치

```
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa

# 파이썬 3.9 버전 설치
sudo apt install python3.9
```

Docker 설치

```
# 패키지 업데이트
sudo apt-get update -y

# 기존에 있던 도커 삭제
sudo apt-get remove docker docker-engine docker.io -y

# 도커 설치
sudo apt-get install docker.io -y

# docker 서비스 실행
sudo service docker start

# /var/run/docker.sock 파일의 권한을 666으로 변경하여 그룹 내 다른 사용자도 접근 가능하게 변경
sudo chmod 666 /var/run/docker.sock

# ubuntu 유저를 docker 그룹에 추가
sudo usermod -a -G docker ubuntu
```

```
# 자동 설치 스크립트
sudo wget -qO- https://get.docker.com/ | sh
```

Jenkins 설치

```
$ docker pull jenkins/jenkins

$ docker run --name jenkins -d -p 8080:8080 -p 50000:50000 -v /home/jenkins:/var/jenki
ns_home -v /var/run/docker.sock:/var/run/docker.sock -u root jenkins/jenkins

$ docker exec -it jenkins /bin/bash

# 비밀번호 확인
$ cat /var/jenkins_home/secrets/initialAdminPassword
```

```
# jenkins container 접속
docker exec -it jenkins /bin/bash

# linux 버전 확인
cat /etc/issue

# Docker 설치
## - Old Version Remove
apt-get remove docker docker-engine docker.io containerd runc
## - Setup Repo
apt-get update
apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/ke
yrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] http
s://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
## - Install Docker Engine
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Jenkins Pipeline

Front-End (React)

```
pipeline{
    agent any

    environment{
        imagename = "dawool-front"
        registryCredential = "b48d2a30-0626-4b3a-bd19-9a6f5b8b9c6f"
        sshCredential = "e4230d01-d0ef-4cf5-abd3-4a71c2db2cab"
        dockerImage = ''
    }

    stages{
        stage('git clone'){
            steps{
                git branch: 'develop',
                credentialsId: 'ea9ec146-c4f4-4c0e-ae6b-59b24aeb349b',
                url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D105.git'
            }
        }
        stage('Build'){
            steps{
```

```
                dir('frontend'){
                    nodejs(nodeJSInstallationName: 'NodeJS 16.13.0'){
                        sh 'npm install'
                        sh 'CI=false npm run build'
                    }
                }
            }
        }
        stage('docker-build'){
            steps{
                echo 'Build Docker'
                dir('frontend'){
                    script{
                        dockerImage = docker.build "betheia/"+imagename
                    }
                }
            }
        }
        stage('docker-push'){
            steps{
                echo 'Push Docker'
                script{
                    docker.withRegistry('', registryCredential){
                        dockerImage.push()
                    }
                }
            }
        }
        stage('docker-run'){
            steps{
                echo 'Pull Docker Image & Docker Image Run'
                sshagent([sshCredential]) {
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er pull betheia/dawool-front'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er ps -q --filter name=front | grep -q . && docker rm -f \$(docker ps -aq --filter nam
e=front)'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er run -d --name front -p 3000:3000 betheia/dawool-front'"
                }
            }
        }
    }
}
```

Back-End(Spring Boot)

```
pipeline {
    agent any

    environment{
        imagename = "dawool-backend"
        registryCredential = "b48d2a30-0626-4b3a-bd19-9a6f5b8b9c6f"
        sshCredential = "e4230d01-d0ef-4cf5-abd3-4a71c2db2cab"
```

```
        dockerImage = ''
    }

    stages{
        stage('git clone') {
            steps {
                git branch: 'develop',
                    credentialsId: 'ea9ec146-c4f4-4c0e-ae6b-59b24aeb349b',
                    url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D105.git'
            }
        }
        stage('build') {
            steps{
                dir('backend/api-server'){
                    sh "chmod +x ./gradlew"
                    sh "./gradlew clean bootJar"
                }
            }
        }
        stage('docker-build'){
            steps{
                echo 'Build Docker'
                dir('backend/api-server'){
                    script{
                        dockerImage = docker.build "betheia/"+imagename
                    }
                }
            }
        }
        stage('docker-push'){
            steps{
                echo 'Push Docker'
                echo "${env.BUILD_NUMBER}"
                script{
                    docker.withRegistry('', registryCredential){
                        dockerImage.push("${env.BUILD_NUMBER}")
                        dockerImage.push("latest")
                        dockerImage
                    }
                }
            }
        }
        stage('docker-run'){
            steps{
                echo 'Pull Docker Image & Docker Image Run'
                sshagent([sshCredential]) {
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er ps -aq --filter name=backend | grep -q . && docker rm -f \$(docker ps -aq --filter
 name=backend)'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er rmi betheia/dawool-backend'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er pull betheia/dawool-backend:latest'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er run -d --name backend -p 8888:8888 betheia/dawool-backend:latest'"
                }
            }
        }
```

```
    }
}
```

Back-End(Django)

```
pipeline {
    agent any

    environment{
        imagename = "dawool-django"
        registryCredential = "b48d2a30-0626-4b3a-bd19-9a6f5b8b9c6f"
        sshCredential = "e4230d01-d0ef-4cf5-abd3-4a71c2db2cab"
        dockerImage = ''
    }

    stages{
        stage('git clone') {
            steps {
                git branch: 'develop',
                    credentialsId: 'ea9ec146-c4f4-4c0e-ae6b-59b24aeb349b',
                    url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22D105.git'
            }
        }
        stage('docker-build'){
            steps{
                echo 'Build Docker'
                dir('backend/django'){
                   script{
                        dockerImage = docker.build "betheia/"+imagename
                    }
                }
            }
        }
        stage('docker-push'){
            steps{
                echo 'Push Docker'
                script{
                    docker.withRegistry('', registryCredential){
                        dockerImage.push()
                    }
                }
            }
        }
        stage('docker-run'){
            steps{
                echo 'Pull Docker Image & Docker Image Run'
                sshagent([sshCredential]) {
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er pull betheia/dawool-django'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er ps -aq --filter name=django | grep -q . && docker rm -f \$(docker ps -aq --filter n
ame=django)'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@j8d105.p.ssafy.io 'dock
er run -d --name django -p 5000:5000 betheia/dawool-django'"
```

```
                }
            }
        }
    }
}
```