포팅메뉴얼

개발 환경

1. **Front**

```
{  "name": "drillapp",  "version": "0.0.1",  "private": true,  "scripts": {    "androi
d": "react-native run-android",    "ios": "react-native run-ios",    "lint": "eslint
.",    "start": "react-native start",    "test": "jest"  },

"dependencies": {
  "@react-native-async-storage/async-storage": "^1.19.3",
   "@react-native-seoul/kakao-login": "^5.3.0",
    "@react-navigation/bottom-tabs": "^6.5.9",
    "@react-navigation/native": "^6.1.9",
    "@react-navigation/native-stack": "^6.9.14",
    "@react-navigation/stack": "^6.3.20",
    "@reduxjs/toolkit": "^1.9.7",
    "aws-sdk": "^2.1486.0",
    "axios": "^1.6.0",
    "react": "18.2.0",
    "react-native": "^0.72.6",
    "react-native-calendars": "^1.1301.0",
    "react-native-config": "^1.5.1",
    "react-native-dotenv": "^3.4.9",
    "react-native-dropdown-select-list": "^2.0.5",
    "react-native-element-dropdown": "^2.10.0",
    "react-native-image-picker": "^7.0.2",
    "react-native-keyboard-aware-scroll-view": "^0.9.5",
    "react-native-modal": "^13.0.1",
    "react-native-progress": "^5.0.1",
    "react-native-safe-area-context": "^4.7.2",
    "react-native-screens": "^3.25.0",
    "react-native-snackbar": "^2.6.2",
    "react-native-splash-screen": "^3.3.0",
    "react-native-vector-icons": "^10.0.0",
    "react-native-video": "^5.2.1",
    "react-redux": "^8.1.3",
    "redux-logger": "^3.0.6",
    "styled-components": "^6.1.0"  },

"devDependencies": {
  "@babel/core": "^7.20.0",
    "@babel/preset-env": "^7.20.0",
    "@babel/preset-typescript": "^7.23.2",
    "@babel/runtime": "^7.20.0",
    "@react-native/eslint-config": "^0.72.2",
```

```
    "@react-native/metro-config": "^0.72.11",
    "@tsconfig/react-native": "^3.0.0",
    "@types/lodash": "^4.14.200",
    "@types/react": "^18.0.24",
    "@types/react-native-dotenv": "^0.2.2",
    "@types/react-native-vector-icons": "^6.4.15",
    "@types/react-native-video": "^5.0.17",
    "@types/react-test-renderer": "^18.0.0",
    "@types/redux-logger": "^3.0.11",
    "@types/styled-components": "^5.1.28",
    "@types/styled-components-react-native": "^5.2.3",
    "babel-jest": "^29.2.1",
    "eslint": "^8.19.0",
    "jest": "^29.2.1",
    "metro-react-native-babel-preset": "0.76.8",
    "prettier": "^2.4.1",
    "react-test-renderer": "18.2.0",
    "typescript": "^4.8.4"  },

  "engines": {    "node": ">=16"  }}
```

2. **Back**

```
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
implementation 'org.springframework.boot:spring-boot-starter-data-redis'
implementation 'org.springframework.boot:spring-boot-starter-jdbc'
implementation 'org.springframework.boot:spring-boot-starter-web'
implementation "org.apache.tomcat.embed:tomcat-embed-jasper"
implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'
implementation 'org.springframework.boot:spring-boot-starter-security'
implementation 'org.springframework.boot:spring-boot-starter-validation'
// 소셜로그인
implementation 'org.springframework.cloud:spring-cloud-starter-openfeign'
implementation group: 'com.google.code.gson', name: 'gson', version: '2.9.0'
implementation group: 'org.bgee.log4jdbc-log4j2', name: 'log4jdbc-log4j2-jdbc4', versi
on: '1.16'
implementation 'com.querydsl:querydsl-jpa'
implementation group: 'com.querydsl', name: 'querydsl-core', version: '5.0.0'
annotationProcessor "com.querydsl:querydsl-apt:${dependencyManagement.importedProperti
es['querydsl.version']}:jpa"
annotationProcessor "jakarta.annotation:jakarta.annotation-api"
annotationProcessor "jakarta.persistence:jakarta.persistence-api"

implementation("io.springfox:springfox-swagger2:3.0.0")
implementation("io.springfox:springfox-swagger-ui:3.0.0")
implementation("io.springfox:springfox-data-rest:3.0.0")
implementation("io.springfox:springfox-bean-validators:3.0.0")
implementation("io.springfox:springfox-boot-starter:3.0.0")

compileOnly 'org.projectlombok:lombok'
runtimeOnly 'com.mysql:mysql-connector-j'
```

```
annotationProcessor 'org.projectlombok:lombok'
testImplementation 'org.springframework.boot:spring-boot-starter-test'
testImplementation 'org.springframework.security:spring-security-test'

implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
implementation 'io.jsonwebtoken:jjwt-impl:0.11.5'
implementation 'io.jsonwebtoken:jjwt-jackson:0.11.5'
implementation 'com.auth0:java-jwt:4.4.0'
runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.2', 'io.jsonwebtoken:jjwt-jackson:0.11.2'
```

3. **AI**

```
absl-py==2.0.0
annotated-types==0.6.0
antlr4-python3-runtime==4.9.3
anyio==3.7.1
attrs==23.1.0
boto3==1.28.71
botocore==1.31.71
certifi==2023.7.22
cffi==1.16.0
charset-normalizer==3.3.2
click==8.1.7
cloudpickle==3.0.0
colorama==0.4.6
contourpy==1.1.1
cycler==0.12.1
exceptiongroup==1.1.3
fastapi==0.103.2
filelock==3.13.1
flatbuffers==23.5.26
fonttools==4.44.0
fsspec==2023.10.0
fvcore==0.1.5.post20221221
h11==0.14.0
httptools==0.6.1
idna==3.4
importlib-resources==6.1.0
iopath==0.1.10
Jinja2==3.1.2
jmespath==1.0.1
kiwisolver==1.4.5
MarkupSafe==2.1.3
matplotlib==3.7.3
mediapipe==0.10.7
mpmath==1.3.0
networkx==3.1
numpy==1.24.4
omegaconf==2.3.0
opencv-contrib-python==4.8.1.78
opencv-python==4.8.1.78
```

```
packaging==23.2
Pillow==10.1.0
portalocker==2.8.2
protobuf==3.20.3
pycocotools==2.0.7
pycparser==2.21
pydantic==2.4.2
pydantic_core==2.10.1
pyparsing==3.1.1
python-dateutil==2.8.2
python-dotenv==1.0.0
PyYAML==6.0.1
requests==2.31.0
s3transfer==0.7.0
six==1.16.0
sniffio==1.3.0
sounddevice==0.4.6
starlette==0.27.0
sympy==1.12
tabulate==0.9.0
termcolor==2.3.0
torch==2.1.0
torchvision==0.16.0
tqdm==4.66.1
typing_extensions==4.8.0
urllib3==1.26.18
uvicorn==0.23.2
watchfiles==0.21.0
websockets==11.0.3
yacs==0.1.8
zipp==3.17.0
```

4. **Jenkins**

```
pipeline {
    agent any
    // Build stage
    //Multi-stage-build를 수행하여 중간 이미지 생성
    stages {
        stage('Build') {
            steps {
                dir('drill') { // 'drill' 디렉토리로 이동
                    script {
                        sh 'docker build --target builder -t drill_builder:latest .'
// 중간 이미지 생성
                    }
                }
            }
        }
        // Docker Build stage
        //최종 Docker 이미지를 생성.
```

```
        stage('Docker Build') {
            steps {
                dir('drill') { // 'drill' 디렉토리로 이동
                    script {
                        sh 'docker build -t drill_back:latest .' // 이미지를 빌드
                    }
                }
            }
        }
        //Blue Health check
        //사용중인 경우 green
        //사용중이 아니면 blue에 신버전 배포

        //Deploy Health check
        //새로 배포한 버전의 Health check를 진행한다.
        //5초 간격으로 10번 진행
        //실패시 종료

        // Finish
        //1. nginx의 리버스 프록시 방향을 새로운 서버로 설정한다.
        //2. 기존의 서버를 종료한다.
        //3. 사용하지 않는 이미지를 삭제한다.
        stage('Blue Health check and Deploy') {
            steps {
                script {
                    sh '''
                    #!/bin/bash



                    echo "http://${ip}:${blue_port} Health check"

                    if timeout 10s curl -s "http://${ip}:${blue_port}" > /dev/null
                    then
                        echo "http://${ip}:${blue_port} Health check success"
                        target_container_name=${green_container_name}
                        target_port=${green_port}
                    else
                        echo "http://${ip}:${blue_port} Health check fail"
                        target_container_name=${blue_container_name}
                        target_port=${blue_port}
                    fi

                    docker run -d --name ${target_container_name} -p ${target_port}:80
60 -u root drill_back:latest



                    echo "deploy health check"

                    for retry_count in \$(seq 60)
                    do
                    if curl -s "http://${ip}:${target_port}" > /dev/null
                    then
                        echo "Deploy Health check success"
                        break
```

```
                    fi

                    if [ $retry_count -eq 60 ]
                    then
                        echo "Deploy Health check failed"
                        exit 1
                    fi
                    sleep 2
                    done



                    echo "finish"

                    if [ $target_port -eq 8060 ]
                    then
                        echo 'set $service_port 8060;' > /etc/nginx/conf.d/service-ur
l.inc
                    else
                        echo 'set $service_port 8061;' > /etc/nginx/conf.d/service-ur
l.inc
                    fi

                    docker restart nginx

                    if [ "${target_port}" -eq "${blue_port}" ]
                    then
                        docker rm -f ${green_container_name} || true
                    else
                        docker rm -f ${blue_container_name} || true
                    fi

                    docker image prune -f
                    '''
                }
            }
        }
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
rue).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 ${Author_I
D}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/uomn1ut56tf3zfijsk84xoq9p
c',
                channel: 'A106_Back_Build'
                )
            }
        }
        failure {
            script {
```

```
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: t
rue).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 ${Author_I
D}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/uomn1ut56tf3zfijsk84xoq9p
c',
                channel: 'A106_Back_Build'
                )
            }
        }
    }
}
```

5. **Docker**

```
# Build
FROM openjdk:11-jdk-slim AS builder

WORKDIR /app
COPY . .

RUN chmod +x ./gradlew
RUN ./gradlew build -x test

# Production
FROM openjdk:11-jdk-slim

COPY --from=builder /app/build/libs/*.jar app.jar
# EXPOSE 8060

ENTRYPOINT ["java", "-Dspring.profiles.active=prd", "-jar", "/app.jar"]
```

6. **Nginx**

```
# HTTP 서버 설정
server {
    # 80 포트에서 들어오는 HTTP 요청을 수신
    listen 80;
    # 요청을 처리할 도메인 이름
    server_name k9a106.p.ssafy.io;
    # 서버 버전 정보 숨기기 (보안상의 이유)
    server_tokens off;
    # 모든 HTTP 요청을 HTTPS로 리다이렉트
    location / {
```

```
        return 301 https://$server_name$request_uri;
    }
}

# HTTPS 서버 설정
server {
    # 443 포트에서 들어오는 HTTPS 요청을 수신
    listen 443 ssl;
    server_name k9a106.p.ssafy.io;
    server_tokens off;
    # 액세스 로그 기록 비활성화
    access_log off;
    # Let's Encrypt로부터 받은 SSL 인증서와 키 파일 경
    ssl_certificate /etc/letsencrypt/live/k9a106.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k9a106.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf; # SSL 설정 포함
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # DH 파라미터 경로


    include /etc/nginx/conf.d/service-url.inc;


    # 기본 요청을 특정 도메인의 서버 그룹으로 프록시
    location / {
        proxy_passhttp://15.164.244.222:$service_port;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Fowarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_redirect off;
    }
}

# 백엔드 서버 그룹 정의
upstream backend_servers {
    server k9a106.p.ssafy.io:8061;
    server k9a106.p.ssafy.io:8062;
    server k9a106.p.ssafy.io:8063;
    server k9a106.p.ssafy.io:8064;
}
```

외부서비스

카카오 소셜 로그인

DB파일

별첨

시연 시나리오

1. 첫 페이지
    - 로컬 회원가입
    - 로컬 로그인 ⇒ 첫 로그인이면 닉네임과 관심 지점 설정 페이지
    - 카카오 로그인 ⇒ 첫 로그인이면 닉네임과 관심 지점 설정 페이지

2. 메인 페이지
    - 지점, 난이도, 코스를 선택하여 랭킹 확인
    - 하단 네비게이션 바를 통하여 원하는 기능으로 이동
      (메인 페이지, 영상 리스트, 영상 촬영, 게시물 업로드, 마이페이지 순서)

3. 영상 리스트 페이지
    - 초기 좋아요 많은 순으로 영상이 나열
    - 유저 이름을 검색하여 원하는 유저의 영상만 확인가능
    - 지점, 코스별로 영상 확인 가능

4. 영상 촬영
    - 카메라 열기 버튼을 클릭하며 카메라가 동작되고 저장하면 핸드촌 갤러리에 저장

5. 게시글 작성
    - ‘영상을 선택해주세요’ 문구를 클릭하며 갤러리 내 영상 선택가능
    - ‘문구를 입력해주세요’ 문구를 클릭하면 게시글 내용 작성 가능
    - 영상의 코스의 지점과 색깔을 선택하여 게시글 업로드 버튼을 클릭

- 영상속 주인공의 코스 완등 여부를 ai가 판단하여 완등했다면 업로드 가능

6. 마이페이지
    - 상단에서 자신의 등급과 다음 등급까지의 경험치를 확인 가능
    - 아래에서 본인이 업로드한 영상 확인 가능