포팅메뉴얼

1. 사용 도구

- **이슈관리** : Notion, Jira
- **형상관리** : GitLab
- **커뮤니케이션** : MatterMost
- **디자인** : Figma
- **CI/CD** : Docker, DockerCompose, Jenkins

2. 개발 도구

2.1 Frontend

- Node.js : 20.15.0
- React : 18.3.1
- ReactNative : 0.76.1
- TypeScript : 5.0.4
- Zustand : 5.0.1

2.2 BackEnd

- Java : 17
- Gradle : 8.10
- SpringBoot : 3.3.5
- JsonWebToken : 0.12.3
- Mysql : 8.0.32

- Swagger(OpenAPI) : 2.1.0
- OpenAi : gpt-4o

3. Settings

1. React

1. package.json

```
{
  "name": "Ssook",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "android": "react-native run-android",
    "ios": "react-native run-ios",
    "lint": "eslint .",
    "start": "react-native start",
    "test": "jest"
  },
  "dependencies": {
    "@react-native-community/datetimepicker": "^8.2.0",
    "@react-native-community/geolocation": "^3.4.0",
    "@react-native-ml-kit/text-recognition": "^1.5.2",
    "@react-native-picker/picker": "^2.9.0",
    "@react-navigation/bottom-tabs": "^6.6.1",
    "@react-navigation/native": "^6.1.18",
    "@react-navigation/stack": "^6.4.1",
    "axios": "^1.7.7",
    "lottie-react-native": "^7.0.0",
    "lucide-react-native": "^0.454.0",
    "react": "18.3.1",
    "react-native": "0.76.1",
    "react-native-fast-image": "^8.6.3",
    "react-native-geolocation-service": "^5.3.1",
    "react-native-gesture-handler": "^2.20.2",
    "react-native-image-picker": "^7.1.2",
    "react-native-location": "^2.5.0",
    "react-native-nfc-manager": "^3.16.0",
    "react-native-permissions": "^5.1.0",
    "react-native-reanimated": "^3.16.1",
    "react-native-safe-area-context": "^4.14.0",
    "react-native-screens": "^3.35.0",
    "react-native-svg": "^15.8.0",
    "react-native-vector-icons": "^10.2.0",
    "react-native-webview": "^13.12.3",
    "zustand": "^5.0.1"
  },
  "devDependencies": {
    "@babel/core": "^7.25.2",
    "@babel/preset-env": "^7.25.3",
    "@babel/runtime": "^7.25.0",
    "@react-native-community/cli": "15.0.0",
    "@react-native-community/cli-platform-android": "15.
0.0",
    "@react-native-community/cli-platform-ios": "15.0.
0",
    "@react-native/babel-preset": "0.76.1",
    "@react-native/eslint-config": "0.76.1",
    "@react-native/metro-config": "0.76.1",
    "@react-native/typescript-config": "0.76.1",
    "@types/react": "^18.2.6",
    "@types/react-test-renderer": "^18.0.0",
    "babel-jest": "^29.6.3",
    "eslint": "^8.19.0",
    "jest": "^29.6.3",
    "prettier": "2.8.8",
    "react-test-renderer": "18.3.1",
    "typescript": "5.0.4"
  },
  "engines": {
    "node": ">=18"
  }
}
```

2. Spring

1. **build.gradle**

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.5'
    id 'io.spring.dependency-management' version '1.1.6'
}

group = 'com.stillalive'
version = '0.0.1-SNAPSHOT'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot
-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot
-starter-security'
    implementation 'org.springframework.boot:spring-boot
-starter-validation'
    implementation 'org.springframework.boot:spring-boot
-starter-web'
```

```
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.h2database:h2'
    runtimeOnly 'com.mysql:mysql-connector-j:8.0.32'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'

    // Swagger
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.1.0'

    // Thymeleaf
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'

    // dotenv
    implementation 'io.github.cdimascio:java-dotenv:5.2.2'

    //JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.12.3'
    implementation 'io.jsonwebtoken:jjwt-impl:0.12.3'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.12.3'

    // Mustache
    implementation 'org.springframework.boot:spring-boot-starter-mustache'

    // Quartz
    implementation 'org.springframework.boot:spring-boot-starter-quartz'
```

```
}

tasks.named('test') {
    useJUnitPlatform()
}
```

2. application.yml

```
server:
    port: ${SERVER_PORT}

    servlet:
        encoding:
            force-response: true

    tomcat:
        connection-timeout: 60s

spring:
    application:
        name: Ssook_BE

    config:
        import: optional:file:.env[.properties]

    web:
        encoding:
            charset: UTF-8
            enabled: true
            force: true
    datasource:
        url: jdbc:mysql://${MYSQL_HOST}:${MYSQL_PORT}/
${MYSQL_DATABASE}?useUnicode=true&characterEncoding=utf8
&characterSetResults=utf8&serverTimezone=Asia/Seoul
        username: ${MYSQL_ROOT_USER}  # ?? ??? ??
        password: ${MYSQL_ROOT_PASSWORD}  # ?? ??? ??
        driver-class-name: com.mysql.cj.jdbc.Driver

    jpa:
        properties:
            hibernate:
                show_sql: false
                format_sql: false
        hibernate:
            ddl-auto: ${SPRING_JPA_HIBERNATE_DDL_AUTO}
            database-platform: org.hibernate.dialect.MyS
QLDialect
            temp:
                use_jdbc_metadata_defaults: false
    devtools:
        restart:
            enabled: false

    jwt:
        secret: ${JWT_SECRET}


    quartz:
        job-store-type: jdbc
        jdbc:
            initialize-schema: always
        properties:
            org:
                quartz:
                    scheduler:
                        instanceName: BalanceScheduler
                    jobStore:
                        class: org.quartz.impl.jdbcjobst
ore.JobStoreTX
                        driverDelegateClass: org.quartz.
impl.jdbcjobstore.StdJDBCDelegate
                        tablePrefix: QRTZ_

                    threadPool:
                        threadCount: 5
```

```
# Swagger Configuration
springdoc:
    api-docs:
        enabled: true
    swagger-ui:
        enabled: true
        path: /swagger-ui.html

# 챗지피티 설정
chatgpt:
    key: ${CHATGPT_API_KEY}  # 환경 변수로 대체
    key2: ${CHATGPT_API_KEY2}  # 환경 변수로 대체
```

---

3. NginX

1. nginx.conf

```
events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        server_name ssookssook.kr;
        return 301 https://$host$request_uri;  # HTTP를
HTTPS로 리다이렉트
    }

    server {
        listen 443 ssl;
        server_name ssookssook.kr;

        ssl_certificate /etc/nginx/certs/fullchain.pem;
        ssl_certificate_key /etc/nginx/certs/privkey.pe
m;

        location /api/v1 {
            proxy_pass http://spring_app:8080;
            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;
            proxy_set_header Authorization $http_authori
zation;
        }

        location /swagger-ui {
            proxy_pass http://spring_app:8080;
            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;
            proxy_set_header Authorization $http_authori
zation;
        }

        location /jenkins {
            proxy_pass http://jenkins:8080;
            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;
        }
    }
}
```

4. Docker, DockerCompose 설정

1. EC2 인스턴스에 접속

```
ssh -i K11E103T.pem ubuntu@k11e103.p.ssafy.io
```

---

2. Docker 설치

1. **기존 설치된 패키지 업데이트**

   ```
   sudo apt update
   sudo apt upgrade -y
   ```

2. **필수 패키지 설치**

   ```
   sudo apt install -y apt-transport-https ca-certificates
   curl software-properties-common
   ```

3. **Docker의 GPG 키 추가**

   ```
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg
   | sudo gpg --dearmor -o /usr/share/keyrings/docker-archi
   ve-keyring.gp
   ```

4. **Docker 저장소 추가**

   ```
   echo "deb [arch=$(dpkg --print-architecture) signed-by=/
   usr/share/keyrings/docker-archive-keyring.gpg] https://d
   ownload.docker.com/linux/ubuntu $(lsb_release -cs) stabl
   e" | sudo tee /etc/apt/sources.list.d/docker.list > /de
   v/null
   ```

5. **Docker 설치**

   ```
   sudo apt update
   sudo apt install -y docker-ce docker-ce-cli containerd.i
   o
   ```

6. **설치 확인**

```
docker --version
```

7. **Docker 실행 권한 부여**

```
sudo usermod -aG docker $USER
newgrp docker
```

3. Docker Compose 설치

1. **Docker Compose 다운로드**

```
sudo curl -L "https://github.com/docker/compose/release
s/latest/download/docker-compose-$(uname -s)-$(uname -
m)" -o /usr/local/bin/docker-compos
```

2. **실행 권한 부여**

```
sudo chmod +x /usr/local/bin/docker-compose
```

3. **설치 확인**

```
docker-compose --version
```

4. Docker Compose & Dockerfile 작성

1. **spring**

```
# 1. Base image
FROM openjdk:17-jdk-alpine AS builder

# 2. Set working directory
WORKDIR /app

# 3. Copy the JAR file into the container
COPY ./build/libs/Ssook_BE-0.0.1-SNAPSHOT.jar app.jar
```

```
# 4. Run the application
ENTRYPOINT ["java", "-jar", "/app/app.jar"]
```

```
services:
  spring_app:
    image: kimdonggeon/stillalive_be:latest
    container_name: spring_app
    expose:
      - "8080"
    env_file:
      - .env  # .env 파일을 참조하여 환경 변수를 로드합니다.
    networks:
      - mynetwork

networks:
  mynetwork:
    external: true
```

2. **mysql**

```
services:
  mysql:
    image: mysql:8.0.32
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: stillalive
      MYSQL_DATABASE: ssook
    ports:
      - "3306:3306"
    networks:
      - mynetwork
    volumes:
      - ./mysql_data:/var/lib/mysql

networks:
  mynetwork:
    external: true
```

3. **jenkins**

```
services:
  jenkins:
    build:
      context: .
      dockerfile: Dockerfile
    image: jenkins-custom  # Jenkins LTS 버전 사용
    container_name: jenkins
    expose:
      - "8080"  # Jenkins 웹 UI 포트
      - "50000"  # 에이전트 통신을 위한 포트
    volumes:
      - ./jenkins_home:/var/jenkins_home  # Jenkins 데이터 볼륨
      - /var/run/docker.sock:/var/run/docker.sock  # 호스트의 Docker 소켓을 연결해 도커 빌드/실행 가능하게 설정
    environment:
      JENKINS_OPTS: --prefix=/jenkins # Jenkins의 URL 경로(prefix)를 /jenkins로 설정
    networks:
      - mynetwork

networks:
  mynetwork:
    external: true
```

4. **nginx**

```
services:
  nginx:
    image: nginx:latest
    container_name: nginx_proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt/live/ssookssook.kr/fullchain.pem:
      - /etc/letsencrypt/live/ssookssook.kr/privkey.pem:/e
    networks:
      - mynetwork

networks:
  mynetwork:
    external: true
```

5. Jenkins 파이프 라인

```
pipeline {
    agent any

    environment {
        REPO_NAME = 'kimdonggeon/stillalive_be'  // Docker Hu
        IMAGE_TAG = "latest"
        DOCKER_CREDENTIALS_ID = ''  // Docker Hub 크레덴셜 ID (
        GIT_CREDENTIALS_ID = ''       // GitLab 크레덴셜 ID (Je
    }

    triggers {
        gitlab(triggerOnPush: true, triggerOnMergeRequest: tr
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'be-develop', url: 'https://lab.s
            }
        }
        stage('Build') {
            steps {
                dir('Ssook_BE') { // Ssook 폴더로 이동
                sh 'chmod +x gradlew'
                sh './gradlew clean build'  // Gradle 사용 시
                }
            }
        }

        stage('Docker Build & Push') {
            steps {
                script {
                    docker.withRegistry('https://registry.hub
                        dir('Ssook_BE'){
                            def image = docker.build("${REPO_
                            image.push()
                        }
                    }
                }
            }
        }

        stage('Deploy') {
            steps {
                script {

                    dir('Ssook_BE'){
                        writeFile file: '.env', text: """
                                                        환경변수
                        """

                        // 기존 컨테이너 중지 및 제거
                        sh "docker rm -f spring_app || true"

                        sh "docker-compose -f docker-compose-
                        sh "docker-compose -f docker-compose-
                    }
                }
            }
        }
    }

    post {
        always {
            cleanWs() // 작업 공간 정리
        }
        success {
            echo 'Deployment successful!'
        }
        failure {
            echo 'Deployment failed.'
        }
    }
}
```