포팅메뉴얼

1. 사용 프로그램 버전

💡 프로젝트에서 사용한 프로그램 버전

- Java : `17`
- Spring Boot : `3.2.2`
- MySQL : `8.0.30`
- JPA : `3.2.2`
- Python : `3.9.7`
- Django : `4.2.11`
- Pytorch : `1.11.0`
- Server : `AWS EC2 Ubuntu 20.0.4`
- Vue : `3.4.21`
- Node.js : `20.11.0`
- Gradle : `8.4`
- nginx : `1.18.0`
- Gitlab CI/CD

2. 빌드

1. 환경 변수

Spring : application.properties (\src\main\resources)

Vue : .env (최상단)

application.properties

```
server.servlet.context-path=/api

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
```

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

decorator.datasource.p6spy.enable-logging=true



kiosk.key=SSE에 사용할 키오스크 키
admin.key=SSE에 사용할 관리자 키

# MySQL
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=데이터베이스 url
spring.datasource.username=유저 이름
spring.datasource.password=유저 패스워드

# JWT setting
jwt.salt=해쉬 솔트값
# 엑세스 토큰(millis)
jwt.access-token.expiretime=토큰 유지 시간
# 리프레시 토큰(millis)
jwt.refresh-token.expiretime=토큰 만료 기간
```

.env

```
# API URL settings for PJT
VITE_VUE_SPRING_URL=스프링으로 보낼 url
```

2. 빌드 방법

1. Front-End

   ```
   npm install
   npm run build
   ```

2. Back-Spring
   - Gradle 실행
   - 서버 실행

3. 배포 시 특이사항

```
없음
```

4. 배포 방법

1. FrontEnd

1. nginx.conf (./frontend/Opensight)

```
server {
    listen       80;
    server_name  j10b104.p.ssafy.io;

    location / {
        root   /usr/share/nginx/html;
```

```
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://${BACKEND_HOST}:${BACKEND_PORT};
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

}
```

2. Dockerfile

```
FROM node:lts-alpine AS build

# Set the working directory in the container
WORKDIR /app

# Copy package.json and package-lock.json to leverage Docker cache
COPY package.json .
COPY package-lock.json .

# Install dependencies
RUN npm ci

# Copy the rest of your Vue.js application source code
COPY . /app

# Build your Vue.js application
RUN npm run build

# Set up a new stage from node:lts-alpine
FROM node:lts-alpine

# Install 'serve' to serve the application
RUN npm install -g serve

# Copy the built application from the previous stage
COPY --from=build /app/dist /app

# Your application will run on port 80
EXPOSE 80

# Serve your app using 'serve'
CMD ["serve", "-s", "/app", "-l", "80"]
```

3. docker-entrypoint.sh

```
#!/bin/sh
set -e

# default.conf.template 파일에서 환경 변수를 대체하고 결과를 default.conf에 저장
envsubst '${BACKEND_HOST} ${BACKEND_PORT}' < /etc/nginx/conf.d/default.conf.template > /etc/ngin
```

```
# 다음 명령어를 실행
exec "$@"
```

2. Backend

1. Dockerfile

```
FROM gradle:8.5-jdk17 AS build

WORKDIR /app

# 라이브러리 설치에 필요한 파일만 복사
COPY build.gradle settings.gradle ./

RUN gradle dependencies --no-daemon

# 호스트 머신의 소스코드를 작업 디렉토리로 복사
COPY . /app

# Gradle 빌드를 실행하여 JAR 파일 생성
RUN gradle clean build --no-daemon

# 런타임 이미지로 OpenJDK 11-jre-slim 지정
FROM openjdk:17.0.1-jdk-slim

# 애플리케이션을 실행할 작업 디렉토리를 생성
WORKDIR /app

# 빌드 이미지에서 생성된 JAR 파일을 런타임 이미지로 복사
COPY --from=build /app/build/libs/*.jar /app/B104.jar

EXPOSE 8080
ENTRYPOINT ["java"]
CMD ["-jar", "B104.jar"]
```

3. CI/CD

1. docker-compose.yml

```
# version: '2.25'
services:
  app:
    image: chiseungoh/backend:latest # Replace with your actual Spring Boot Docker image tag
    ports:
      - "8080:8080"
    container_name: backend
    depends_on:
      - mysql
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/bank?serverTimezone=Asia/Seoul&characterEnc
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: SSAFY
      TZ: Asia/Seoul
      # Add other environment variables here ss
    networks:
      - docker-network
```

```
  mysql:
    image: mysql:8.0.30 # Replace with your actual MySQL Docker image tag
    environment:
      MYSQL_ROOT_PASSWORD: SSAFY
      MYSQL_DATABASE: bank
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql
    container_name: mysql
    command: --default-time-zone='+09:00'
    networks:
      - docker-network

  frontend:
    image: chiseungoh/frontend:latest # Replace with your actual frontend Docker image tag
    ports:
      - "8081:80"
    networks:
      - docker-network
    depends_on:
      - app
    container_name: frontend
    environment:
      TZ: Asia/Seoul

networks:
  docker-network:
    external:
      name: docker-network
    driver: bridge
    labels:
      com.docker.compose.network: "true"

volumes:
  mysql-data:
    driver: local
```

2. .gitlab-ci.yml

```
build_backend:
  stage: build
  image: docker:19.03.12
  services:
    - docker:19.03.12-dind
  script:
    - echo "Building backend Docker image..."
    - cd backend
    - docker build -t $CI_REGISTRY_IMAGE/backend:$CI_COMMIT_REF_SLUG .
    # - docker push $CI_REGISTRY_IMAGE/backend:$CI_COMMIT_REF_SLUG
  only:
    - develop

build_frontend:
  stage: build
  image: docker:19.03.12
  services:
```

```
    - docker:19.03.12-dind
  script:
    - echo "Building frontend Docker image..."
    - cd frontend/OpenSight
    - docker build -t $CI_REGISTRY_IMAGE/frontend:$CI_COMMIT_REF_SLUG .
    # - docker push $CI_REGISTRY_IMAGE/frontend:$CI_COMMIT_REF_SLUG
  only:
    - develop

deploy:
  stage: deploy
  image: docker:latest
  # before_script:
  # - apk add --no-cache py-pip python3-dev libffi-dev openssl-dev gcc libc-dev rust cargo make
  # - pip install docker-compose 111

  script:
    - echo "Deploying application..."
    # - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
    # - docker pull $CI_REGISTRY_IMAGE/backend:$CI_COMMIT_REF_SLUG
    # - docker pull $CI_REGISTRY_IMAGE/frontend:$CI_COMMIT_REF_SLUG
    # - cd ..
    # - docker-compose down
    # 123
    - docker-compose down
    - docker-compose up -d
  only:
    - develop
```

3. 외부 서비스 정보

1. 소셜 인증

- 네이버 개발자 센터 어플리케이션 등록
  - redirect URI 등록
  - client_id, client_secret 발급
  - 이용할 계정 등록

- Vue

```
function naverSocialLogin() {
  window.location.href = 'https://nid.naver.com/oauth2.0/authorize?response_type=code&client_id=
}
```

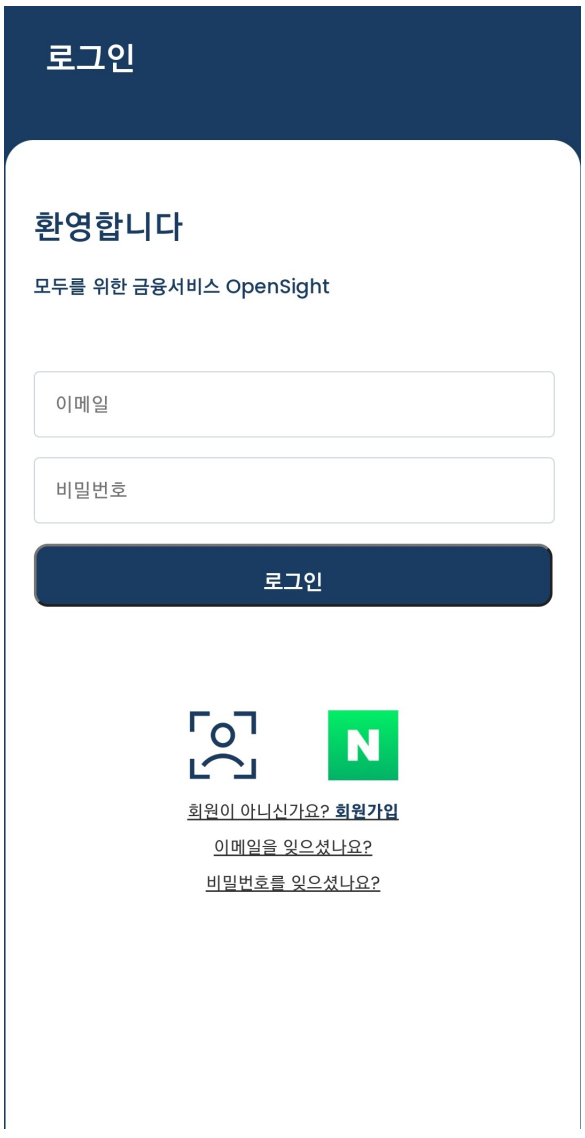
- Spring

```
코드 참조
```

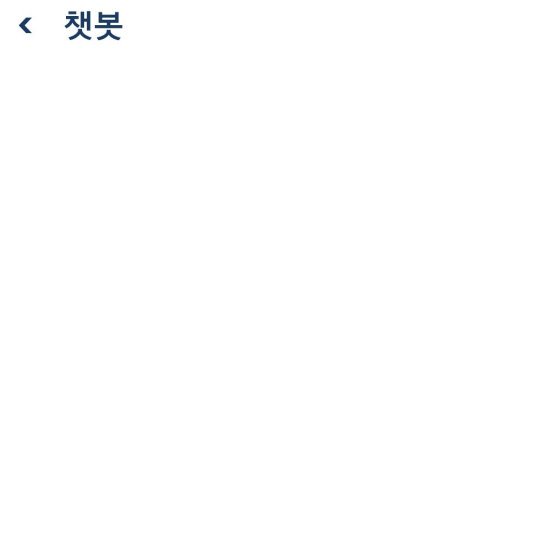
4. 시연 시나리오

1. 전맹 시각장애인 페르소나

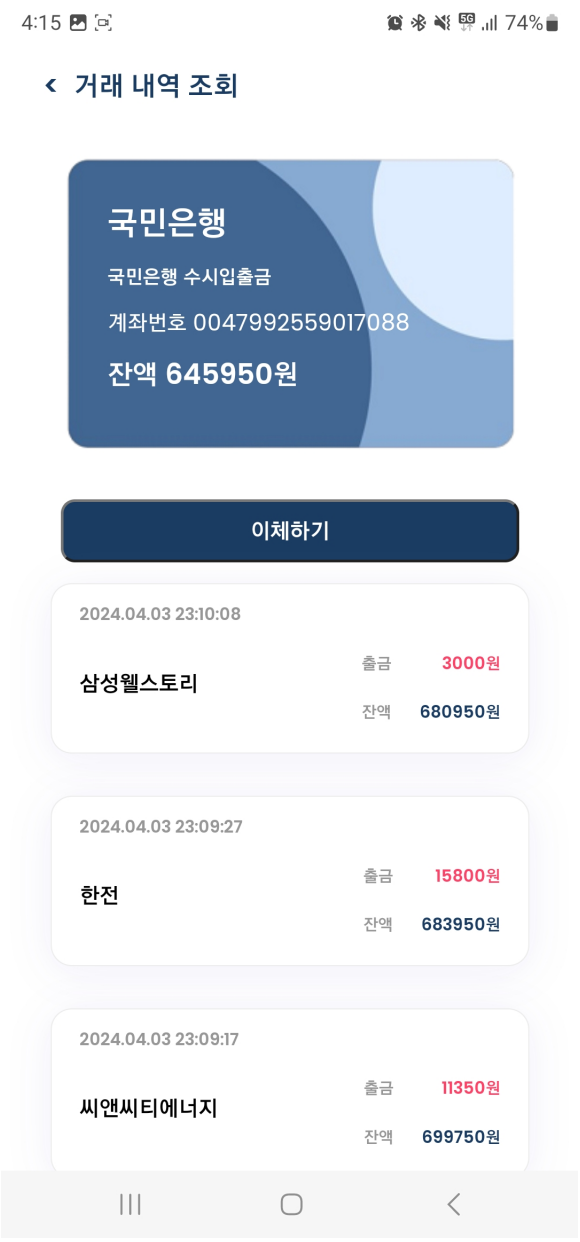
- 얼굴인식 로그인



- 챗봇 버튼 클릭
- 음성인식으로 은행업무 명령



- 거래내역조회 확인



2. 저시력 시각장애인 및 일반인 페르소나

- 챗봇 버튼 클릭
- 문자로 은행업무 명령
- 계좌이체
- 얼굴인식으로 본인확인
- 계좌이체 성공