porting 메뉴얼

상태	In progress
tags	설계

frontend dependencies

```
"dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "@types/file-saver": "^2.0.5",
    "@types/jest": "^27.5.2",
    "@types/navermaps": "^3.6.4",
    "@types/node": "^16.18.24",
    "@types/react": "^18.0.38",
    "@types/react-date-range": "^1.4.4",
    "@types/react-dom": "^18.0.11",
    "@types/react-slick": "^0.23.10",
    "aos": "^2.3.4",
    "axios": "^1.4.0",
    "browser-image-compression": "^2.0.2",
    "chart.js": "^4.3.0",
    "date-fns": "^2.30.0",
    "dotenv": "^16.0.3",
    "file-saver": "^2.0.5",
    "html2canvas": "^1.4.1",
    "moment": "^2.29.4",
    "qrcode.react": "^3.1.0",
    "react": "^18.2.0",
    "react-chartjs-2": "^5.2.0",
    "react-cookie": "^4.1.1",
    "react-date-range": "^1.4.0",
    "react-datepicker": "^4.11.0",
    "react-dom": "^18.2.0",
    "react-kakao-maps-sdk": "^1.1.8",
    "react-page-slides": "^0.1.3",
    "react-router-dom": "^6.10.0",
    "react-scripts": "5.0.1",
    "react-slick": "^0.29.0",
    "recoil": "facebookexperimental/Recoil.git#nightly",
    "recoil-persist": "^4.2.0",
    "sass": "^1.62.0",
    "scss": "^0.2.4",
    "slick-carousel": "^1.8.1",
    "styled-components": "^5.3.10",
    "tailwindcss-animation-delay": "^1.0.7",
    "typescript": "^4.9.5",
    "web-vitals": "^2.1.4"
  },
```

backend gradle

```
plugins {
  id 'java'
  id 'org.springframework.boot' version '2.7.11'
  id 'io.spring.dependency-management' version '1.0.15.RELEASE'
}

group = 'com.flowery'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

configurations {
  compileOnly {
    extendsFrom annotationProcessor
  }
}
```

```
repositories {
  mavenCentral()
}

dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
  implementation 'org.springframework.boot:spring-boot-starter-web'
  compileOnly 'org.projectlombok:lombok'
  runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'
  annotationProcessor 'org.projectlombok:lombok'
  testImplementation 'org.springframework.boot:spring-boot-starter-test'

  implementation 'com.google.zxing:core:3.4.1'
  implementation 'com.google.zxing:javase:3.4.1'


  implementation platform('com.amazonaws:aws-java-sdk-bom:1.11.1000')
  implementation 'com.amazonaws:aws-java-sdk-s3:1.12.67'
  implementation 'org.springframework.boot:spring-boot-starter-security'
  implementation 'commons-io:commons-io:2.11.0'    /* Apache commons-io */
  implementation group: 'commons-fileupload', name: 'commons-fileupload', version: '1.4' /* Apache Commons FileUpload */

  /* 문자 발송 */
  implementation 'net.nurigo:sdk:4.2.7'
  /* redis */
  implementation 'org.springframework.boot:spring-boot-starter-data-redis'
  implementation group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.5.13'
  // [JWT]
  implementation 'io.jsonwebtoken:jjwt:0.9.1'
}

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

backend application.properties

```
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb: //mariaDB 주소
spring.datasource.username= // mariadb id
spring.datasource.password= // mariadb password

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

cloud.aws.s3.bucket = //s3 버킷값 입력

cloud.aws.credentials.access-key = //s3용 aws 키 값 입력
cloud.aws.credentials.secret-key = //s3용 aws 시크릿 값 입력
cloud.aws.region.static = ap-northeast-2

server.servlet.context-path=/api

# Redis server ??
spring.redis.host = //사용하실 서버 dns 혹은 ip
spring.redis.port = 6379

naver-cloud-sms.accessKey = // 네이버 sms용 엑세스 키 값

naver-cloud-sms.secretKey = // 네이버 sms용 엑세스 시크릿 값

naver-cloud-sms.serviceId = // 네이버 sms용 서비스 아이디

naver-cloud-sms.senderPhone = // 네이버 송신자 전화번호

spring.jwt.key = // jwt용 시크릿 값
spring.jwt.live.atk = // jwt access token 생존 시간
spring.jwt.live.rtk = // jwt refresh token 생존 시간
```

docker images

- redis image
- frontend image
- backend image
- flask image
- mariadb image
- (ci cd 용) jenkins image

backend docker file

```
FROM adoptopenjdk/openjdk11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

frontend docker file

```
FROM node:18.13.0 as build-stage
WORKDIR /var/jenkins_home/workspace/flowery/frontend
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /var/jenkins_home/workspace/flowery/frontend/build /usr/share/nginx/html
COPY --from=build-stage /var/jenkins_home/workspace/flowery/frontend/deploy_conf/nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 3000
CMD ["nginx", "-g","daemon off;"]
```

flask docker file

```
# base image
FROM python:3.11.3

# set the working directory in the container
WORKDIR /app

# copy the Flask app code to the container
COPY . /app

RUN python -m venv venv
ENV VIRTUAL_ENV=/app/venv
ENV PATH="$VIRTUAL_ENV/bin:$PATH"

# install necessary packages
RUN apt-get update
RUN apt-get install -y libgl1-mesa-glx

RUN pip3 install --upgrade pip
RUN pip3 install --root-user-action=ignore requests
RUN pip3 install flask
RUN pip3 install flask_cors
RUN pip3 install flask_sqlalchemy
RUN pip3 install pymysql
```

```
RUN pip3 install torch torchvision torchaudio
RUN pip3 install PILLOW
RUN pip3 install opencv_python
RUN pip3 install pandas
RUN pip3 install requests
RUN pip3 install psutil
RUN pip3 install pyyaml
RUN pip3 install tqdm
RUN pip3 install matplotlib
RUN pip3 install seaborn
RUN pip3 install openai
RUN pip3 install boto3


# set environment variables
ENV FLASK_APP=flowery
ENV FLASK_RUN_PORT=5000

# expose the port that Flask will run on
EXPOSE 5000

# run the Flask app
CMD [ "flask", "run", "--host=0.0.0.0", "--port=5000" ]
```

frontend 폴더 내부의 nginx.conf 파일 (docker 내부 nginx로 작동)

```
server {
    listen       3000;
    listen  [::]:3000;
    // sever_name은 자신의 dns 혹은 host 주소로
    server_name  flowery.duckdns.org;
    client_max_body_size 100M;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
       client_max_body_size 100M;
    }

    location /api {
        proxy_pass http://flowery.duckdns.org:8080;
        proxy_redirect     off;
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        client_max_body_size 100M;
    }

    location /flask {
        proxy_pass http://flowery.duckdns.org:5000;
        #proxy_redirect     off;
        #proxy_set_header   Host $host;
        #proxy_set_header   X-Real-IP $remote_addr;
        #proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        client_max_body_size 200M;
    }

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

HTTPS 설정용 외부 nginx 파일

```
server {
  listen 80; #80포트로 받을 때
  server_name flowery.duckdns.org; #도메인주소, 없을경우 localhost
  return 301 https://flowery.duckdns.org$request_uri;
```

```
}
server {
  listen 443 ssl http2;
  server_name flowery.duckdns.org;

  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/flowery.duckdns.org/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/flowery.duckdns.org/privkey.pem;

  location / { # location 이후 특정 url을 처리하는 방법을 정의(여기서는 / -> 즉, 모든 request)
    proxy_pass http://flowery.duckdns.org:3000; # Request에 대해 어디로 리다이렉트하는지 작성. 8443 -> 자신의 springboot app이사용하는 포트
    proxy_set_header Host $http_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;
  }

}
server {
    if ($host = flowery.duckdns.org) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  listen 80;
  server_name flowery.duckdns.org;
    return 404; # managed by Certbot


}
```

EC2 서버에 터미널에서 Redis docker 설치

```
docker pull redis
docker run -it -d --rm -p 6379:6379 --name redis redis

docker pull mariadb
docker run -p 3306:3306 --name mariadb -e MARIADB_ROOT_PASSWORD={비밀번호} -d mariadb
```

EC2 서버에 올리고 나서 터미널에서 Docker build & run

```
// (springboot) backend image build 및 실행
docker build -t backimg ./backend
docker run -it -d --rm -p 8080:8080 --name backimg backimg

// flask image build 및 실행
docker build --no-cache -t flaskimg ./ai
docker run -it -d --rm -p 5000:5000 --name flaskimg flaskimg

// fronted image build 및 실행
docker build --no-cache -t frontimg ./frontend
docker run -it -d --rm -p 3000:3000 --name frontimg frontimg
```

정상 결과

```
ubuntu@ip-172-26-15-41:~$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED             STATUS
8422a4321a51   frontimg              "/docker-entrypoint.…"   23 minutes ago      Up 23 minutes
a2df93cda220   backimg               "java -jar /app.jar"     26 minutes ago      Up 26 minutes
0098a61311dd   redis                 "docker-entrypoint.s…"   About an hour ago   Up About an hour
17ab7d845ac2   flaskimg              "flask run --host=0.…"   7 hours ago         Up 7 hours
38898c18c45c   mariadb               "docker-entrypoint.s…"   2 weeks ago         Up 2 weeks
1fc9cbe94568   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   2 weeks ago         Up 2 weeks
```