포팅 메뉴얼

Re:Code 포팅 매뉴얼

I. 개요

1. 프로젝트 개요

Re:Code는 **BOJ 문제 제출 기록(성공/실패)**을 연동하여

GPT API 기반으로 자동 생성된 **알고리즘 오답노트**를 제공하는 SNS 서비스입니다.

- SNS 형식의 피드 및 탐색 제공
- 문제 유형별 오답노트 기록, 통계 및 분석 제공

2. 주요 기능

1. 사용자의 BOJ 제출 내역 불러오기 (Chrome Extension 활용)
2. 실패-성공 코드 비교 기반 오답노트 자동화
3. 코드 중심의 피드 콘텐츠 제공

3. 사용 도구

- **이슈 관리**: JIRA
- **형상 관리**: GitLab
- **커뮤니케이션**: Notion, Mattermost
- **디자인**: Figma
- **CI/CD**: Docker, AWS EC2

4. 개발 환경

- **Frontend**: React 19, Vite, TypeScript, Tailwind CSS, Zustand
- **Backend**: Java 21, Spring Boot 3.5.4, Spring Security, JWT
- **DB/Infra**: MySQL 9.4.0, Ubuntu 20.04, Docker

5. 외부 서비스

- **BOJ 제출 기록 연동 (Chrome Extension 활용)**
- **GPT API** (코드 비교 분석: 알고리즘, 시간복잡도, 공간복잡도, 추가/삭제된 로직 등)

6. Gitignore 처리 항목

- 환경변수 파일 (.env, application.yml)
- DB 비밀번호, JWT secret key
- GPT API key
- 기타 보안 관련 설정

II. 빌드 & 실행

1. 환경 변수 설정

Frontend (.env)

```
VITE_API_BASE_URL=http://localhost:8080
```

Backend (back.env)

```
GMS_API_KEY=S13P11A507-a6fb9568-81ef-49df-9c5d-246de3786ecc
JWT_SECRET=K3v!pA9zQ#yXr7WcLmN6eTb$J1uVo@MfRd2CgHs8ZnEtBx5YqPwL0dUiG^Sk4OhY
```

2. 프론트엔드 빌드

```
npm install
npm run dev   # 개발 실행
npm run build # 프로덕션 빌드
```

3. 백엔드 빌드

```
./gradlew build     # jar 생성
./gradlew bootRun   # 실행
```

4. docker-compose.yml

```
version: "3.8"

services:
  backend:
    # 이미지 빌드용 Dockerfile 경로 추가
    build: ./back
    shm_size: "2g"
    image: recode-back
    container_name: recode-back
    env_file:
      - ./back.env
    ports:
      - "8080:8080"
    networks:
      - app-network
    environment:
      - TZ=Asia/Seoul

  frontend:
    # 이미지 빌드용 Dockerfile 경로 추가
    build: ./front
    image: recode-front
    container_name: recode-front
    ports:
      - "80:80"
    networks:
      - app-network
    environment:
      - VITE_REST_API_URL=http://backend:8080
      - TZ=Asia/Seoul
    depends_on:
      - backend

networks:
```

```
  app-network:
    driver: bridge
```

4. Dockerfile

- FE

```
# 배포용 dockerfile
# [1단계] 빌드
# 1. build 환경 설정
# node.js 22 lts img
FROM node:20-alpine AS builder

# 2. 작업 디렉토리 설정
WORKDIR /app

# 3. 의존성 설치 (package.json, package-lock.json 복사)
COPY package*.json ./
RUN npm install

# 4. source code copy
COPY . .

# 5. vite app build
RUN npm run build

# [2단계] 배포
# 6. nginx로 빌드된 정적 파일 서빙
FROM nginx:alpine

# (필요시) nginx 설정 파일 복사
# 프론트 라우팅 (react router) 위해 try_files 설정 포함을 권장
COPY nginx.conf /etc/nginx/conf.d/default.conf

# 7. 빌드된 파일을 nginx 기본 웹서버 경로로 복사
COPY --from=builder /app/dist /usr/share/nginx/html

# 8. nginx 기본 포트 노출
EXPOSE 80

# 9. Nginx 포어그라운드에서 실행
CMD [ "nginx", "-g", "daemon off;" ]
```

BE

```
FROM eclipse-temurin:21-jdk

# 필수 런타임 라이브러리(noble: t64들 포함)
RUN apt-get update && apt-get install -y \
  curl unzip ca-certificates \
  libnss3 libxss1 libasound2t64 libatk-bridge2.0-0t64 libgtk-3-0t64 \
  libdrm2 libgbm1 libxshmfence1 libu2f-udev libvulkan1 libcups2t64 \
  fonts-liberation \
  libx11-6 libxcomposite1 libxrandr2 libxdamage1 libxrender1 \
  libxi6 libxtst6 libxkbcommon0 libxext6 libx11-xcb1 \
  libpango-1.0-0 libpangocairo-1.0-0 libatspi2.0-0t64 \
  libglib2.0-0 \
 && rm -rf /var/lib/apt/lists/*
```

```
# Chrome for Testing + Chromedriver (버전 고정)
RUN set -eux; \
  CFT_VER="129.0.6668.70"; \
  curl -fsSLo /tmp/chrome.zip  "https://storage.googleapis.com/chrome-for-testing-public/${CFT_VER}/linux64/chrome-linux64.zip"; \
  curl -fsSLo /tmp/driver.zip  "https://storage.googleapis.com/chrome-for-testing-public/${CFT_VER}/linux64/chromedriver-linux64.zip"; \
  unzip -qo /tmp/chrome.zip -d /opt; \
  unzip -qo /tmp/driver.zip -d /opt; \
  ln -sf /opt/chrome-linux64/chrome /usr/bin/google-chrome; \
  ln -sf /opt/chromedriver-linux64/chromedriver /usr/bin/chromedriver; \
  rm -f /tmp/*.zip

# 앱
COPY build/libs/recode-0.0.1-SNAPSHOT.jar /app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

5. 실행 방법

- **로컬 실행**:
  - FE → `npm run dev`
  - BE → `./gradlew bootRun`
- **Docker 실행**:

```
docker-compose up -d --build
```

III. 배포

1. 서버 환경

- AWS EC2 (Ubuntu 20.04)
- Docker
- Nginx

2. 배포 절차

1. EC2에 GitLab에서 프로젝트 clone
2. 프론트엔드 → docker-compose.yml 및 front/Dockerfile 기반으로 Docker 이미지 생성 후 배포
3. 백엔드 → docker-compose.yml 및 back/Dockerfile 기반으로 Docker 이미지 생성 후 배포
4. Docker 컨테이너 실행

IV. DB

1. DB 초기 설정

- MySQL 9.4.0
- DB 이름: `recode`
- 사용자/비밀번호는 application.properties 에 명시

2. DB 덤프 파일

Dump20250817.zip

V. 외부 서비스 설정

1. BOJ 제출 기록 연동

- Chrome Extension 설치 → 제출 기록 수집
- 수집된 제출 기록을 기반으로 성공/실패 코드 매칭

2. GPT API 연동

- OpenAI GPT API Key 발급 필요
- 백엔드 `application.yml`에 key 등록
- 코드 차이(알고리즘/시간복잡도/로직 등) 분석 후 오답노트 생성

VI. 시연 시나리오

1. **로그인/회원가입** → JWT 인증
2. **Chrome Extension으로 BOJ 계정 연동**
3. **제출 기록 기반 오답노트 자동 생성 (GPT 분석)**
4. **피드 / 탐색 페이지** → 노트 공유 및 댓글·좋아요·팔로우
5. **마이페이지** → 나의 오답노트/댓글/좋아요 관리 및 태그별 빈도 통계 확인

VII. 참고 자료

- 요구사항 명세서
  - https://docs.google.com/spreadsheets/d/190IxhaF_4bI-IhAyCfAw_IEfoDjf7-zHGuIjwXz_bvo/edit?usp=sharing
- 와이어프레임
  - https://www.figma.com/design/FPq9g3IAN8eEcYsFGU4xry/%ED%94%84%EB%A1%9C%ED%86%A0%ED%83%EB%A0%88%ED%8D%BC%EB%9F%B0%EC%8A%A4?node-id=0-1&p=f&t=i9zSTIaSJiQTsj1w-0
- ERD
  - https://www.erdcloud.com/d/FdTChLcMTNSEw4N2Y
- API 명세서 (Swagger)
  - API 명세