

2024학년도 1학기

소프트웨어공학개론 (SWE3002_42)

차수영 교수님

성균관대 1등 맛집 추천 앱

먹구루

맛집



최종 발표 시작하기

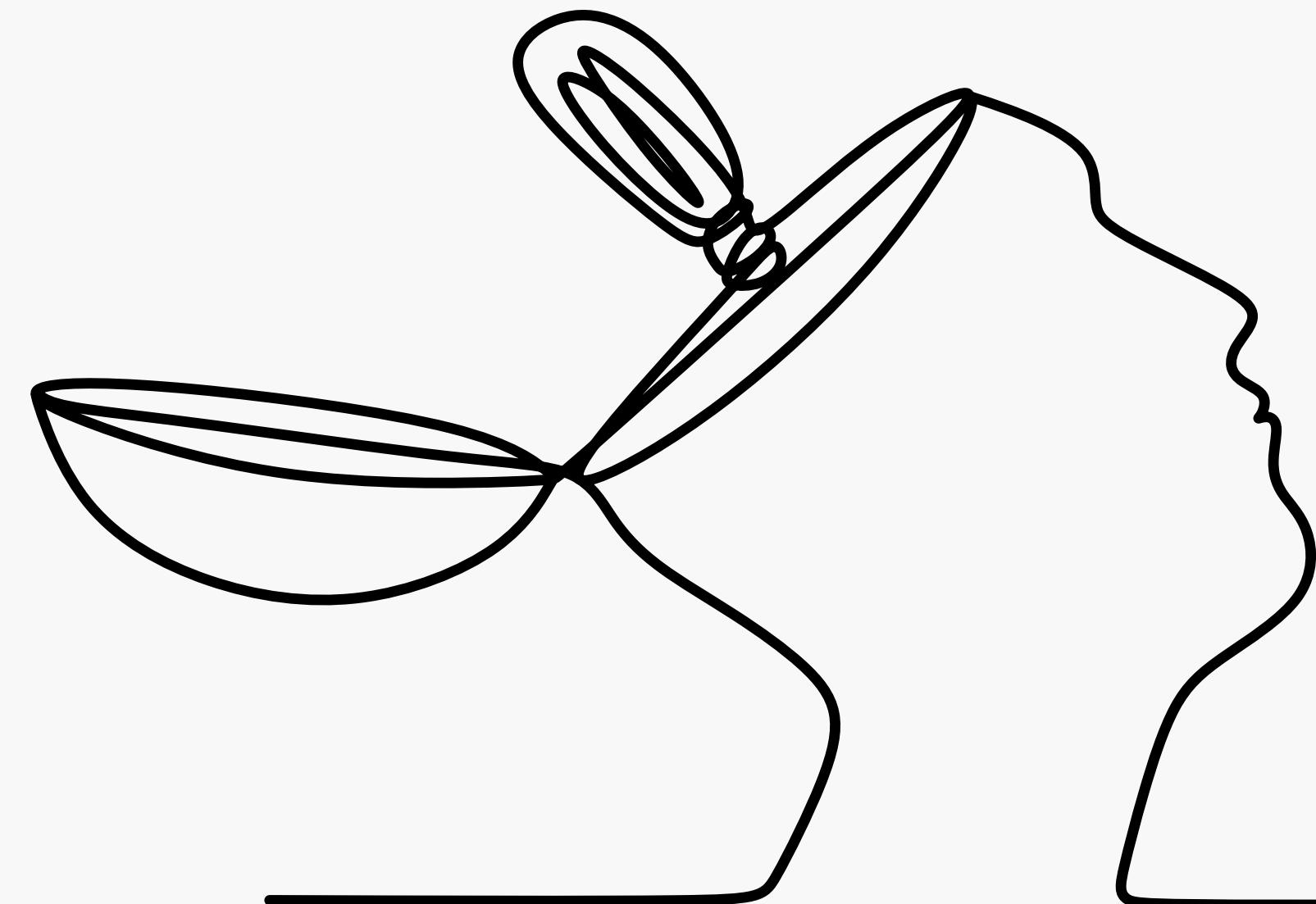
TEAM 1

2020312049 송새론
2019312751 김광원
2022310931 김민서
2017313665 김태훈
2020314841 박정호
2017315471 신경덕
2018310541 엄승주

목차

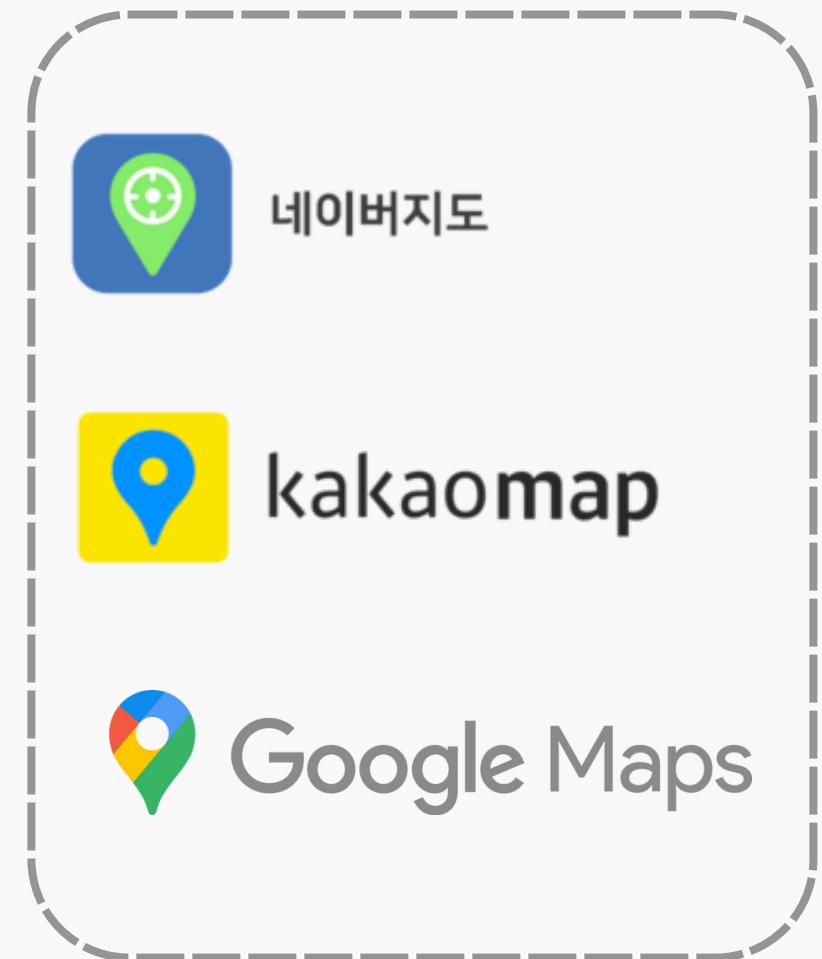
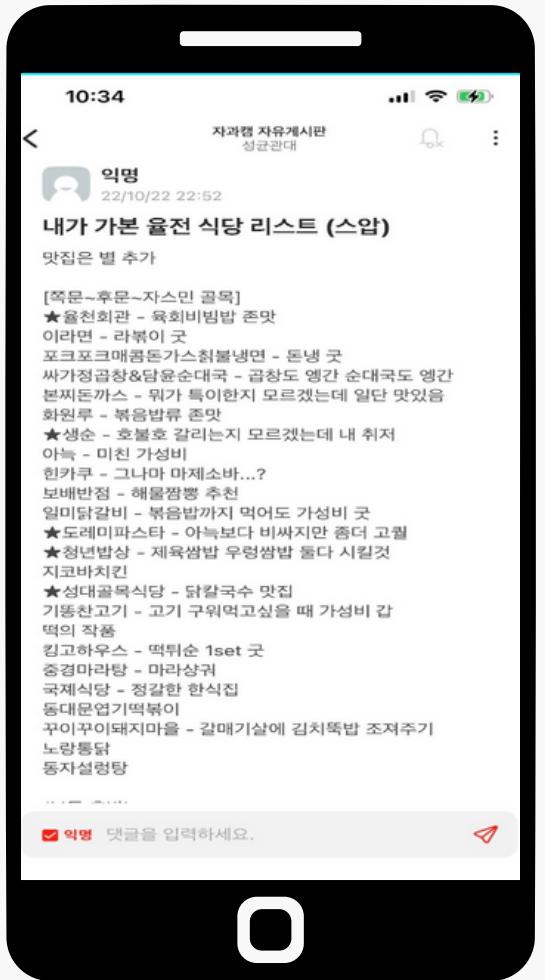


1. 프로젝트 동기
2. 프로젝트 목표
3. 프로젝트 진행 과정
4. 구현 결과
5. 시연 영상
6. 테스트
7. 결론



1. 프로젝트 정리

1. 프로젝트 동기



커뮤니티



학생들의
솔직한 리뷰

지도 앱



편리한 지도 탐색,
다양한 정보/리뷰

“기존 서비스의 장점을 모아
성균관대 학생을 위한
맛집 추천 서비스를
만들어보자”

→ AI 추천 서비스

→ 믿을만한 리뷰 시스템

→ 편리하고 다양한 검색 시스템

먹구
드꾸



2. 프로젝트 목표

2. 프로젝트 목표

1

기능

- 유저 인증 토큰 발급
- 세부 검색
- AI 기반 추천
- 리뷰 생성/관리
- 지도 기반 서비스

2

시스템

- 컨테이너화
- 무중단 배포
- MSA 기반 서비스 개발
- 개인정보 암호화

3

에자일 개발 프로세스

- Github: 코드/배포 관리
- JIRA 티켓/이슈 관리
- Notion: 개발 문서 관리
- Discord: 개발 관련 소통/알림



3. 프로젝트

진행 과정



3. 프로젝트 진행 과정

멤버

Front-End Developer

- 엄승주
- 박정호
- 김민서
- 신경덕

Back-End Developer

- 신경덕
- 김광원
- 김태훈

UI & UX Designer

- 김민서
- 송새론

Manager

- (전체) 송새론
- (개발) 신경덕

3. 프로젝트 진행 과정

AGILE APPROACH



Notion

기술 스택

- 백엔드
 - 언어: Kotlin
 - 프레임워크: Spring Boot, Spring MVC
 - 라이브러리: JPA, QueryDSL
 - 테스트: Kotest, TestContainers
 - 빌드툴: Gradle
 - Github Actions
 - Docker, Docker Compose
 - AWS EC2
- Frontend
- Database: MySQL

개발 문서 관리

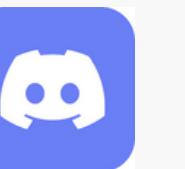


Jira

Team1 - KAN 보드

할 일	진행 중	완료
프로필 스크린 UI	UI 수정 KAN-24	KAN-110
	UI/UX 디자인 KAN-5	KAN-5
	프론트 코트 통합 및 정리 KAN-102	KAN-102
	리스트 정렬기능 KAN-98	KAN-98
	UI 정리 및 수정 KAN-105	KAN-105
	리뷰 작성, 삭제시, Restaurant에 avgRating, review Count변경 KAN-99	KAN-99
	리뷰 페이징 기능 추가 KAN-94	KAN-94
	아플로고 디자인 KAN-7	KAN-7
	검색페이지 KAN-95	KAN-95

티켓 & 이슈 관리



Discord

fe-pr-빠꾸기

- [meokgu-skku/fe] Pull request opened: #21 [KAN-105] UI 개선 및 이미지 문제 해결
 - UI 수정할 부분 수정
 - 폰트 추가 및 변경
 - 이미지 파일 추가 및 빌드시 사라지지 않도록 조치
- [fe:KAN-105] 1 new commit
6c94ecf Fix - sjsjmine129
- [GitHub] 오늘 오전 12:47
sinkyoungdeok [meokgu-skku/fe] Pull request review submitted: #21 [KAN-105] UI 개선 및 이미지 문제 해결
디자인부분이라 리뷰는 힘들겠네요 😊 가시정
- [GitHub] 오늘 오전 10:15
sjsjmine129 [meokgu-skku/fe] Pull request closed: #21 [KAN-105] UI 개선 및 이미지 문제 해결

개발 관련 소통 & 알림

4. 구현 결과

- 아기택배
- 어플리케이션 기능



4. 구현 결과

기술 스택

BACK-END (MAIN)

- 언어: KOTLIN
- 프레임워크: SPRING BOOT, SPRING MVC, SPRING DATA JPA
- 라이브러리: JPA, QUERYDSL, KT-SEARCH
- 테스트: KOTEST, TESTCONTAINERS, MOCKK
- 빌드툴: GRADLE
- CI/CD 툴: GITHUB ACTIONS
- 컨테이너: DOCKER, DOCKER-COMPOSE
- 데이터베이스: MYSQL, REDIS, ELASTICSEARCH
- 모니터링: KIBANA
- 클라우드 서비스: AWS EC2, AWS SES
- 컨벤션: KTLINT

BACK-END (BATCH)

- 언어: PYTHON
- 프레임워크: SELENIUM
- 데이터베이스: MYSQL, REDIS, ELASTICSEARCH
- 컨테이너: DOCKER, DOCKER-COMPOSE
- CI/CD 툴: GITHUB ACTIONS
- API: CHAT GPT4 API, NCP GEOCODE API

BACK-END (AUTOCOMPLETE)

- 언어: GOLANG
- CI/CD 툴: GITHUB ACTIONS
- 컨테이너: DOCKER, DOCKER-COMPOSE
- 데이터베이스: REDIS

BACK-END (IMAGE-UPLOADER)

- 언어: PYTHON
- 클라우드 서비스: LAMBDA, API GATEWAY, S3
- 모니터링: AWS CLOUD WATCH

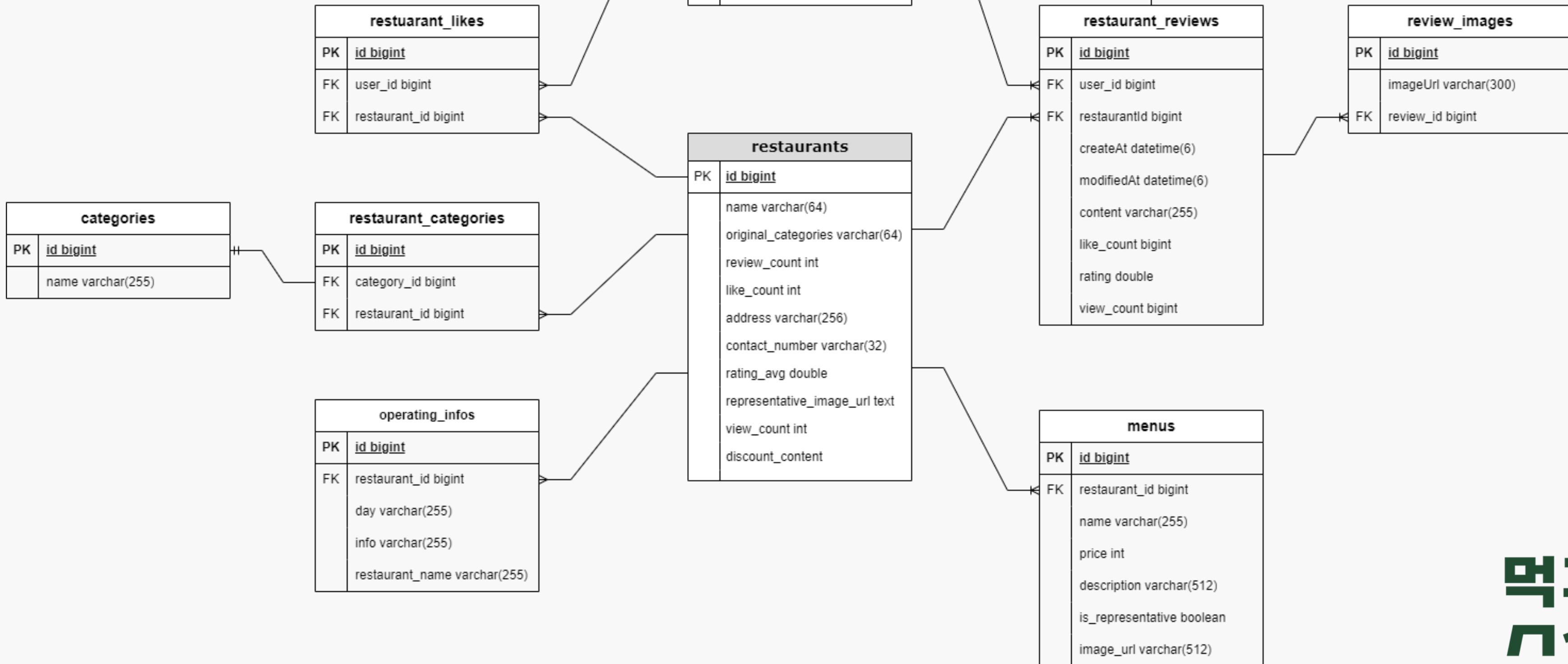
APP

- 언어: JAVASCRIPT
- 지원 환경: ANDROID, IOS
- 프레임워크: REACT NATIVE
- 빌드 프레임워크: REACT-NATIVE CLI

4. 구현 결과

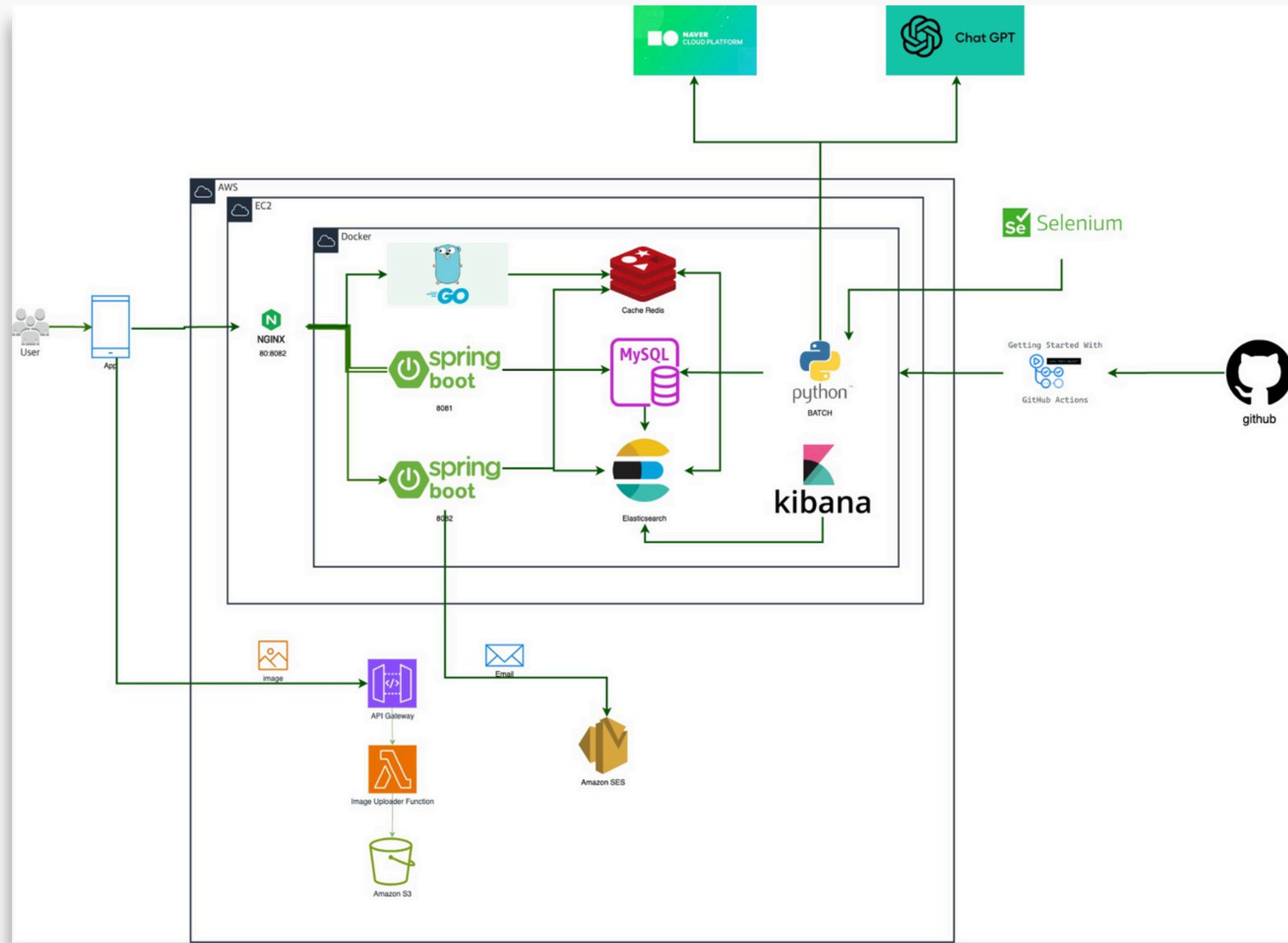
ER 다이어그램

MySQL



4. 구현 결과

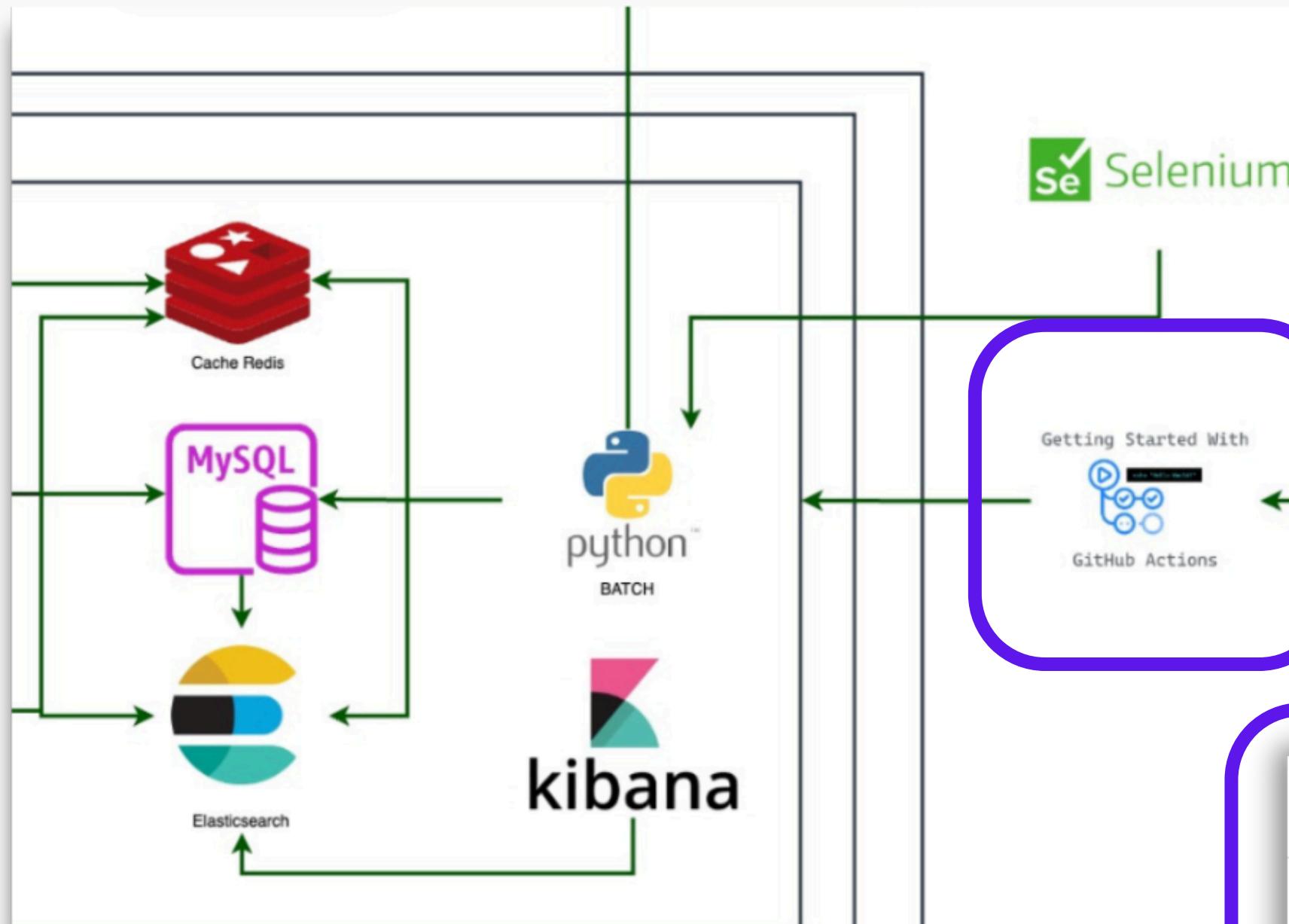
아키텍쳐



AWS EC2
AWS SES
AWS Lambda
AWS S3
Nginx
Spring boot
MySQL
Redis
ElasticSearch
Kibana
Github Action
Selenium
Chat GPT4
NCP geocode API

4. 구현 결과

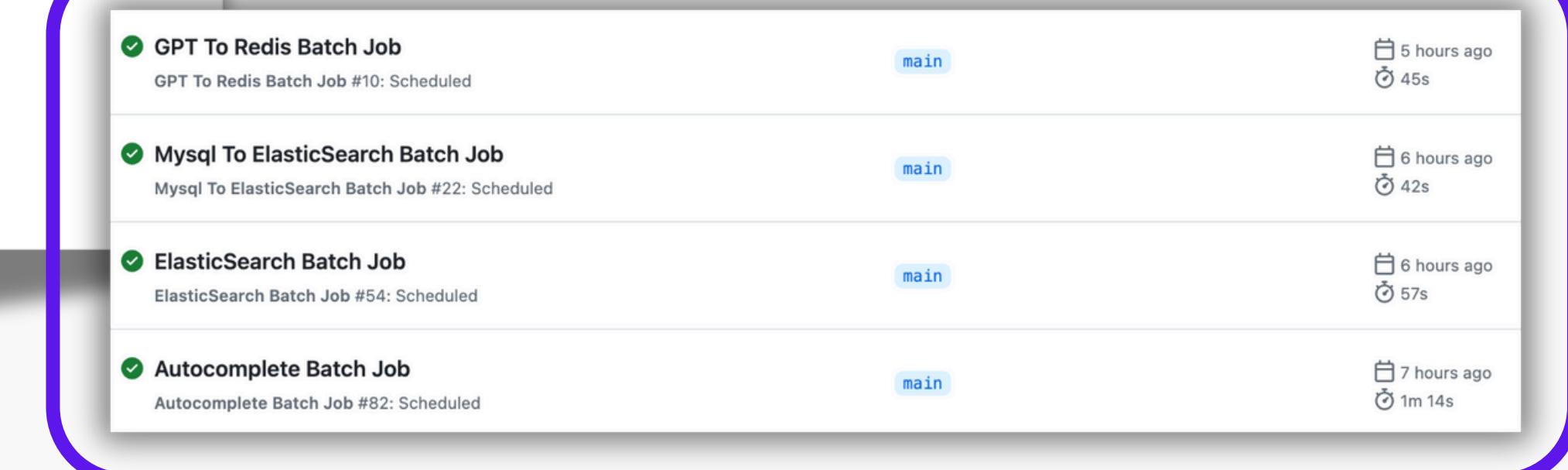
아키텍처 (CRAWLING & BATCH)



음식점 정보 Crawling (with Selenium)

NCP geocode API

restaurant.csv, menu.csv



4. 구현 결과

아키텍처 (CRAWLING & BATCH)

GPT To Redis Batch Job	main	5 hours ago
GPT To Redis Batch Job #10: Scheduled		45s
Mysql To ElasticSearch Batch Job	main	6 hours ago
Mysql To ElasticSearch Batch Job #22: Scheduled		42s
ElasticSearch Batch Job	main	6 hours ago
ElasticSearch Batch Job #54: Scheduled		57s
Autocomplete Batch Job	main	7 hours ago
Autocomplete Batch Job #82: Scheduled		1m 14s

[Autocomplete Batch Job]

음식점 정보들의 자동완성 기능을 제공하기 위해 Redis에 해당 정보 적재

[ElasticSearch Batch Job]

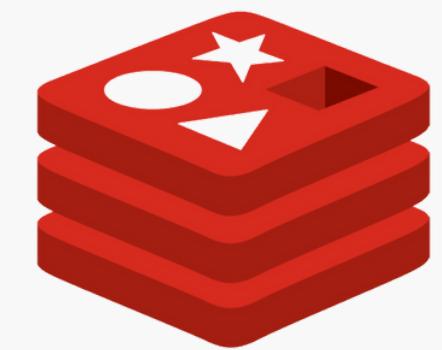
다양한 검색 조건과 검색 Quality 향상을 위해 Elastic Search에 해당 정보 적재

[Mysql To ElasticSearch Batch Job]

수정된 음식점 리뷰 개수, 좋아요 개수 등을 Elastic Search에 반영

[GPT To Redis Batch Job]

사용자별 음식점 추천 시스템을 도입하기 위해 GPT 및 프롬프트 엔지니어링 적용 후 Redis에 적재

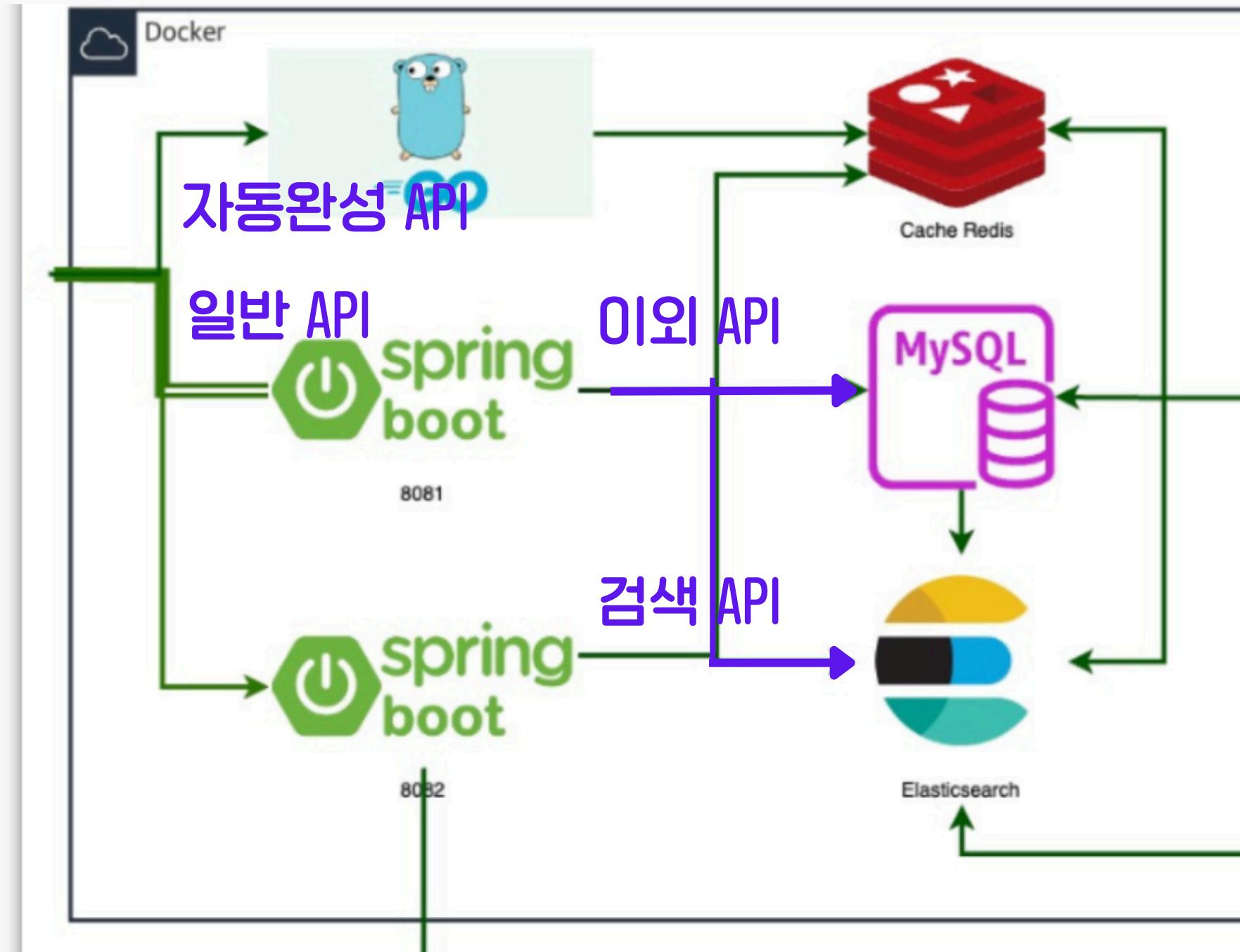


- 추천 시스템
- 자동완성

- 검색 기능 향상
- 주기적인 Update

4. 구현 결과

아키텍처 (REVERSE PROXY)



Spring Boot →

GO Server →

Kibana →

nginx.conf

```
server {
    listen 80;
    server_name example.com;

    include /etc/nginx/default.d/*.conf;

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

    location / {
        proxy_pass $service_url;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
    }
}

server {
    listen 3000;
    server_name example.com;

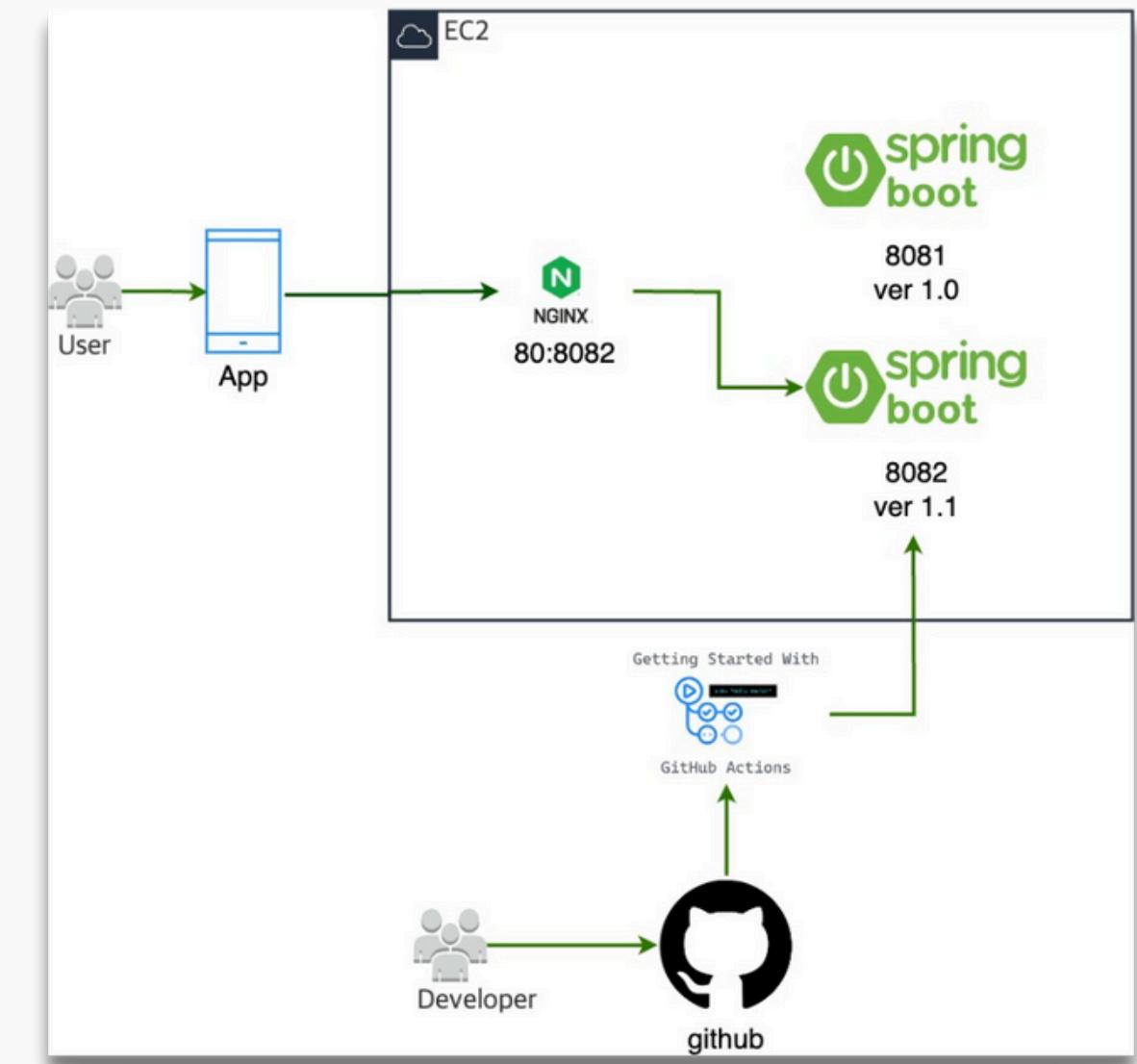
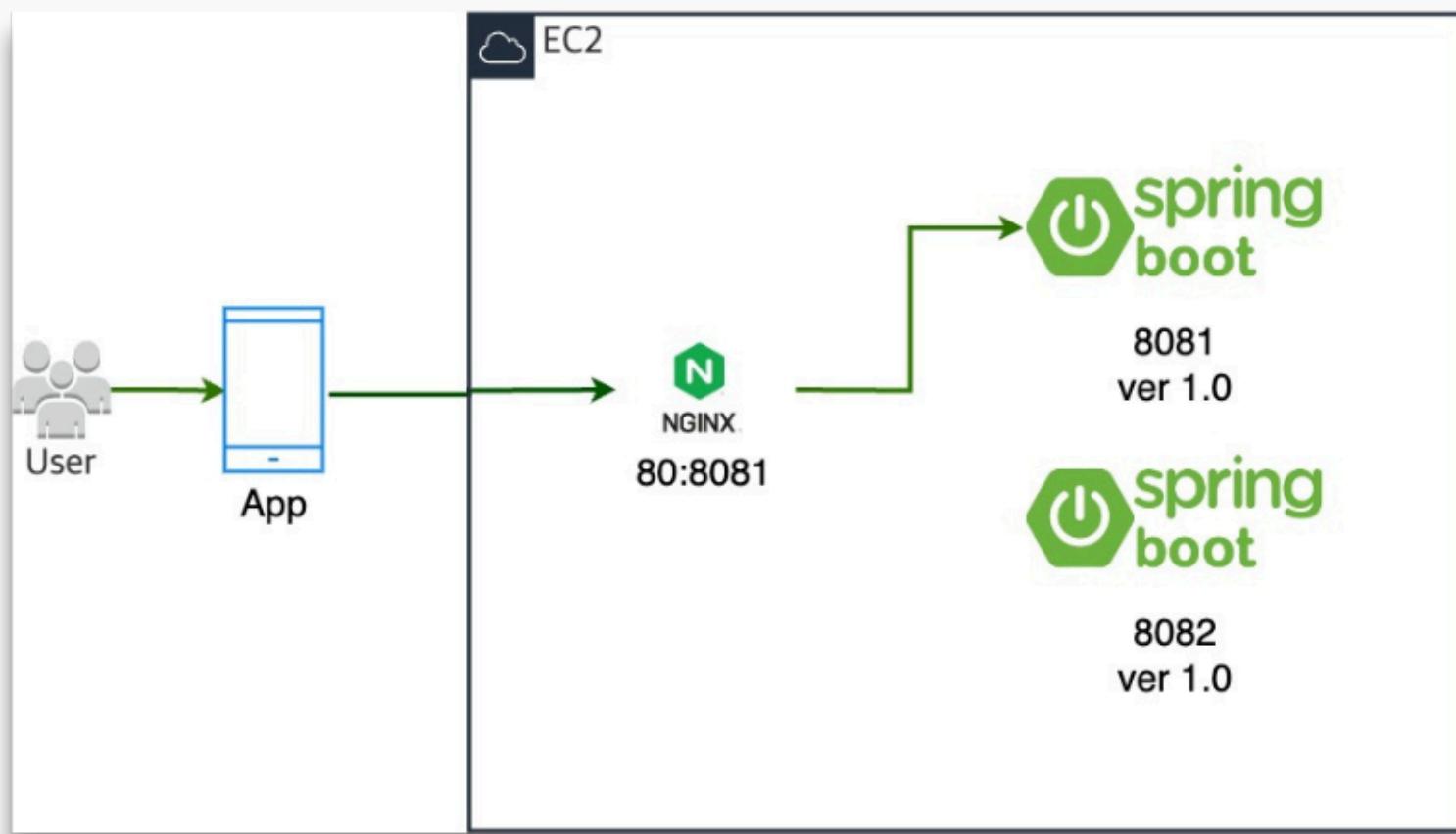
    location / {
        proxy_pass http://localhost:8090;
    }
}

server {
    listen 5600;
    server_name example.com;

    location / {
        proxy_pass http://localhost:5601;
    }
}
```

4. 구현 결과

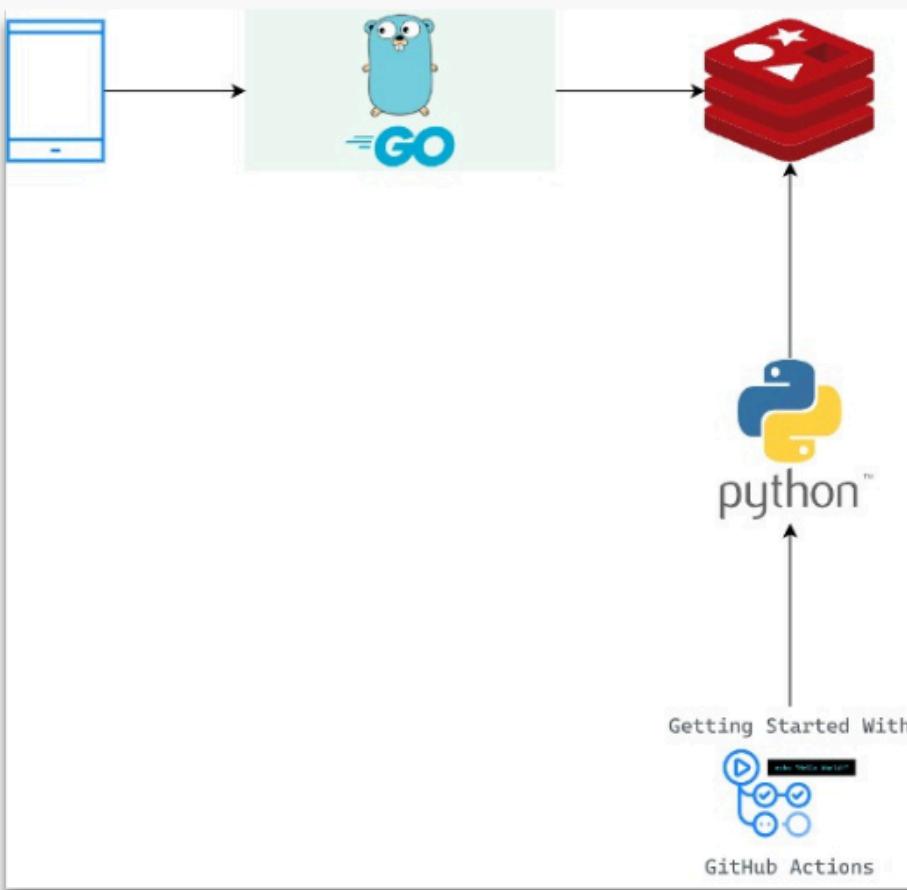
아키텍처 (무중단 배포)



1. 개발자 코드 수정 및 배포 trigger 요청
2. 운영 중인 서버 확인(8081)
3. 미운영 중인 서버 버전 업데이트(8082)
4. nginx reload

4. 구현 결과

아키텍처 (자동완성)



10만건 데이터에서
20~30ms 응답 속도

```

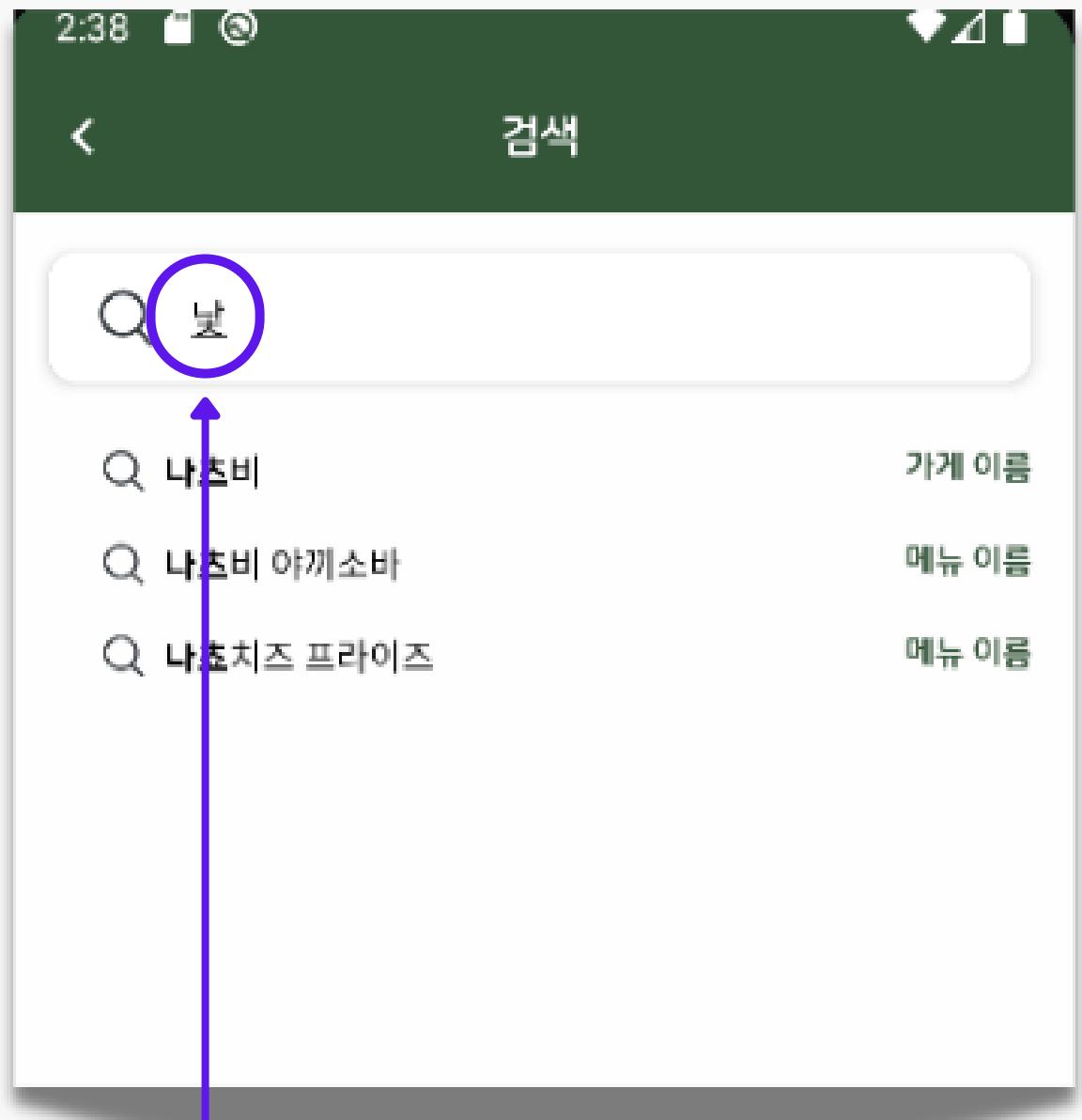
LOG 검색어 : 모
LOG 자동완성 api start
LOG 자동완성 api end (duration: 31ms)
LOG 검색어 : 모
LOG 자동완성 api start
LOG 자동완성 api end (duration: 34ms)
LOG 검색어 : 목
LOG 자동완성 api start
LOG 자동완성 api end (duration: 28ms)
LOG 검색어 : 목
LOG 자동완성 api start
LOG 자동완성 api end (duration: 29ms)
LOG 검색어 : 목구
LOG 자동완성 api start
LOG 자동완성 api end (duration: 18ms)
LOG 검색어 : 목궁
LOG 자동완성 api start
LOG 자동완성 api end (duration: 16ms)
LOG 검색어 : 목구머
LOG 자동완성 api start
LOG 자동완성 api end (duration: 18ms)
LOG 검색어 : 목구멍
LOG 자동완성 api start
LOG 자동완성 api end (duration: 20ms)

```

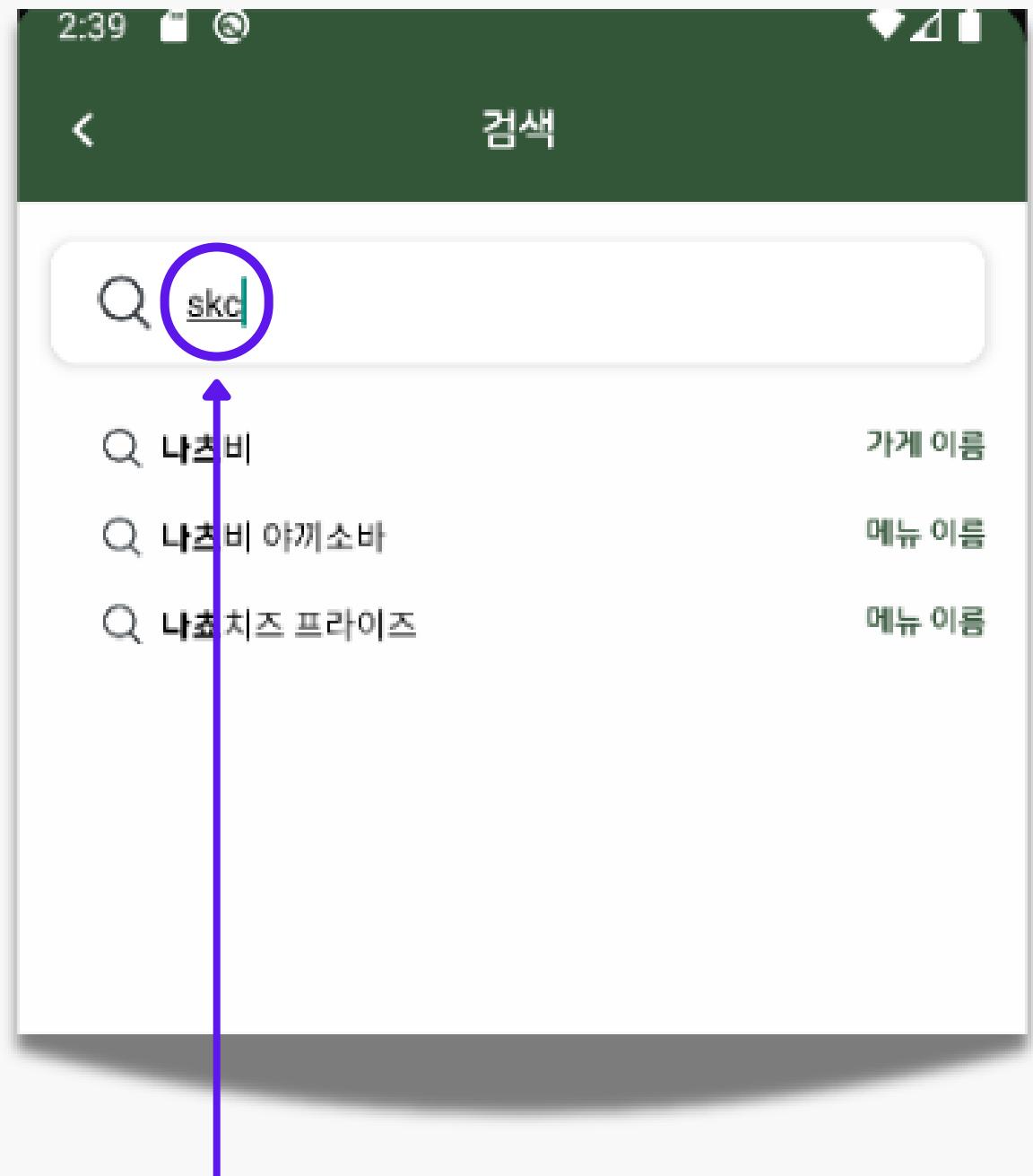
1. 음식점 이름, 메뉴 이름, 카테고리 데이터를 CSV 파일에서 읽어들이기
2. 이 데이터를 초성, 중성, 종성으로 분해
3. 분해된 데이터를 기반으로 영어, 한국어 타자 데이터를 생성
4. 이를 통해 새로운 버전의 Redis 데이터를 생성
5. 생성된 데이터를 TTL(Time-To-Live) 2일로 설정하여 Redis에 저장

4. 구현 결과

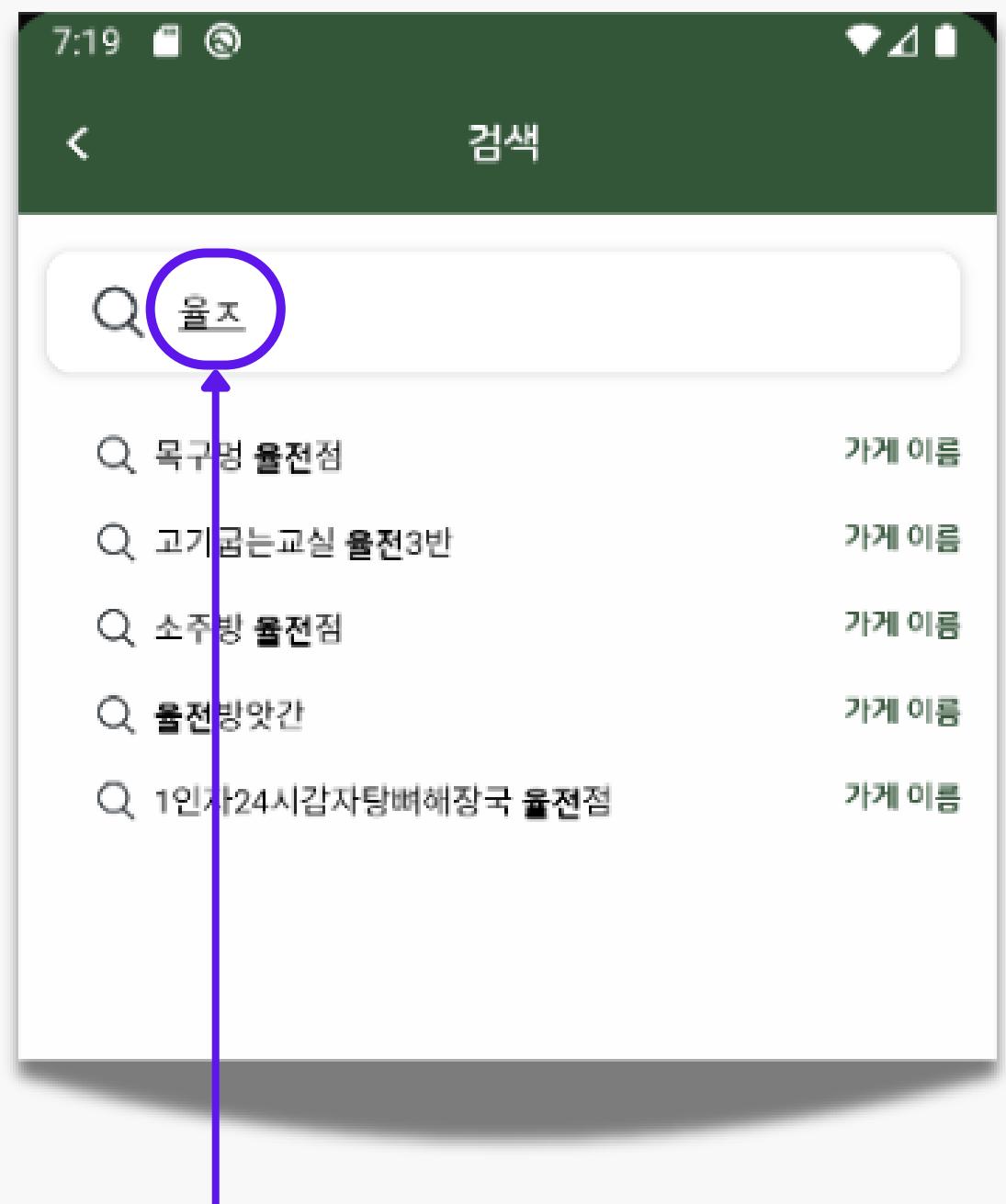
아키텍처 (자동완성)



한국어타자 검색



영어타자 검색

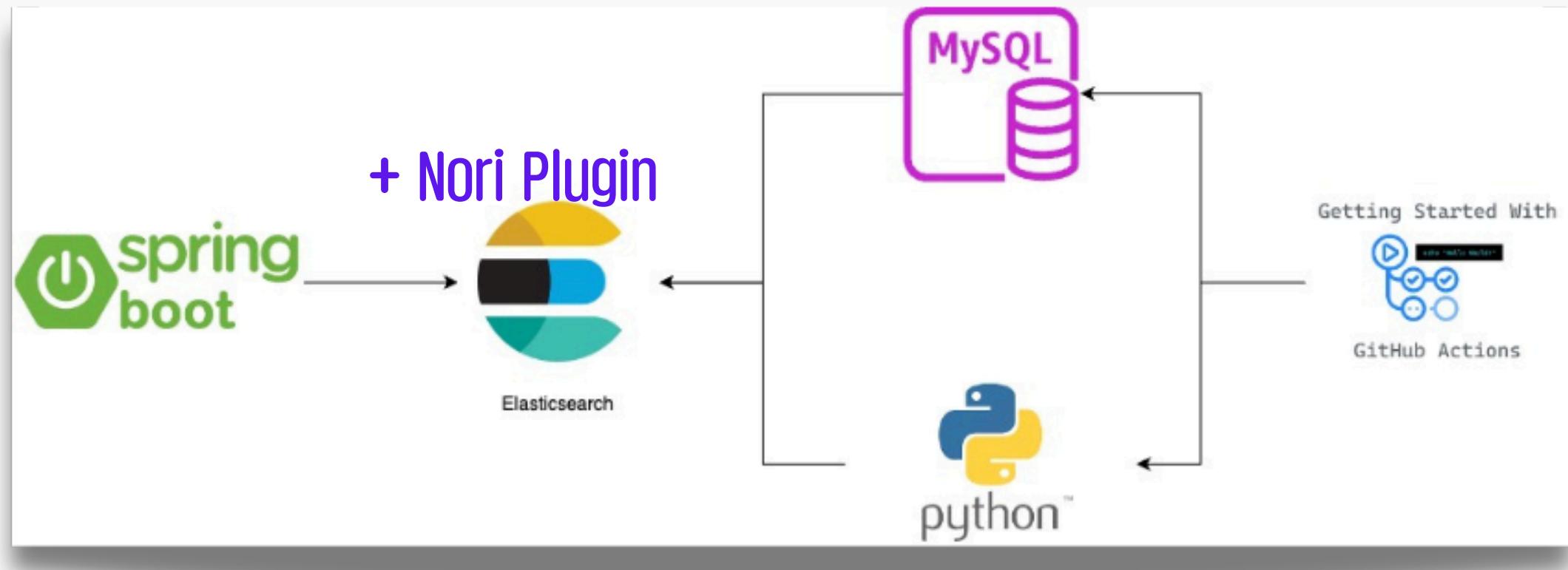


띄어쓰기 매칭

먹구
드꾸

4. 구현 결과

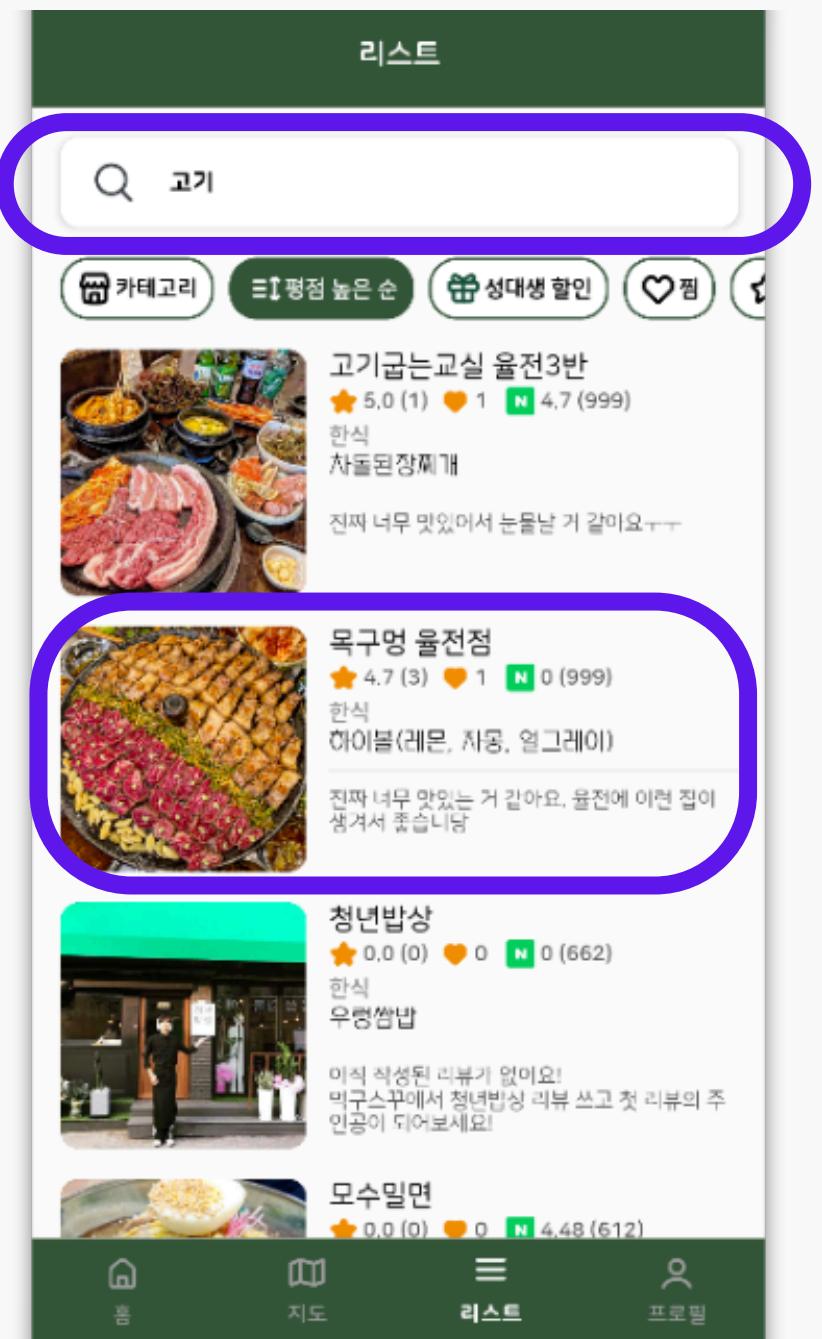
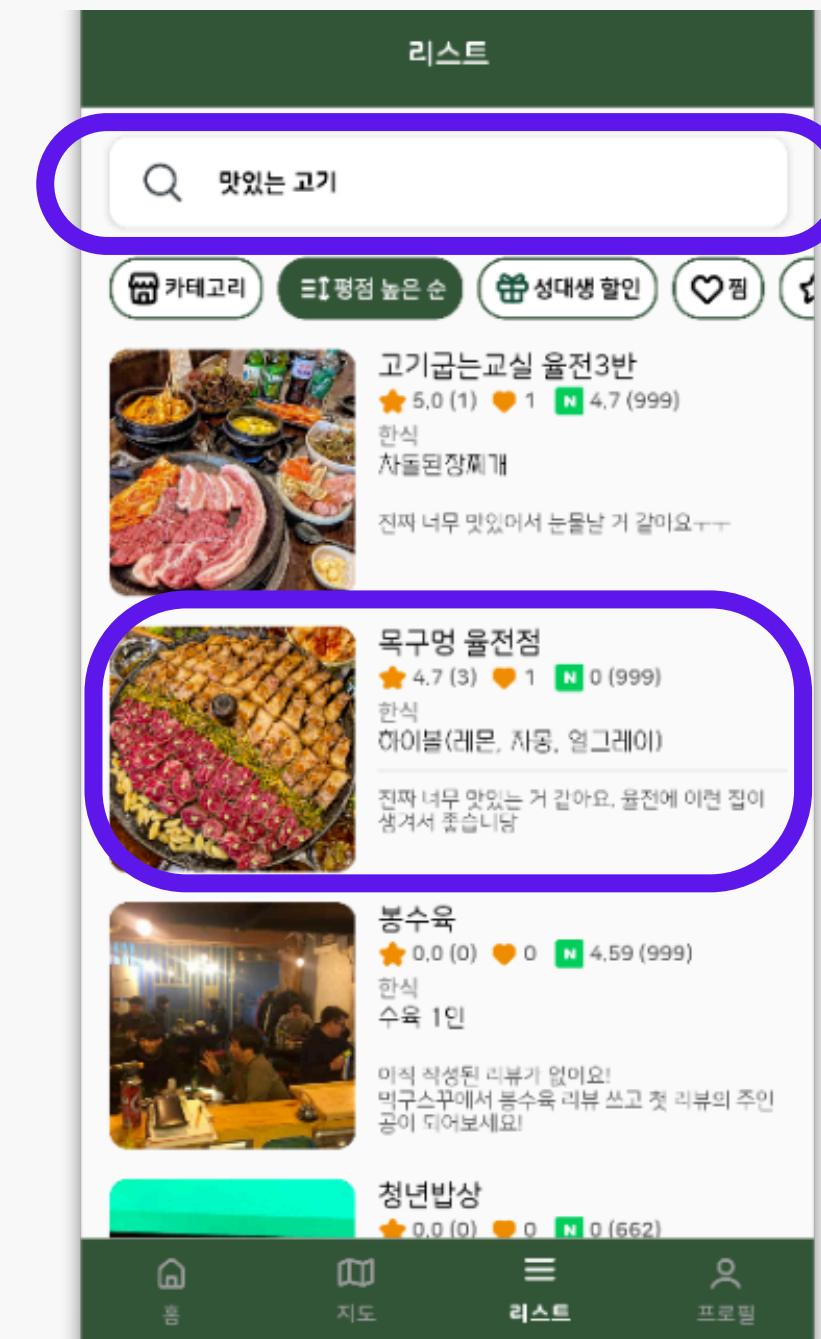
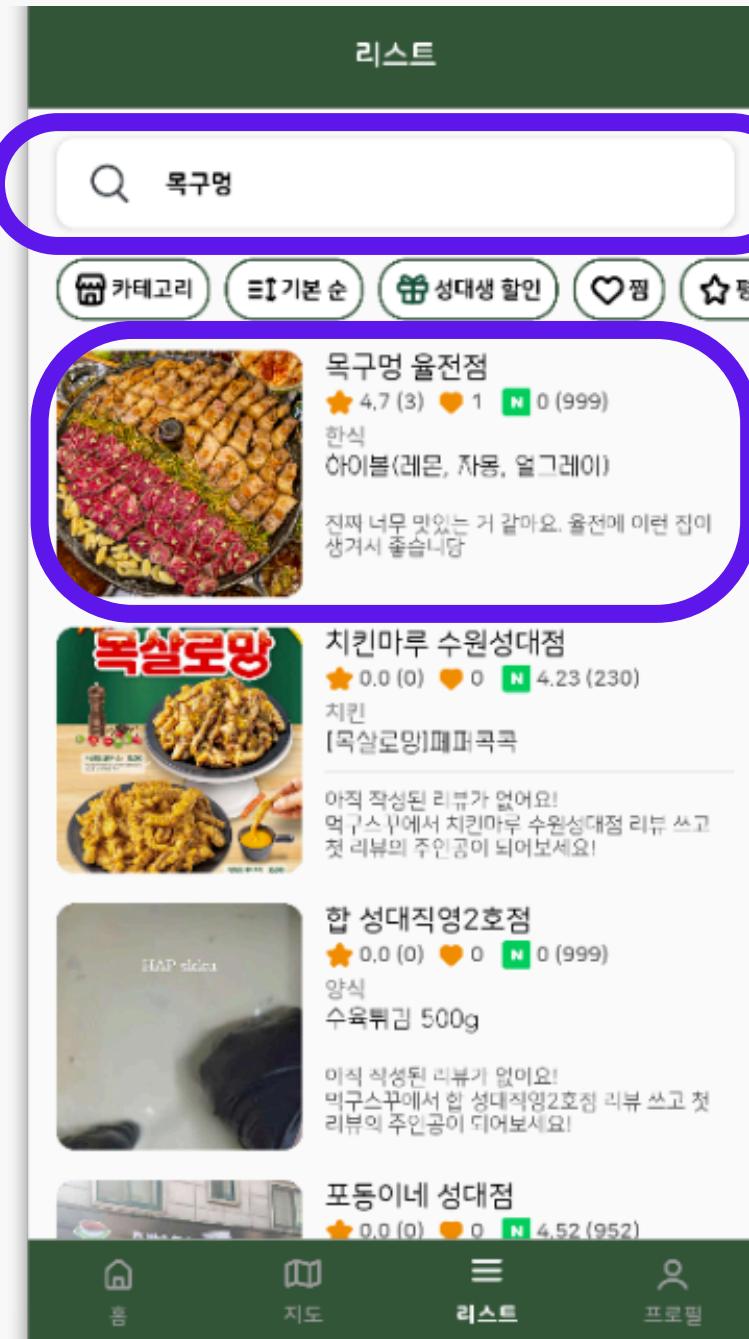
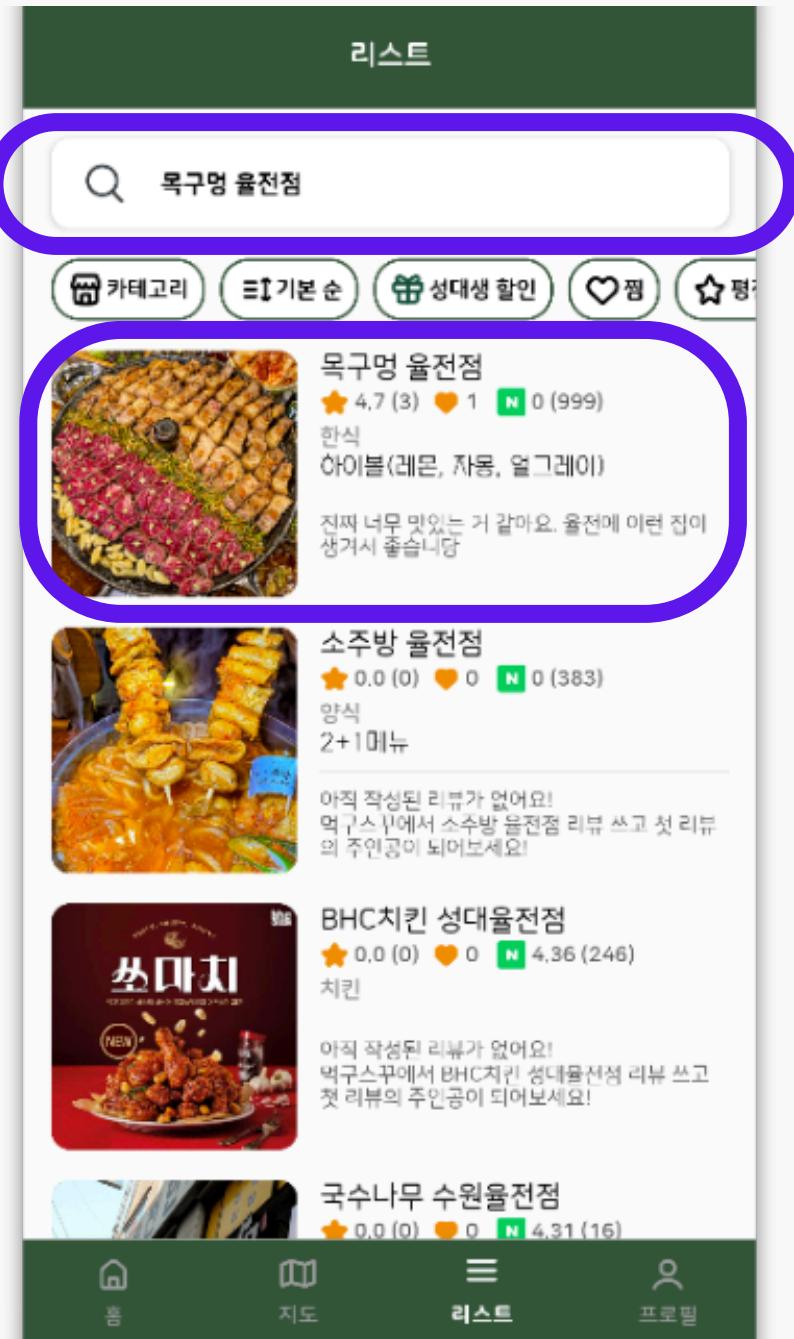
아키텍처 (검색)



1. 음식점, 메뉴, 카테고리 데이터를 csv에서 읽어서 ES에 저장
2. 사용자가 남긴 좋아요 갯수, 댓글 갯수 등을 mysql에서 불러와 es에 저장
3. 1, 2번을 통해 저장한 데이터는 새로운 index에 저장
4. 3번을 통해 만들어진 index로 alias를 변경
5. 기존에 사용중이던 index는 제거

4. 구현 결과

아키텍처 (검색)



먹구
드꾸

4. 구현 결과

아키텍처 (검색)

```

should(
  match( field: "name", request.query) { this: MatchQueryConfig
    boost = 0.1
  },
  match( field: "category", request.query) { this: MatchQueryConfig
    boost = 0.03
  },
  match( field: "original_category", request.query) { this: MatchQueryConfig
    boost = 0.03
  },
  nested { this: NestedQuery
    path = "menus"
    query = bool { this: BoolQuery
      should(
        match( field: "menus.menu_name", request.query) { this: MatchQueryConfig
          boost = 0.01
        },
        match( field: "menus.description", request.query) { this: MatchQueryConfig
          boost = 0.01
        }
      )
    }
  }
)
minimumShouldMatch( value: 1)

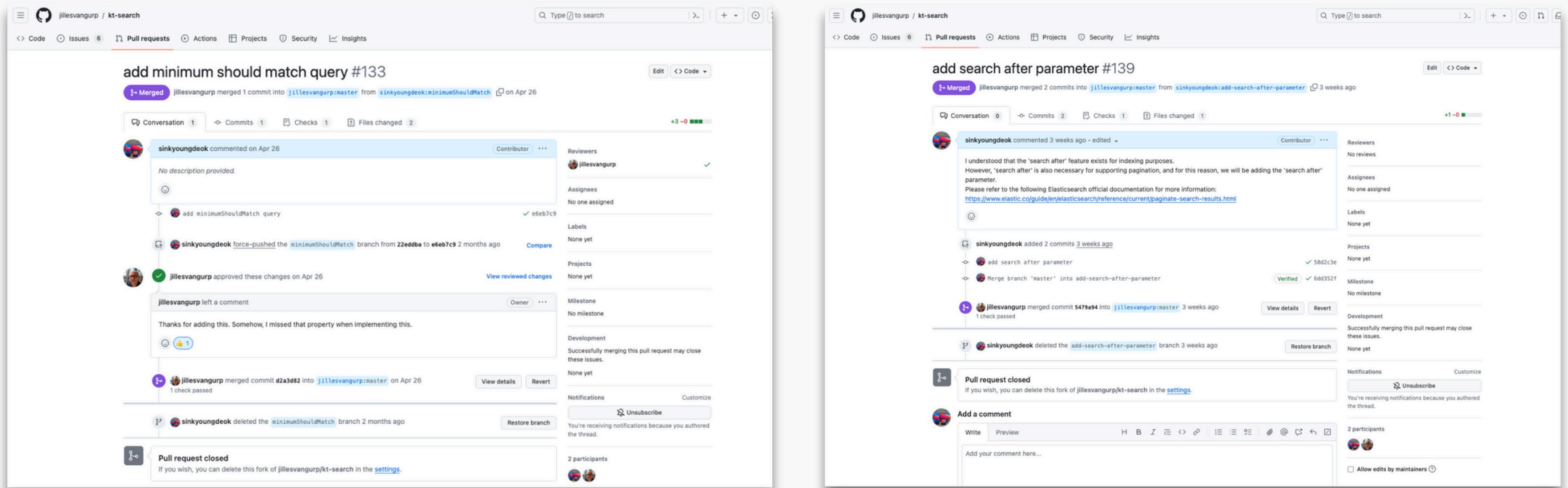
```

1. 이름 매칭 점수: 0.1
2. 카테고리 매칭 점수: 0.03
3. 메뉴이름 매칭 점수: 0.01
4. 메뉴설명 매칭 점수: 0.01

위 점수를 합산한 결과를 기반으로
매칭된 음식점을 상위로 노출

4. 구현 결과

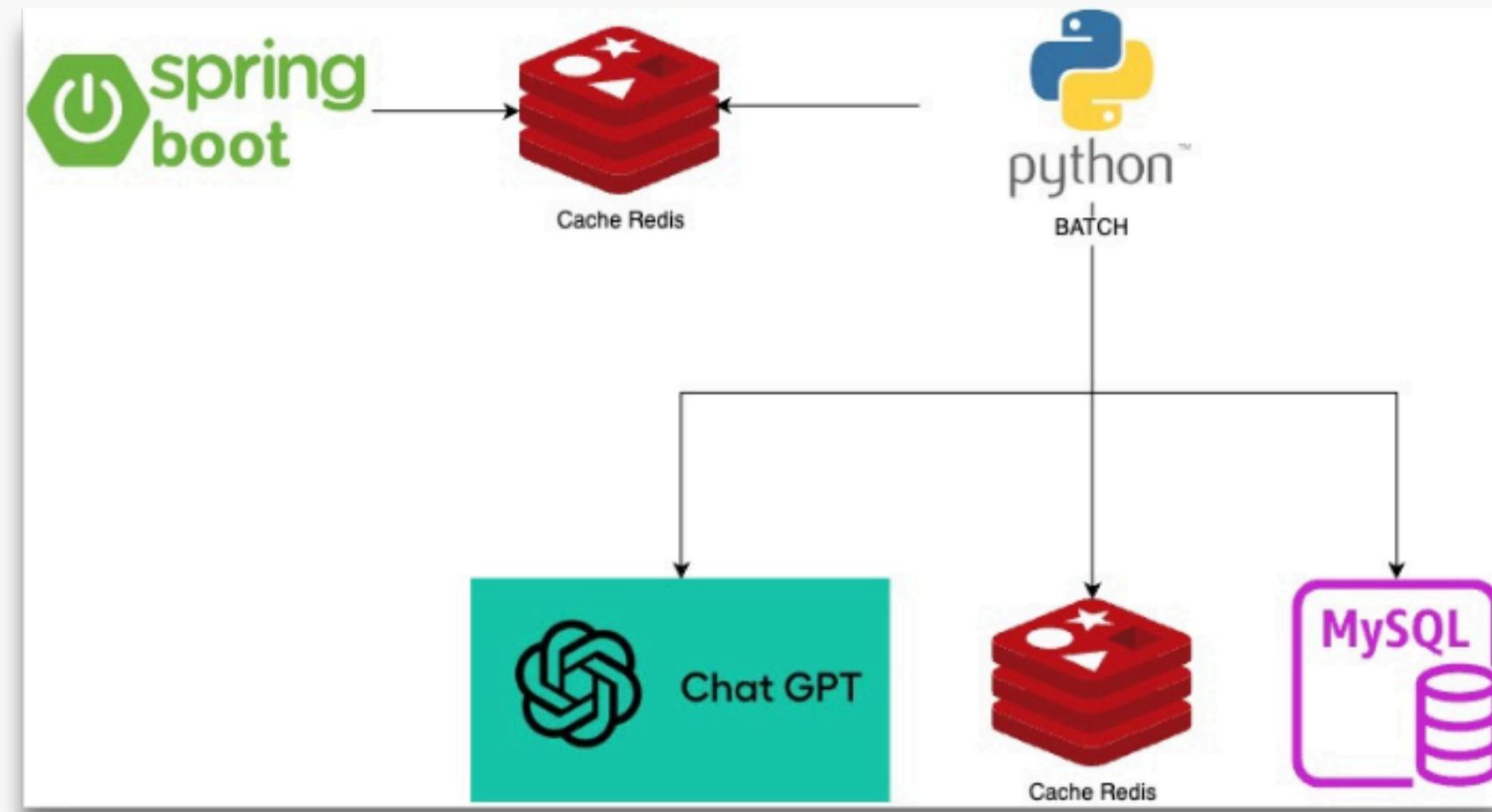
아키텍처 (검색)



kotlin elasticsearch 핵심 라이브러리 오픈소스에 기여

4. 구현 결과

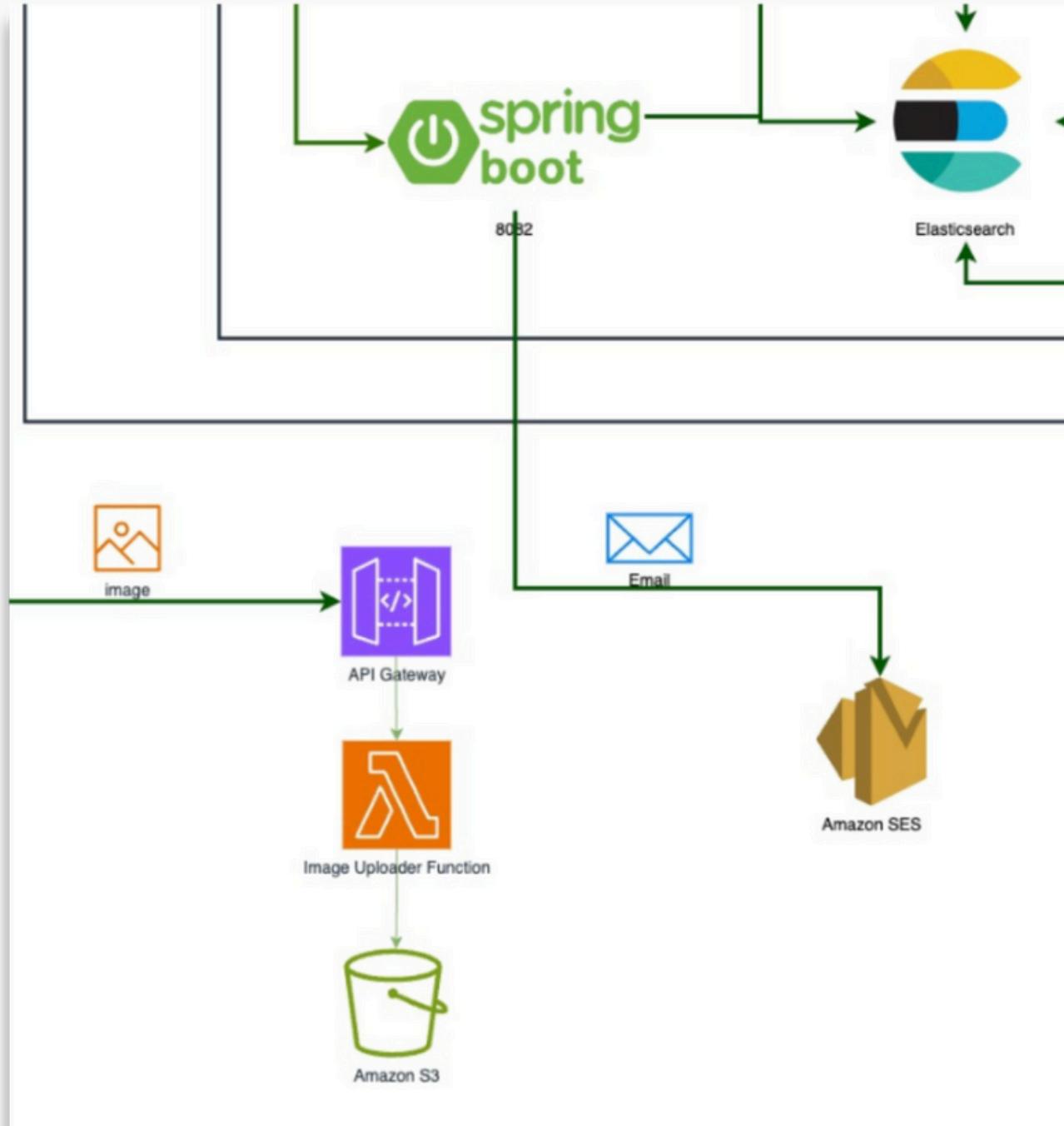
아키텍처 (CHAT GPT)



1. mysql에서 네이버 평점, 네이버 리뷰 기반으로 필터링해옴
2. mysql에서 유저가 찜한 음식점을 가져옴
3. redis에서 유저가 검색한 검색어를 가져옴
4. 1번 데이터와 2, 3번 데이터를 비교하여 유저에게 추천할 음식점들을 선별
5. ttl(Time-To-Live)를 3일로 설정하여 선별한 데이터를 redis에 저장

4. 구현 결과

아키텍처 (ETC..)



이미지 저장

[이미지 유형]

- 사용자 개인 프로필 Image
- 리뷰 Image

[이미지 저장 로직]

1. 이미지 업로드 시행
2. Lambda Call
3. Lambda S3에 이미지 저장
4. 저장된 이미지 주소를 DB에 기록

이메일 인증

성균관대 재학생만 사용할 수 있는
인증/인가 로직 진행

[이메일 인증 로직]

1. Spring Boot API서버 서비스 로직 호출
2. AWS SES (이메일 전송 AWS 서비스)
에서 이메일 전송
3. DB에 이메일 인증 번호 저장
4. 인증번호 대조 및 가입 절차 진행

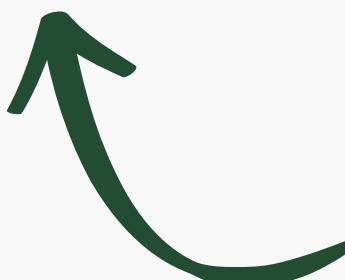
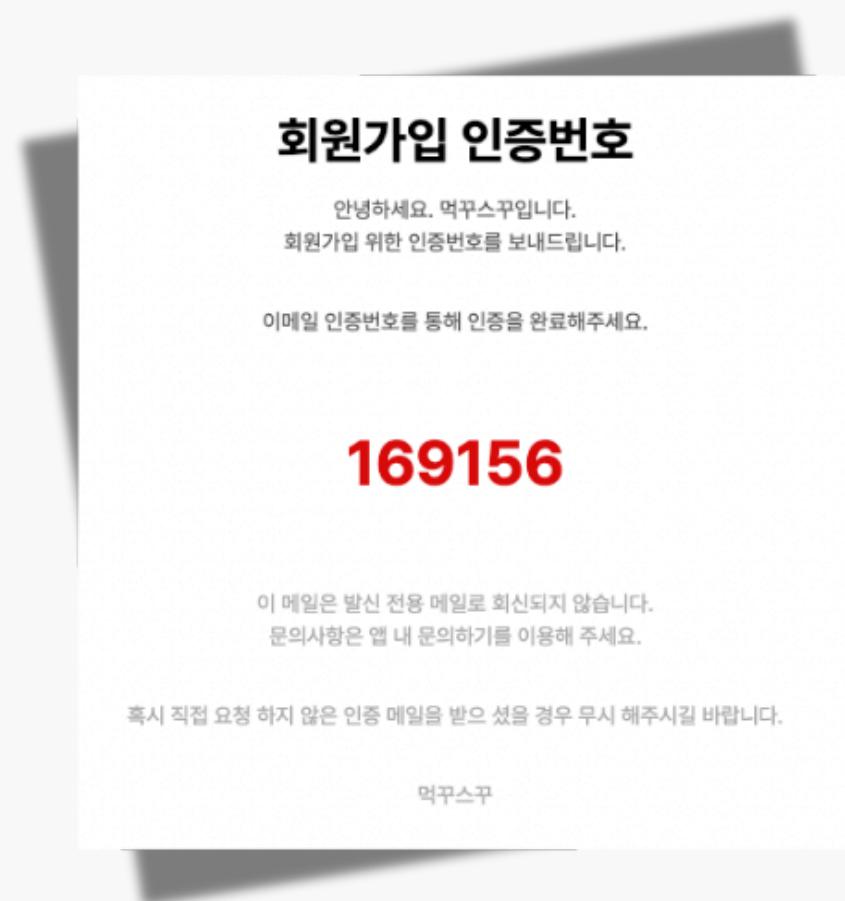
4. 구현 결과

어플리케이션 기능

- 성균관대 학생만을 위한 계정 생성 및 로그인
- GPT를 기반으로한 사용자 맞춤형 음식점 추천 시스템
- 사용자 검색 편의를 위한 자동완성 기능
- 다양한 필터를 통한 음식점 검색 기능
- 키워드를 이용한 쿼리 기반 음식점 추천 검색 기능
- 사용자의 현재 위치를 기반으로 한 음식점 지도와 음식점 리스트
- 음식점 세부 정보 및 리뷰 조회 및 작성
- Kingo-Pass: 성균관대와 제휴를 맺은 음식점 리스트
- 사용자 마이페이지

4. 구현 결과

- 성균관대 학생만을 위한 계정 생성 및 로그인



성균관대 공식 이메일을 통해
성균인 인증을 할 수 있어요.

오직 성균인만 사용 가능한 앱!

< 회원가입

이름
송새론

이메일 주소
saron01@g.skku.edu

비밀번호
.....

비밀번호 확인
.....

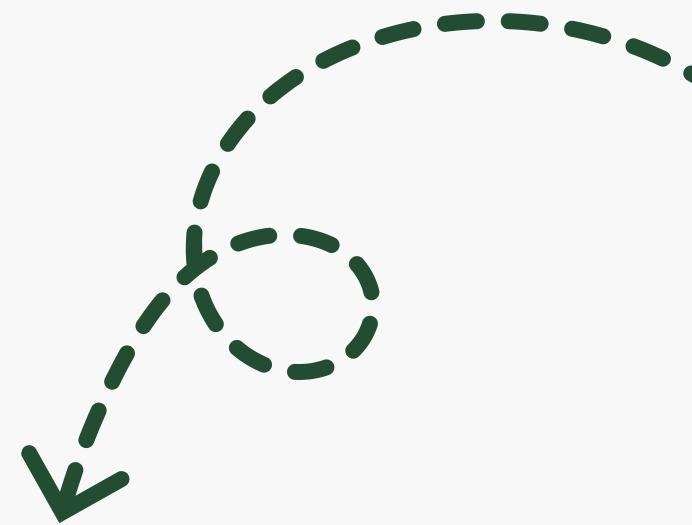
회원가입

가입하시면 이용약관 및 개인정보 보호정책에
자동으로 동의하게 됩니다.

☰ ☐ <

4. 구현 결과

- GPT 기반 사용자 맞춤형 음식점 추천 시스템



쓸데없는 고민 NO!
 사용자 맞춤형으로 음식을 추천해드려요

홈

먹구스꾸's 오늘의 픽



보리네주먹고기
★ 5.0 (0) ● 0 ■ 4.8 (999)

한식
 주먹고기 200g

아직 작성된 리뷰가 없어요!
 먹구스꾸에서 보리네주먹고기 리뷰 쓰고
 첫 리뷰의 주인공이 되어보세요!

카테고리 >

 한식	 양식	 일식	 중식	 분식
 치킨	 피자	 버거	 아시안	 카페

성대생 할인, 킹고패스



자명문

음료 한정 테이크 아웃
30% 가격 할인



정성식탁

테이블 당 음료수 or
에이드 제공



정든닭발

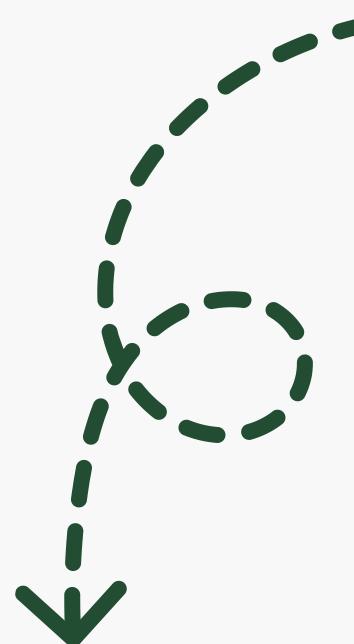
대왕 달걀찜
제공! 테이블
주류 or 음료수

홈 지도 리스트 프로필


 먹구
스꾸

4. 구현 결과

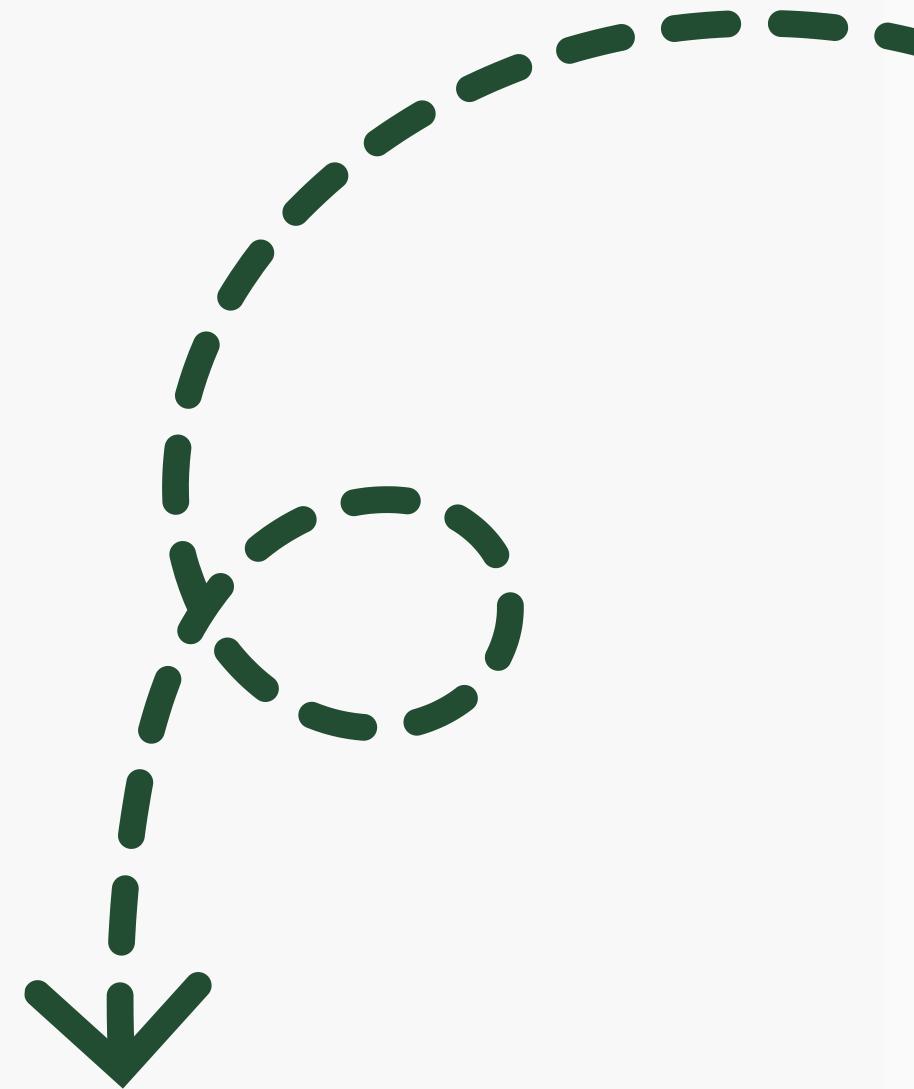
- 다양한 필터를 통한 음식점 검색 가능



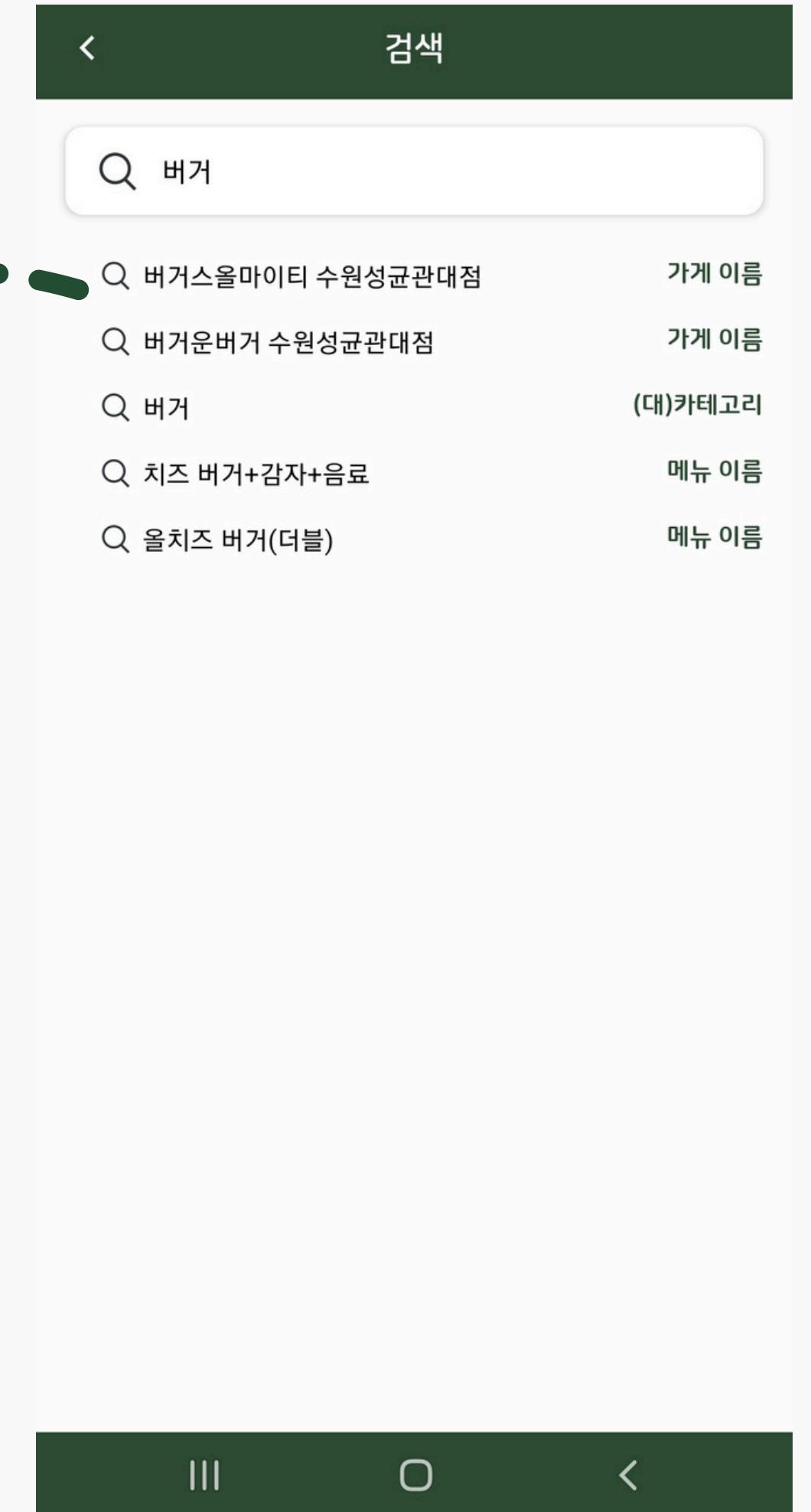
여러가지 필터를 통해 원하는 식당을 쉽게 찾을 수 있어요

4. 구현 결과

- 키워드를 이용한 쿼리 기반 음식점 추천 검색 기능



자동완성 기능을 통해
간단한 키워드로 쉽게 검색할 수 있어요



검색

버거

버거스올마이티 수원성균관대점
가게 이름

버거운버거 수원성균관대점
가게 이름

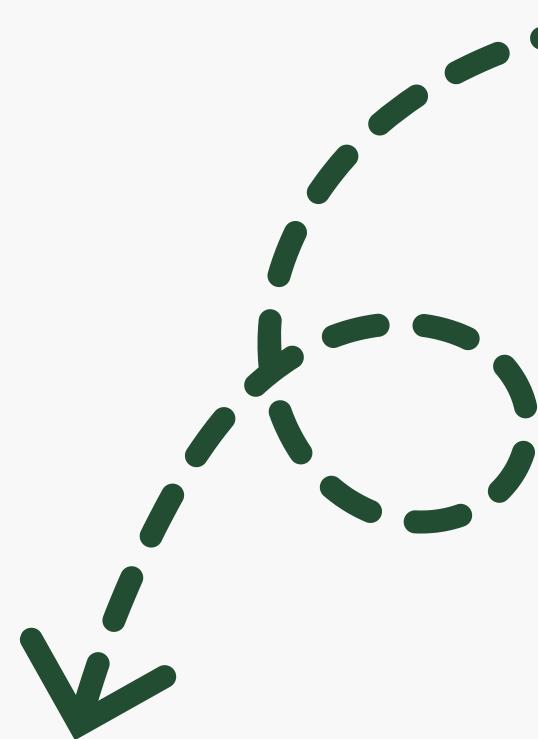
버거
(대)카테고리

치즈 버거+감자+음료
메뉴 이름

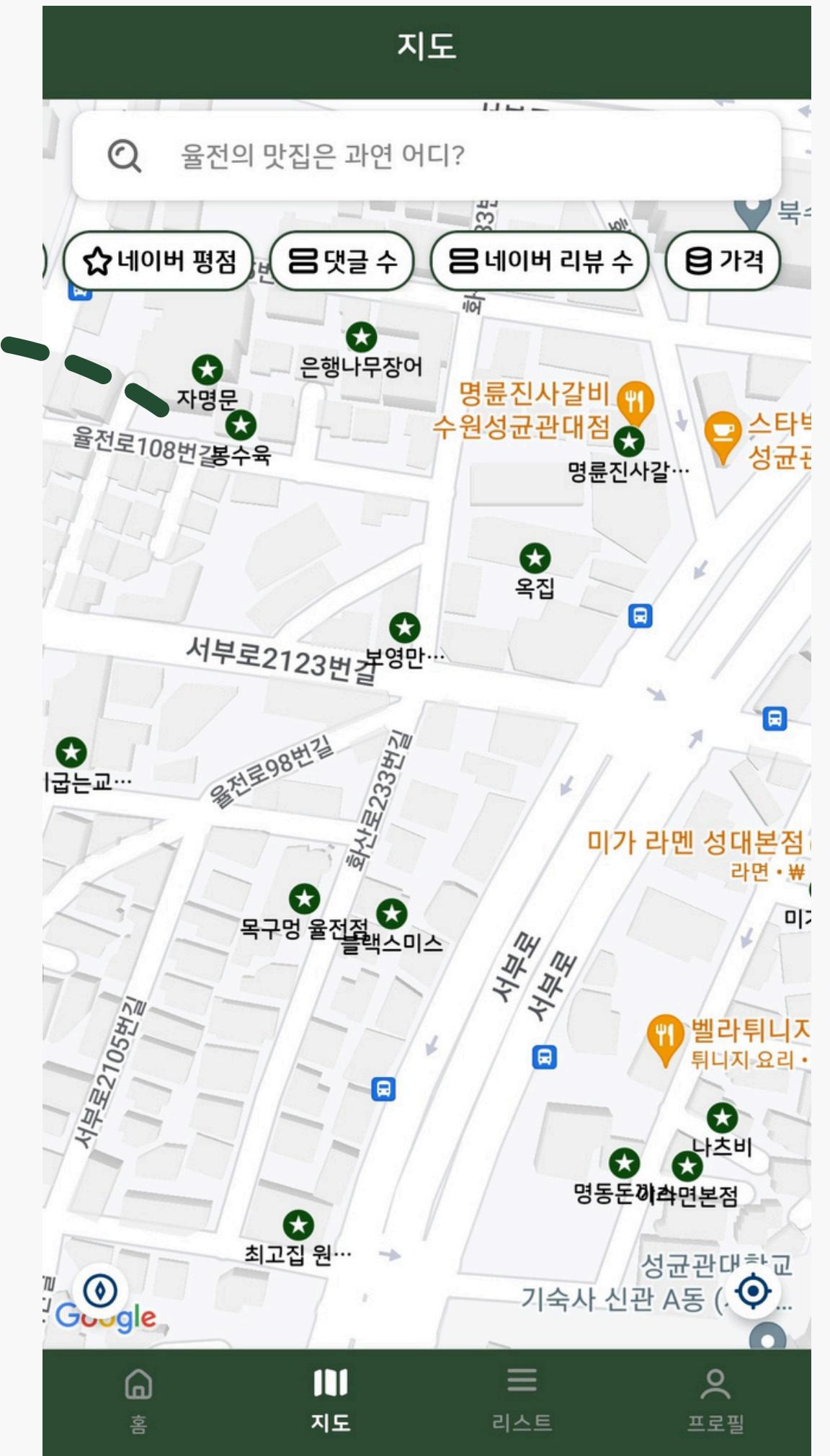
올치즈 버거(더블)
메뉴 이름

4. 구현 결과

- 사용자의 **현재 위치를 기반으로 한 음식점 지도와 음식점 리스트**

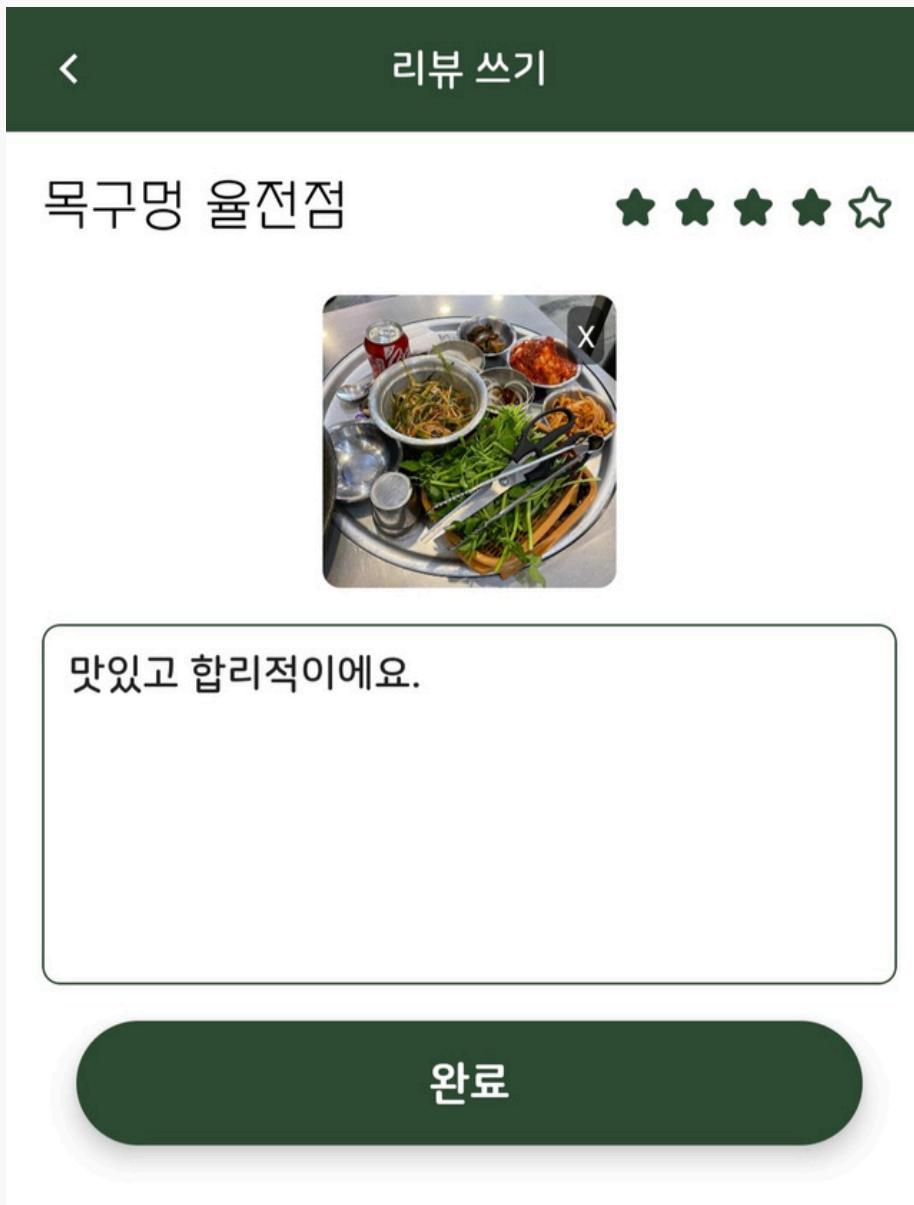


나의 **현재 위치를 기준으로**
성대 주변 맛집을 쉽게 확인할 수 있어요

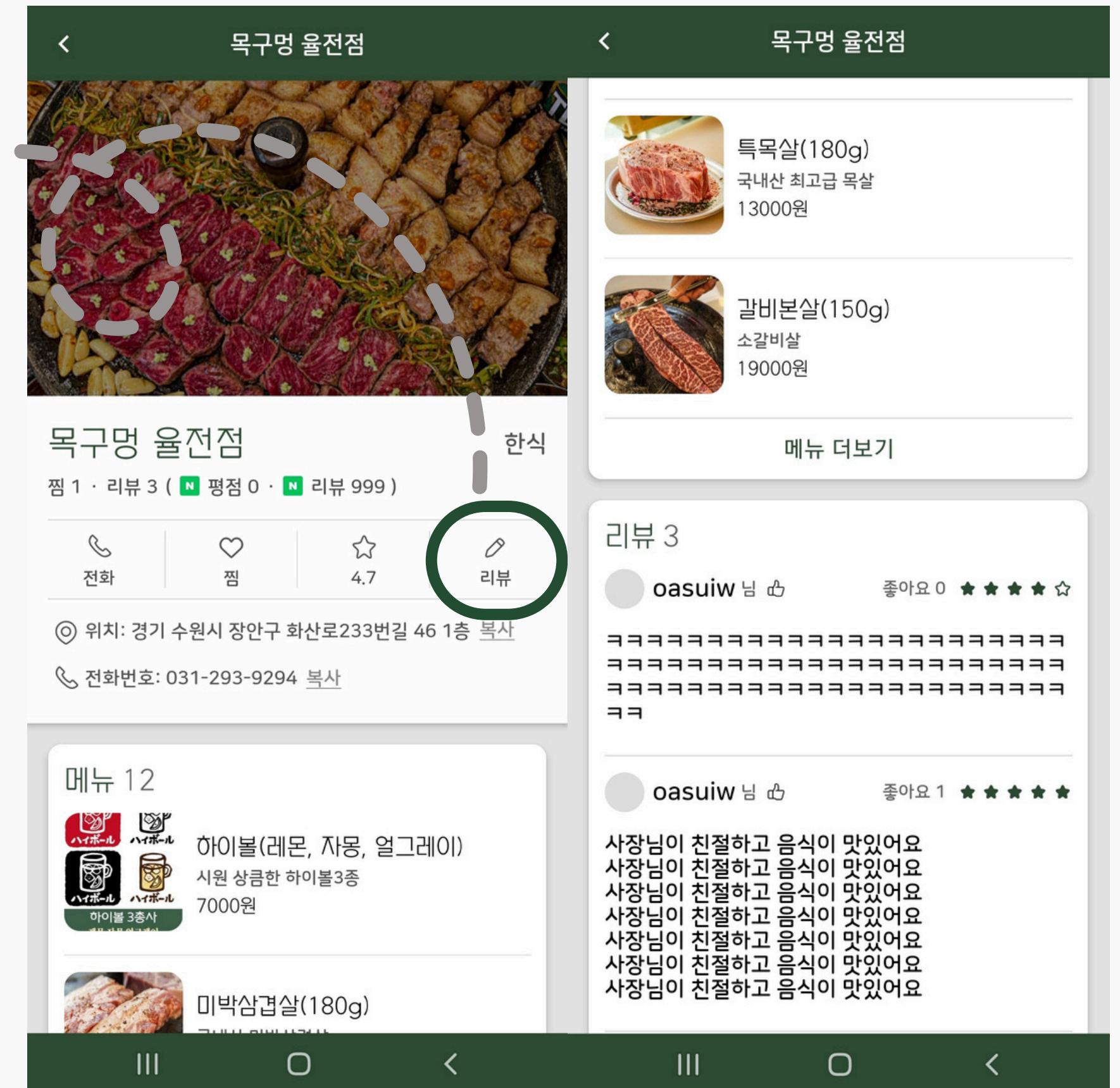


4. 구현 결과

- 음식점 세부 정보 및 리뷰 조회 및 작성



음식점의 정보를 한눈에 확인할 수 있고
리뷰를 작성할 수 있어요

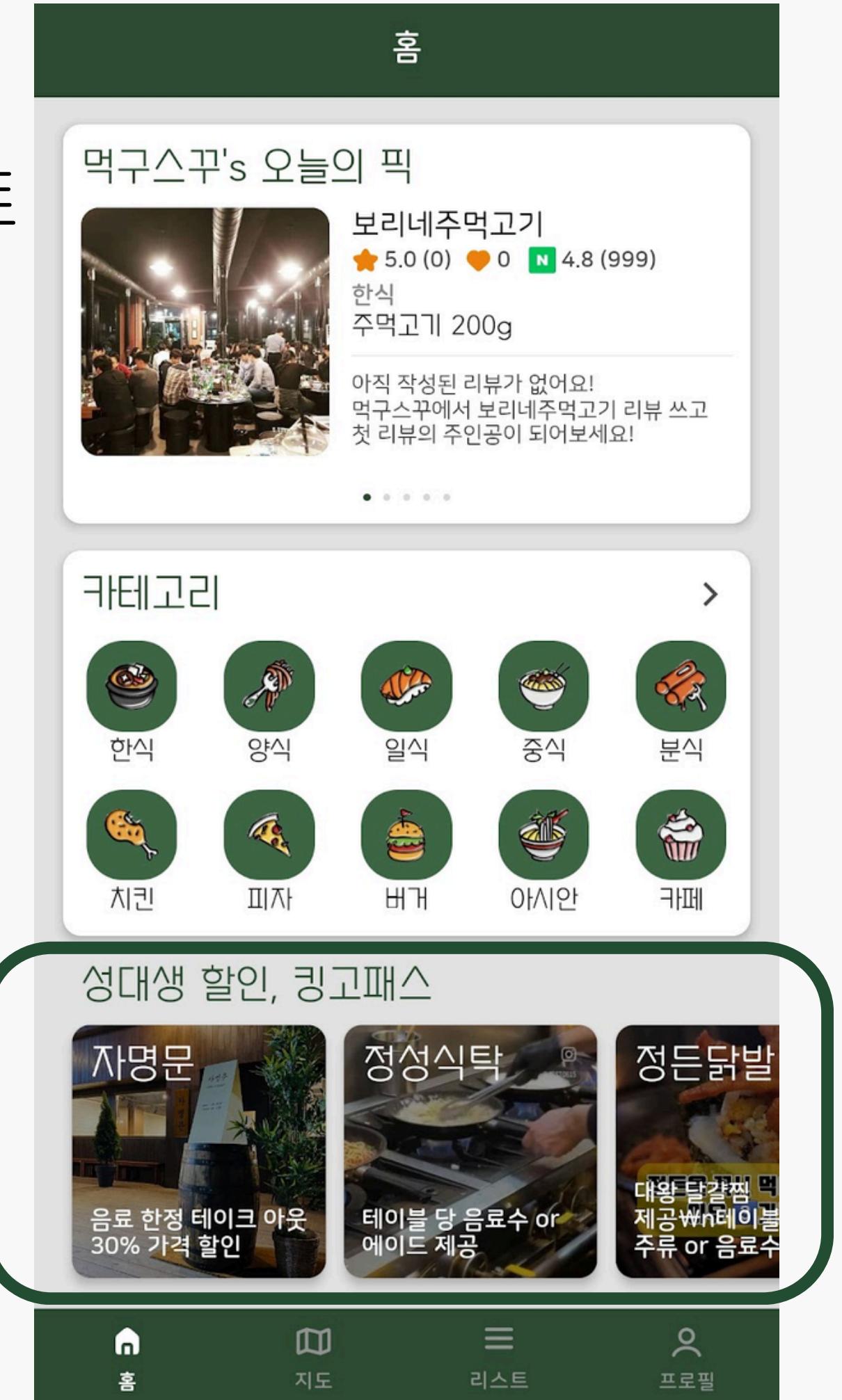


박구
한국

4. 구현 결과

- **Kingo-Pass:** 성균관대와 제휴를 맺은 음식점 리스트

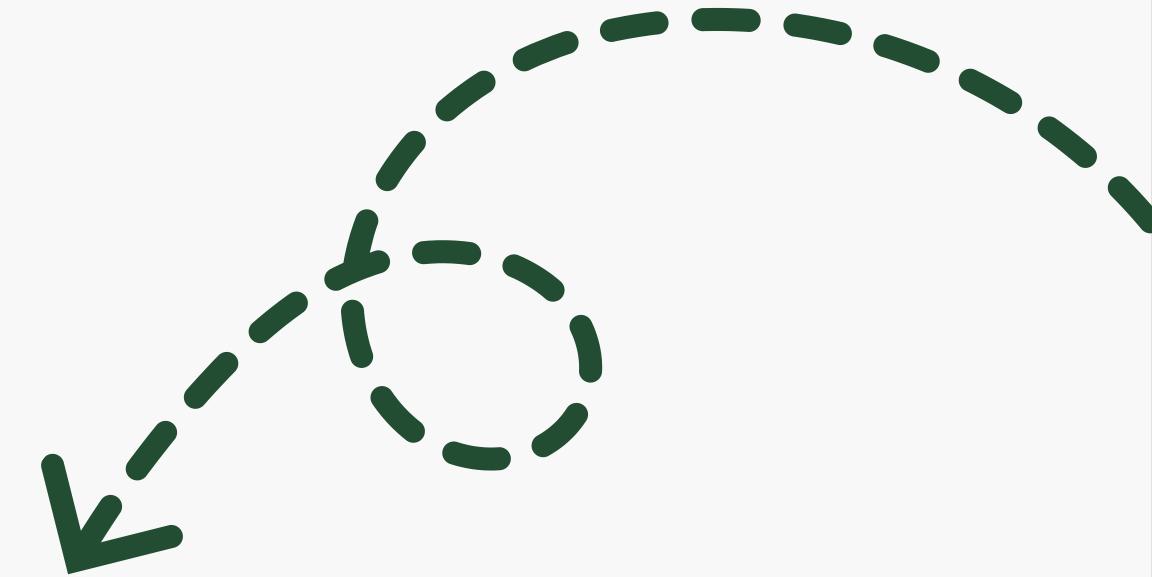
Kingo-Pass 로 할인 가능한 음식점을 쉽게 찾고
할인된 가격으로 만나세요!

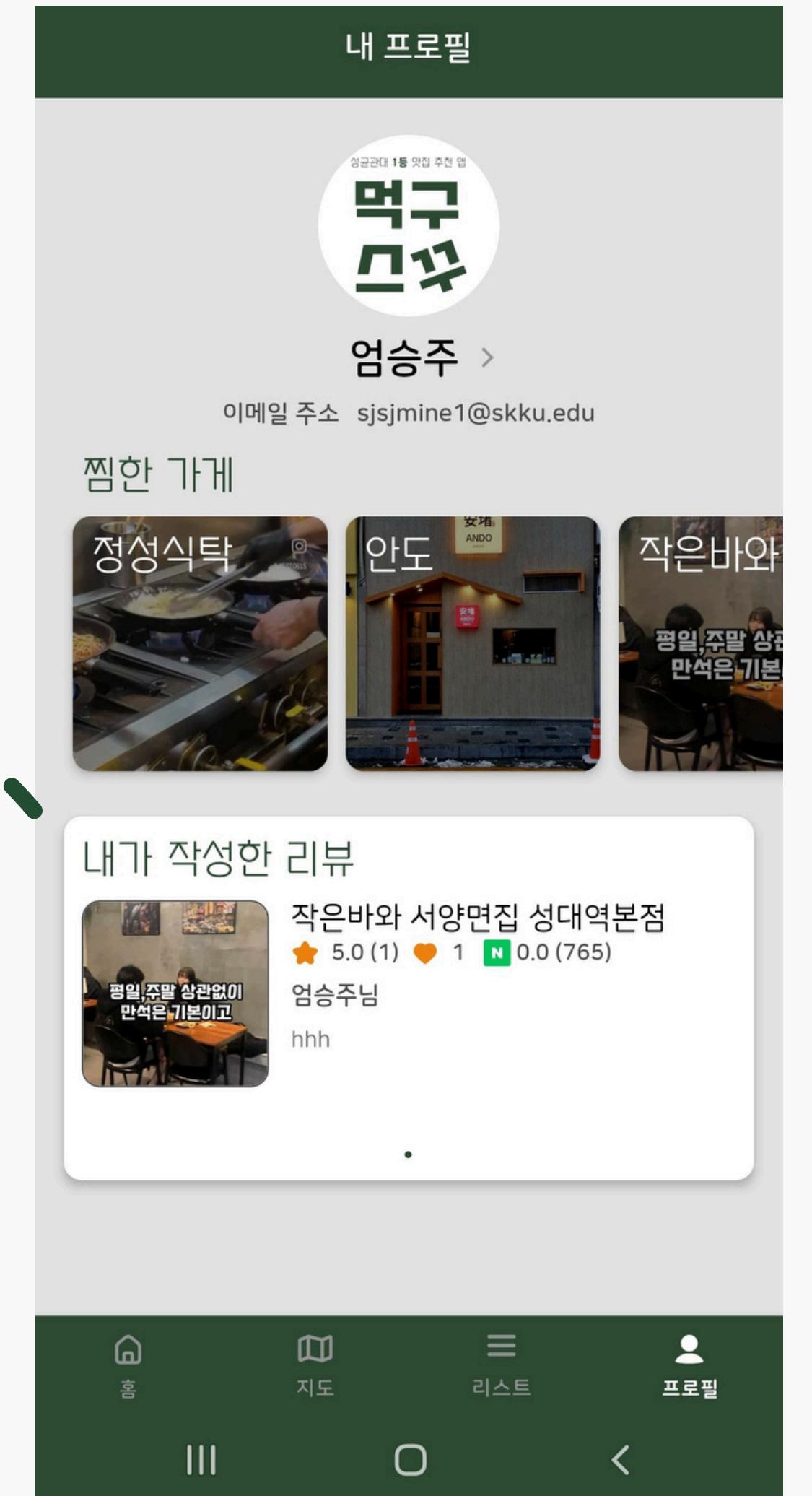
The screenshot shows the Kingo-Pass app interface. At the top, there's a dark green header bar with the word "홈" (Home) in white. Below it is a white card for "먹구스꾸's 오늘의 픽" (Meatgu's Today's Pick), featuring a photo of a restaurant interior, the dish "보리네주먹고기" (Bori Neju Mebeoggi), a rating of 5.0 (0 reviews), 0 likes, and 4.8 (999 reviews). It also lists "한식 주먹고기 200g". A note says "아직 작성된 리뷰가 없어요! 먹구스꾸에서 보리네주먹고기 리뷰 쓰고 첫 리뷰의 주인공이 되어보세요!" (No reviews yet! Write a review for Bori Neju Mebeoggi on Meatgu and become the protagonist of the first review!). Below this is a section for "카테고리" (Categories) with icons for 한식, 양식, 일식, 중식, 분식, 치킨, 피자, 버거, 아시안, and 카페. At the bottom, a box highlights "성대생 할인, 킹고패스" (Student Discount, KingoPass) with three offers: "자명문 음료 한정 테이크 아웃 30% 가격 할인" (Zamyeonmun Beverage Limited Takeout 30% Price Discount), "정성식탁 테이블 당 음료수 or 에이드 제공" (Jungseongsitdak Table Beverage or Aid Supply), and "정든닭발 대왕 달걀찜 제공 테이블 주류 or 음료수" (Jeondan Dakbal Large King Egg Stew Supply Table Liquor or Beverage).

4. 구현 결과

- 사용자 마이페이지



마이페이지에서
내가 **찜한가게, 작성한 리뷰를**
확인할 수 있어요



내 프로필

성균관대 1동 맛집 추천 앱
먹구
드꾸

엄승주 >

이메일 주소 sjsjmine1@skku.edu

찜한 가게

정성식탁
안도
작은바와

내가 작성한 리뷰

작은바와 서양면집 성대역본점
★ 5.0 (1) ♥ 1 N 0.0 (765)
엄승주님
hhh

홈 지도 리스트 프로필

☰ ⌂ ⌄ <

**먹구
드꾸**

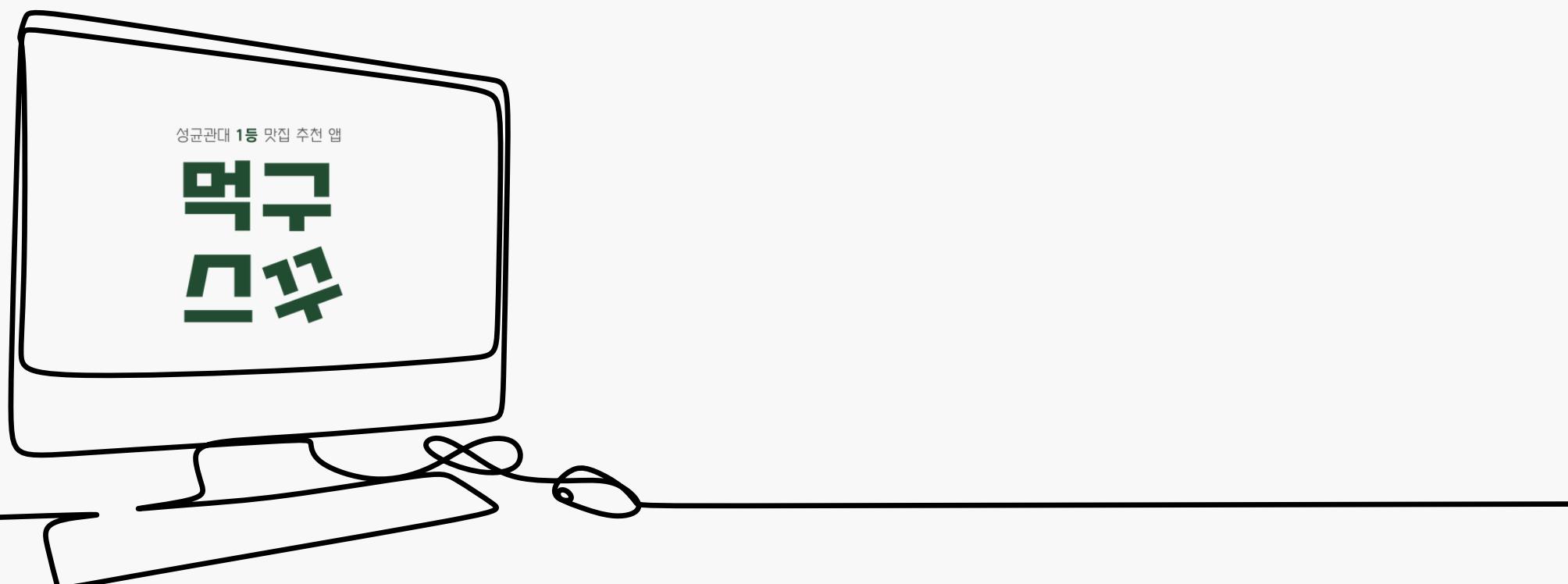
성균관대 1등 맛집 추천 앱

먹구
드루

5. 시연 영상

먹구
드루

5. 시연영상



송시

이메일 주소

비밀번호

비밀번호 확인

영어, 숫자, 특수문자가 포함되어야 합니다.

송새론人 송승현 송소희 ...

1	2	3	X
. .	-		
4	5	6	완료
ㄱㅋ	ㄴㄹ	ㄷㅌ	
ㅂㅍ	ㅅㅎ	ㅈㅊ	, ?!
#!1	한/영	□○	,
		[]	





6. 태그리

6. 테스트

기능 테스트

Unit Test

함수의 기능들을 독립적으로 테스트

Integration Test

실제 API 요청을 가정하여 진행하는 전체 비즈니스 로직 테스트

CI

운영 서버에 배포하기 이전에 진행하는 Code - Build - Test

사용성 테스트

Usability Test

사용자의 실제 서비스 사용 테스트

6. 테스트

기능 테스트 UNIT TEST

```

describe( name: "Review" ) -> this: DescribeSpecContainerScope

    describe( name: "decrementLikeCount" ) { this: DescribeSpecContainerScope
        it( name: "should decrement like count when it is greater than zero" ) { this: TestScope
            // Given
            val user = User(
                id = 1L,
                nickname = "testuser",
                profileImageUrl = "http://example.com/profile.jpg"
            )

            val review = Review(
                user = user,
                restaurantId = 10L,
                content = "Great restaurant!",
                rating = 4.5,
                likeCount = 5,
                viewCount = 100
            )

            // When
            review.decrementLikeCount()

            // Then
            review.likeCount shouldBe 4
        }
    }
}

```

함수의 로직을 독립적으로 테스트

[decrementLikeCount() 함수 테스트]

Review 좋아요 수를 감소하는 함수 테스트

'Review' 도메인에서,
좋아요 개수가 5개인 Review가 존재하고,
좋아요 개수를 감소시키는 함수를 실행했을 때,
좋아요 개수가 4가 되는가?

6. 테스트

기능 테스트 INTEGRATION TEST

```

beforeEach { it: TestCase
    setUpUser( email: "test@gmail.com", userRepository)
}

describe( name: "#deleteReview basic test") { this: DescribeSpecContainerScope
    it( name: "when existed review delete should return success") { this: TestScope
        // given
        val restaurant = restaurantRepository.save(
            RestaurantUtil.generateRestaurantEntity(
                name = "restaurant"
            )
        )

        val review = reviewRepository.save(
            ReviewUtil.generateReviewEntity(
                restaurantId = restaurant.id,
                user = userRepository.findByEmail( email: "test@gmail.com") ?: throw Exception()
            )
        )

        // when
        val result = mockMvc.perform(
            delete( urlTemplate: "$baseUrl/reviews/${review.id}")
        ).also { it: ResultActions
            println(it.andReturn().response.contentAsString)
        }

        .andExpect(status().isOk)
        .andExpect(jsonPath( expression: "$.result").value( expectedValue: "SUCCESS"))
        .andReturn()
    }
}

```

actualResult.result shouldBe CommonResponse.Result.SUCCESS

실제 API 요청을 가정하여 진행하는 전체 비즈니스 로직 테스트

[존재하는 리뷰를 삭제하는 Request API를 처리하는 테스트]
 자신이 작성한 Review를 삭제하는 API 요청이 들어오면,
 성공적으로 삭제되었는지 응답함을 테스트

먼저 사용자가 로그인이 되었다고 가정
 사용자가 Review를 작성한 상황에서,
 해당 Review를 삭제하는 APIEndPoint로 삭제 요청을 보냈을 때,
 성공적으로 삭제되었다는 응답을 보낸다.

6. 테스트

기능 테스트

CONTINUOUS INTEGRATION

```

name: Ktlint
on: pull_request

jobs:
  build-and-check:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set Up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'adopt'
      - name: Grant Execute Permission For Gradlew
        run: chmod +x gradlew
      - name: Ktlint & Detekt Check
        run: ./gradlew check -x test
  
```

```

name: Test
on:
  pull_request:

env:
  AWS_ACCESS_KEY: ${{ secrets.AWS_ACCESS_KEY }}
  AWS_SECRET_KEY: ${{ secrets.AWS_SECRET_KEY }}

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set Up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'adopt'
      - name: Grant Execute Permission For Gradlew
        run: chmod +x gradlew
      - name: Test With Gradle
        run: ./gradlew test koverXmlReport
      - name: Add coverage report to PR
        id: kover
        uses: mi-kas/kover-report@v1
        with:
          path: ${{ github.workspace }}/build/reports/kover/report.xml
          token: ${{ secrets.GITHUB_TOKEN }}
          title: Code Coverage
          update-comment: true
          min-coverage-overall: 10
          min-coverage-changed-files: 10
          coverage-counter-type: LINE
  
```

Pull Request마다 CI 진행

1. 개발한 코드의 Lint 체크
2. Compile 및 Test Code 검증
3. TestCode Coverage 검증

✓ [KAN-39] Review 작성 API & 테스트코드 초안 작성
 Test #53: Pull request #31 opened by goathoon

✗ [KAN-39] Review 작성 API & 테스트코드 초안 작성
 Ktlint #53: Pull request #31 opened by goathoon

[KAN-39] PR에서 Lint 오류로 CI가 Fail된 모습

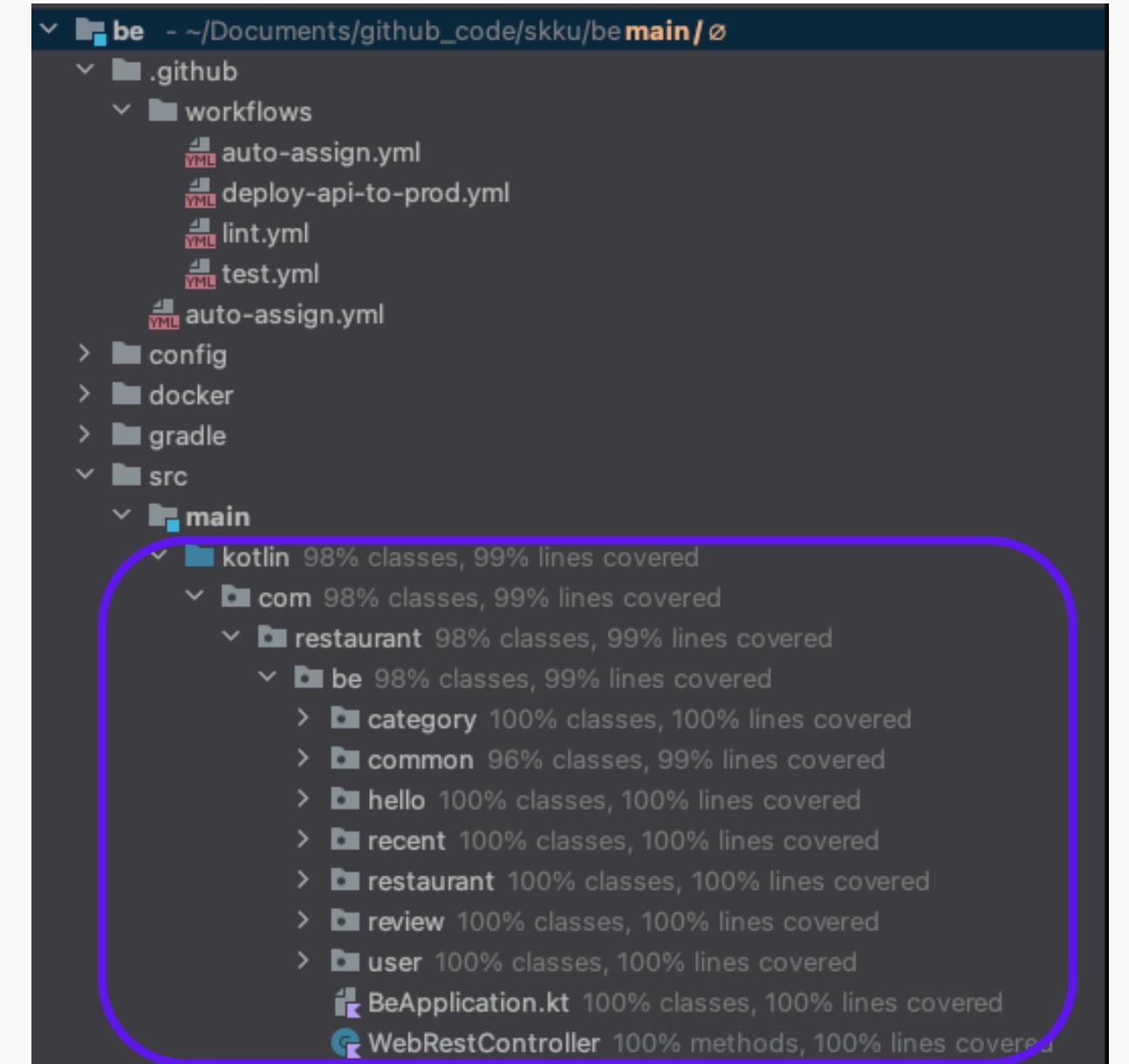
6. 테스트

기능 테스트

github-actions bot commented 2 days ago · edited

Code Coverage

File	Coverage	...
src/main/kotlin/com/restaurant/be/common/config/JwtSecurityConfig.kt	99.54%	
src/main/kotlin/com/restaurant/be/common/config/SecurityConfig.kt	100.00%	
src/main/kotlin/com/restaurant/be/common/exception/GlobalExceptionHandler.kt	100.00%	
src/main/kotlin/com/restaurant/be/common/exception/ServerException.kt	100.00%	
src/main/kotlin/com/restaurant/be/common/jwt/JwtFilter.kt	96.00%	
src/main/kotlin/com/restaurant/be/common/jwt/TokenProvider.kt	98.99%	
src/main/kotlin/com/restaurant/be/common/redis/RedisRepository.kt	100.00%	
src/main/kotlin/com/restaurant/be/common/response/CommonResponse.kt	100.00%	
src/main/kotlin/com/restaurant/be/common/response/ErrorCodes.kt	100.00%	
src/main/kotlin/com/restaurant/be/common/response/Token.kt	100.00%	
src/main/kotlin/com/restaurant/be/user/domain/service/SignInUserService.kt	100.00%	
src/main/kotlin/com/restaurant/be/user/domain/service/SignUpUserService.kt	100.00%	
Total Project Coverage	98.56%	



메인 서버 테스트커버리지 약 99%
 주요 사용 코드 테스트 커버리지 100%

6. 테스트

사용성 테스트 정성적 평가

A군

(남, 대학교 3학년)

- 검색 조건이 다양해서 **만족**
- 학교 주변 맛집 정보가 많아서 **유용함**
- 추천이 개인의 니즈를 얼마나 반영하는지는 **잘 모르겠음**

B군

(남, 대학교 1학년)

- 식당 정보가 아직 유통 한정인 게 **아쉬움**
- UI 가 친숙해서 사용이 **편리함**
- 세부 검색 기능 **만족**
- AI한테 **추천 식당 물어보는 기능**이 있으면 좋겠음

C양

(여, 대학교 4학년)

- UI가 편리하지만 **차별성 부족**
- 식당, 리뷰 정보가 많아야 **유용**하다고 생각

- 다양의 식당/리뷰 데이터 확보**가 서비스에 중요
- AI 채팅 기반 식당 추천 기능 추가 필요



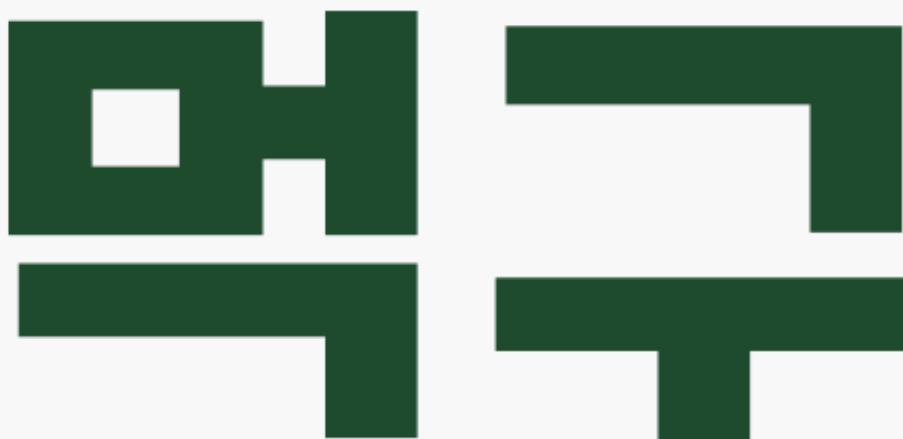
7. 풀

7. 결론



“기존 서비스의 장점을 모아 성균관대 학생을 위한 맛집 추천 서비스를 만들어보자”

성균관대 1등 맛집 추천 앱



- AI 추천 서비스
- 믿을만한 리뷰 시스템
- 편리하고 다양한 검색 시스템



성균관대 1등 맛집 추천 앱

이상으로

먹구
드구

발표를

마끼겠습니다.

감사합니다.



Q&A