1조 PickTogether

| 먹거리 공동구매 웹 서비스 위치기반 & AI 추천

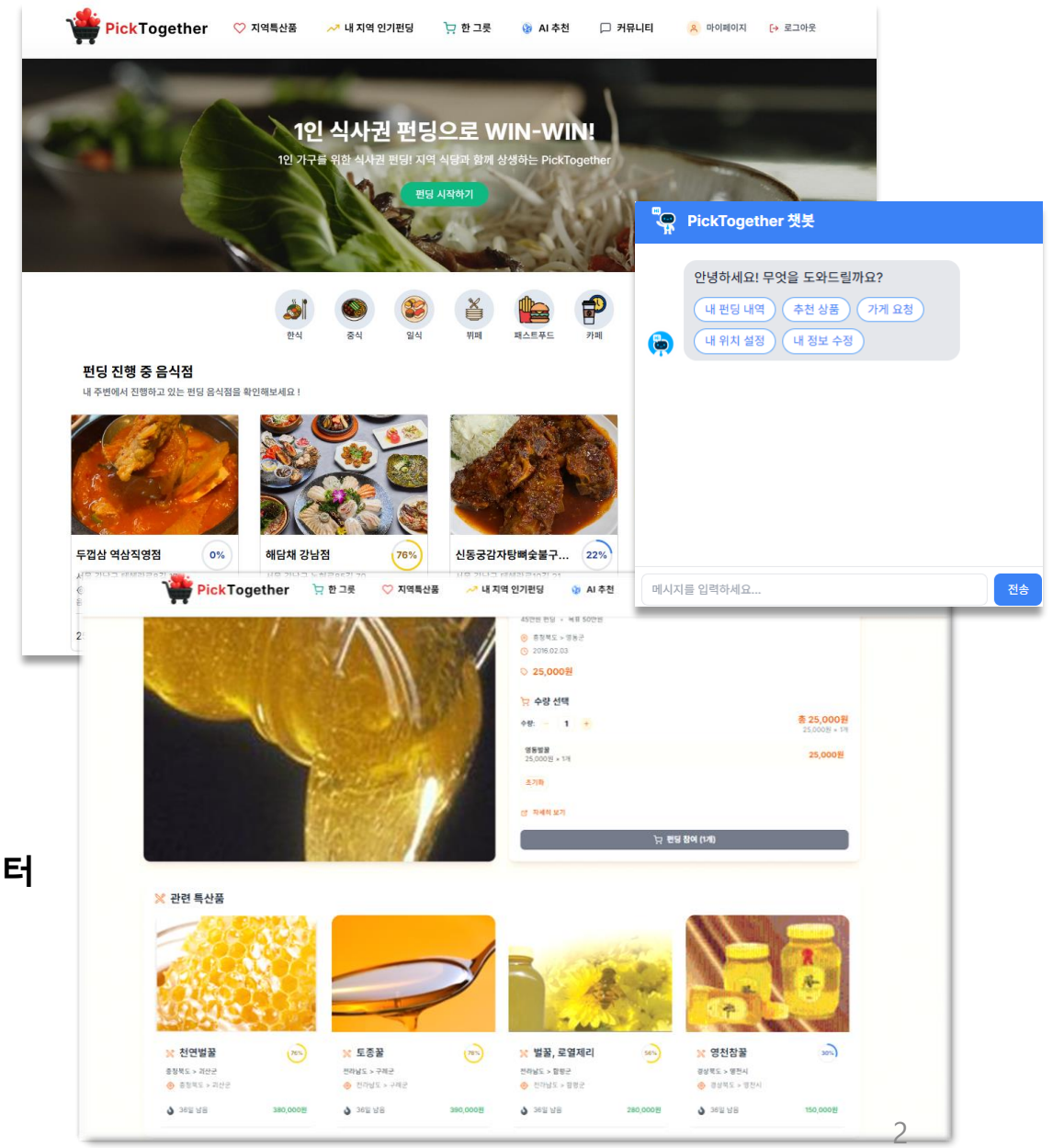
- GitHub

고인한: https://github.com/inhan99
김찬영: https://github.com/chanO4135
고윤호: https://github.com/KessokuMAS
송승찬: https://github.com/bannana-key
서민서: https://github.com/bannana-key

picktogether

| 먹거리 공동구매 웹 서비스 위치기반 & AI 추천

개발 기간	2025.08.
플랫폼	Web(반응형)
개발 인원	5명 (팀장 김찬영 팀원 고인한, 고윤호, 서민서, 송승찬)



개발 환경

언어	Java, JavaScript, TailWindCSS, Python
서버	Tomcat, Aws
프레임워크	Spring Framework 3.1.1, MySQL
DB	MariaDB, Amazon RDS
IDE	VSCode
API, 라이브러리	Kakao/Naver/Google Login API, Kakao Map API, 공공데이터 포털API, PortOneAPI, Langchain_openai, PyTorch

picktogether

2025 소비트렌드 키워드에 따르면, 소비자들은 **합리적이고 무해한 일상 소비와 상생 구조**를 추구함.

주목해야할 핵심 트렌드 키워드

#아보하
평범한 일상 속 합리적 소비 실현

무해력 & 공진화 전략
음식점과 소비자 모두가 이익을 얻는 상생구조

토핑경제
단순 아닌 펀딩 참여 + 할인 혜택이라는 토핑을 얹음
소비자가 단구매가순 소비자가 아니라 참여자이자 기여자가 되는 구조

페이스테크
위치 기반 + 선호 메뉴 분석 → 맞춤형 AI 추천
단순 기술이 아닌, 생활에 밀착된 개인화 경험 제공

picktogether

1인 가구를 위한 **한그릇 펀딩** 시스템

1인 가구 증가 → 대량 구매의 어려움, 잦은 **식품 낭비**

식당 최소 주문 인원 제한 → 혼자 외식하기 어려움

장거리 인기 맛집 접근 제한 → 직접 방문 또는 주문이 번거로움

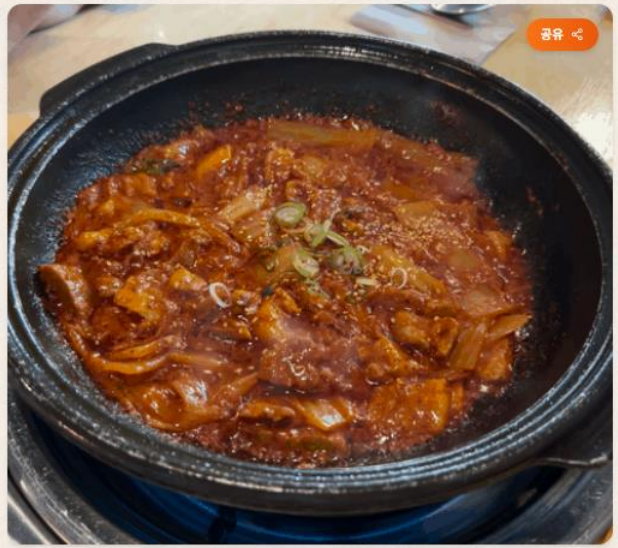
출처 : https://kostat.go.kr/board.es?mid=a10301010000&bid=10820&act=view&list_no=434103

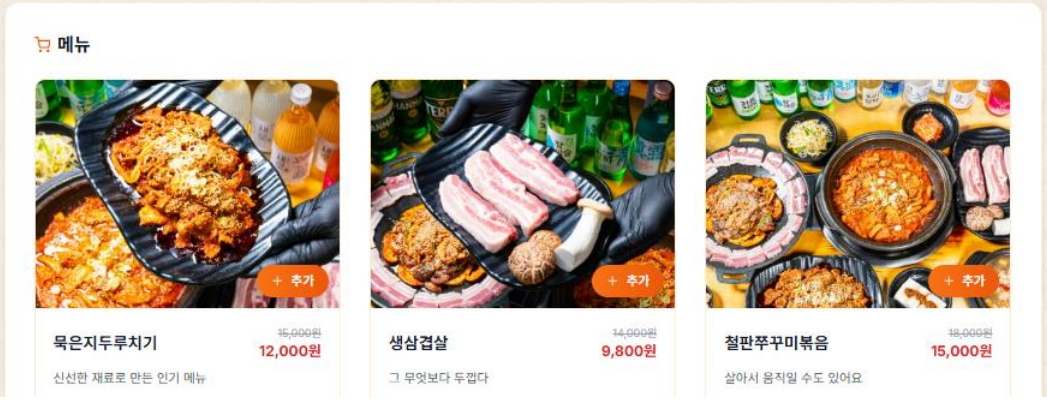
소상공인 한계 → 복잡한 입점 절차, 단발성 이벤트로 지속 매출 확보 어려움

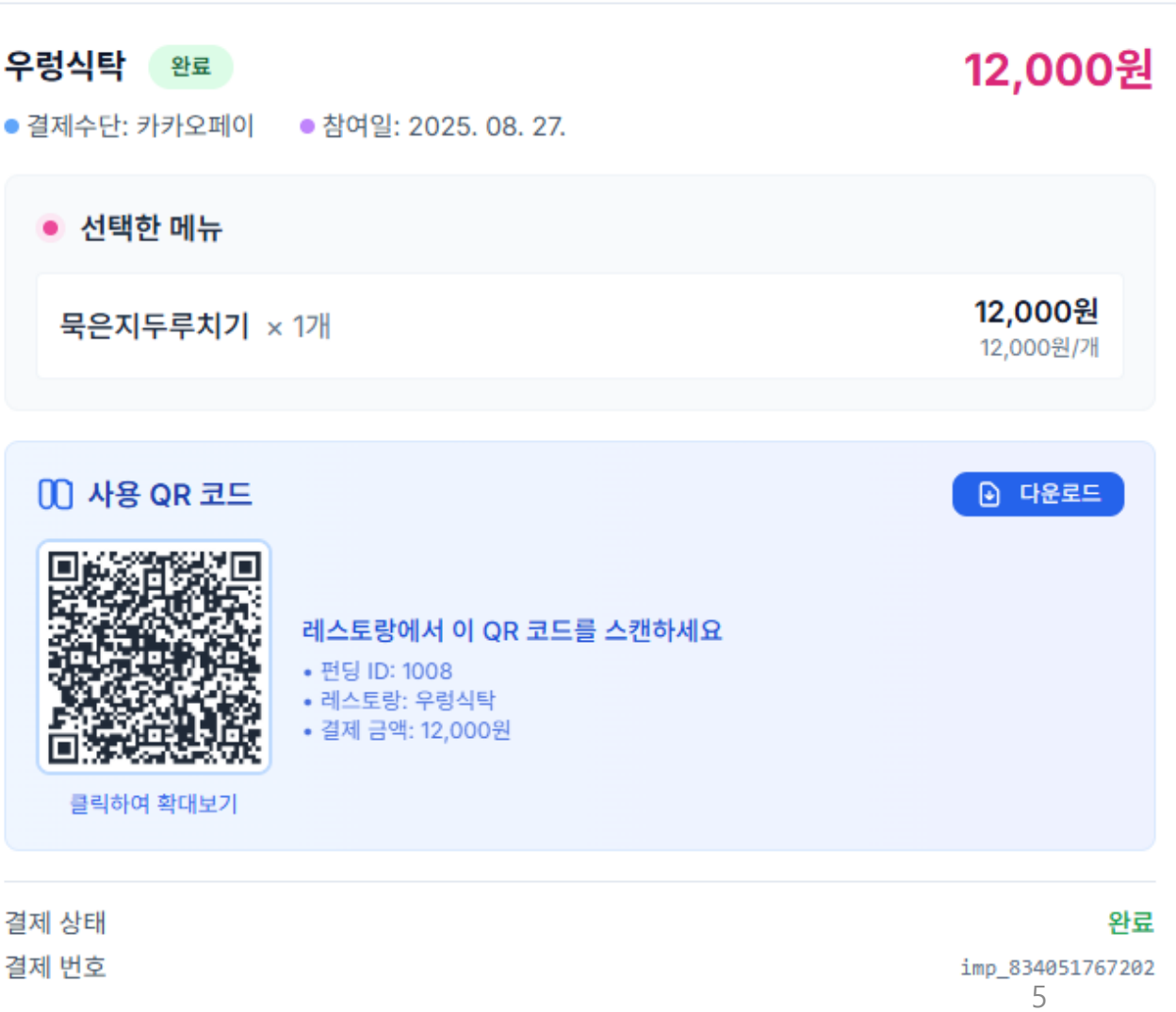
현행 서비스의 한계 → 기존 공동구매 서비스는 정기배송·리워드 중심 →지역 식당·오프라인 식사권 기반 상품은 미흡

picktogether

| 먹거리 공동구매 웹 서비스 위치기반 & AI 추천







picktogether

| 서비스 기획 및 방향성 설계(1/5)

진행 순서

기획

서비스 기획 → 요구 사항 정리

분석/설계

서비스 용어 정리 → IA 설계 → ERD 설계 → WBS 작성

이행

와이어프레임 → 개발환경 구축 → Git 연동 (SourceTree) → 핵심 기능 구현 (예 : 펀딩 시스템 구현)

→ 일반 기능 구현 (예 : 시큐리티, 게시판 등) → 리팩토링 (예 : 중복함수제거 등) → 성능개선 (예 : 무한스크롤 추가 등) → ASW 호스팅 및 서비스 배포

picktogether

| 서비스 기획 및 방향성 설계(2/5)

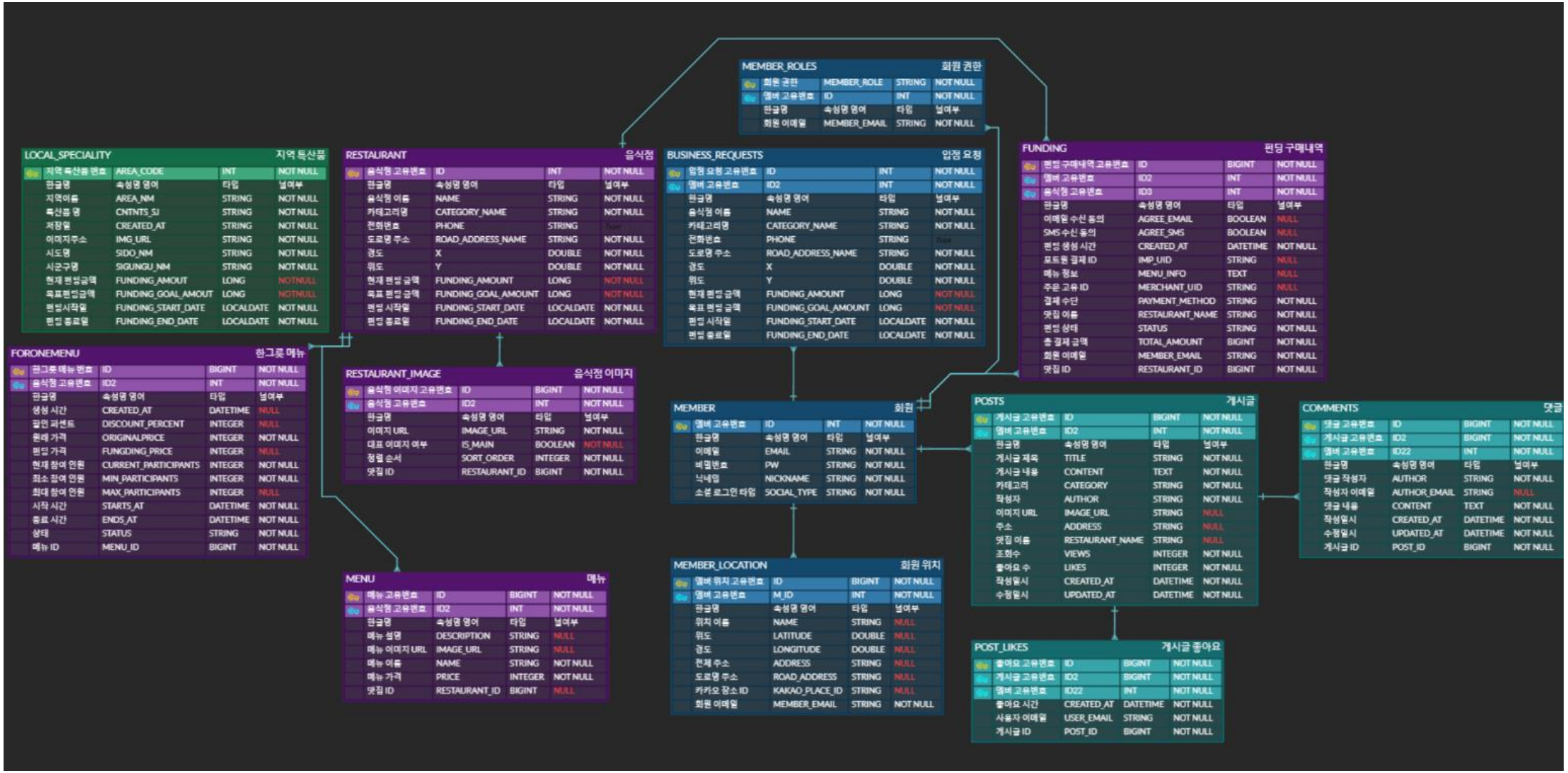
WBS

최초작성일	2025.08.01	최종수정일	2025.08.25		
구분	업무 내용	세부 업무	시작일	종료일	소요일
착수	주제 선정	주제 선정	2025.08.01	2025.08.03	3
분석	요구사항 분석	요구사항 분석	2025.08.03	2025.08.04	2
		요구사항 정의	2025.08.04	2025.08.05	2
설계	ERD 설계	ERD 설계	2025.08.05	2025.08.06	2
	표준 용어 정의	표준 용어 정의	2025.08.06	2025.08.06	1
	IA 설계	IA 설계	2025.08.07	2025.08.09	3
개발	개발환경 세팅	프로젝트 생성&샘플코드 작성	2025.08.09	2025.08.11	3
	핵심 기능 구현	프론트앤드 핵심 기능 구현	2025.08.11	2025.08.20	10
		백앤드 핵심 기능 구현	2025.08.13	2025.08.20	8
	일반 기능 구현	프론트앤드 일반 기능 구현	2025.08.16	2025.08.23	8
		백앤드 일반 기능 구현	2025.08.16	2025.08.23	8
리팩토링	성능개선	무한스크롤 등	2025.08.21	2025.08.26	6
검수	검수 확인	통합테스트	2025.08.23	2025.08.29	4
배포	AWS배포	AWS배포	2025.08.30	2025.08.31	2

picktogether

| 서비스 기획 및 방향성 설계(3/5)

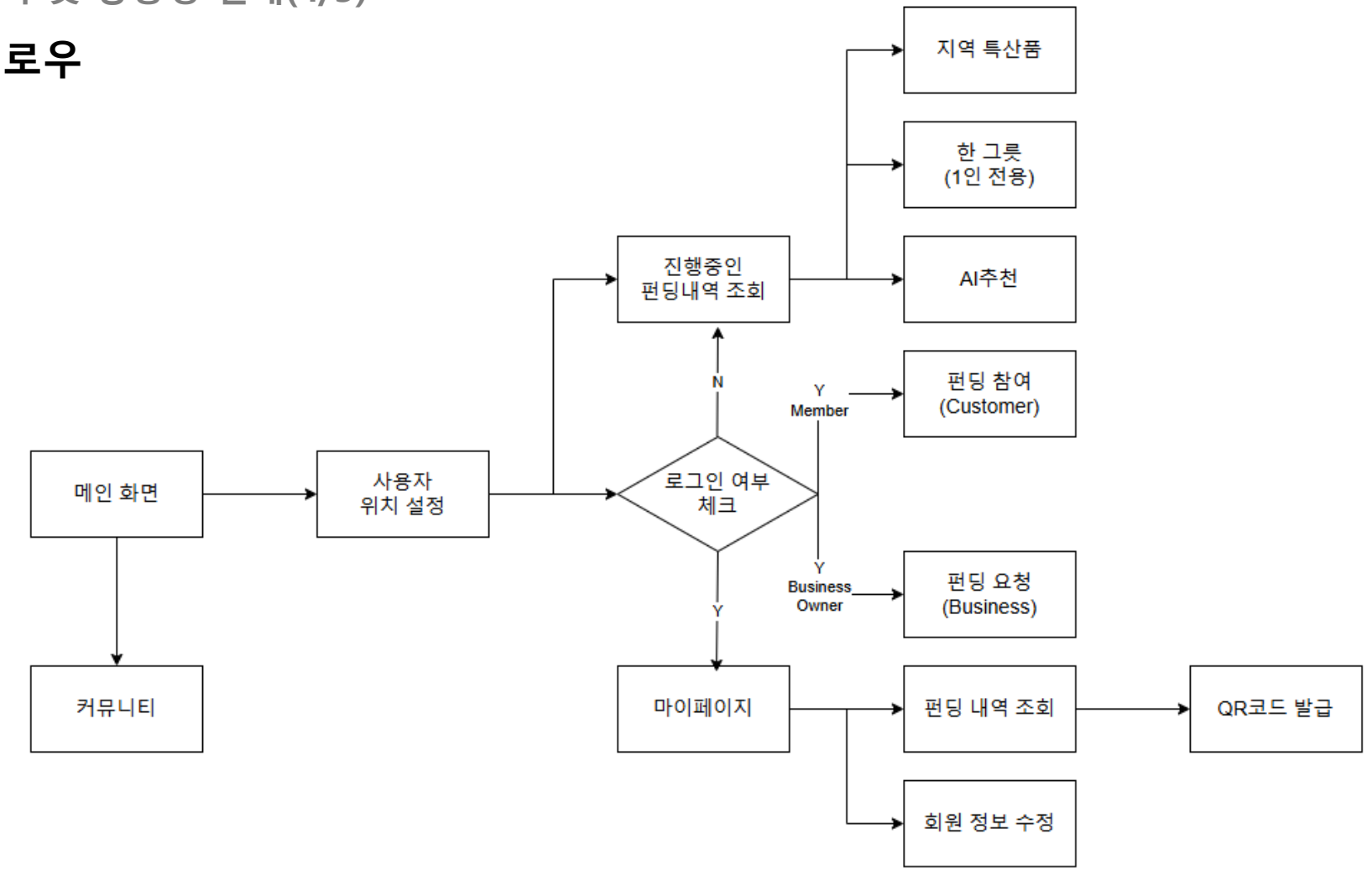
ERD 도출



picktogether

| 서비스 기획 및 방향성 설계(4/5)

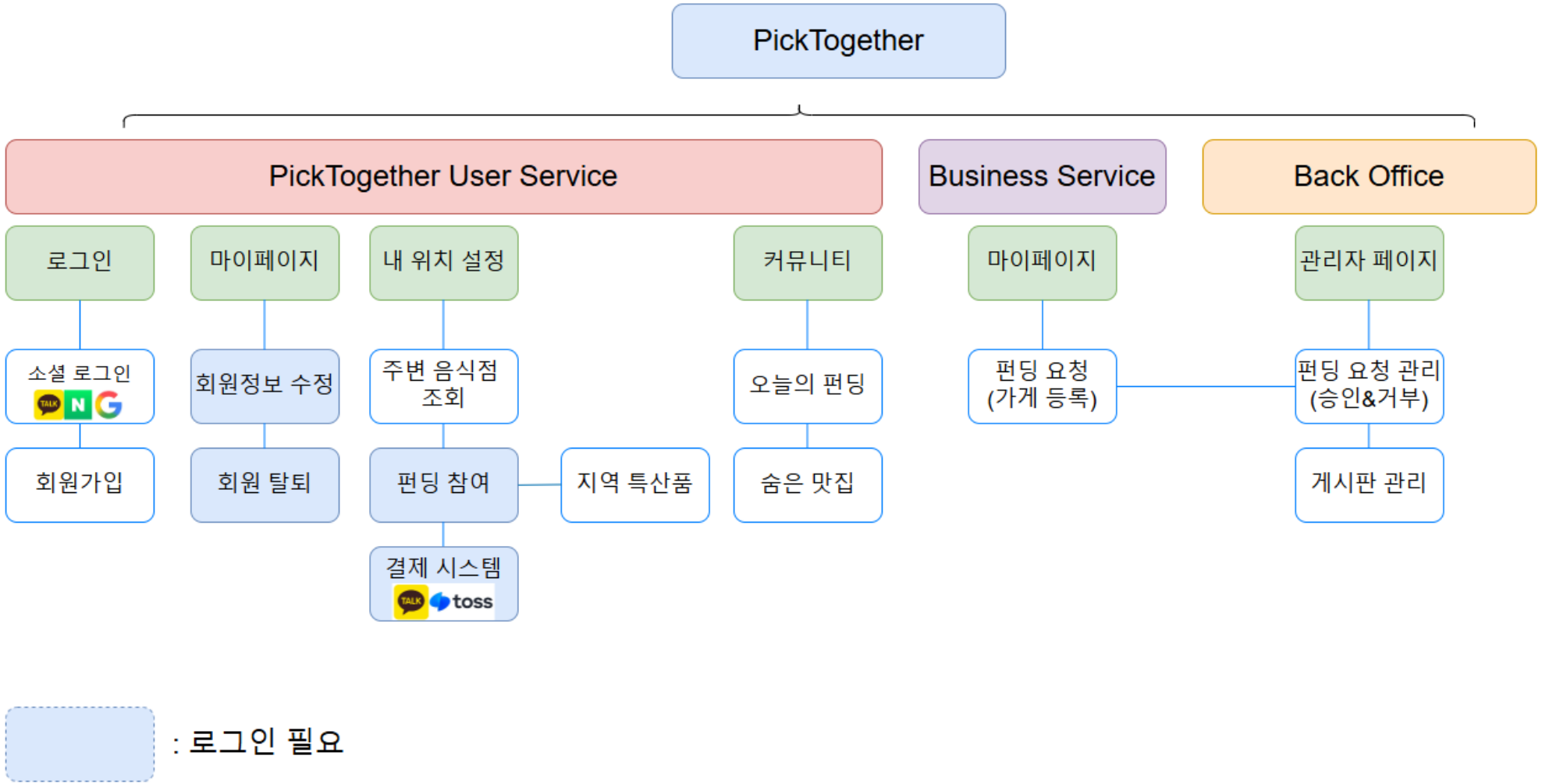
사용자 플로우



picktogether

| 서비스 기획 및 방향성 설계(5/5)

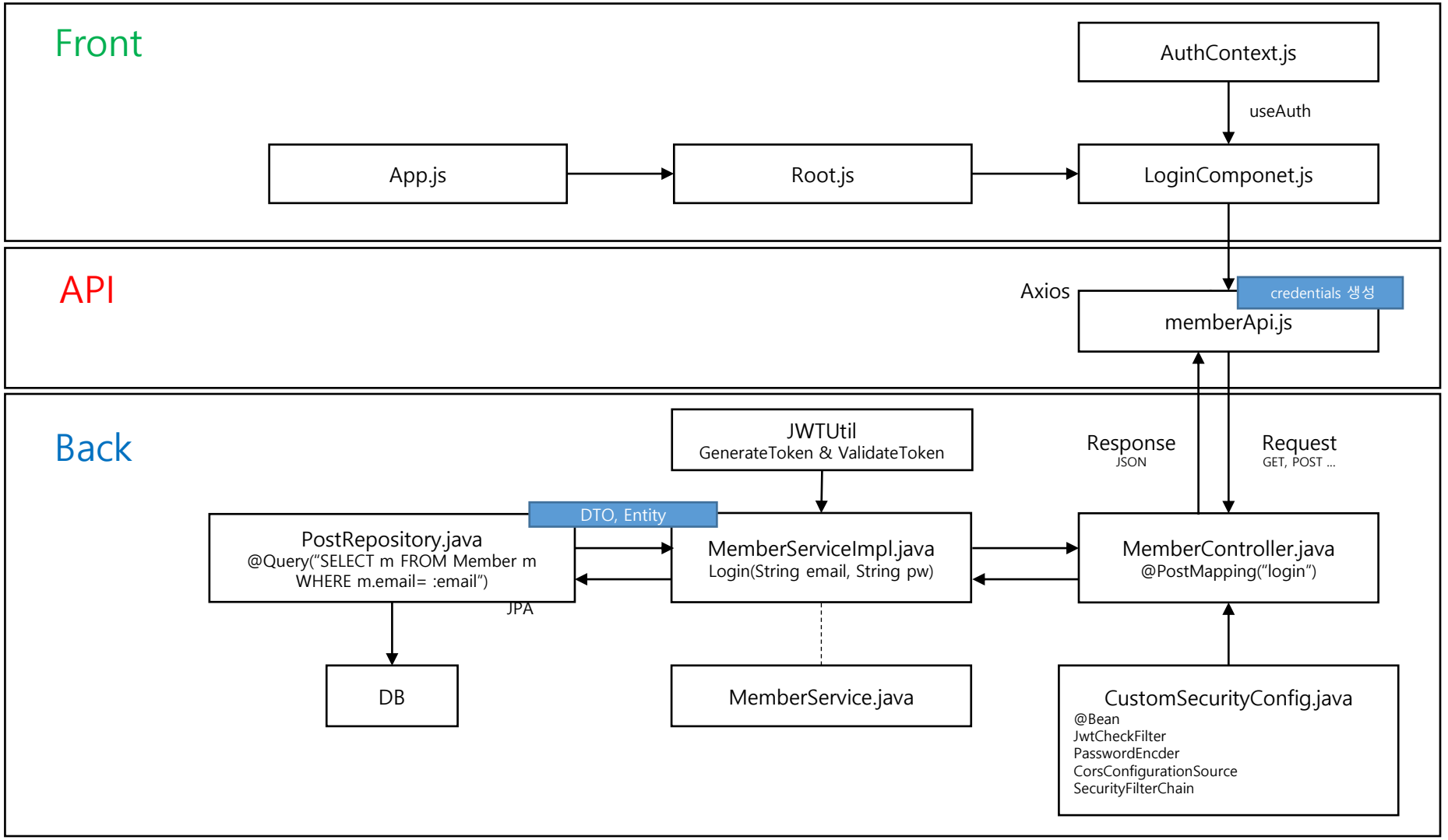
IA 설계



picktogether

| 프로젝트 주요 기능

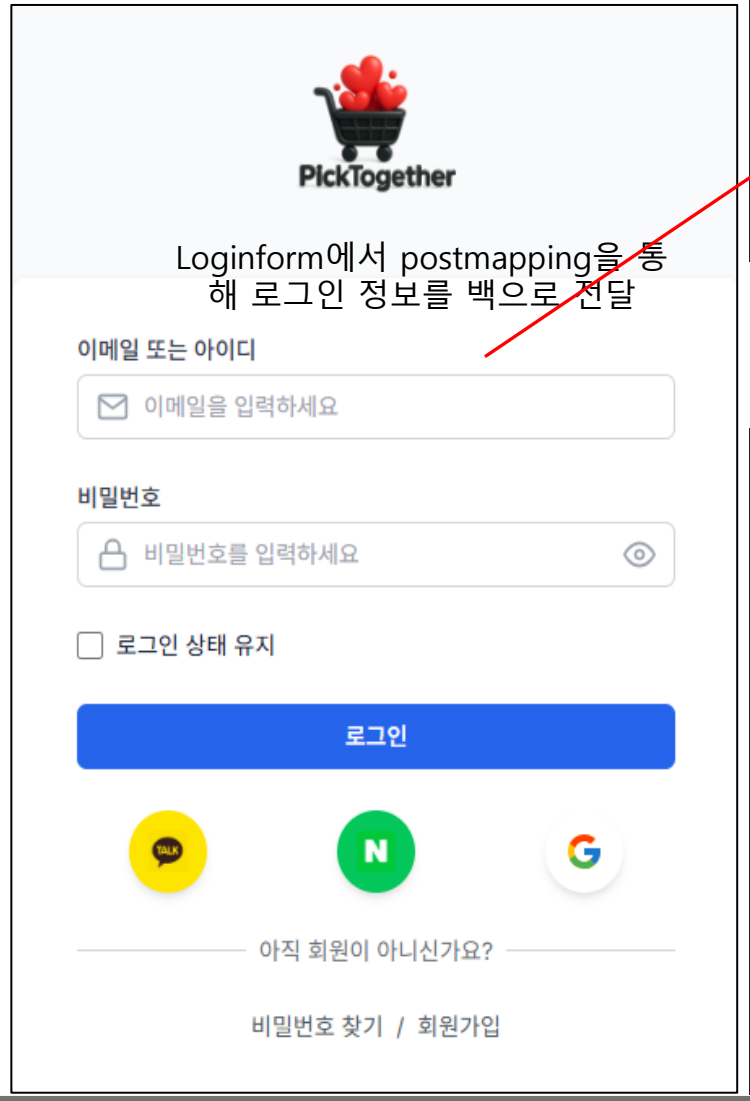
로그인 & 회원가입



picktogether

| 프로젝트 주요 기능

로그인 & 회원가입



Loginform에서 postmapping을 통해 로그인 정보를 백으로 전달

Controller

```
@RequestMapping("/api/member")
@RequiredArgsConstructor
@Log4j2
public class MemberController {

    private final MemberService memberService;

    @PostMapping("/login")
    public ResponseEntity<Map<String, Object>> login(@RequestBody Map<String, String> request) {
        try {
            log.info("로그인 요청: " + request);

            String email = request.get(key:"email");
```

Repository를 통해 DB에 접근해 email & pw 검증 후 JWT토큰 생성

Service

```
@Override
public Map<String, Object> login(String email, String pw) {
    log.info("로그인 시도: " + email);

    // 이메일로 회원 조회
    Member member = memberRepository.findByEmail(email
            .orElseThrow(() -> new RuntimeException(message:"존재하지 않는 회원입니다."));

    // 비밀번호 검증
    if (!passwordEncoder.matches(pw, member.getPw())) {
        throw new RuntimeException(message:"비밀번호가 일치하지 않습니다.");
    }

    // 역할 목록을 문자열 리스트로 변환
    List<String> roleNames = member.getMemberRoleList().stream()
            .map(MemberRole::name)
            .collect(Collectors.toList());

    // JWT 토큰 생성
    Map<String, Object> claims = new HashMap<>();
    claims.put(key:"email", member.getEmail());
    claims.put(key:"pw", member.getPw());
    claims.put(key:"nickname", member.getNickname());
    claims.put(key:"socialType", member.getSocialType());
    claims.put(key:"roleNames", roleNames);

    String accessToken = jwtUtil.generateToken(claims, min:60); // 60분 유효

    // MemberDTO 생성
```

JPA를 활용하여 객체와 데이터베이스 간의 매핑을 자동화

Repository

```
public interface MemberRepository extends JpaRepository<Member, String> {

    @Query("SELECT m FROM Member m WHERE m.email = :email")
    Optional<Member> findByEmail(@Param("email") String email);
```

Database

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conter

email	nickname	pw	social_type	delete_a
aaa@aaa.com	일반회원	$2a$10$1ifh72/i631B6VjIn..gUelN6Ib2OJ4ZZiyS..	NULL	0
admin@picktogether.com	관리자	$2a$10$vQjMHUTjEpm5MzXB1kn20eM5.ol84gIp..	NULL	0
business@picktogether.com	소상공인	$2a$10$3qb4T72hXwbAHWSLdIvAseoaAAUi4o...	NULL	0
ish979797@naver.com	송승찬영	$2a$10$VpXJFXVZcFJ3u/xovCXkH.YA9GAL3mR...	KAKAO	0

비밀번호는 PasswordEncoder를 통해 단방향 해시 함수(ByCrypt)로 변환하여 저장

picktogether

| 프로젝트 주요 기능

로그인 & 회원가입

Registerform에서 postmapping을 통해
회원가입 정보를 백으로 전달

Controller

```java
@PostMapping("/register")
public ResponseEntity<Map<String, Object>> register(@RequestBody Map<String, String> request) {
    try {
        log.info("회원가입 요청: " + request);

        String email = request.get(key:"email");
```

Service

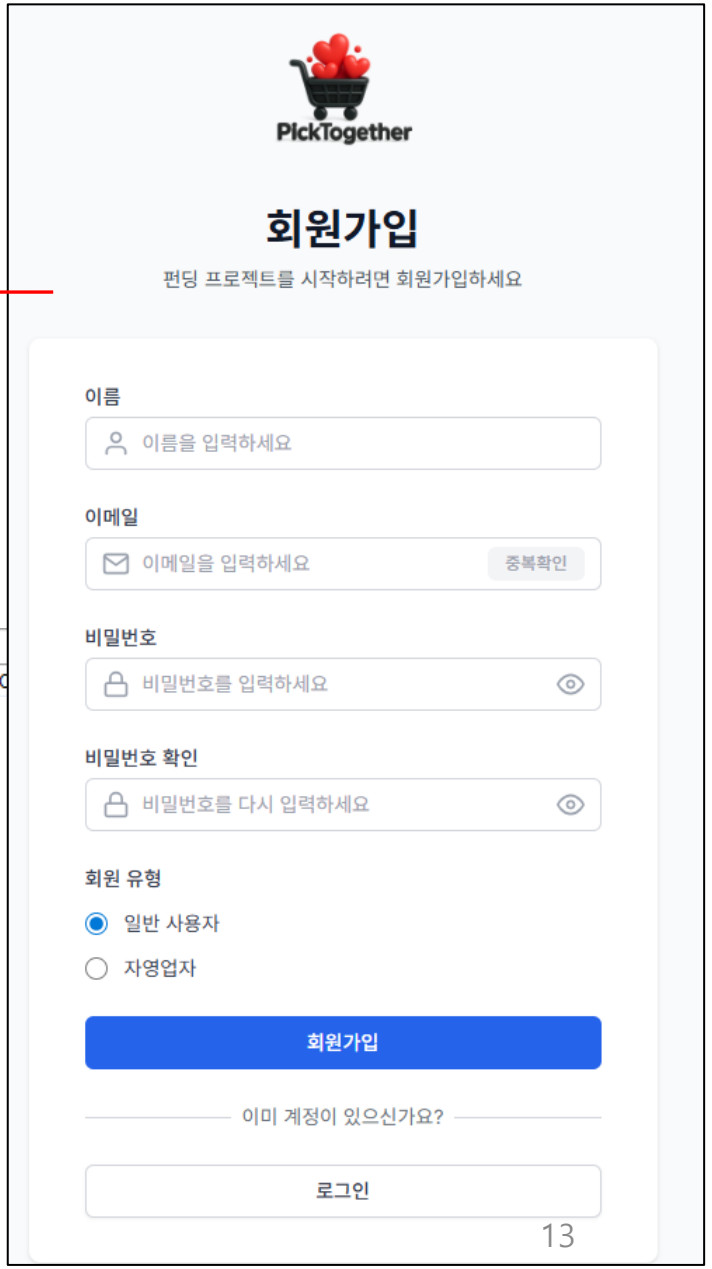
```java
@Override
public Map<String, Object> register(String email, String pw, String nickname, String memberType) {
    log.info("회원가입 시도: " + email + ", 회원유형: " + memberType);

    // 이메일 중복 확인
    if (memberRepository.existsByEmail(email)) {
        throw new RuntimeException(message:"이미 존재하는 이메일입니다.");
    }

    // 비밀번호 암호화
    String encodedPw = passwordEncoder.encode(pw);

    // 회원 생성
    Member member = Member.builder()
            .email(email)
            .pw(encodedPw)
            .nickname(nickname)
            .socialType(socialType:null)  // 일반회원은 null
            .deleteAccount(deleteAccount:0)  // 명시적으로 0으로 설정
            .build();

    // 회원 유형에 따른 역할 설정
    if ("BUSINESS_OWNER".equals(memberType)) {
        member.addRole(MemberRole.BUSINESS_OWNER);
        log.info("자영업자 역할로 회원가입: " + email);
    } else {
        // 기본 역할은 USER
        member.addRole(MemberRole.USER);
        log.info("일반 사용자 역할로 회원가입: " + email);
    }

    // 저장 전에 명시적으로 기본값 설정
    if (member.getDeleteAccount() == null) {
        member.setDeleteAccount(deleteAccount:0);
    }

    // 저장
    memberRepository.save(member);
```

Builder를 통해 Entity생성 후 DB에 저장

email	nickname	pw
aaa@aaa.com	일반회원	$2a$10$1ifh72/i631B6VjIn..gUelN6Ib2C

member_email	member_role_list
ish979797@naver.com	USER
aaa@aaa.com	USER

회원 유형에 따른 member_role 저장



picktogether

| 프로젝트 주요 기능

위치 기반

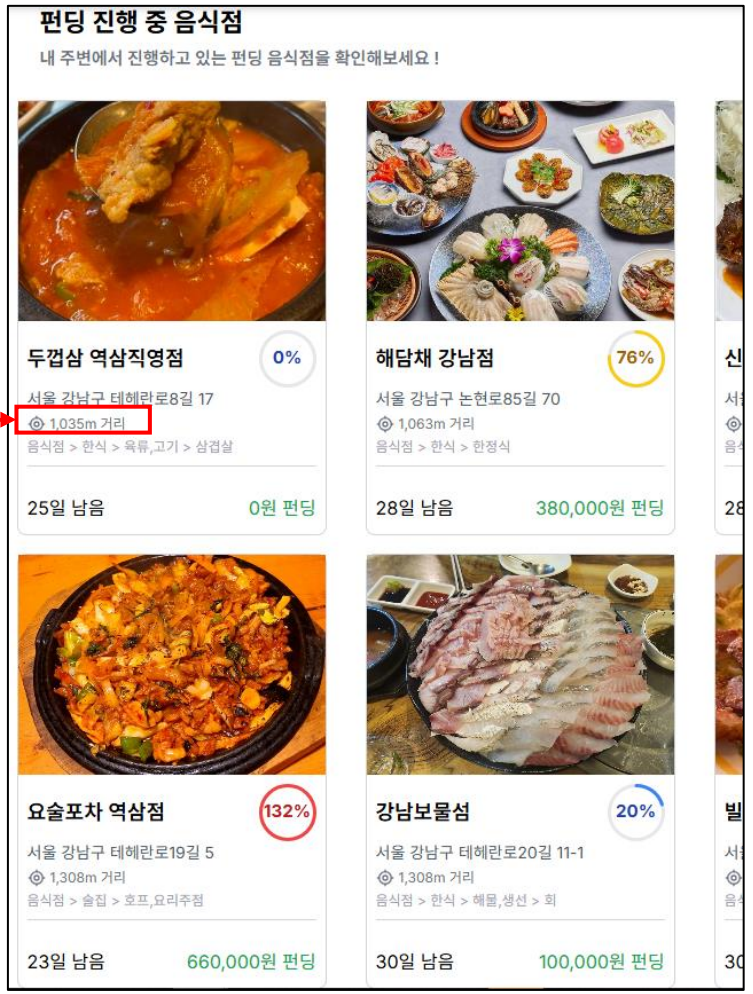
주소 등록시 자동으로 주소목록에 추가

```
// 1) Kakao Maps SDK 로드
useEffect(() => {
  if (window.kakao?.maps) return;

  const script = document.createElement("script");
  script.src = `https://dapi.kakao.com/v2/maps/sdk.js?appkey=${KAKAO_MAP_CONFIG.MAP_API_KEY}&autoload=false&libraries=services`;
  script.async = true;
  script.onload = () => {
    window.kakao.maps.load(() => {
      console.log("Kakao Maps load Complete");
    });
  };
  document.head.appendChild(script);

  return () => {
    if (script.parentNode) script.parentNode.removeChild(script);
```

KakaoMap Api를 활용해
나의 주소지를 설정

```
SELECT
  r.id                                  AS restaurantId,
  r.name                                AS name,
  r.road_address_name                   AS roadAddressName,
  r.place_url                           AS placeUrl,
  r.category_name                       AS categoryName,
  r.funding_amount                      AS fundingAmount,
  r.funding_goal_amount                 AS fundingGoalAmount,
  LEAST(100, GREATEST(0,
      ROUND((r.funding_amount * 100.0) / NULLIF(r.funding_goal_amount, 0))
  ))                                    AS fundingPercent,
  ST_Distance_Sphere(POINT(r.x, r.y), POINT(:lng, :lat)) AS distance,
```

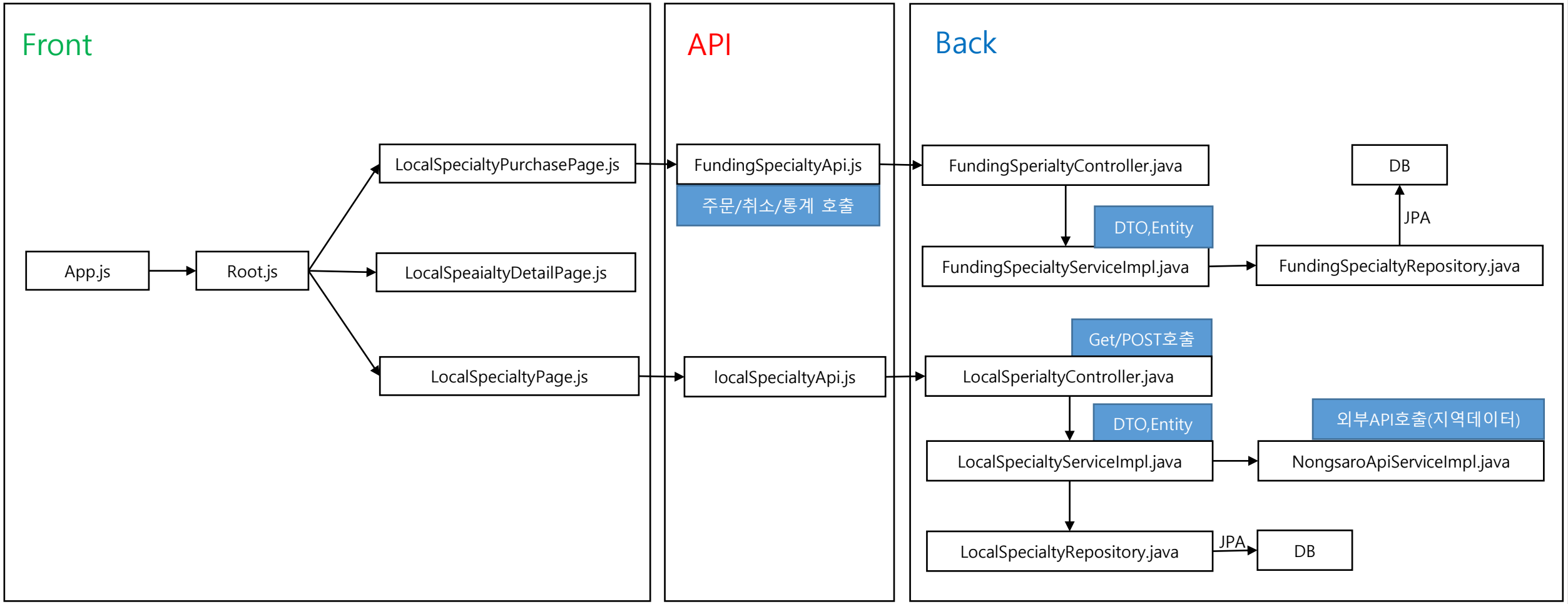
입력한 나의 주소지를 기반으로 위도와 경도를
이용해 해당 음식점까지의 거리를
JPQL으로 계산 후 리턴



picktogether

| 프로젝트 주요 기능

지역 특산품



picktogether

| 프로젝트 주요 기능

지역 특산품

농사로API를 활용해 지역 특산품 데이터를 CSV파일로 변환 후 DB에 저장

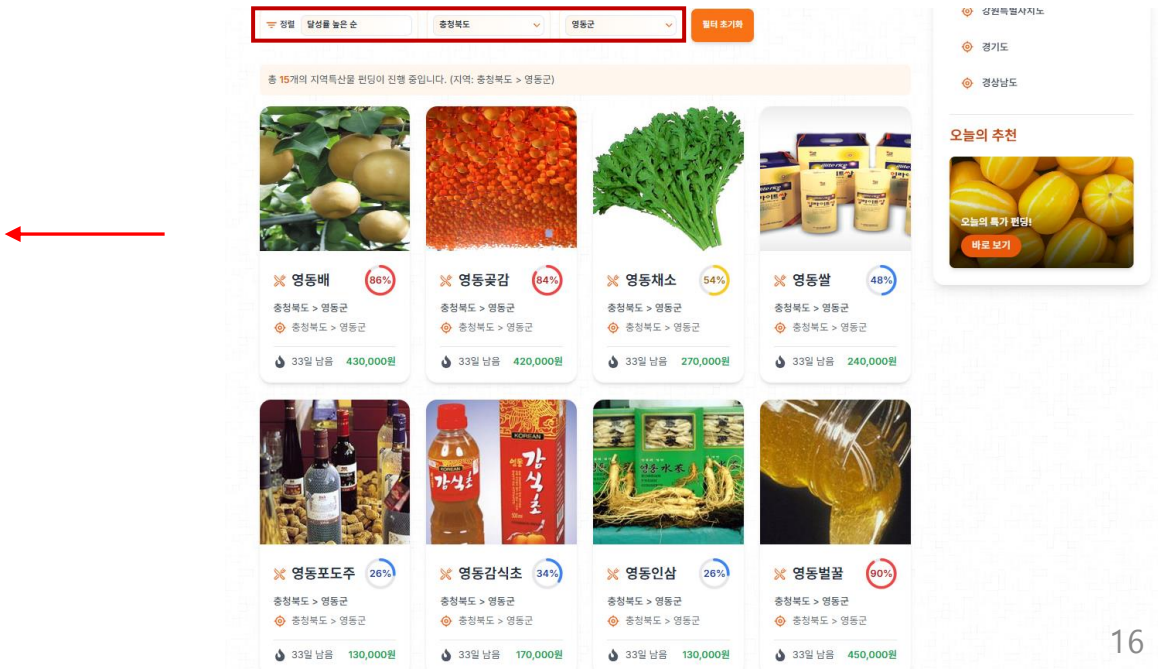
local_specialty_20250818.csv

src > test > java > com > backend > repository > local_specialt

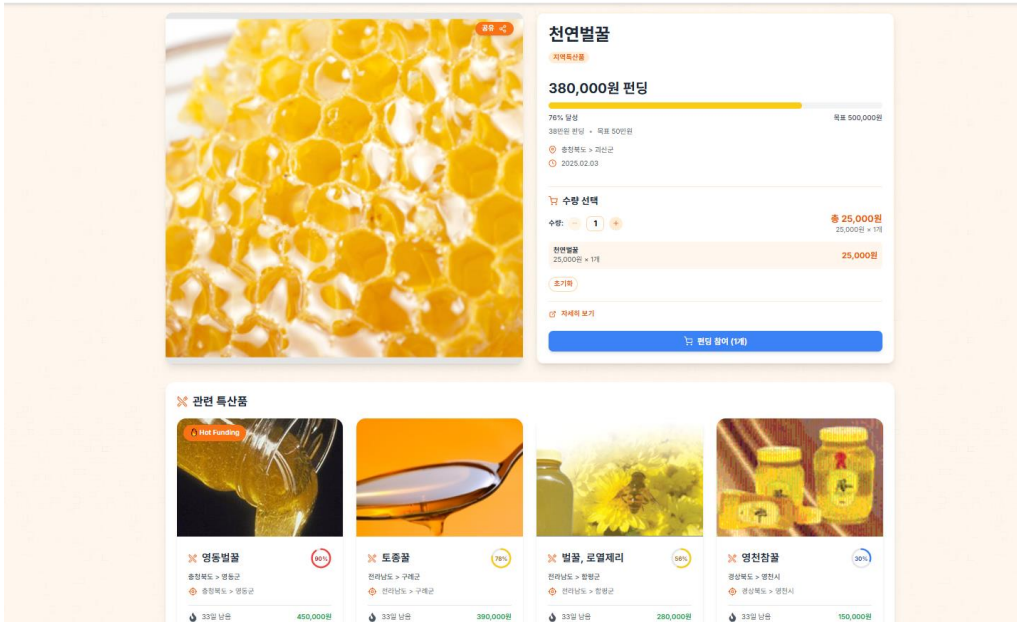
```
1  콘텐츠번호,콘텐츠제목,지역명,이미지URL,서비스일자,링크URL,지
2  257836,철마 무농약 인증쌀,부산광역시 > 기장군,http://www.n
3  257835,장안 검정쌀,부산광역시 > 기장군,http://www.nongsaro
4  257834,옥수수,부산광역시 > 기장군,http://www.nongsaro.go.
5  257833,배,부산광역시 > 기장군,http://www.nongsaro.go.kr/c
6  257832,당근,부산광역시 > 기장군,http://www.nongsaro.go.kr
7  257831,딸기,부산광역시 > 기장군,http://www.nongsaro.go.kr
8  257830,토마토,부산광역시 > 기장군,http://www.nongsaro.go.
```

id	area_code	area_nm	cntnts_no	cntnts_sj	created_at	funding_
1	2671000000	부산광역시 > 기장군	257836	철마 무농약 인증쌀	2025-08-18	150000
2	2671000000	부산광역시 > 기장군	257835	장안 검정쌀	2025-08-18	180000
3	2671000000	부산광역시 > 기장군	229085	기장채소	2025-08-18	270000
4	2671000000	부산광역시 > 기장군	229084	기장흑미	2025-08-18	220000
5	2671000000	부산광역시 > 기장군	229083	기장붕장어	2025-08-18	140000
6	2671000000	부산광역시 > 기장군	229082	기장곰장어	2025-08-18	170000
7	2671000000	부산광역시 > 기장군	229081	기장생갈치	2025-08-18	240000
8	2671000000	부산광역시 > 기장군	229080	철마한우	2025-08-18	230000
9	2671000000	부산광역시 > 기장군	229079	기장멸치	2025-08-18	430000
10	2671000000	부산광역시 > 기장군	229078	기장다시마	2025-08-18	160000
11	2671000000	부산광역시 > 기장군	229077	기장미역	2025-08-18	400000
12	2771000000	대구광역시 > 달성군	229086	구지 오이	2025-08-18	390000
13	2771000000	대구광역시 > 달성군	215051	현풍 단감	2025-08-18	280000
14	2771000000	대구광역시 > 달성군	105115	화원 설화마늘	2025-08-18	380000

지역 특산품 리스트 페이지
펀딩 참여율 & 지역별 필터링



지역 특상품 상세 페이지 연관된 특상품 리스트 추출



picktogether

| 프로젝트 주요 기능

AI 추천

사용자의 주문내역을 시각화 된 차트로 표현

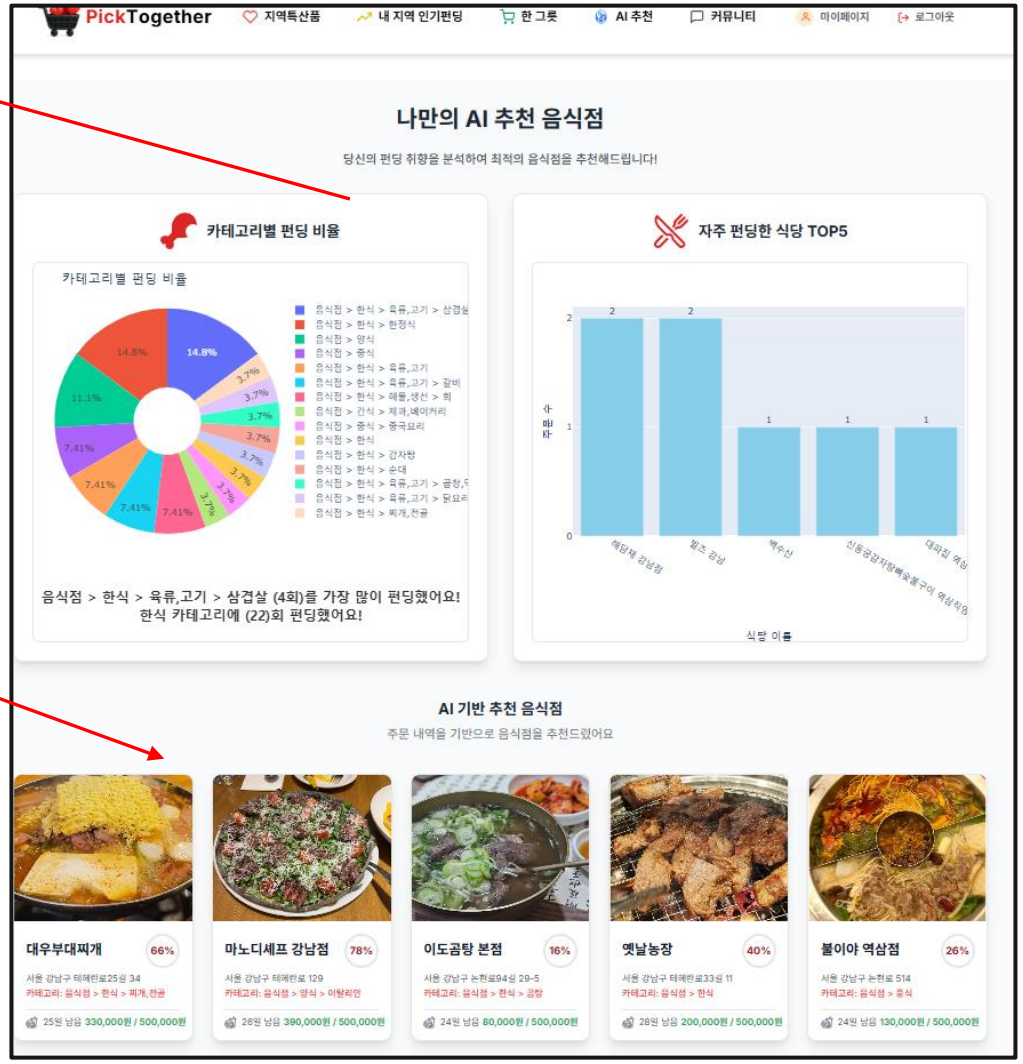
Pytorch모델을 이용해 사용자의 주문 내역 데이터를 학습

```python
# DB에서 주문 기록 불러오기
sql = """
SELECT member_email, restaurant_id, 1 as label
FROM funding
"""
df = pd.read_sql(sql, conn)
conn.close()

# 사용자/아이템 ID를 숫자로 인코딩
user_enc = LabelEncoder()
item_enc = LabelEncoder()
df["user_id"] = user_enc.fit_transform(df["member_email"])
df["item_id"] = item_enc.fit_transform(df["restaurant_id"])

num_users = df["user_id"].nunique()
num_items = df["item_id"].nunique()

# Torch Dataset
users = torch.tensor(df["user_id"].values, dtype=torch.long)
items = torch.tensor(df["item_id"].values, dtype=torch.long)
labels = torch.tensor(df["label"].values, dtype=torch.float32)

class SimpleRec(nn.Module):
    def __init__(self, num_users, num_items, emb_dim=32):
        super().__init__()
        self.user_embed = nn.Embedding(num_users, emb_dim)
        self.item_embed = nn.Embedding(num_items, emb_dim)
        self.fc = nn.Linear(emb_dim*2, 1)

    def forward(self, u, i):
```

```python
# 추천 API
@app.get("/recommend/deep")
def recommend_restaurants(user_email: str):
    if user_email not in user2idx:
        return {"message": "알 수 없는 유저"}

    user_id = torch.tensor([user2idx[user_email]])
    scores = []
    for item, idx in item2idx.items():
        item_id = torch.tensor([idx])
        with torch.no_grad():
            score = model(user_id, item_id).item()
        scores.append((item, score))
```

학습한 데이터를 기반으로 음식점을 점수화하여 상위 10개 리스트 출력

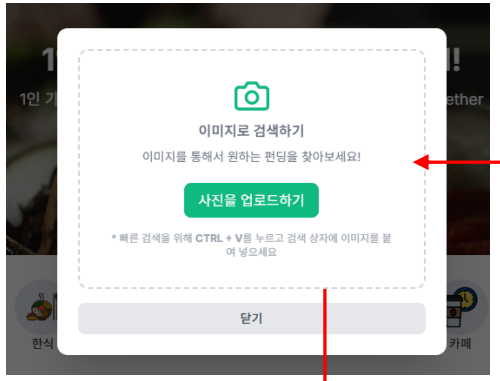
```
[Epoch 5/10] Loss: 0.5452
[Epoch 6/10] Loss: 0.5031
[Epoch 7/10] Loss: 0.4636
[Epoch 8/10] Loss: 0.4262
[Epoch 9/10] Loss: 0.3905
[Epoch 10/10] Loss: 0.3565
Training: 100%

=== 추천 음식점 TOP 10 ===
- 41 (score=0.9353)
- 23 (score=0.8934)
- 40 (score=0.8749)
- 25 (score=0.8739)
- 29 (score=0.8525)
```



picktogether

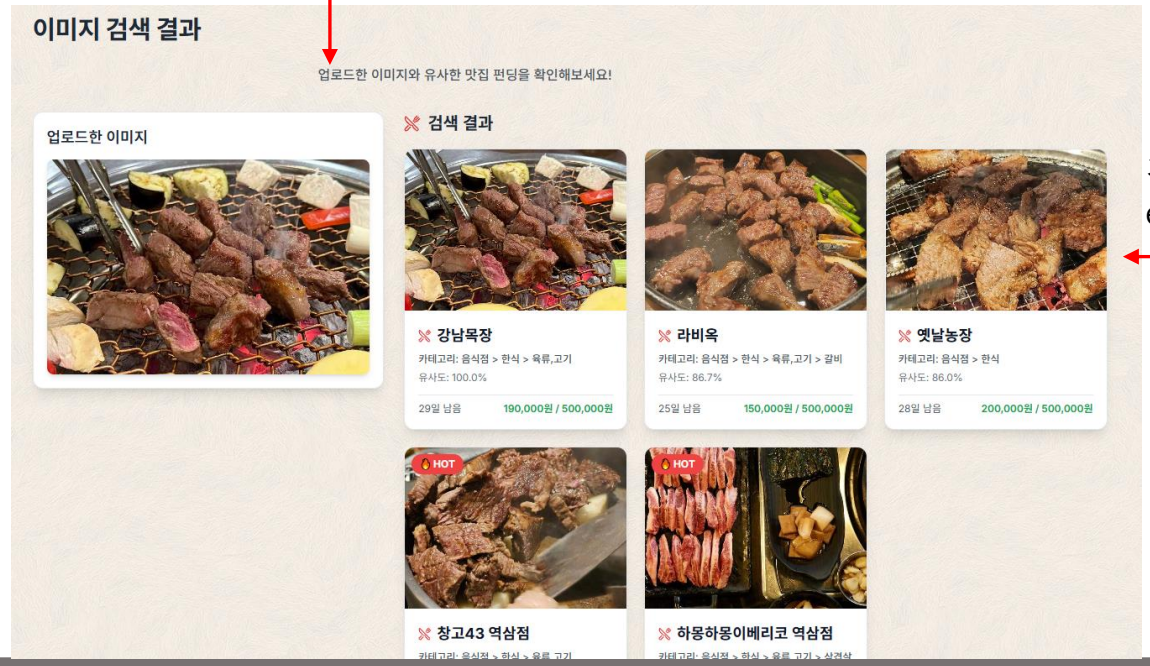
| 프로젝트 주요 기능

이미지 분석




```python
@app.post("/search/image")
async def search_by_image(file: UploadFile = File(...)):
    image = Image.open(file.file)
    query_vec = clip_model.encode(image, convert_to_numpy=True)  # float32

    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT restaurant_id, image_url, embedding FROM restaurant_image_embed
    rows = cursor.fetchall()
    conn.close()

    # 유사도 계산
    results = []
    for row in rows:
        restaurant_id = row[0]
        image_url = row[1]
        emb = np.array(json.loads(row[2]), dtype=np.float32)  #  dtype 맞추기
        score = util.cos_sim(query_vec, emb).item()
        results.append({
            "restaurant_id": restaurant_id,
            "image_url": image_url,
            "score": score
```

업로드한 이미지를 embedding하여 기존에
저장되어있던 벡터값과 비교하여 유사도 측정



검색할 이미지를
embedding하여 벡터값으로 저장

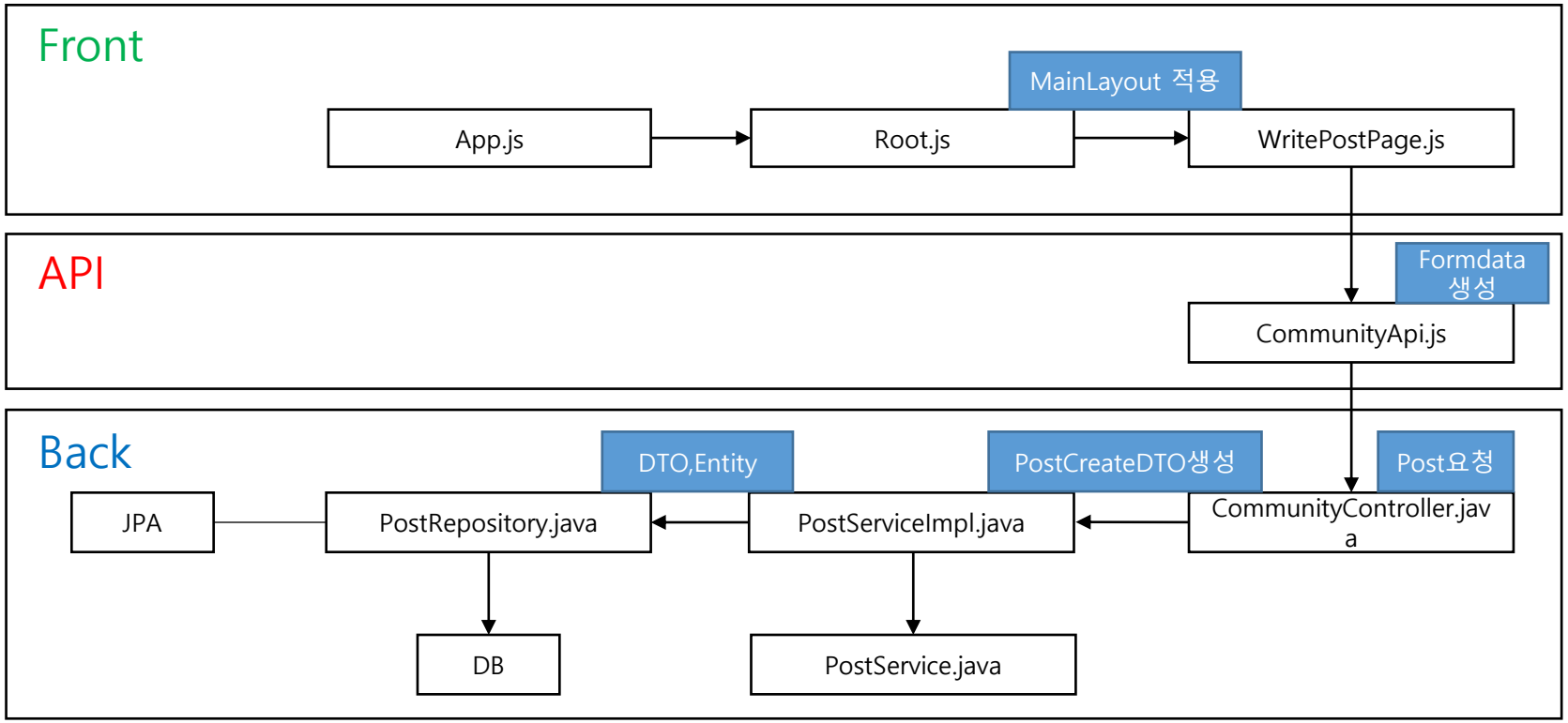
```python
from PIL import Image
from sentence_transformers import SentenceTransformer

# CLIP 모델 로드
model = SentenceTransformer("clip-ViT-B-32")

embeddings = []
folder = "./images"

# 파일명 정렬 (예: 1.jpg ~ 45.jpg 순서대로)
files = sorted(
    [f for f in os.listdir(folder) if f.lower().endswith((".jpg", ".jpeg", ".png", ".JPG"))],
    key=lambda x: int(os.path.splitext(x)[0])  # '1.jpg' → 1
)

for file in files:
    path = os.path.join(folder, file)
    img = Image.open(path)
```

id	embedding	image_url	restaurai
1	[0.3453162908554077, 0.3559839129447937, ...	1.JPG	1
2	[-0.11513492465019226, 0.523566722869873...	2.JPG	2
3	[0.012780934572219849, 0.188527181744575...	3.JPG	3
4	[0.17855489253997803, 0.2846080064773559...	4.JPG	4
5	[0.44429028034210205, 0.525867760181427, ...	5.JPG	5
6	[-0.639532208442688, 0.1231064647436142, ...	6.JPG	6
7	[0.2740914821624756, -0.0162894576787948...	7.JPG	7
8	[-0.24770885705947876, 0.581369936466217...	8.JPG	8
9	[-0.30124783515930176, 0.624628067016601...	9.JPG	9
10	[-0.34646478295326233, 0.320001304149627...	10.JPG	10
11	[-0.03946602717041969, 0.915158569812774...	11.JPG	11
12	[0.3253169655799866, 0.6832049489021301, ...	12.JPG	12
13	[-0.19920113682746887, -0.09301383048295...	13.JPG	13
14	[-0.2272803783416748, 0.4343204498291015...	14.JPG	14
15	[-0.08029647171497345, -0.07529798150062...	15.JPG	15

picktogether

| 프로젝트 주요 기능

커뮤니티



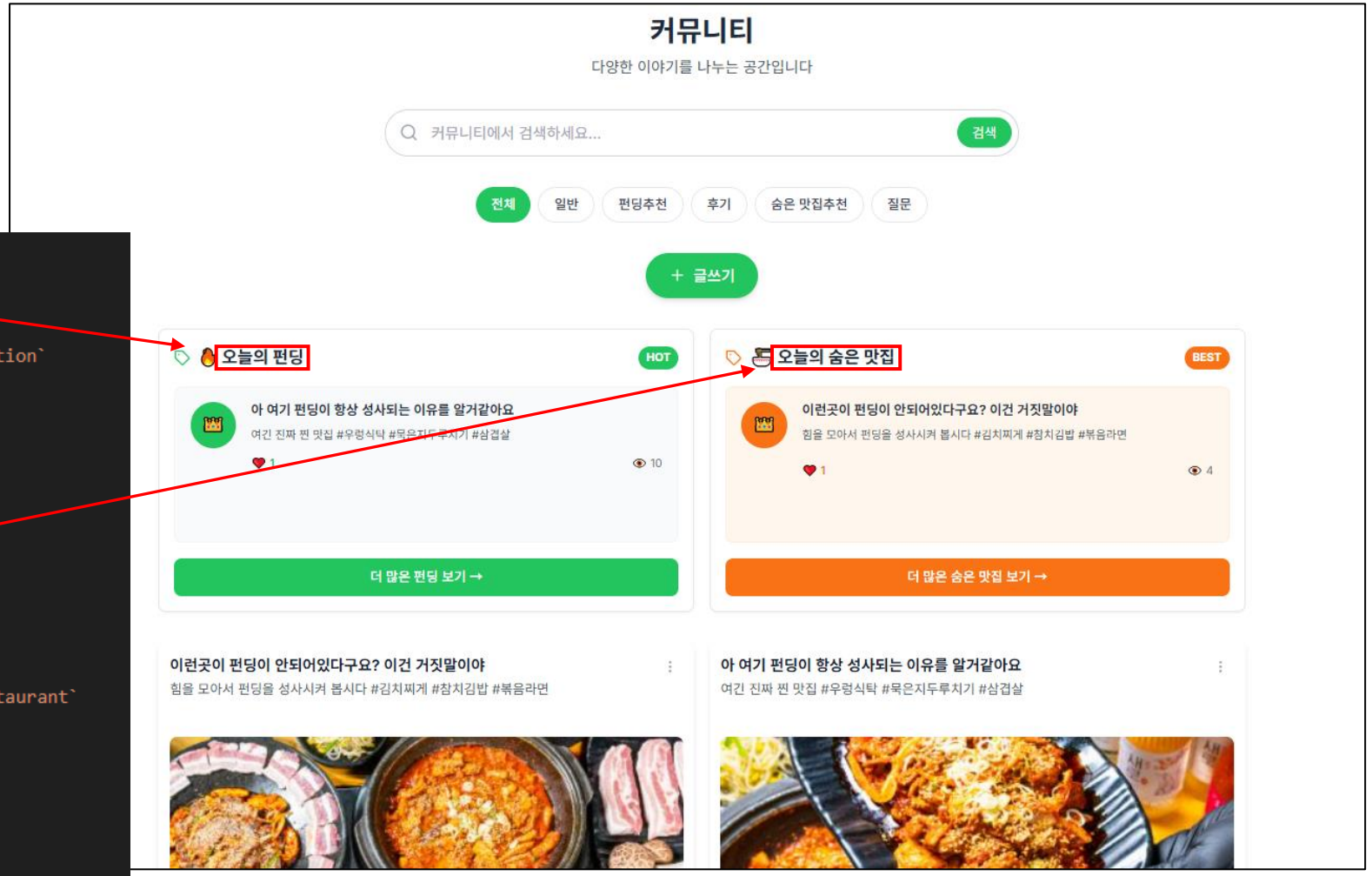
picktogether

| 프로젝트 주요 기능

커뮤니티

게시판 내부에서 조회수를 기반으로 오늘의 펀딩과
오늘의 숨은 맛집 선정
선정된 게시글은 커뮤니티 상단에 노출

```
// 오늘의 추천 가져오기
getTodayRecommendation: async () => {
  try {
    const response = await axios.get(
      `${API_SERVER_HOST}${API_BASE_URL}/posts/today-recommendation`
    );
    return response.data;
  } catch (error) {
    console.error("오늘의 추천 조회 실패:", error);
    throw error;
  }
},

// 오늘의 숨은 맛집 가져오기
getTodayHiddenRestaurant: async () => {
  try {
    const response = await axios.get(
      `${API_SERVER_HOST}${API_BASE_URL}/posts/today-hidden-restaurant`
    );
    return response.data;
  } catch (error) {
    console.error("오늘의 숨은 맛집 조회 실패:", error);
    throw error;
  }
},
};
```

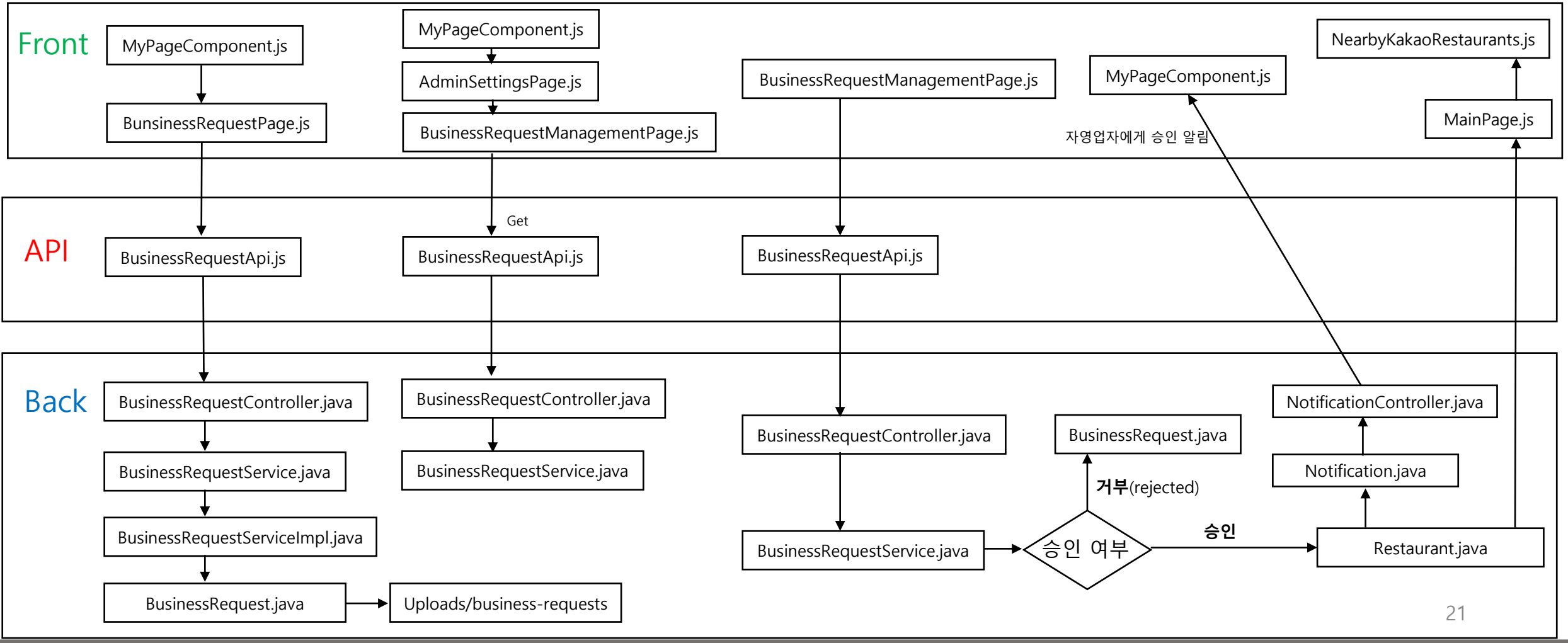


picktogether

| 프로젝트 주요 기능

관리자 페이지

API 호출: GET /api/restaurants/nearby
↓
RestaurantController.java의 getNearby() 메서드
↓
RestaurantService.java의 getNearby() 메서드
↓
승인된 가게들만 필터링하여 반환
↓
메인 페이지에 승인된 가게들 표시



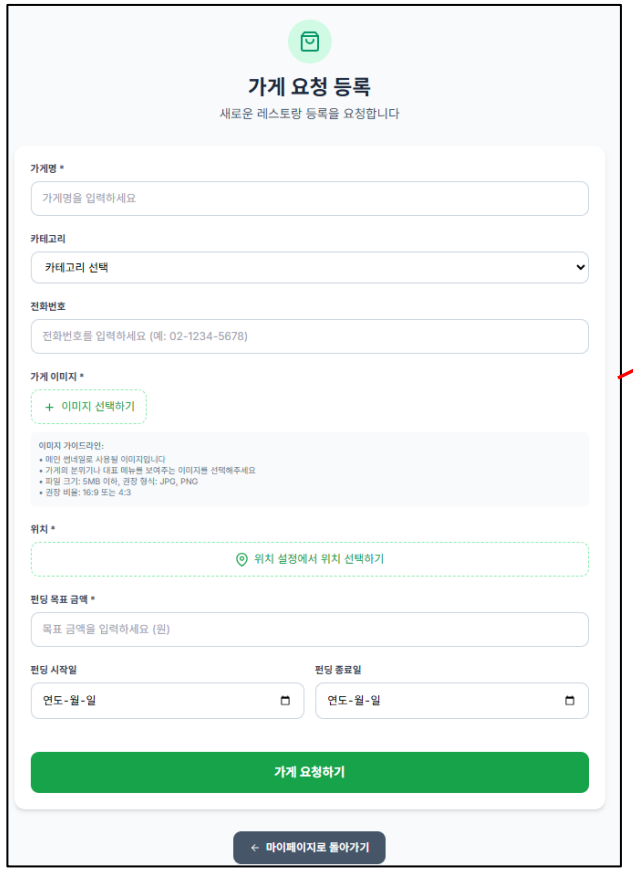
picktogether

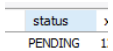
| 프로젝트 주요 기능

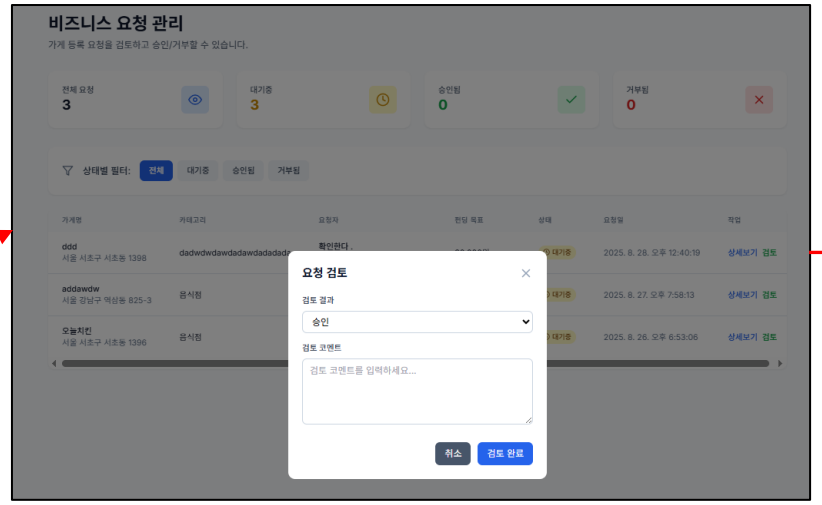
관리자 페이지

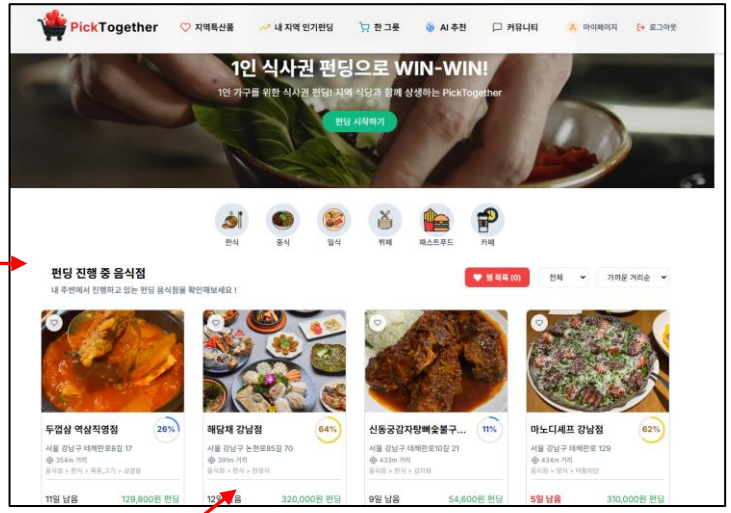
가게요청 상태

```
PENDING(description:"대기중"),
APPROVED(description:"승인됨"),
REJECTED(description:"거부됨");
```









레스토랑 도메인

id	category_name	distance	funding_amount	funding_end_date	funding_goal_amount	funding_start_date	name	phone	place_url	road_address_nam
1	음식점 > 중식 > 중국요리	880	310000	2025-08-25	500000	2025-08-26	대가방 본점	02-544-6336	http://place.map.kakao.com/7815078	서울 강남구 봉은사

요청 승인후 레스토랑 도메인에
업데이트되어 메인에 반영

가게요청 도메인

id	category_name	created_at	funding_end_date	funding_goal_amount	funding_start_date	image_url	name	phone	place_url
1	음식점	2025-08-26 18:53:06.951842	2025-08-29	9047000	2025-08-15	uploads/business-requests/76637408-b62d-467...	오늘치킨	030646	https://map.kakao.com/link/map/37.496427699...

picktogether

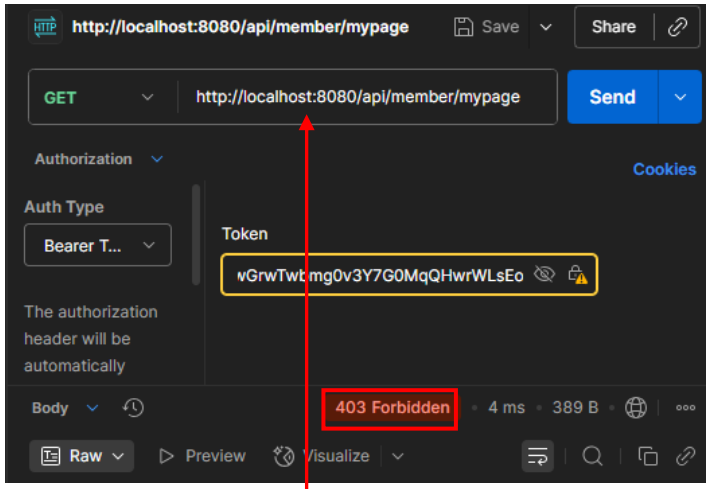
| 문제해결

Security

문제 발생

로그인 후 토큰값을 이용해 Postman으로
Controller 테스트를 진행했을때 403 에러가 문제 발생

Postman



MemberController

```
@GetMapping("/mypage")
public ResponseEntity<MemberDTO> getCurrentMember(Authentication authentication
    // 인증 객체에서 username(email) 꺼내기
    String email = authentication.getName();
    MemberDTO member = memberService.getMemberByEmail(email);

    if (member == null) {
        return ResponseEntity.notFound().build();
    }
```

발생 원인

JwtCheckFilter

```
@Override
protected boolean shouldNotFilter(HttpServletRequest request) {
String path = request.getRequestURI();

if (request.getMethod().equalsIgnoreCase(anotherString:"OPTIONS") || "/favicon.ico".equals(path)) {
    return true; // 예외
}
if (path.startsWith(prefix:"/api/member/")) return true;
```

shouldNotFilter에 api/member/ 경로를 등록해 놓아서
SecurityContextHolder 안에Authentication 객체가 세팅되지 않음

CustomSecurityConfig

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    log.info(message:"----------------------security config----------------------------");

    http
        .cors(cors -> cors.configurationSource(corsConfigurationSource()))
        .csrf(csrf -> csrf.disable())
        .sessionManagement(sm -> sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .formLogin(form -> form.disable())
        .authorizeHttpRequests(auth -> auth
            .requestMatchers(HttpMethod.OPTIONS, ...patterns:"/**").permitAll()
            .requestMatchers(...patterns:"/uploads/**").permitAll() // 이미지 파일 접근 허용
            .requestMatchers(...patterns:"/api/member/mypage").authenticated()
            .requestMatchers(...patterns:"/api/member/locations").authenticated()
```

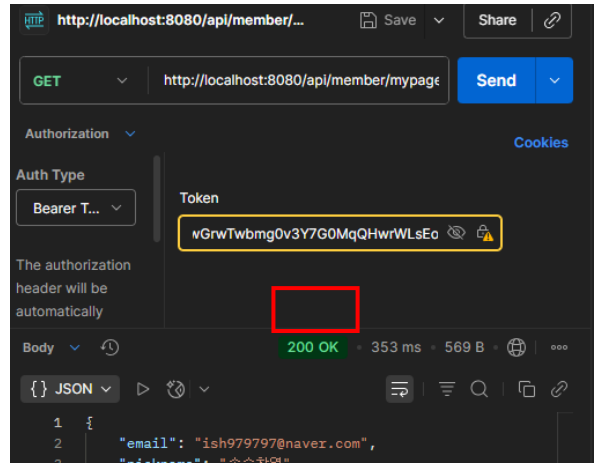
해결 방식

JwtCheckFilter에 통과시킬 경로를
세분화하여 해결

JwtCheckFilter

```
// 공개 엔드포인트만 명시
String[] PUBLIC_PREFIXES = {
    "/api/member/login",
    "/api/member/register",
    "/api/member/check-email",
    "/api/member/delete-account",
    "/api/member/kakao",
    "/api/member/google",
    "/api/member/naver",
    "/api/test/",
    "/api/products/view/",
    "/api/restaurants",
    "/api/community",
    "/api/local-specialty",
    "/api/local-specialties", // 새로운 지역특산물 API 추가
    "/api/funding-specialty", // 지역특산품 구매 API 추가
    "/uploads" // 이미지 파일 접근 허용
};

for (String prefix : PUBLIC_PREFIXES) {
    if (path.startsWith(prefix)) {
        return true; // 공개 → 필터 스킵
    }
```

Postman



picktogether

| 문제해결

JPA

문제 발생

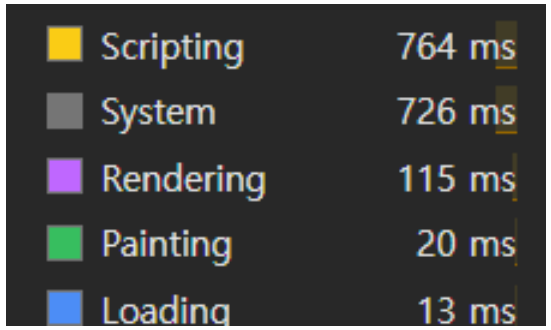
음식점 descriptions을 조회하는 과정에서
N+1문제 발생

```
Hibernate:
    select
        rd1_0.id,
        rd1_0.description,
        rd1_0.restaurant_id
    from
        restaurant_description rd1_0
    where
        rd1_0.restaurant_id=?
Hibernate:
    select
        rd1_0.id,
        rd1_0.description,
        rd1_0.restaurant_id
    from
        restaurant_description rd1_0
    where
        rd1_0.restaurant_id=?
Hibernate:
    select
        rd1_0.id,
        rd1_0.description,
        rd1_0.restaurant_id
    from
        restaurant_description rd1_0
    where
        rd1_0.restaurant_id=?
```

발생 원인

```
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "restaurant_id", nullable = false)
private Restaurant restaurant;
```

엔티티를 처음 조회할때 LAZT처리로 인해 연관객체를
불러오지 않지만 실제로 그 연관객체에 접근할때 각
Resutarnt마다 별도의 select 쿼리문이 발생

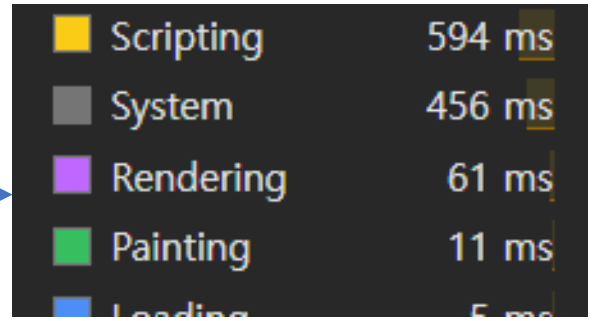


해결 방식

```
@EntityGraph(attributePaths = {"descriptions"})
List<Restaurant> findAll();
```

EntityGraph를 이용해
N+1문제 해결

```
Hibernate:
    select
        rd1_0.id,
        rd1_0.description,
        rd1_0.restaurant_id
    from
        restaurant_description rd1_0
    where
        rd1_0.restaurant_id=?
2025-08-28T20:26:29.114+09:00  INFO 15420 --- [backend] [
```



Rendering시간 40%이상 감소

picktogether

| 느낀점

구분	내용
잘한 점	1차 프로젝트의 부족하다 느낀 점을 개선하여 초반 설계를 잘 잡아 놓아서 각자 구현한 기능들을 합쳤을 때 큰 오류 문제들이 없이 서버가 잘 돌아갔습니다. 챗봇 AI기능을 추가하여 사용자 결제 과정 편의성을 높였고 카카오 로그인,카카오 맵 API를 활용하여 로그인 편의성 및 간편한 주소지 설정이 가능하도록 개선했습니다. 1시간 캐시 설정으로 반복 요청 시 성능 향상하였습니다.
어려움 극복	공공데이터포털을 사용하여 API를 가져오는 상황에서 CSV파일로 변환을 시킬때 기존API형태가 XML형식으로 저장이돼어있어서 변환이 어려웠지만 팀원에 적극적인 도움을 통해 원하는 API를 가져올수있었습니다. Securtity관련 코드로 인해 데이터를 받아오지 못하는 에러를 로그를 찍어가며 발견하고 Security관련 기능을 맡은 팀원들과 함께 프로세스를 확인하여 권한을 받는 방법을 숙지하였고 문제를 해결했습니다.
아쉬웠던 점	여러 기능들을 구현하면서 플랫폼 상 많은 이미지들을 가져와야 했었는데 이미지 폴더를 제대로 관리해놓지않아 원하는 이미지를 찾는데 시간소모가 걸렸던 것이 아쉬운점으로 남았습니다
깨달은 점	개발과정 중 해결되지 않는 문제들은 최대한 빠르게 팀원들에게 도움을 요청해야 시간을 절약할 수 있다는 것을 알게 되었고, 알지못한 라이브러리들을 접해보면서 기능에 새로움을 발견하여 다음 프로젝트에는 더 좋은 결과물을 만들어 내야겠다고 깨달았습니다.

감사합니다.