III. 공개SW프로젝트 결과보고서

프로젝트명 :

LLM기반 사용자 대화형 검색 및 결제 방식 추천 서비스

교과목명	공개SW프로젝트
담당교수	송수환
팀 장	박주영
팀 원	강동경
	고유지
	서형선
	신유진

시연 영상	프로젝트 전체 시연 영상 https://youtu.be/OgTnuBROMvE ETL Pipeline 시연 영상 https://youtu.be/LxLEkmTSveA
배포 URL	https://github.com/dongguk-team3/team3-project
테스트계정	

□ 결과보고서 요약

프로젝트 추진배경 및 필요성	카드사·통신사·가맹점별 할인 혜택은 매우 다양하지만 정보가 분산되어 있어 사용자는 자신이 보유한 카드나 멤버십을 활용해 받을 수 있는 혜택을 정확히 파악하기 어렵다. 특히 매장 선택 과정에서 위치, 가격, 분위기 등 다양한 조건을 고려하면서 동시에 할인 적용 여부까지 비교하기란 현실적으로 부담이 크다. 이러한 불편함을 해소하기 위해 자연어 기반 대화 방식으로 사용자의 의도를 이해하고 사용자의 보유 혜택 정보를 기반으로 최적의 매장과 할인 결제 방식을 찾아주는 서비스가 필요하다.
프로젝트 최종목적 및 세부 목표	본 프로젝트의 최종 목적은 사용자가 사전에 입력한 카드·통신사·멤버십 정보를 기반으로 가장 높은 혜택을 받을 수 있는 매장과 결제 방식을 LLM과의 대화형 인터페이스를 통해 안내하는 서비스 구축이다. 이를 위해 LLM 기반 자연어 질의 응답 및 의도 파악, 위치 기반 매장 검색과 할인 정보의 통합, 사용자 프로필 기반 혜택 계산 로직 구현, RAG 기술을 활용한 추천 결과의 신뢰성 확보, Flutter 기반 모바일 앱 개발, MCP 기반 마이크로서비스 연동을 세부 목표로 설정하여 수행했다.
기대효과	본 서비스는 사용자가 보유한 혜택을 자동으로 매칭해 제시함으로써 최적의 할인 혜택을 지원해 합리적인 소비를 촉진하고 경제적 효용성을 높인다. 또한 LLM 기반 대화형 검색, 외부 API 연동, 혜택 정산 로직, 마이크로서비스 아키텍처 구축 등 최신 기술이 통합된 서비스로서 서비스 확장과 재사용이 용이한 기술적 기반을 확보했다는 점에서 의미가 있다. 향후 멤버십·제휴 포인트·구독 서비스 등 데이터 수집 범위 확장을 통해 금융·쇼핑·여행 등 다양한 산업으로 발전할 잠재력을 갖추고 있으며 마켓플레이스·제휴 마케팅·핀테크 영역과 연계할 경우 실질적 사업화 가능성도 기대된다.
키워드 (Keyword) (국문)	대화형 검색 할인 혜택 추천 사용자 프로필 기반 추천 LLM 마이크로서비스
키워드 (Keyword) (영문)	Conversational Search Discount Recommendation User Profile-Based Recommendation Large Language Model Microservices Architecture

목 차

제 1 장 프로젝트 과제의 필요성

제 1 절 과제 개요 및 추진 배경

1. 현황 및 문제의식

가. 할인 혜택 정보의 분산

현대 소비자는 카드사, 통신사, 멤버십, 제휴 구독 서비스 등 다양한 채널에서 할인 혜택을 제공받고 있다. 그러나 각 플랫폼이 독립적으로 운영되고 있어 혜택 정보를 종합적으로 파악하기 어렵다. 예를 들어 외식·카페 이용 시 사용자는 지도 앱에서 매장을 찾은 뒤 카드사 앱에서 제휴 여부를 확인하고, 리뷰 플랫폼에서 분위기와 만족도를 검토해야 한다. 이러한 분산된 구조는 소비자의 시간과 노력을 요구하며, 결과적으로 적용 가능한 혜택을 놓치거나 활용하지 못하는 상황을 반복적으로 유발한다. 이는 소비자가 얻을 수 있는 경제적 혜택이 소비 단계에서 실질적으로 적용되지 못하는 구조적 문제를 의미한다.

나. 기존 검색 서비스의 한계

기존의 키워드 기반 검색 엔진은 단일 조건을 조회할 때는 충분히 유용하지만 사용자가 여러 요구사항을 동시에 제시하는 상황에는 적절히 대응하지 못한다. 예를 들어 “강남에서 조용한 분위기의 카페 중 내가 가진 신한카드로 할인되는 곳”과 같이 분위기(정성적 요소)와 할인·위치(정량적 요소)를 함께 고려해야 하는 요청은 현재 검색 시스템에서 정확히 처리되기 어렵다. 이로 인해 사용자는 다양한 플랫폼을 오가며 직접 정보를 비교·선별해야 하고, 이러한 반복적인 과정은 사용자 경험을 저하시킬 뿐만 아니라 소비 과정의 효율성도 크게 떨어뜨린다.

2. 과제 필요성

사용자가 보유한 혜택 정보를 손쉽게 활용할 수 있도록 카드·통신사·멤버십 정보를 기반으로 최적의 할인 결제 방식을 자동으로 매칭해 주는 서비스가 요구된다. 현재는 매장을 찾고 할인 여부를 확인하기 위해 여러 플랫폼을 번갈아 확인해야 하는 번거로움이 존재하며 이러한 과정은 시간 소비뿐만 아니라 혜택을 놓치는 문제로 이어질 수 있다. 따라서 위치 정보와 할인 정보를 하나의 시스템에서 확인할 수 있도록 통합하고 사용자가 보유한 혜택 정보를 미리 입력해 두면 별도의 설정 없이도 가장 큰 혜택을 자동으로 안내받을 수 있도록 지원하는 방식은 소비자의 부담을 크게 줄일 수 있다.

또한 복잡한 메뉴 선택이나 필터 설정 없이 자연어로 대화하듯 질문하기만 하면 매장 추천과 할인 안내를 동시에 제공받을 수 있는 대화형 시스템은 사용 편의성을 크게 향상시킬 수 있다. 이를 위해서는 사용자의 요구를 이해하는 모델과 매장 추천·혜택 계산 기능이 함께 동작하는 구조가 필요하다. 이러한 접근 방식은 소비자가 혜택을 스스로 찾아다니는 기존 방식에서 벗어나 서비스가 사용자에게 필요한 혜택을 능동적으로 제시해 주는 사용자 경험을 제공한다.

결과적으로 위치 기반 매장 검색, 사용자 프로필 기반 혜택 계산, 할인 정보 자동 매칭, LLM 기반

대화형 인터페이스를 통합한 서비스는 기존 소비 도우미 서비스에서는 제공되지 않는 차별화된 기능이며, 경제적 혜택 측면과 사용자 경험(UX) 측면 모두에서 높은 실효성을 가진다. 따라서 본 프로젝트는 분산된 할인 정보 접근성을 개선하고, 사용자가 실제로 보유한 혜택을 놓치지 않고 활용할 수 있도록 돕는 실질적인 솔루션을 제공한다는 점에서 수행의 필요성이 충분히 존재한다.

제 2 절 과제 목적 및 내용

1. 과제 목적

본 프로젝트의 목적은 분산되어 있는 위치 정보와 할인 정보를 통합하여, 사용자에게 가장 유리한 소비 선택지를 대화형 방식으로 제공하는 서비스를 구현하는 데 있다. 기존의 매장 검색 서비스는 단순한 정보 나열에 머무르며, 사용자가 실제로 보유한 카드·통신사 혜택과 연결되지 못해 실질적인 절감 효과를 제공하지 못했다.

본 프로젝트는 사용자가 사전에 입력한 혜택 정보를 기반으로 할인 금액을 자동 계산하고, LLM을 활용해 일상적인 대화형 요청만으로 매장 추천과 혜택 안내를 동시에 받을 수 있는 개인화 소비 도우미 서비스 구축을 목표로 하였다.

2. 과제 내용

프로젝트는 MCP(Model Context Protocol) 기반의 마이크로서비스 구조를 적용하여, 기능 단위로 서버를 분리한 후 이를 통합하는 방식으로 개발되었다. 각 구성 요소는 다음과 같다.

가. LLM 기반 의도 파악 및 필터링 (Main Server)

사용자의 자연어 질문(“강남역 조용한 파스타집 추천해줘”)에서 핵심 키워드(예: 지역, 업종, 분위기)를 자동 추출하고, 시스템 범위를 벗어난 요청(Prompt Injection 등)을 감지하여 차단하는 필터링 로직을 구축하였다.

나. 위치 및 할인 데이터 수집 서버 (Location & Discount Server)

위치 정보: 네이버 지도 API를 연동하여 매장 데이터를 수집하고, 정수형 좌표 데이터를 표준 실수형 형식으로 변환하는 전처리 로직을 구현하였다.

할인 정보: 웹에서 크롤링한 비정형 데이터를 계산가능한 할인 금액·할인율 중심의 수치 데이터로 표준화하여 추천 알고리즘이 활용할 수 있도록 정제하였다.

다. 추천 엔진 (Recommendation Server)

데이터 통합 : 매장 정보와 할인 정보를 매장명 기반으로 통합하였다.

정렬 : 사용자의 위치를 기반으로 한 '거리순' 정렬결과와, '내 할인 혜택순' 두가지 기준으로 랭킹을 산출할 수 있도록 알고리즘 구현하였다.

최적화 : 모바일 환경 최적화를 위해 10개 단위로 끊어서 전송하는 페이지네이션(무한 스크롤) 기능 적용하였다.

라. RAG Pipeline

단순한 조건 검색에 그치지 않도록, 리뷰 데이터를 임베딩하여 저장한 뒤 사용자의 감성적 요구(예: 분위기 좋은, 조용한, 카공하기 좋은)에 가장 가까운 매장을 찾아내는 검색 증강 생성(RAG) 기술을 구현하였다.

3. 과제 수행 방법

프로젝트는 기능 단위로 단계적으로 확장·검증하는 방식으로 진행되었다.

가. 아키텍처 설계 및 데이터 규격 정의

시스템을 유연하게 확장하기 위해 MCP 아키텍처를 도입하고, 각 서버(Location, Discount, Rec) 간 주고받을 데이터 규격(JSON Schema)을 먼저 정의하여 협업 효율을 높였습니다.

나. 데이터 파이프라인 구축 (ETL)

네이버 API와 크롤링 봇을 통해 원천 데이터를 확보하고, 이를 추천 알고리즘이 이해할 수 있는 표준 포맷으로 정제하는 과정을 자동화했습니다.

다. 테스트 및 사용자 경험 중심의 검증

개발된 알고리즘을 터미널 환경에서 시나리오별(거리 우선 vs 할인 우선)로 테스트하여 논리적 오류를 수정하고, 최종 단계에서 Flutter 앱과 통합하여 실제 사용 경험을 점검하였다.

제 3 절 과제 범위

1. 사업적 측면

가. 타겟 및 서비스 범위 설정 (F&B MVP)

본 프로젝트의 범위는 광범위한 소비 영역 중 결제 빈도가 가장 높고 할인 혜택 정보가 파편화되어 있는 외식·카페(F&B) 분야를 최우선 적용 대상으로 설정하였다. 서울 주요 상권을 중심으로 사용자가 보유한 카드·통신사·멤버십 혜택을 실시간으로 매칭해 주는 MVP(Minimum Viable Product) 모델을 구축하여 식비 절감 효과를 검증하는 데 집중하였다.

나. 핵심 가치

단순한 정보 나열이 아닌 사용자의 자연어 질문에 대해 가장 경제적인 결제 방식을 제안하는 소비 보조 기능을 핵심 가치로 삼았다. 이를 통해 사용자는 복잡한 검색 과정 없이 금전적 혜택과 편의성을 동시에 누릴 수 있도록 하였다. 쇼핑, 여행, 구독 등 타 도메인으로의 확장은 기술적 안정성이 확보된 이후의 로드맵으로 설정하고 본 과제에서는 F&B 영역에서의 완결성 있는 사용자 경험(End-to-End UX) 구현에 주력하였다.

2. 연구적 측면

가. MCP 아키텍처 기반의 시스템 통합

단일 서버 구조가 아닌, 기능별로 독립된 3개의 서버(Location, Discount, Recommendation)를 LLM이 지휘하는 MCP 아키텍처를 설계하고 검증하였다. 이를 통해 각 모듈의 독립성을 보장하면서도, 복잡한 사용자 질의를 유연하게 처리할 수 있는 확장 가능한 시스템 구조를 확립하였다.

나. 데이터 정규화 및 추천 알고리즘 구현

API 기반의 실시간 위치 데이터(네이버)와 웹 크롤링 기반의 정적 할인 데이터(DB)라는 이질적인 데이터 소스를 하나의 표준 포맷으로 통합하는 전처리 기술을 연구하였다. 또한, 물리적 거리와 경제적 혜택이라는 두개의 가치를 사용자의 필요에 따라 최적화하여 정렬하는 추천 알고리즘을 구현하여 추천의 신뢰성을 확보하였다.

다. RAG 및 LLM 활용성 검증

단순한 데이터 검색을 넘어, 리뷰 데이터를 벡터화하여 사용자의 감성적 의도("조용한", "분위기 좋은")를 파악하는 RAG 파이프라인을 구축하였다. 이를 통해 LLM이 거짓된 정보(Hallucination) 없이 정확한 사실과 자연스러운 설명을 동시에 제공할 수 있다는 점을 확인하였다.

제 2 장 프로젝트 목적 및 내용

제 1 절 프로젝트의 최종 목적

본 프로젝트의 최종 목적은 분산된 위치 정보와 할인 혜택 정보를 LLM 기술로 통합하여 사용자 개인에게 가장 경제적인 소비 선택지를 실시간으로 제안하는 서비스를 구축하는 것이다. 기존의 검색 서비스는 단순히 매장의 위치나 평점 정보만을 나열하는 데 그쳤다면 본 서비스는 사용자가 사전에 등록한 결제 프로필(카드·통신사·멤버십)과 매장의 할인 정책을 실시간으로 대조하여 실질적인 체감 혜택 금액을 산출한다. 이를 통해 사용자는 복잡한 검색 과정 없이 자연어 대화만으로 지금 내 카드로

할인받을 수 있는 가장 가까운 맛집을 추천받을 수 있으며 이는 번거로운 검색 과정을 대신 처리해주어 사용자의 편의성을 높이고 최적의 할인을 통해 실질적인 가계 비용 절감 효과를 제공한다.

제 2 절 프로젝트의 세부 목표

가. MCP 아키텍처 기반의 데이터 통합 시스템 구축

- **마이크로서비스 분리:** 위치(Location), 할인(Discount), 추천(Recommendation) 등 각기 다른 기능을 수행하는 서버를 독립적으로 구축하여 시스템의 유연성과 확장성을 확보한다.
- **데이터 형식 통합(Data Integration):** 실시간 API에서 제공하는 위치 데이터와, 웹 크롤링을 통해 DB에 적재된 정적 할인 데이터를 매장명 기준으로 매핑하고 표준화된 포맷으로 변환하는 전처리 파이프라인을 구현한다.

나. 프로필 기반 추천 엔진 개발

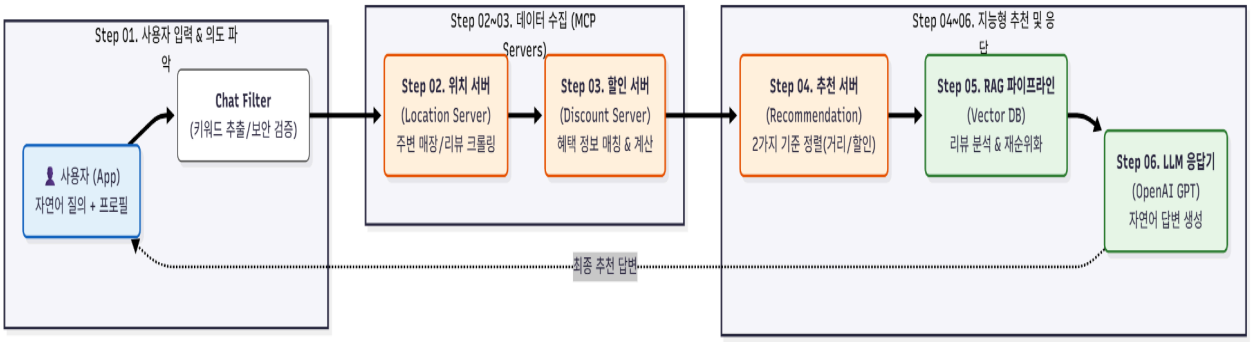
- **프로필 매칭:** 사용자가 보유한 카드 및 멤버십 정보를 기반으로 적용 가능한 혜택을 필터링하는 로직을 구현한다.
- **정렬 알고리즘(Sorting):** 사용자의 상황에 맞는 물리적 거리와 경제적 혜택 이라는 두가지 기준으로 정렬된 결과를 제공하고, 대량의 데이터를 효율적으로 처리하기 위한 페이지네이션(Pagination) 기능을 구현한다.

다. LLM 및 RAG 기반의 지능형 인터페이스 구현

- **의도 파악 및 필터링:** 사용자의 자연어 질문("강남역 조용한 파스타집")에서 핵심 키워드(지역, 업종, 분위기)를 추출하고, 서비스 범위 밖의 질문을 차단하는 보안 체계를 구축한다.
- **소비맥락 분석(RAG):** 단순 스펙 비교를 넘어, 매장 리뷰 데이터를 벡터화(Embedding)하여 저장하고, 사용자의 감성적 니즈와 유사도가 높은 매장을 2차적으로 선별하는 검색 증강 생성 기술을 적용한다.

제 3 절 프로젝트의 개발 및 연구 내용

제 1항 메인 파이프라인 구성도



제 2 항 Chat Filter PipeLine

가. 개요 및 목표

- **목표**
  1) 사용자 입력을 검증하고 보안 위협(Prompt Injection)을 차단.
  2) LLM과 규칙 기반 방식을 결합하여 핵심 키워드(장소 타입, 속성, 지역)를 추출하는 파이프라인 구현.

나. 입출력 데이터 정의

- **입력(Input)**
  1) **User Query:** 사용자 입력 질문
  2) **User Profile:** 사용자 프로필 정보 (통신사, 카드, 멤버십)

- **출력(Output)**
  1) **Validation Result:** 입력 검증 결과 (성공/실패)
  2) **Keywords:** 추출된 키워드 (장소 타입, 속성, 지역)
  3) **MCP Ready Flag:** 다음 단계(Location Server) 진행 가능 여부

다. 세부 동작 시퀀스

- **입력 검증 (Input Validation)**
  1) **정규화:** 입력 텍스트를 정리하고 길이를 500자로 제한하여 시스템 과부하를 방지
  2) **Prompt Injection 탐지:** 정규 표현식 기반으로 악의적인 프롬프트 패턴 탐지하여 보안 위협을 사전에 차단
     - 탐지 패턴: “이전 지시 무시”, “시스템 프롬프트”, “역할 변경” 등
  3) **키워드 필터링:** 허용/차단 키워드 리스트를 통한 도메인 검증
     - 허용 키워드: 음식점, 카페, 맛집, 할인 등
     - 차단 키워드: 코딩, 프로그래밍, 정치, 주식 등

4) **프로필 검증:** User profile 필수 필드 확인

- **키워드 추출 (Keyword Extraction)**
  1) **하이브리드 추출 방식:** Gemini API와 규칙 기반 방식을 결합
     - 1차: Gemini 2.0 Flash API를 통한 자연어 이해 기반 추출
     - 2차 (Fallback): API 실패 시 정규 표현식 기반 규칙 추출
  2) **추출항목**
     - place_type: 장소 타입 (카페, 한식, 일식, 치킨 등 25개 카테고리)
     - attributes: 장소 속성 (맛있는, 분위기좋은, 가성비좋은 등 21개 속성)
     - location: 지역 정보 (강남역, 홍대, 동국대학교 등 70개 지역)
- **응답 포맷팅**
  1) **MCP Ready 플래그 설정:** place_type이 추출되면 다음 단계 진행 가능
  2) **통합 응답 구조:** 검증 결과, 키워드, 프로필을 하나의 딕셔너리로 반환

라. 문제점 및 해결방안

- **문제점 1: Gemini API 호출 실패 시 전체 파이프라인이 중단되는 문제 발생**
  1) **해결방안: Fallback 메커니즘 구현**
     - Gemini API 실패 시 자동으로 규칙 기반 추출 방식으로 전환
     - 정규 표현식 기반 패턴 매칭을 통해 안정적인 키워드 추출 보장
     - 25개 장소 타입, 21개 속성, 70개 지역에 대한 패턴 사전 구축

- **문제점 2: Prompt Injection 공격에 취약하여 시스템 프롬프트를 우회하는 시도가 가능**
  1) **해결방안: 다층 보안 검증 구현**
     - 1단계: 정규 표현식 기반 악의적 패턴 탐지 (8개 패턴)
     - 2단계: 허용/차단 키워드 화이트리스트/블랙리스트 검증
     - 3단계: 입력 길이 제한 및 정규화를 통한 추가 방어
     - 검증 실패 시 즉시 파이프라인 중단 및 안전한 에러 메시지 반환

제 3 항 Location Server

가. 개요 및 목표

- **목표**
  1) 위치 정보를 기반으로 주변 음식점 및 카페를 검색.
  2) 실시간 리뷰 데이터를 수집하여 사용자에게 제공하는 통합 시스템 구축.

나. 입출력 데이터 정의

- **입력(Input)**

1) **Location:** 위치 좌표 (latitude: 위도, longitude: 경도)
2) **Place Type:** 장소 유형
3) **Attributes:** 속성 리스트

■ **출력(Output)**

1) **Stores:** 가게명 리스트
   - 예) ["장충동커피", "기브온 카페인바", "포우즈"]
2) **Reviews:** 가게별 리뷰 데이터
   - 예) {
   "장충동커피": ["커피 퀄리티가 좋아요", "가성비 좋네요"],
   "기브온 카페인바": ["디저트가 훌륭합니다", "커피 향이 좋아요"]
   }
3) **Distances:** 가게 거리 정보 (거리 계산용)
4) **Locations:** 가게 위치 정보 (위도, 경도)

다. 세부 동작 시퀀스

■ **요청 파싱 및 MCP 연결**

1) **데이터 전달 :** LocationModule로부터 전달받은 사용자 좌표(Latitude, Longitude), 업종(Category), 검색 속성 데이터를 MCP 프로토콜(Stdio)을 통해 격리된 서브 프로세스(location_server.py)로 전달.

■ **검색 쿼리 구성 및 좌표 기반 검색**

1) 전달받은 좌표와 업종 키워드를 결합하여 검색 쿼리를 생성.
   - 방식 A (nearby_reviews.py): 좌표를 searchCoord 파라미터로 Smart Around API에 전달하여 반경 기반 검색.
   - 방식 B (query_to_naver.py): 텍스트 주소가 있으면 Forward Geocoding으로 좌표 획득. 네이버 검색 API에 검색어와 좌표(lat/lng)를 함께 전달.

■ **네이버 로컬 검색 수행**

1) **API 호출 :** 생성된 쿼리를 바탕으로 Naver Place Search API 호출하여 매장 데이터 확보.

■ **데이터 정제 및 구조화**

1) **노이즈 제거 :** API 응답 데이터의 노이즈를 제거하고
2) **정규화 :** 후속 프로세스(Discount Server, RAG)에서 사용하기 용이하도록 매장명, 표준 주소, 카테고리, X/Y 좌표 등 핵심 식별 정보만을 추출하여 정규화된 딕셔너리 리스트로 변환.

■ **리뷰 크롤링**

1) **상세 정보 수집 :** 식별된 매장들의 상세 정보(Place ID 기반 리뷰, 평점, 키워드 태그)를 Playwright 브라우저로 크롤링.

■ **최종 응답 생성**

1) **직렬화 및 반환 :** 매장 리스트(Stores)와 매장별 리뷰 데이터(Reviews)를 JSON 객체로 직렬화(Serialization)한 후, 표준 출력 스트림(Stdout)을 통해 메인 MCP Client로 반환.

라. 문제점 및 해결방안

■ **문제점 1: 동기식 순차 처리에 따른 응답 지연**

1) 원인

- [지오코딩 → 네이버 검색 API 호출 → (반복) 개별 매장 상세/리뷰 수집] 과정이 순차적으로 이루어짐.
- nearby_reviews.py에서 검색된 매장 리스트(N개)를 순회하며 추가 정보를 수집할 때, N번의 네트워크 I/O가 발생하여 전체 응답 시간이 매장 수에 비례해 급격히 늘어남.

2) 해결방안

- **비동기 병렬 처리:** 개별 매장의 리뷰 및 상세 정보를 수집하는 과정을 asyncio.gather 등을 활용하여 병렬로 동시에 요청.
- **검색 결과 캐싱:** '강남역 + 맛집'과 같이 빈번하게 요청되는 쿼리와 그 결과값을 Redis나 인메모리에 캐싱. 동일한 지역/카테고리 요청 시 API 호출 없이 즉시 응답하여 네이버 API 호출 속도 향상.

■ **문제점 2 : 키워드 매칭의 정확도 한계**

1) 원인

- query_to_naver.py는 단순히 "행정구역명 + 카테고리(예: 역삼동 중국집)" 형태로 쿼리를 조합.
- 사용자가 "조용한 분위기의 데이트 장소"를 원했을 때, LLM이 '데이트'나 '분위기' 같은 속성을 추출하더라도 네이버 검색 API의 쿼리에 이를 단순 결합하면 검색 결과가 없거나 엉뚱한 결과가 나올 확률이 높음.

2) 해결방안

- **쿼리 확장:** LLM을 활용해 단일 쿼리 대신 연관 검색어 리스트(예: "역삼동 파스타", "역삼동 스테이크", "역삼동 와인바")를 생성하여 다각도로 검색을 수행한 뒤 결과를 합침.
- **메타데이터 필터링:** 네이버에서 1차로 넓게 검색된 결과 (Over-fetching)를 가져온 후, 코드 레벨에서 리뷰 텍스트나 태그에 사용자가 원하는 속성(예: '조용한', '주차가능')이 포함되어 있는지 필터링.

제 4 항 Discount Server

가. 개요 및 목표

■ **목표**

1) 웹상에 흩어진 비정형 할인 정보(텍스트)를 정기적으로 수집하여 계산 가능한 정형 데이터(수치)로 변환해 DB에 적재하는 ETL(Extract, Transform, Load) 파이프라인 구축.
2) 추천 시스템이 활용할 수 있도록 할인 정보를 표준화된 API로 제공

나. 입출력 데이터 정의

- **입력 (Input)**
  1) **Stores:** 조회 대상 매장명 리스트.
  2) **User Profile :** 사용자 프로필
- **출력 (Output)**
  1) **Discounts:** 매장별 할인 정보가 담긴 표준화된 JSON 리스트.
     - 포함 정보: `providerType` (제공자), `discountType` (할인 유형: PERCENT/AMOUNT/PER_UNIT), `discountAmount` (할인값),constraints(제약 조건), `isDiscount`(할인/적립 구분), `appliedByUserProfile`(사용자 적용 가능 여부) 등

다. 세부 동작 시퀀스 – 할인 정보 제공 MCP 서버

- **MCP 클라이언트의 호출**
  1) **요청 입력 수신:** 사용자 프로필과 매장 이름 목록 전달받음.
  2) **DB 커넥션 풀 초기화:** PostgresSQL 접근을 위해 커넥션 풀을 생성하고 연결을 준비.
- **할인 정보 서비스 실행**
  1) **입력 데이터 정규화:** 전달 받은 유저 프로필 및 매장명 리스트를 내부 스키마에 맞춰 정규화.
  2) **할인 DB 조회:** 각 매장에 적용 가능한 모든 할인 프로그램을 조회.
  3) **사용자 프로필 매칭:** appliedByUserProfile 필드를 추가해 사용자에게 실제로 적용 가능한 할인만 true로 표시.
- **할인 정보 반환**
  1) **할인 결과 구성 및 응답:** 이용 가능한 할인 프로그램 데이터를 정리하여 표준화된 JSON 형식으로 변환한 뒤 MCP 클라이언트에게 반환.
  2) **DB 커넥션 풀 정리:** 요청 처리가 완료되면 커넥션 풀을 반환 및 해제하여 리소스 정리.

라. 세부 동작 시퀀스 – ETL 파이프라인

- **데이터 수집 (Crawling – Extract)**
  1) **타겟 선정:** 주요 프랜차이즈, 통신사, 카드사 웹사이트 및 제휴 페이지 리스트 확보.
  2) **크롤링 수행:** 각 웹사이트의 AJAX 엔드포인트를 직접 호출하거나 BeautifulSoup4를 활용하여 정적/동적 웹페이지의 혜택 텍스트 수집.

- **데이터 정규화 (Normalization – Transform)**
    1) **구조화:** regex 및 LLM을 "SKT 멤버십 제시 시 10% 할인 또는 적립"과 같은 자연어 텍스트를 분석.
    2) **수치 변환:** 추천 엔진 연산을 위해 `discountType: "PERCENT"`, `amount: 10`과 같은 수치형 데이터로 매핑.
    3) **스키마 매칭:** 다양한 형태의 혜택 정보를 통일된 JSON 스키마(Schema)로 변환.
- **데이터 적재 및 제공 (Load & Serve)**
    1) **DB 저장:** 정제된 데이터를 PostgreSQL 데이터베이스에 적재 (Upsert 방식 적용).
    2) **API 응답:** Recommendation Server의 요청 시, 해당 매장의 유효한(Valid) 할인 규칙을 조회하여 반환.

마. 문제점 및 해결방안

- **문제점 1: 데이터의 유효성 유지**
    1) **원인:** 웹사이트의 정보가 변경되었으나 DB에 반영되지 않아 사용자에게 만료된 혜택을 안내할 위험 존재.
    2) **해결방안: 유효기간 기반 관리 및 자동 DB 갱신**
        - 데이터 수집 시 `validFrom`, `validTo` 필드를 파싱하여 유효 기간을 명시.
        - 주기적인 크롤링 스케줄러를 가동하여 만료된 정보를 갱신하거나 비활성화 처리.
- **문제점 2: DB 입출력의 지연**
    1) **원인:** ETL 파이프라인 실행 시 DB 연결을 계속 생성하여 트래픽이 급증해 서버 과부하.
    2) **해결방안: DB Connection Pool 구현 및 사용**
        - DB 연결을 ETL 파이프라인 실행 시 미리 일정 개수만큼 생성하고 파이프라인이 실행되는 동안 재사용.
        - 지연 시간 및 서버 부하감소.

제 5 항 Recommendation Engine

가. 개요 및 목표

- **목표**
    1) 사용자 프로필과 할인 정보를 기반으로 매장을 2가지 기준(개인화, 거리) 으로 정렬.
    2) 최적의 추천 결과를 생성하는 엔진 구현.

나. 입출력 데이터 정의

- **입력(Input)**
    1) **Stores:** 매장 이름 리스트
    2) **Discounts by Store:** Discount Server로부터 받은 매장별 할인 정보
    3) **User Profile:** 사용자 프로필 (통신사, 카드, 멤버십)
    4) **User Location:** 사용자 위치 좌표 (위도, 경도)

5) **Stores Detail:** 매장 상세 정보 (좌표 포함)

- **출력(Output)**
  1) **Personalized Recommendations:** 사용자 프로필과 매칭되는 할인이 있는 매장 순위
  2) **By Distance:** 거리순 매장 순위

다. 세부 동작 시퀀스

- **데이터 전처리 (Data Preparation)**
  1) **할인 정보 추출:** Discount Server의 응답에서 매장별 할인 리스트 추출
  2) **거리 계산:** 하버사인 공식을 사용하여 사용자 위치와 매장 간 거리 계산 (미터 단위)
  3) **좌표 정규화:** 네이버 API의 정수형 좌표를 실수형으로 변환 (1000000 또는 10000000으로 나누기)
  4) **할인 금액 계산:** 기준 금액 12,000원 기준으로 각 할인의 실제 금액 계산
     - PERCENT: 퍼센트 할인 (최대 할인 금액 제한 적용)
     - AMOUNT: 정액 할인
     - PER_UNIT: 단위별 할인 (예: 1000원당 150원, 최대 3000원)

- **개인화 추천 (Personalized Recommendations)**
  1) **프로필 매칭:** 사용자 프로필과 할인 조건 비교
     - appliedByUserProfile 필드 우선 확인 (Discount Server가 이미 매칭한 경우)
  2) **Provider Type별 매칭 로직**
     - TELCO: 통신사명 일치 확인 (대소문자 무시, 부분 일치 허용)
     - CARD/PAYMENT: 카드명 일치 확인 (부분 일치 허용)
     - MEMBERSHIP: 멤버십명 일치 확인 (부분 일치 허용)
     - STORE: 무조건 매칭 (매장 자체 할인)
     - 필터링: 프로필과 맞는 할인이 있는 매장만 선택
     - 정렬: 프로필과 맞는 할인 금액이 큰 순으로 정렬 (동일 시 거리순)
     - 순위 부여: 1위부터 순차적으로 rank 할당

- **거리순 추천 (By Distance)**
  1) **정렬:** 사용자 위치에서 가까운 순으로 정렬 (거리 정보 없는 매장은 후순위 배치)
  2) **정보 포함:** 모든 할인 정보를 all_benefits에 포함 및 순차적 rank 할당

- **결과 최적화 및 반환 (Optimization & Response)**
  1) **데이터 병합:** 정렬된 리스트(Personalized / Distance / Total)를 통합.
  2) **페이지네이션 (Pagination)**: 모바일 환경의 데이터 로딩 속도 최적화를 위한 슬라이싱(Slicing) 수행.
     - 클라이언트 요청(`Page`, `Limit`)에 맞춰 전체 데이터 중 해당 구간(예: 1~10위)만 추출.

- HasNext 플래그를 계산하여 무한 스크롤(Infinite Scroll) 가능 여부 반환.

3) **최종 응답 생성:** 메타데이터(총 개수, 현재 페이지 등)와 함께 JSON 포맷으로 반환.

라. 문제점 및 해결방안

- **문제점 1: 중복 로직 발생 및 정보 혼란 야기**
  1) **원인**
     - Discount Server의 `appliedByUserProfile` 필드 미활용으로 인한 재검증 로직 중복.
     - 사용자가 받을 수 없는 할인 정보를 노출하여 사용자 혼란 발생.
  2) **해결 방안: 우선순위 기반 매칭 및 선택적 노출 전략**
     - `appliedByUserProfile` 필드 최우선 확인 (True인 경우 즉시 매칭 판단).
     - 선택적 정보 노출: Personalized 섹션은 매칭된 할인만, Total Discount 섹션은 세부 정보를 숨김 처리하여 UX 개선.
- **문제점 2: 좌표 형식 불일치로 인한 거리 계산 오류**
  1) **원인:** 네이버 API의 좌표 형식(정수형)과 표준 좌표 형식(실수형) 간 불일치.
  2) **해결방안: 좌표 정규화 로직 구현**
     - 좌표 값이 1000 이상인 경우 자동 감지.
     - 위도/경도 범위를 고려하여 적절한 나눗셈(1000000 또는 10000000)

제 6 항 RAG Pipeline

가. 개요 및 목표

- **목표**
  1) 위치/할인 추천 결과와 리뷰를 벡터화(Vectorization)하여 저장.
  2) 사용자 질의에 맞는 매장을 Top-K로 검색하고 LLM 답변 컨텍스트를 생성하는 RAG 파이프라인 구현.

나. 입출력 데이터 정의

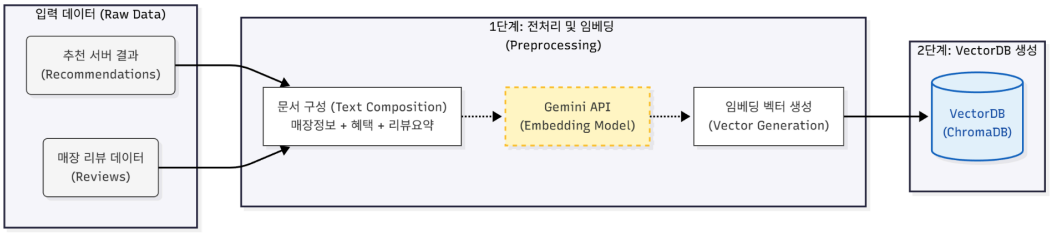
- **구성요소**
  1) **RAG Pipeline :** 전체 프로세스 제어 및 로직 수행
  2) **VectorDB :** 고차원 벡터 데이터 저장소 (ChromaDB 활용)

- **입력(Input)**
  1) **Recommendations:** 추천 서버의 결과 (할인순/거리순 store list)
  2) **Reviews:** 가게별 reviews가 포함된 텍스트 list
  3) User Query, User profile, User ID

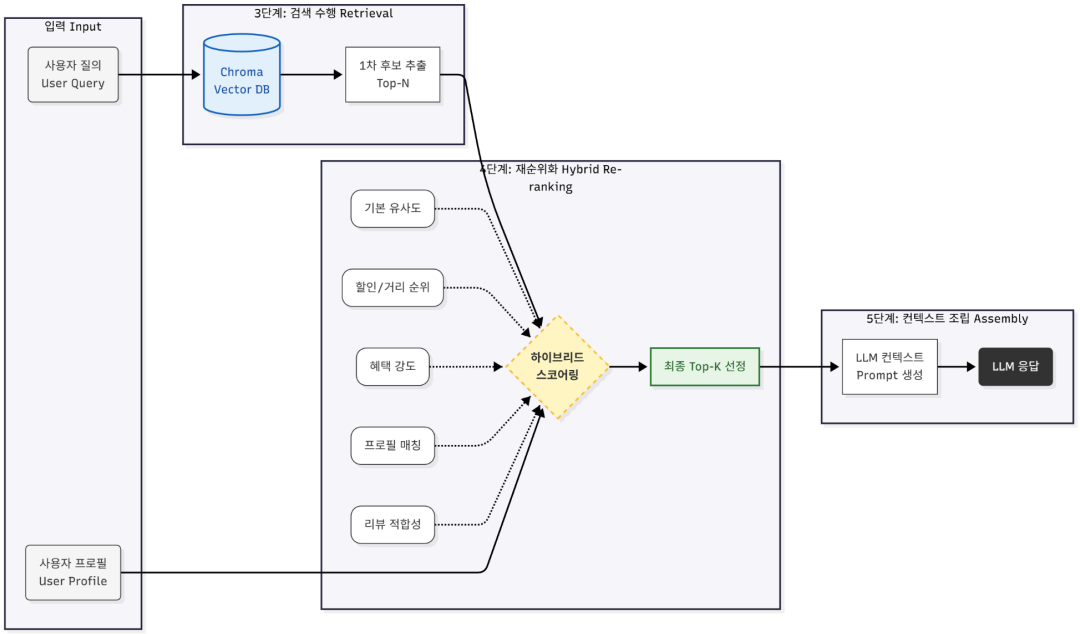
- **출력(Output)**
  1) **LLM Context:** RAG 검색 및 재순위화가 완료된 매장 정보와 리뷰 요약본이 포함된

LLM 응답용 Context

다. RAG Pipeline 세부 동작 시퀀스



- **전처리 및 임베딩:**
  1) **데이터 정규화:** 추천 서버의 매장 데이터와 리뷰 데이터를 RAG 파이프라인이 처리하기 쉬운 포맷으로 변환.
  2) **문서 구성:** 매장의 위치, 할인혜택, 리뷰 요약 정보를 하나의 텍스트 덩어리로 결합
  3) **벡터 생성:** 결합된 텍스트를 Gemini Embedding API를 사용하여 고차원 벡터(Embedding Vector)로 변환
  4) **VectorDB 생성:** 임베딩 된 문서 데이터를 ChromaDB 라이브러리를 통해 VectorDB로 저장함.



- **검색 수행**
  1) **유사도 검색:** VectorDB와 User Query간의 유사도 검색을 수행하여 이에 맞는 매장정보가 담긴 1차 후보를 추출.
- **재순위화(Re-ranking)**

1) **하이브리드 스코어링:** 유사도 검색 결과의 정확도를 높이기 위해 5가지 가중치 요소를 반영한 자체 스코어링 알고리즘을 적용.
    - 기본 유사도, 할인/거리 순위, 혜택 강도, 프로필 매칭, 리뷰 적합성

■ **컨텍스트 조립**

1) 최종 선정된 Top-K 매장 선정

2) **프롬프트 변환:** 선정된 매장의 상세 정보와 리뷰 요약본을 LLM이 이해할 수 있는 자연어 프롬프트 형태로 변환.

라. 문제점 및 해결방안

■ **문제점 1: 단순 유사도 검색의 한계 (정보 반영 미흡)**

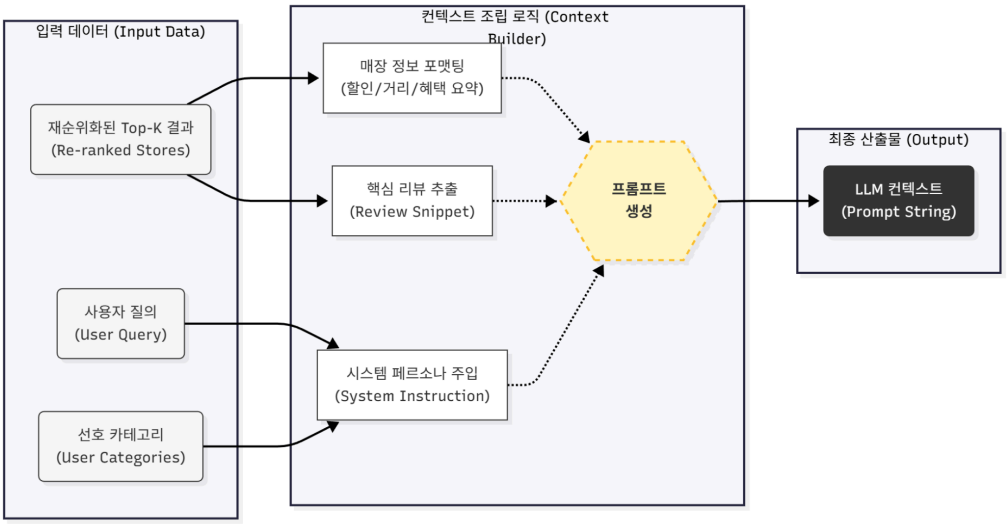
1) **원인**: 위 동작 시퀀스의 3.검색 수행 결과에서 단순 유사도 계산 방식만으로는 할인 정보나 사용자 프로필, 소비 맥락을 올바르게 반영하지 못하는 문제가 발생함.

2) **해결방안: 하이브리드 재순위화(Re-ranking) 스코어링 알고리즘 구현**
    - 단순 검색 결과를 보정하기 위해, 아래와 같이 5가지 핵심 요소를 가중 합산하는 자체 스코어링 수식을 설계 및 적용함.
    - $Re - ranked = Sim(Q, D) + \alpha rank + \beta benefit + \gamma profile + \delta reviev$
    - 기본 유사도 (Sim(Q, D)): User Query(Q)와 VectorDB 내 문서(D) 간의 임베딩 유사도 (L2 Distance).
    - 할인/거리 순위 가중치 (alpha * Rank): Recommendation Server에서 넘어온 정량적 순위가 높을수록 가산점 부여.
    - 혜택 강도 반영 (beta * Benefit): 적용되는 할인율(%)이나 할인 금액(원)이 클수록 높은 점수 부여.
    - 프로필 매칭 (gamma * Profile): 사용자의 통신사, 카드사, 멤버십 정보가 혜택 조건과 일치할 경우 추가 점수 부여 (개인화).
    - 리뷰 적합성 (delta * Review): User Query와 매장 실제 리뷰 텍스트 간의 직접적인 의미적 유사도(Cosine Similarity)를 재계산하여 반영.

■ **문제점 2: LLM 응답 품질 저하 (환각 및 부정확한 정보)**

1) **원인:** 검색된 원시 데이터(Raw Data)를 그대로 LLM에 입력할 경우, 정보 과잉이나 형식 불일치로 인해 부적절한 답변이 생성됨.

2) 해결방안: LLM 전용 컨텍스트 조립 (Context Assembly)
    - 매장 정보 포맷팅: 할인 혜택과 거리 정보를 LLM이 읽기 쉬운 문장 형태로 변환.
    - 핵심 리뷰 추출: 검색된 리뷰 중 사용자 질의와 가장 관련된 핵심 문장만을 선별하여 포함.
    - 시스템 페르소나 주입: "당신은 맛집 추천 비서입니다"와 같은 역할 정의 및 사용자 선호 카테고리(가성비, 분위기 등)를 반영하여 최종 프롬프트 완성.



제 7 항 LLM Responder

가. 개요 및 목표

- **목표**
  1) RAG 파이프라인에서 전달받은 최적의 매장 정보(Context)를 바탕으로 사용자의 의도에 부합하는 자연어 답변 생성.
  2) 딱딱한 데이터 리스트를 "친절한 비서" 페르소나를 통해 사용자 친화적인 메시지로 변환.

나. 입출력 데이터 정의

- **구성요소**
  1) **LLM Responder:** 프롬프트 보강 및 API 호출 담당 모듈.

- **입력(Input)**
  1) **User Query:** 사용자의 최초 자연어 질문.
  2) **LLM Context:** RAG 모듈에서 재순위화 및 조립이 완료된 프롬프트.
  3) **Filter Result:** 이전 단계(Chat Filter)에서 추출된 핵심 키워드(장소, 속성, 지역 등)

- **출력(Output)**
  1) **Final Response:** 사용자의 의도에 부합하는 자연어 추천 답변 및 할인 요약 정보

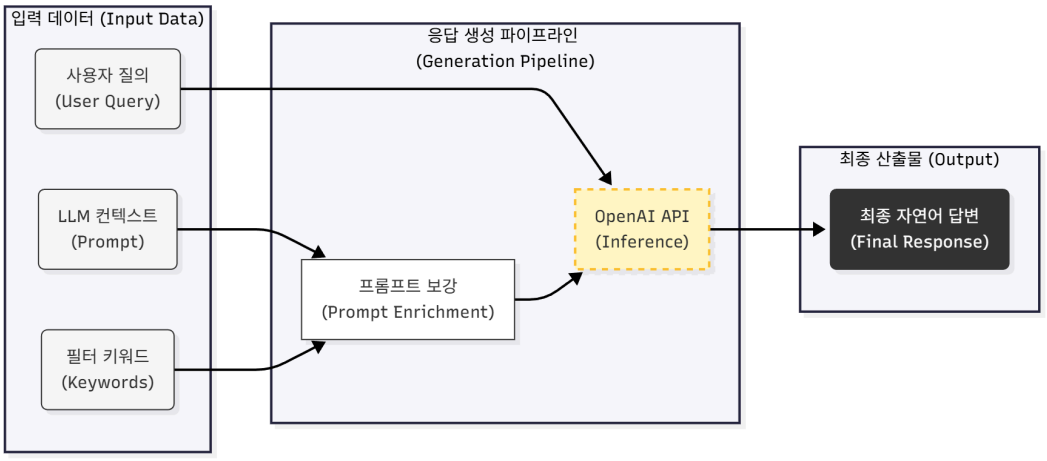
다. 세부 동작 시퀀스

- **프롬프트 보강 (Prompt Enrichment)**

1) **키워드 재주입:** RAG Context에 Filter Result(핵심 키워드)를 명시적으로 다시 포함시켜 LLM이 사용자의 원래 의도(예: "조용한 분위기")를 잊지 않도록 강화.
2) **제약 조건 설정:** "제공된 Context 내에서만 답변할 것", "없는 사실을 지어내지 말 것" 등의 Hallucination 방지 지침 추가.

- **LLM API 호출 (Inference)**
  1) **모델 호출:** 보강된 프롬프트를 OpenAI GPT 모델(gpt-5.1-chat-latest)에 전송.
  2) **파라미터 최적화:** 답변의 창의성 조절을 위해 Temperature 값을 조정(사실 기반 답변을 위해 낮게 설정).

- **응답 생성 및 후처리 (Response Generation)**
  1) **자연어 변환:** JSON 형태의 데이터를 "OO님, 찾으시는 강남역 근처 파스타 맛집은..."과 같은 대화체로 변환.
  2) **할인 정보 요약:** 추천된 매장의 핵심 할인 혜택(예: "신한카드 10% 할인 가능")을 강조하여 답변 하단에 요약 배치.



라. 문제점 및 해결방안

- **문제점 1: 정보 왜곡 및 환각(Hallucination) 가능성**
  1) **원인:** LLM이 Context에 없는 내용을 그럴싸하게 지어내거나 할인율 숫자를 틀리게 말하는 현상 발생 가능.
  2) **해결방안: 엄격한 시스템 프롬프트(System Prompt) 적용**
     - "너는 주어진 데이터(Context)만을 근거로 답변해야 한다"는 강력한 지시문(Instruction) 삽입.
     - 숫자 데이터(할인율, 거리)는 LLM이 계산하지 않고, Recommendation Server에서 계산된 값을 그대로 인용하도록 강제.

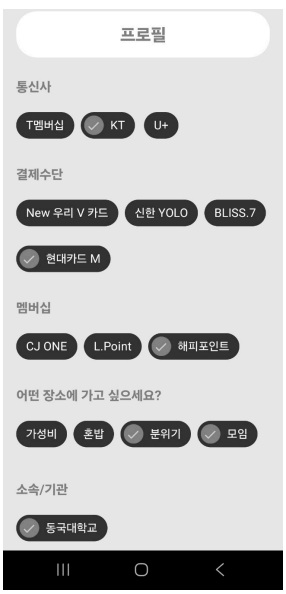
■ **문제점 2: 일관성 없는 답변 톤앤매너**

1) **원인:** 매번 호출할 때마다 어떨 때는 딱딱하게 어떨 때는 지나치게 가볍게 답변하는 문제.

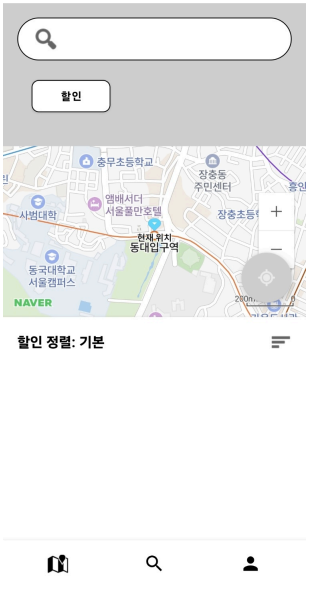
2) 해결방안: 페르소나(Persona) 고정

- 시스템 프롬프트에 "당신은 전문적이고 친절한 맛집 추천 컨시어지입니다"라는 역할을 명확히 정의하여 일관된 톤 유지.

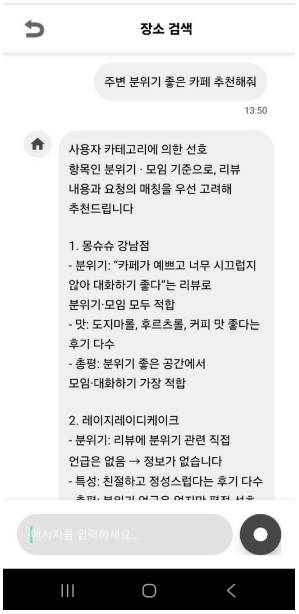
제 4 절 프로젝트의 주요 결과물



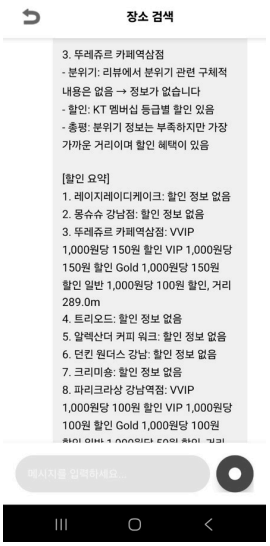
프로필 입력



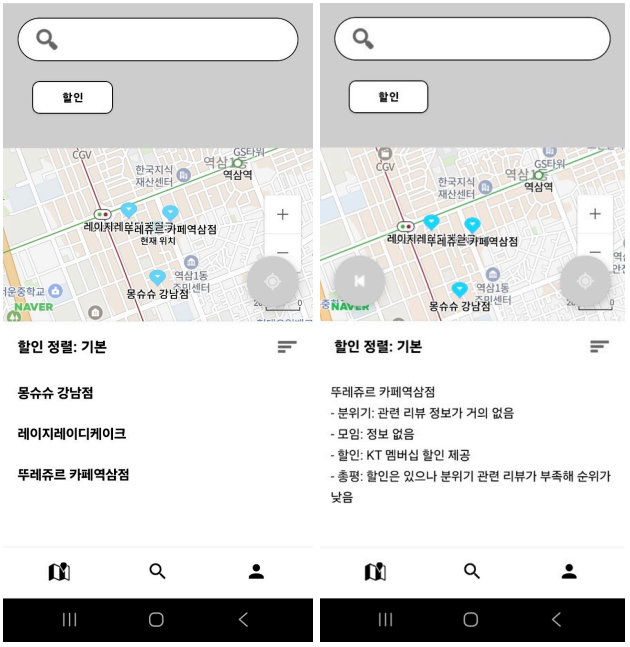
메인 화면



채팅 입력 화면1



채팅 입력 화면2



지도 결과물 화면1 지도 결과물 화면2

제 3 장 추진 일정

제 1 절 추진전략 및 방법

1. 추진 목표

본 프로젝트는 기술적 실증과 사용자 가치 검증이라는 두 가지 핵심 목표를 달성하기 위해 추진되었다.

- **기술적 목표:** LLM과 마이크로서비스를 연동하는 MCP 아키텍처의 유연성과 확장성을 검증하고, API(실시간)와 DB(정적)라는 이질적인 데이터 소스를 표준화하여 통합하는 견고한 백엔드 파이프라인을 구축한다.
- **서비스 목표:** 복잡한 검색 없이도 사용자의 카드/통신사 혜택을 높은 정확도로 매칭해 주는 MVP를 출시하여 실질적인 가계 비용 절감 효과를 입증한다.

2. 추진 전략 및 방법

가. 마이크로서비스 기반 병렬 개발 전략

단일 서버 개발 시 발생할 수 있는 의존성 문제를 해결하기 위해 시스템을 Location, Discount, Recommendation 3개의 독립적인 서버로 분리하였다. 이를 통해 팀원들이 각자의 전문 분야(API 연동, 크롤링, 알고리즘)에 집중하여 병렬적으로 개발을 진행함으로써 개발 속도와 코드 품질을 동시에 확보하였다.

나. 데이터 접근성을 고려한 전략

- **(초기 계획)** 사용자의 과거 결제 내역을 학습하여 소비 패턴을 예측하는 방식을 고려하였으나 개인 금융 데이터 확보의 진입 장벽과 예측 모델의 불확실성이 높다고 판단하였다.
- **(변경 전략)** 사용자가 직접 입력한 보유 혜택 프로필을 기반으로 할인 룰을 대조하는 알고리즘으로 전환하였다. 이를 통해 데이터 확보 문제를 해결함과 동시에 추천 결과의 신뢰도와 정확도를 높은 수준으로 끌어올리는 실용적인 전략을 채택하였다.

다. 단계별 통합

- **1단계 :** 각 서버의 핵심 로직(검색, 크롤링, 정렬)을 개별적으로 구현하고 단위 테스트(Unit Test)를 수행하여 기능적 오류를 제거하였다.
- **2단계 :** 서버 간 통신 규격(JSON Schema)을 확정하고 데이터 형식이 달라도 시스템이 중단되지 않는 방어적 프로그래밍을 적용하여 통합 테스트를 수행하였다.
- **3단계 :** 기능 구현이 완료된 백엔드 위에 LLM(RAG)과 모바일 앱(Flutter)을 연동하여 사용자가 기술을 체감할 수 있는 직관적인 서비스를 완성하였다.

제 2 절 추진체계

1. 프로젝트 계획

세부 개발 내용	세부 추진일정(주)										비 고
	1	2	3	4	5	6	7	8	9	10	
프로젝트 개발 계획 수립											
사용자 애플리케이션											
Location/Discount 서버											
Recommendation 서버											
Main Server & LLM											
RAG 시스템											
Flutter UI											
통합 및 최적화											
프로젝트 개발 마무리 및 고도화 작업											
분기별 진척률(%)	50%					100%					

2. 프로젝트 실행

가. 애자일(Agile) 기반 마이크로서비스 개발

- 팀원들이 각자 맡은 서버(Location, Discount, Rec)를 독립적으로 개발하는 병렬 처리 방식(Agile)을 채택하여 개발 속도를 단축시켰다.
- 매주 정기 회의를 통해 각 서버의 API 입출력 규격을 맞추고 변경 사항을 즉시 반영하는 스프린트(Sprint) 방식을 운영하였다.

나. 협업 도구

- **GitHub:** 소스 코드 버전 관리 및 Issue Tracker를 활용한 버그 추적.
- **Notion:** API 명세서(JSON Schema) 및 회의록 공유.
- **Discord:** 실시간 소통 및 서버 연동 테스트 결과 공유.

3. 품질 관리

가. 단계별 테스트

- **단위 테스트 (Unit Test):** 각 서버별 핵심 로직(거리 계산, 할인율 파싱 등)이 정상 작동하는지 검증하는 테스트 코드를 작성 및 수행하였다. (예: test_logic.py, test_distance.py)
- **통합 테스트 (Integration Test):** mcp_client.py를 통해 모든 서버가 연결된 상태에서 데이터가 끊김 없이 흐르는지 검증하였다.

나. 데이터 무결성 및 예외 처리

- **방어적 프로그래밍:** 외부 데이터(네이버 좌표, 웹 크롤링 텍스트)의 형식이 예고 없이 변경되더라도 서버가 중단되지 않도록 try-except 블록과 데이터 타입 자동 변환 로직을 적용하여 시스템 안정성을 확보하였다.
- **스키마 검증:** 서버 간 주고받는 데이터가 약속된 필드(latitude, discountAmount 등)를 포함하고 있는지 상시 모니터링하였다.

4. 프로젝트 모니터링 및 제어

가. 리스크 관리

- **초기 문제:** 사용자의 과거 소비 내역 데이터 확보가 불가능하여 패턴 분석 모델 구현에 차질 발생.
- **대응:** 불확실한 예측 모델 대신, 사용자가 직접 입력한 보유 혜택 프로필을 기반으로 하는 정렬형 매칭 모델로 신속하게 전환하여 프로젝트의 실현 가능성과 정확도를 높였다.

나. 성능 최적화

- **문제 식별:** 다수의 매장 데이터를 한 번에 로딩할 때 앱 반응 속도 저하 우려.
- **조치:** Recommendation 서버에 페이지네이션 기능을 도입하여 데이터를 10개씩 분할 전송하도록 개선함으로써 모바일 환경에서의 응답 속도를 최적화하였다.

제 4 장 프로젝트의 활용 방안 및 기대효과

제 1 절 기대효과 및 수익성

1. 기대효과

가. 소비자의 번거로움 해소 및 실질적 혜택 제공

여러 앱을 오가며 할인 정보를 일일이 찾아야 했던 사용자의 수고와 시간을 획기적으로 줄여준다. 특히 사용자가 가진 카드로 실제로 얼마를 할인받을 수 있는지 금액으로 계산해 줌으로써 사용자가 모르고 지나쳤던 혜택을 꼼꼼히 챙겨 합리적인 소비를 할 수 있도록 돕는다.

나. 소상공인 가게의 발견 및 골목상권 활성화

대형 프랜차이즈에 비해 홍보가 어려워 혜택이 잘 알려지지 않았던 동네 작은 가게들의 정보도 통합 검색 결과에 노출시킨다. 이를 통해 사용자는 숨겨진 혜택 가맹점을 쉽게 찾을 수 있게 되고 결과적으로 소비자와 지역 소상공인을 연결하여 골목상권 활성화에 기여한다.

2. 수익성

가. 맞춤형 광고 및 제휴

단순히 배너를 띄우는 것이 아니라 신한카드를 쓰는 사람에게 신한카드 혜택이 있는 매장을 먼저 보여주는 방식의 맞춤형 광고가 가능하다. 이는 불특정 다수에게 노출하는 것보다 광고 효과가 훨씬 높을 것으로 예상되며 향후 카드사나 통신사와의 제휴를 통해 수익을 창출할 수 있다.

나. 소비 데이터 활용 비즈니스

사용자가 어떤 위치에서 어떤 혜택을 찾는지에 대한 데이터는 금융권이나 리테일 기업에게 매우

중요한 정보가 된다. 이러한 오프라인 소비 맥락 데이터를 분석하여 트렌드 리포트를 발행하거나 기업에 데이터를 제공하는 방식으로 비즈니스 모델을 확장할 수 있다.

제 2 절 기술적 측면

가. 확장성 높은 MCP 아키텍처 구현

모든 기능을 하나의 서버에 넣지 않고, 위치, 할인, 추천 기능을 각각 독립적인 서버로 나누어 LLM으로 연결하는 MCP 구조를 성공적으로 구현하였다. 덕분에 나중에 쇼핑이나 여행 같은 새로운 기능을 추가할 때도 기존 시스템을 건드리지 않고 블록을 끼우듯이 쉽게 확장할 수 있는 기반을 마련하였다.

나. 데이터 통합 및 안정성 기술 확보

실시간으로 변하는 지도 데이터(API)와 고정된 할인 데이터(DB)를 하나로 합치고 텍스트로 된 할인 정보를 계산 가능한 숫자로 바꾸는 자동화 파이프라인(ETL)을 구축하였다. 특히 데이터 형식이 서로 달라도 에러가 나지 않도록 처리하는 방어적 프로그래밍 기술을 적용하여 시스템이 멈추지 않고 안정적으로 구동되도록 하였다.

다. RAG 기술을 통한 정확한 추천

단순히 조건만 맞추는 게 아니라 리뷰 데이터를 분석하여 "분위기 좋은", "조용한" 같은 감성적인 질문까지 이해하는 검색 증강 생성(RAG) 기술을 적용하였다. 이를 통해 AI가 없는 말을 지어내는 환각 현상을 줄이고 사용자가 믿을 수 있는 정확한 답변을 제공하는 기술적 토대를 확보하였다.

제 3 절 경제, 산업적 측면

가. 합리적인 소비 문화 확산

물가가 비싼 현대 사회에 소비자가 귀찮아서 놓치기 쉬운 혜택을 꼼꼼하게 챙겨주는 도구를 제공함으로써 가격 대비 만족도를 중시하는 합리적인 소비 문화를 만드는 데 기여한다.

나. 골목상권 활성화 지원

광고비가 부족해 혜택을 알리기 어려웠던 동네 작은 가게들의 정보도 사용자의 위치를 기반으로 적극적으로 노출해 준다. 이는 대형 플랫폼에만 의존하지 않고도 지역 소상공인이 소비자와 만날 수 있는 새로운 기회를 제공하여 지역 경제 활성화에 도움이 된다.

제 4 절 활용 방안

가. 다양한 생활 영역으로 서비스 확장

현재 구축된 F&B(식음료) 추천 기술을 응용하면 영화, 주유, 호텔 등 생활 전반으로 서비스를 넓힐 수 있다. 예를 들어 "이번 주말 여행 코스 짜줘"라고 물으면 숙박 할인과 맛집 할인을 동시에 묶어서 추천해 주는 종합 생활 비서 서비스로 발전 가능하다.

나. Open API 제공을 통한 생태계 확장

본 프로젝트를 통해 구축한 정제된 할인 정보와 추천 알고리즘을 API 형태로 공개할 수 있다. 이렇게 되면 다른 스타트업이나 개발자들이 핀테크 앱이나 가계부 앱을 만들 때 해당 기능을 쉽게 가져다 쓸 수 있어 더 큰 서비스 생태계를 조성하는 데 기여할 수 있다.

제 5 장 참여인력

제 1 절 업무 분담

[표 3] 참여인력 및 담당분야

NO.	성명	소속			담당분야	참여도(%)
		학과	학번	학년		
1	강동경	AI융합학부	2022113180	3	RAG 기반 질의 응답 시스템 설계 및 구현, MCP Client 설계 및 구현	100
2	고유지	컴퓨터AI학부	2023113574	3	안드로이드 UI/UX, 위치기반 리뷰 크롤링, 검색 쿼리 구성 및 좌표 기반 검색	100
3	박주영	컴퓨터공학전공	2022111585	3	추천 서비스, 데이터 통합	100
4	서형선	컴퓨터AI학부	2022113581	3	안드로이드 UI/UX, Prompt 필터링 및 키워드 추출, 개인화 추천 알고리즘	100
5	신유진	컴퓨터공학전공	2021111986	3	할인 정보 데이터베이스 모델링, ETL 파이프라인 구축, 데이터 정규화 및 통합 관리 로직 설계	100

제 2 절 팀원 별 수행 성과

가. 강동경 (RAG 및 MCP Client 총괄)

- **RAG 기반 질의 응답 시스템 구축:** 사용자의 자연어 질문과 매장 리뷰 데이터를 벡터화하여 저장하는 Vector DB를 구축하고 질문의 맥락에 가장 적합한 매장을 검색하는 RAG 파이프라인을 구현하여 LLM 답변의 정확도를 대폭 향상시켰다.
- **MCP Client 아키텍처 설계:** Location, Discount, Recommendation 등 개별 마이크로서비스를 통합 관리하는 중앙 관제 서버(MCP Client)를 설계 및 구현하여 서버 간의 원활한 데이터 통신과 시스템 확장성을 확보하였다.

나. 고유지 (UI/UX 및 위치 데이터 담당)

- **직관적인 안드로이드 UI/UX 구현:** Flutter 프레임워크를 활용하여 사용자가 대화형 인터페이스를 통해 쉽고 편하게 정보를 얻을 수 있도록 앱 화면을 구성하고 지도 및 리스트 뷰를 최적화하여 사용자 경험(UX)을 개선하였다.
- **위치 기반 서비스 및 리뷰 데이터 수집:** 사용자의 실시간 위치 정보를 처리하는 로직을 앱에 구현하고 추천 결과의 신뢰도를 높이기 위해 주요 포털의 매장 리뷰 데이터를 수집하는 전처리 모듈을 개발하였다.

다. 박주영 (할인 서버 및 데이터 통합)

- **할인 정보 데이터 파이프라인 구축:** 주요 프랜차이즈별 할인 혜택 정보를 수집하고 이를 추천 시스템이 활용 가능한 표준 JSON 포맷으로 구조화하여 DB에 적재하는 전처리 과정을 수행하였다.
- **거리 기반 추천 및 최적화:** 사용자 위치 기반의 실시간 거리순 정렬 알고리즘을 구현하고 대량의 데이터 처리 시 서버 부하를 줄이기 위해 데이터를 분할 전송하는 페이지네이션 기능을 개발하여 모바일 응답 속도를 최적화하였다.
- **시스템 안정성 확보:** 이질적인 데이터 소스간의 형식 불일치로 인한 오류를 방지하기 위해 데이터 타입을 자동으로 감지하고 보정하는 방어적 프로그래밍 기법을 적용하여 서버의 결함 허용 능력을 강화하였다.

라. 서형선 (입력 처리 및 할인 추천 로직)

- **Prompt 필터링 파이프라인 구축:** 사용자의 입력값에서 지역, 메뉴, 분위기 등 핵심 키워드를 추출하는 전처리 로직과 악의적인 프롬프트를 차단하는 보안 필터링 시스템을 구현하여 서비스 안정성을 확보하였다.
- **할인 기반 추천 알고리즘 고도화:** 사용자의 프로필(카드/통신사)과 할인 정보를 매칭하여 실질적인 혜택 금액을 산출하고 이를 기준으로 최적의 매장을 우선 노출하는 할인율순 정렬 알고리즘을 구현하여 개인화 추천의 정확도를 높였다.
- **앱 연동 및 상태 관리:** 추천 결과를 안드로이드 앱의 UI/UX 흐름에 맞춰 연동하고 상태 관리

로직을 최적화하여 매끄러운 사용자 경험을 완성하였다.

마. 신유진 (데이터 파이프라인 및 DB 구축)

- **할인 정보 ETL 파이프라인 구축:** 웹상의 비정형 할인 정보를 수집하고 계산 가능한 수치 데이터로 변환하여 적재하는 크롤링-데이터정규화-데이터베이스 흐름의 주기적인 자동 데이터 수집 파이프라인을 구축하였다.
- **데이터베이스 설계:** 외부 요청에 따라 반환할 할인 정보를 저장하기 위해 제3정규화를 기반으로 할인 서비스에서 사용할 데이터베이스를 설계하고 구축하였다.
- **데이터 정규화 및 통합 관리:** 서로 다른 형식을 가진 외부 데이터들을 표준화된 스키마로 정규화하여 PostgreSQL 데이터베이스에 저장함으로써 데이터의 무결성을 보장하고 조회 속도를 최적화하였다.

제 6 장 참고문헌

- 가자마 마사히로 외, *추천 시스템 입문: GNN, RNN부터 트랜스포머까지 최신 딥러닝 기술을 활용한 추천 시스템 구축하기*, 한빛미디어, 2023.
- 마크 리처즈, 닐 포드, *소프트웨어 아키텍처 101: 엔지니어링 접근 방식으로 배우는 소프트웨어 아키텍처 기초*, 한빛미디어, 2021.
- Lewis, P., et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- Sam Newman, *Building Microservices: Designing Fine-Grained Systems*, O'Reilly Media, 2nd Edition, 2021.