AWS 1조 - N1S3

공대영 - 조장, AI 챗봇 관련 담당
김인환 - 발표자, AI 챗봇 관련 담당
박시은 - 프론트, 백엔드, 서비스 인프라 구성
서호준 - 프론트, 백엔드, 서비스 인프라 구성

travEIWith

aws

목차

- travElWith 소개(배경, 기능, 차별성)
- 시연(개요, 영상)
- 메인 아키텍처(bedrock제외)
- 백엔드 아키텍처(챗봇, 리뷰, 베드락)
- Agent 구성(Agent, Action group)
- trouble 해결
- 향후 과제/ 보완점

aws

배경

SBS - 10명 중 7명은 "국내 여행"…인기 1위 지역 어디?

국민일보 - 고물가 시대 여름휴가 키워드는 ‘국내로, 싸게, 짧게’

SBS Biz - 제주여행 인기 악화일로…관심도·점유율 7년 만에 '최저'

트렌드는 영원하지 않다!

aws

travElWith 기능 소개

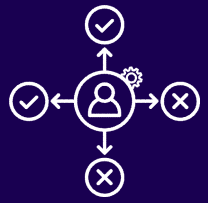
1) 여행지 추천 LLM 기반 챗봇 (Main)
   a) 질문 답변 - 사용자가 입력한 정보를 바탕으로 응답 생성
      i. 여행지 정보 제공 - 역사, 문화, 축제 등 일반 정보 제공
      ii. 여행지 및 축제 추천 - 테마, 계절, 지역 등을 고려한 추천
      iii. 여행 플래너 - 사용자의 일정을 고려한 계획 생성

2) 리뷰 게시판
   a) 익명 기반의 작성, 조회, 수정, 삭제 가능 게시판
   b) 리뷰 작성 시, 관련 내용을 HTTP API를 통한 리뷰 랭킹 업데이트

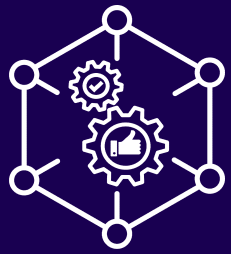
3) 실시간 인기 여행지
   a) 리뷰 기반 평점이 높은 여행지 순위

aws

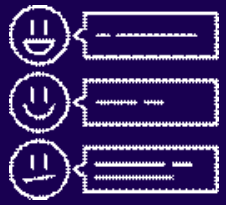
travElWith 차별성



여행지 선택 도움 - 개인화



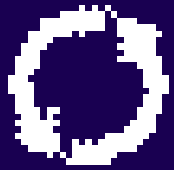
순위 계산식



리뷰 분석 - 긍정/부정

RAG를 통한 추천

리뷰 실시간 반영



사용자 참여로 서비스 고도화

aws

Chatbot concept

소개

이름은 ‘With’
20년 경력의 국내 여행 가이드
친절하고 예의바름

Prompt

사용자의 효율적 의사결정 도움
최신 정보 제공에 노력
관련 주제에 한해 응답
Hallucination 최소화

Dataset

전국관광지표준정보데이터
전국문화축제표준데이터

시연 개요

단기 날씨 확인

사용자 :
- 여행 계획을 위한 날씨 정보 요구
  - 특정 지역 + (특정 일자 or n일 뒤)

응답 예상 정보 :
- 오전/오후 요약된 날씨
- 여행을 위한 조언

축제 일정 확인

사용자 :
- 여행 계획을 위한 축제 정보 요구
  - 특정 지역 or 특정 기간

응답 예상 정보 :
- 축제 이름, 일정, 축제 내용
- 여행을 위한 조언

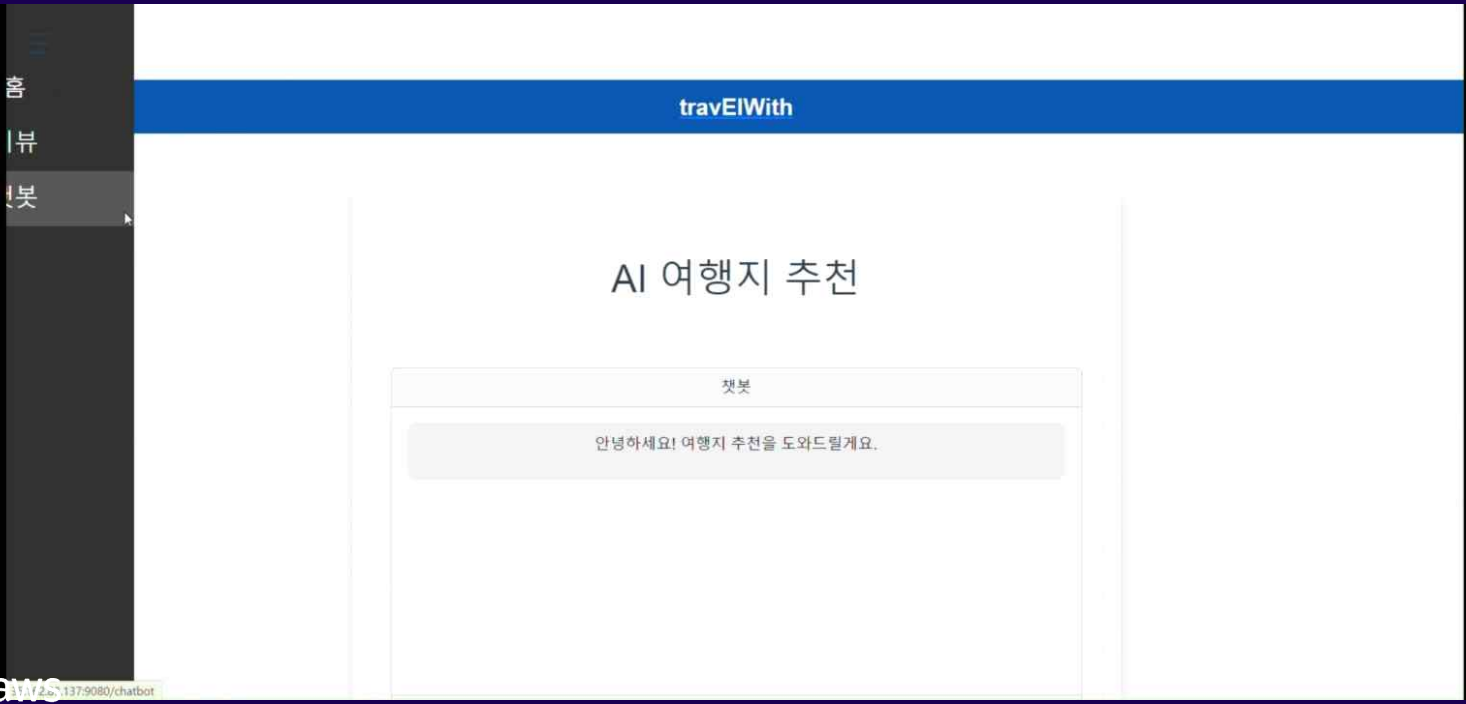
인기 관광지 순위 확인

사용자 :
- 여행 계획을 위한 정보 조회
  - 특정 여행지 순위 or 상위 5등

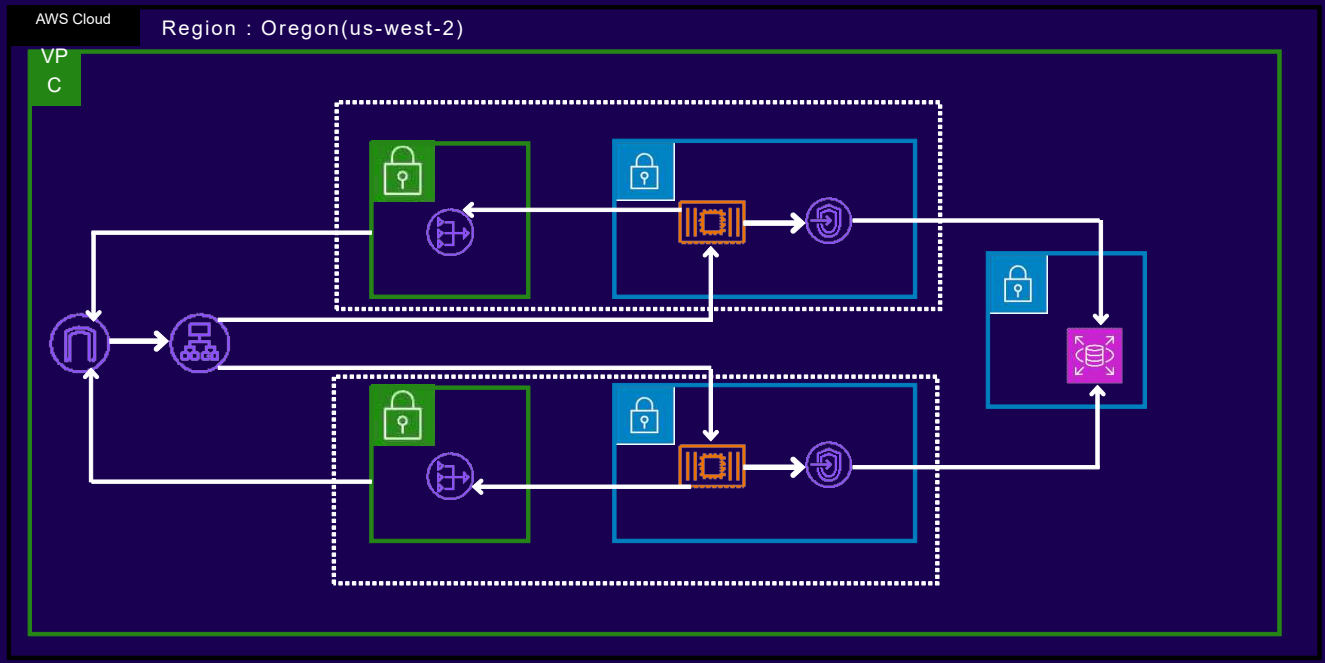
응답 예상 정보 :
- 해당 여행지의 순위 + 추가 정보
  or 상위 5개 여행지 목록

aws

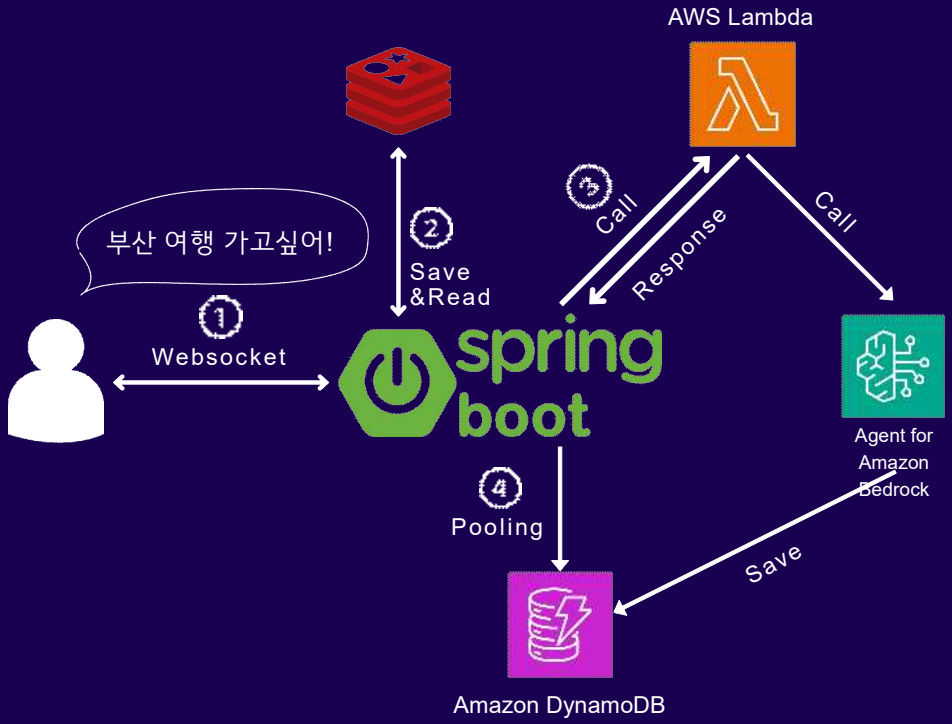
시연 동영상



Architecture

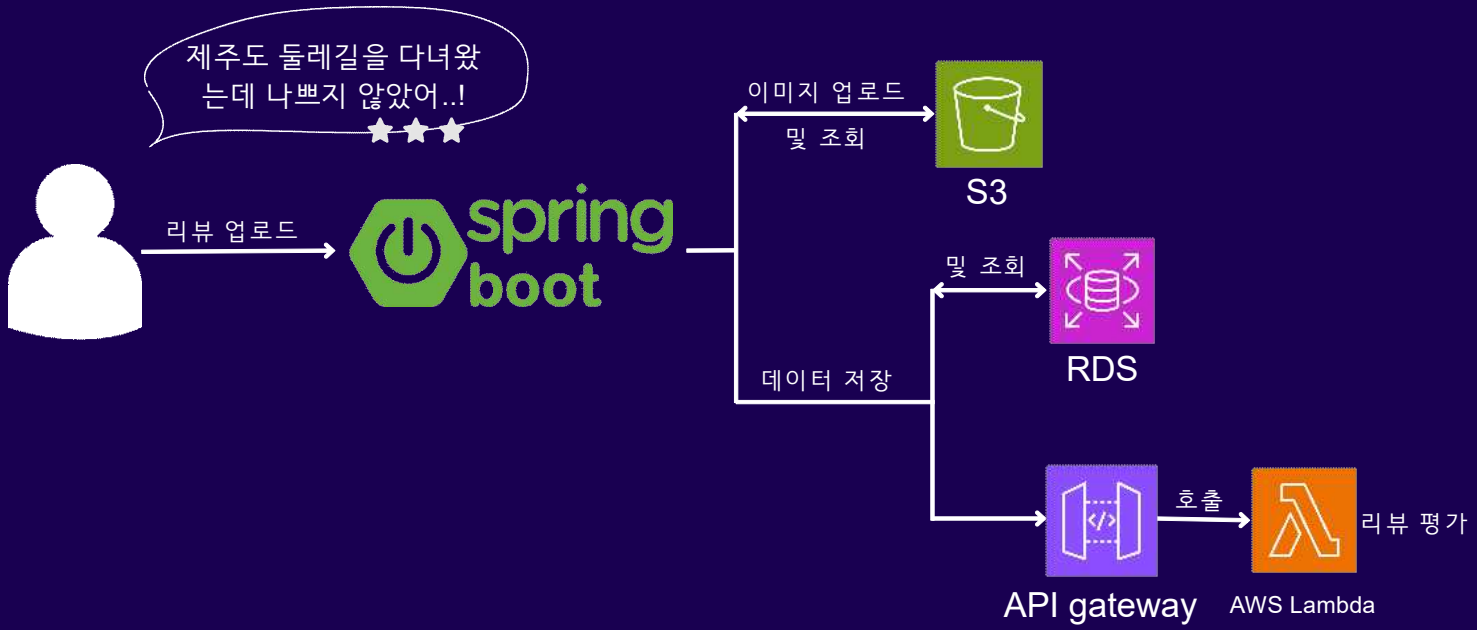


Backend

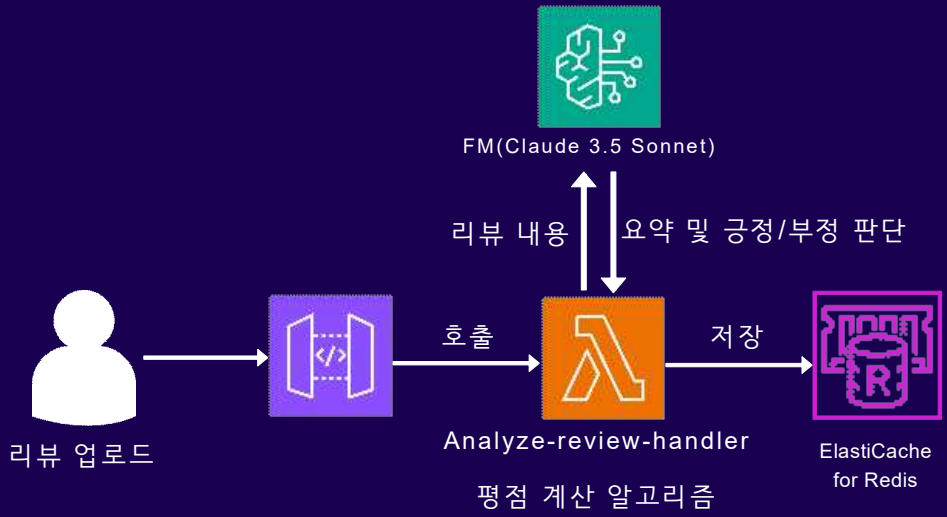


1. Pooling을 위한 Websocket 통신
2. Redis를 통한 이전 대화 내역 저장
3. Lambda를 통한 Agent 호출
4. DDB를 조회하여 답변 수신

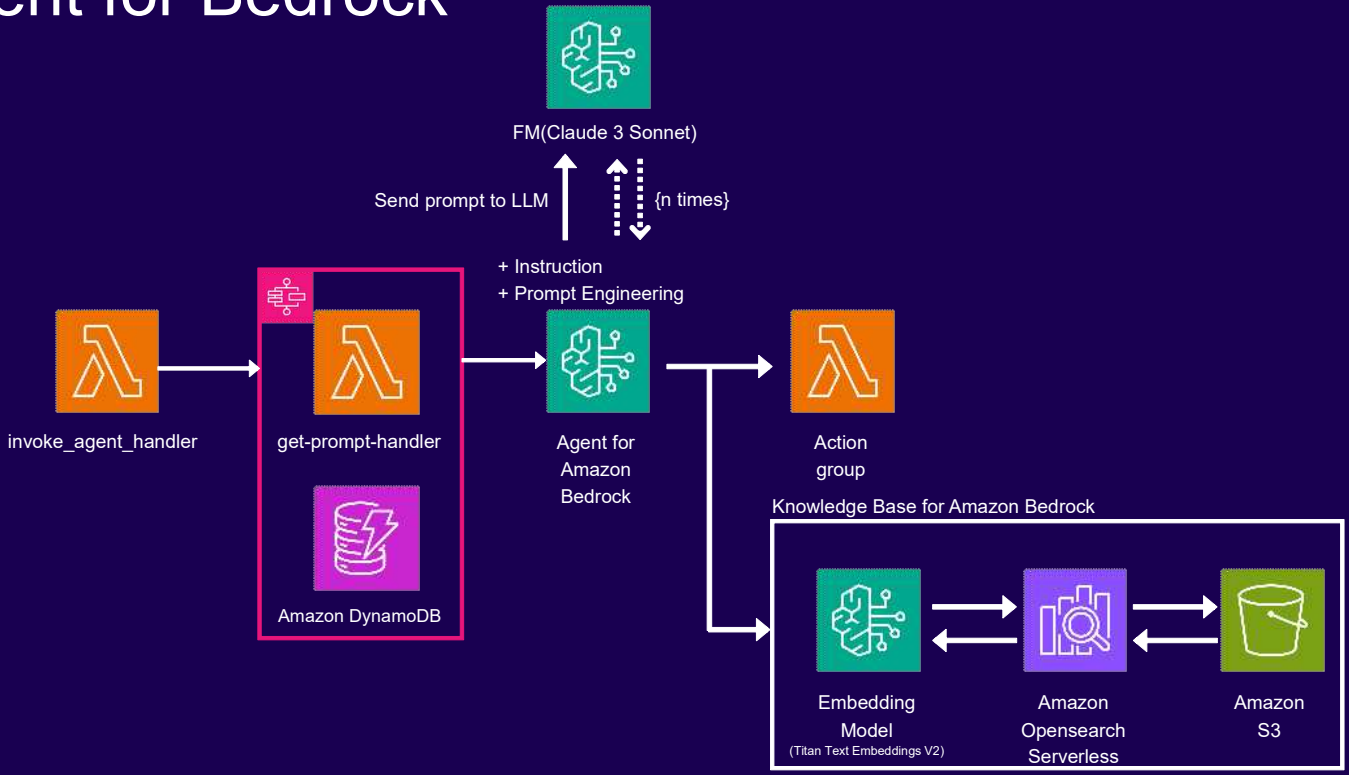
Review


제주도 둘레길을 다녀왔는데 나쁘지 않았어..!
리뷰 업로드
spring boot
이미지 업로드
및 조회
S3
및 조회
RDS
데이터 저장
호출
리뷰 평가
API gateway
AWS Lambda

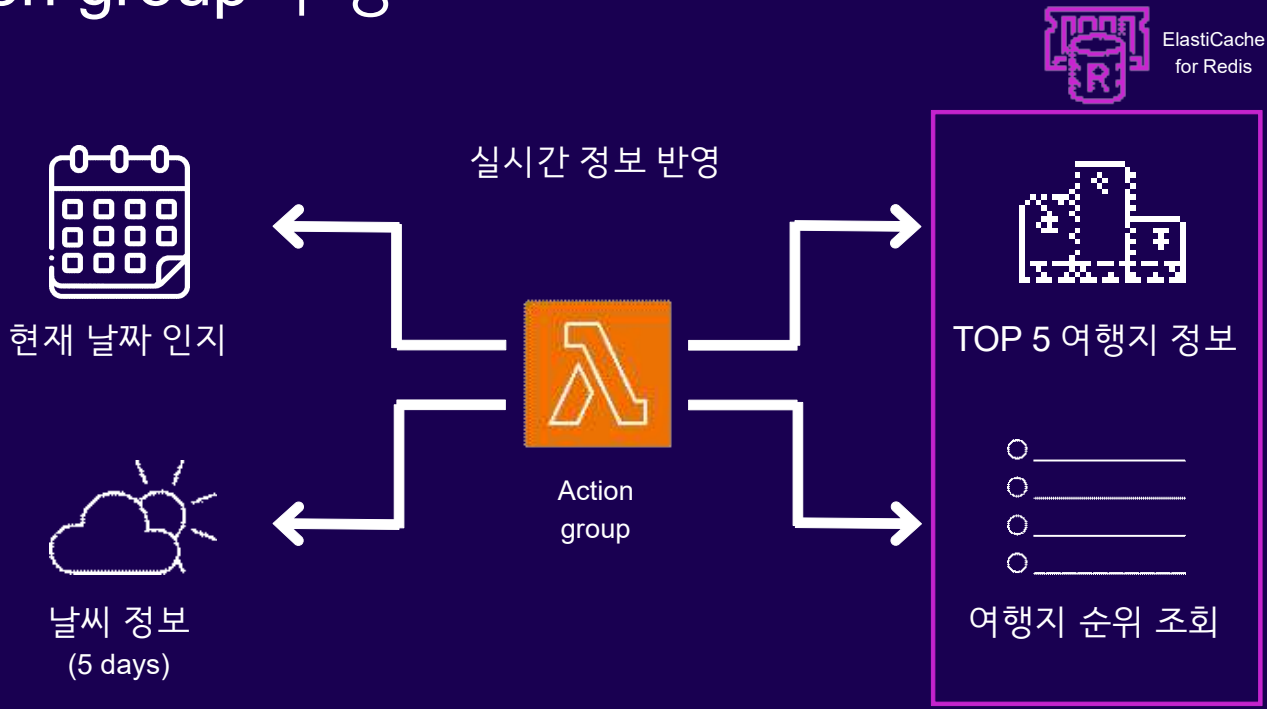
Ranking system



Agent for Bedrock



Action group 구성



ChatBot 구성

- 리전 : 오레곤(us-west-2)
- 모델 : Claud 3.0 Sonnet v1.0
- 에이전트 : AWS Bedrock Agent
- KB : AWS KnowledgeBase, OpenSearch Serverless VectorDB
- 데이터셋 : 전국관광지정보표준데이터, 축제정보 데이터셋

기술 문제 해결 - part 1

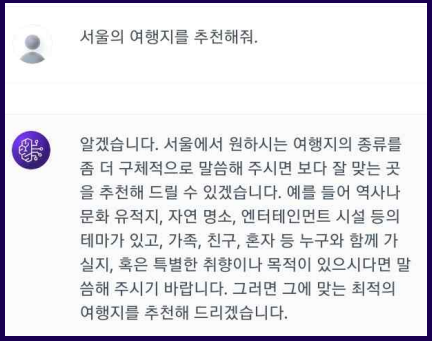
Dataset 구성

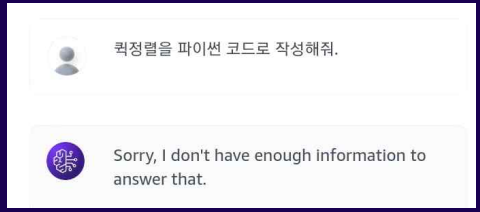
- 불필요 칼럼 제거, 중복 데이터 삭제, 병합 등 기본 전처리
- 관광지명+관광지소개 내용을 바탕으로 키워드 도출.
- Metadata 첨부

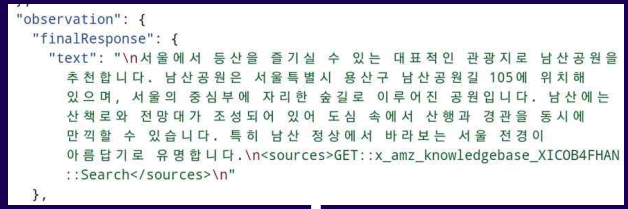
기술 문제 해결 - part 2

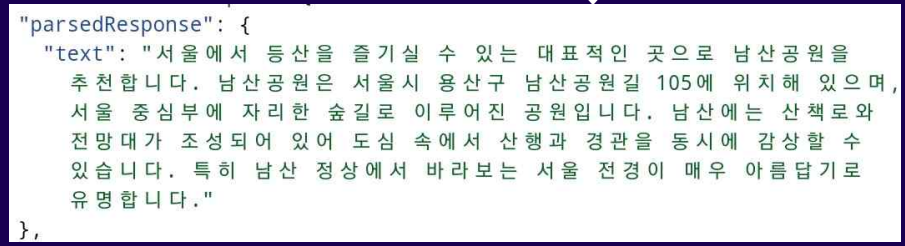
Agent 응답 관련 Advanced Prompt Customize

- 질의에서 정보가 부족한 경우, 서비스와 관련없는 질의, 불필요한 메세지 등을 Advanced Prompt를 활용해 해결하였음







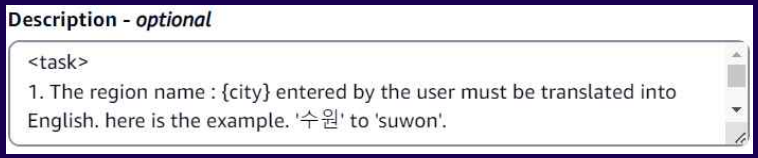


기술 문제 해결 - part 3

Action Group

- 외부 API에 전달할 prameter는 영문인데 반해 사용자는 한글로 질의하게 됨(날씨 API)
  - Action group Description에 예시를 포함해 제시함으로써 해결함

- 토큰 최적화
  - responseBody return 시 불필요한 부분을 Lambda 함수 내에서 최대한 제거




```
summarized_weather = []
for date, info in weather_info.items():
    morning_summary = summarize_weather("AM", info['morning'])
    if morning_summary:
        summarized_weather.append(f"Date:{date}, {morning_summary}")

    afternoon_summary = summarize_weather("PM", info['afternoon'])
    if afternoon_summary:
        summarized_weather.append(f"Date:{date}, {afternoon_summary}")

return "\n".join(summarized_weather)
```

기술 문제 해결 - part 4

Agent 호출 방식

문제점 1 : Backend에서 Agent를 호출하는 작업은 하기에 너무 밀결합된 Architecture가 됨
해결 : Lambda 함수를 통해 호출하도록 하는 Microservice 모듈화를 진행

- Api gateway를 trigger로, HTTP POST Api를 호출해서 Lambda 함수를 call
- 이 Lambda 함수가 Agent를 invoke하고 결과를 받도록 구현

But 문제점 발생. API GW에서는 최대 30초까지의 Timeout Limit이 존재. 에이전트의 질의 시간이 길어지면 통신이 종료

기술 문제 해결 - part 4

Agent 호출 방식

문제점 2 : API GW
Timeout
해결 : Agent를 호출하고 결과를 받아오는 과정을 비동기식으로 처리

- 람다 함수를 2개로 분할, HTTP Post API를 받는 Invoke-agent-handler Lambda에서 step function을 invoke해서 비동기식으로 다음 람다 함수를 진행
- get-prompt-handler 람다 및 상태머신에선 실제로 Agent와 통신을 진행하고 그 질의 결과 정보를 DynamoDB로 putItem하게 정의
- 백엔드에서 지속적으로 풀링하면서 session id가 같고 질의시간보다 더 나중에 수신된 DynamoDB Item을 식별하고 가져와서 프론트엔드에 띄우는 식으로 구현

향후 과제 및 보완점

Agent의 질의 속도를 개선

- Latency를 최소화 하도록 최적화
- Agent의 응답속도를 높이기 위한 방법
  - Dataset을 작은 단위로 쪼개 Metadata를 활용하는 방식?
  - 벡터 DB나 Chunking 방식의 변화

현재 시스템은 완전관리형이라 최적화에 여러 문제점이 존재
그래서 LangChain을 사용하는 방향으로 보완 가능할 것으로 예상