SW개발/HW제작 설계서

프로젝트 명 : 공연예술통합 정보 제공 AI 챗봇 서비스

2022. 07. 10

챗지니어스 – 김가현 김수민 이승은 허영은
Mentor 서지훈

수행 단계별 주요 산출물

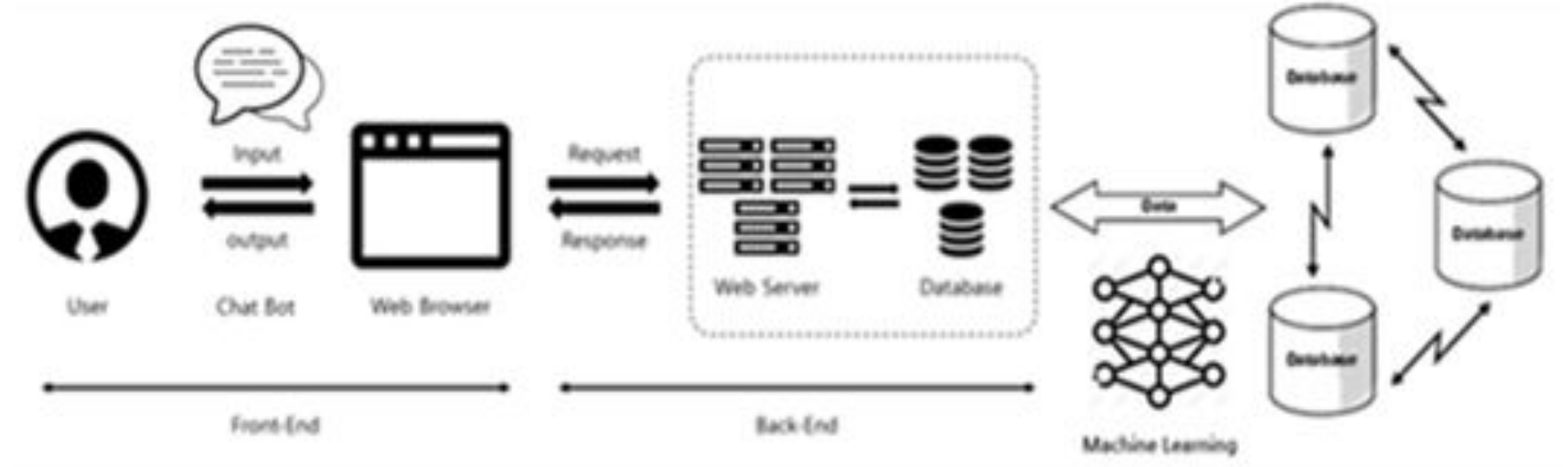
단계	산출물	일반 ·모바일 APP ·Web 등	응용 소프트웨어 ·빅데이터 ·인공지능 ·블록체인 등	응용 하드웨어 ·IoT ·로봇 ·드론 등
환경 분석	시장/기술 환경 분석서	△	△	△
	설문조사 결과서	△	△	△
	인터뷰 결과서	△	△	△
요구사항 분석	**요구사항 정의서**	○	○	○
	유즈케이스 정의서	△	△	△
아키텍처 설계	**서비스 구성도(시스템 구성도)**	○	○	○
	서비스 흐름도(데이터 흐름도)	△	○	△
	UI/UX 정의서	△	△	△
	하드웨어/센서 구성도	-	-	○
기능 설계	**메뉴 구성도**	○	○	○
	화면 설계서	○	○	△
	엔티티 관계도	○	○	△
	기능 처리도(기능 흐름도)	○	○	○
	알고리즘 명세서/설명서	△	○	○
	데이터 수집처리 정의서	-	○	-
	하드웨어 설계도	-	-	○
개발 / 구현	**프로그램 목록**	○	○	○
	테이블 정의서	○	○	△
	핵심 소스코드	○	○	○

※ ○ 필수, △ 선택

| 요구사항 정의서

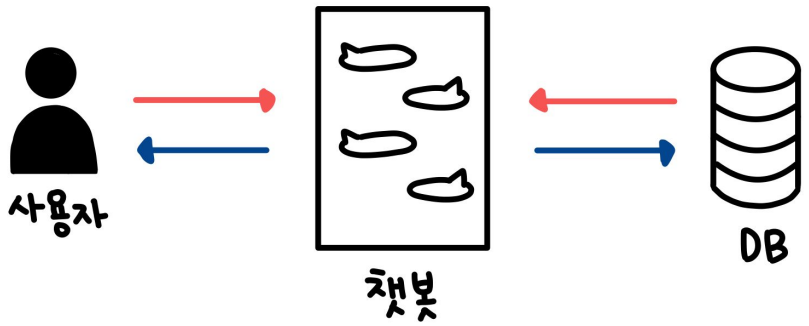
구분	기능	설명
S/W	제목에 따른 정보 제공	DBMS에 있는 공연 제목에 따른 data를 사용자가 검색할 수 있다.
	가격에 따른 공연 추천	가격대를 지정하여 그 가격대에 맞는 공연을 추천할 수 있다.
	공연 기간에 따른 공연 추천	기간을 지정하여 기간 동안 공연중인 공연을 추천할 수 있다.
	장르에 따른 공연 추천	장르에 따라 DB의 data를 이용하여 공연을 추천할 수 있다.
	지역에 따른 공연 추천	전국 지역에 따라 DB의 data를 이용하여 공연을 추천할 수 있다.
	인기 순위	현재 공연 중인 공연의 인기 순위를 제공할 수 있다.

| 서비스 구성도 – 서비스 시나리오



1. 사용자가 챗봇에 공연예술 정보에 대한 질문 입력 (front-end)
2. Dialogflow에서 자연어 처리 분석 후 웹 서버에 데이터 요청
3. 서버에서 Database에 저장된 데이터를 브라우저에 전달 (back-end)
4. 전달받은 데이터를 통해 Dialogflow에서 답변을 만들어 챗봇에 전달
5. 챗봇에서 사용자에게 답변 출력 (front-end)

| 서비스 흐름도



<제목에 따른 정보 제공>

1. 사용자가 정보를 제공받고 싶은 공연의 제목을 챗봇에 입력한다.
2. 해당 공연에 대한 정보를 데이터베이스에서 검색한다.
3. 정보가 존재하면 공연 장르, 공연 상태, 공연 시작일, 공연 종료일, 공연 포스터, 공연시설명을 챗봇에 제공한다.
4. 챗봇에서 정보를 출력한다.

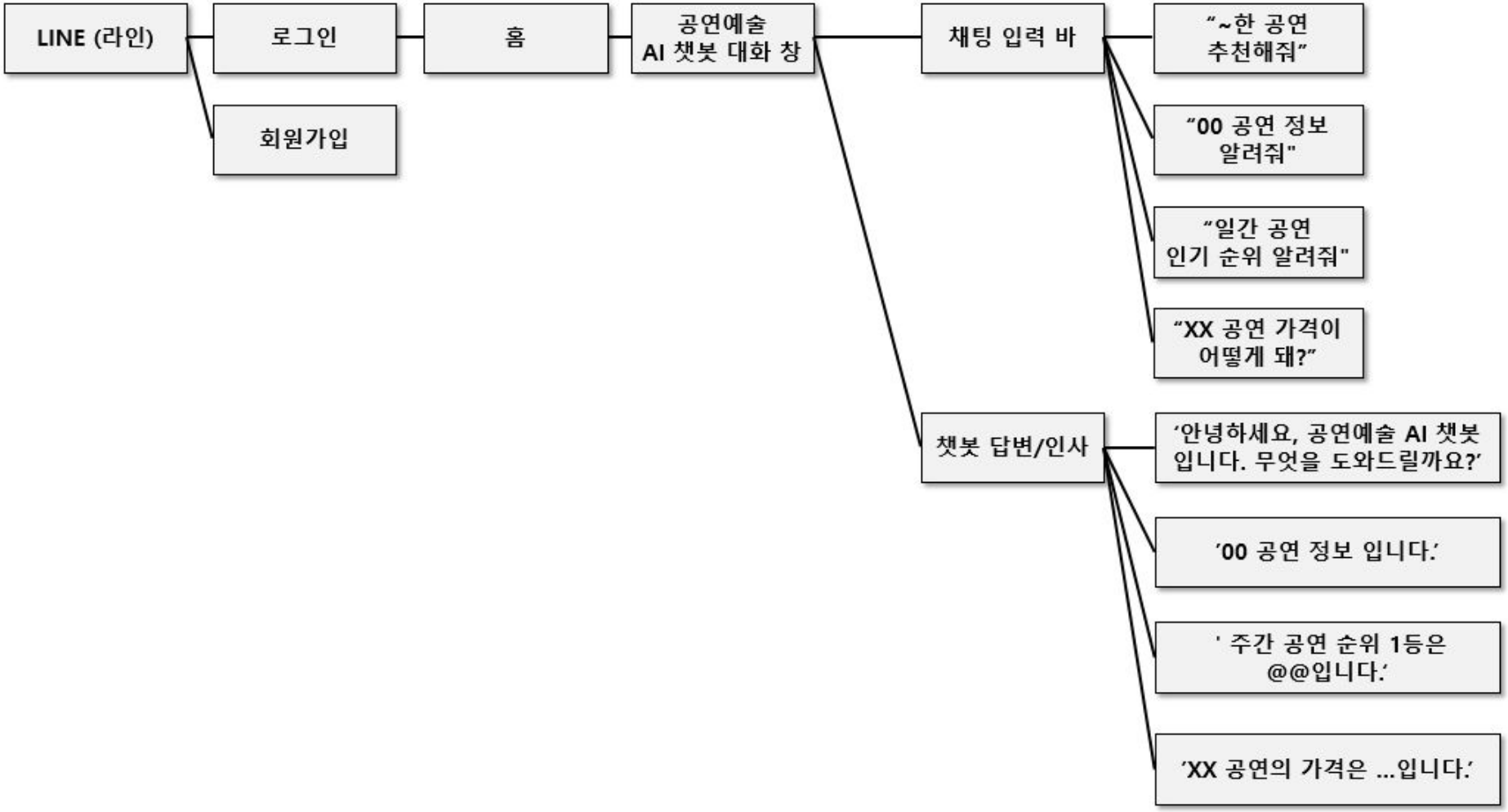
<가격, 공연 기간, 장르, 지역에 따른 공연 추천>

1. 사용자가 원하는 가격, 공연 기간, 장르, 지역을 입력한다.
2. 해당 정보에 대한 공연이 존재하는지 데이터베이스에 검색한다.
3. 해당하는 공연이 존재하면 공연 제목과 정보를 챗봇에 제공한다.
4. 챗봇에서 정보를 출력한다.

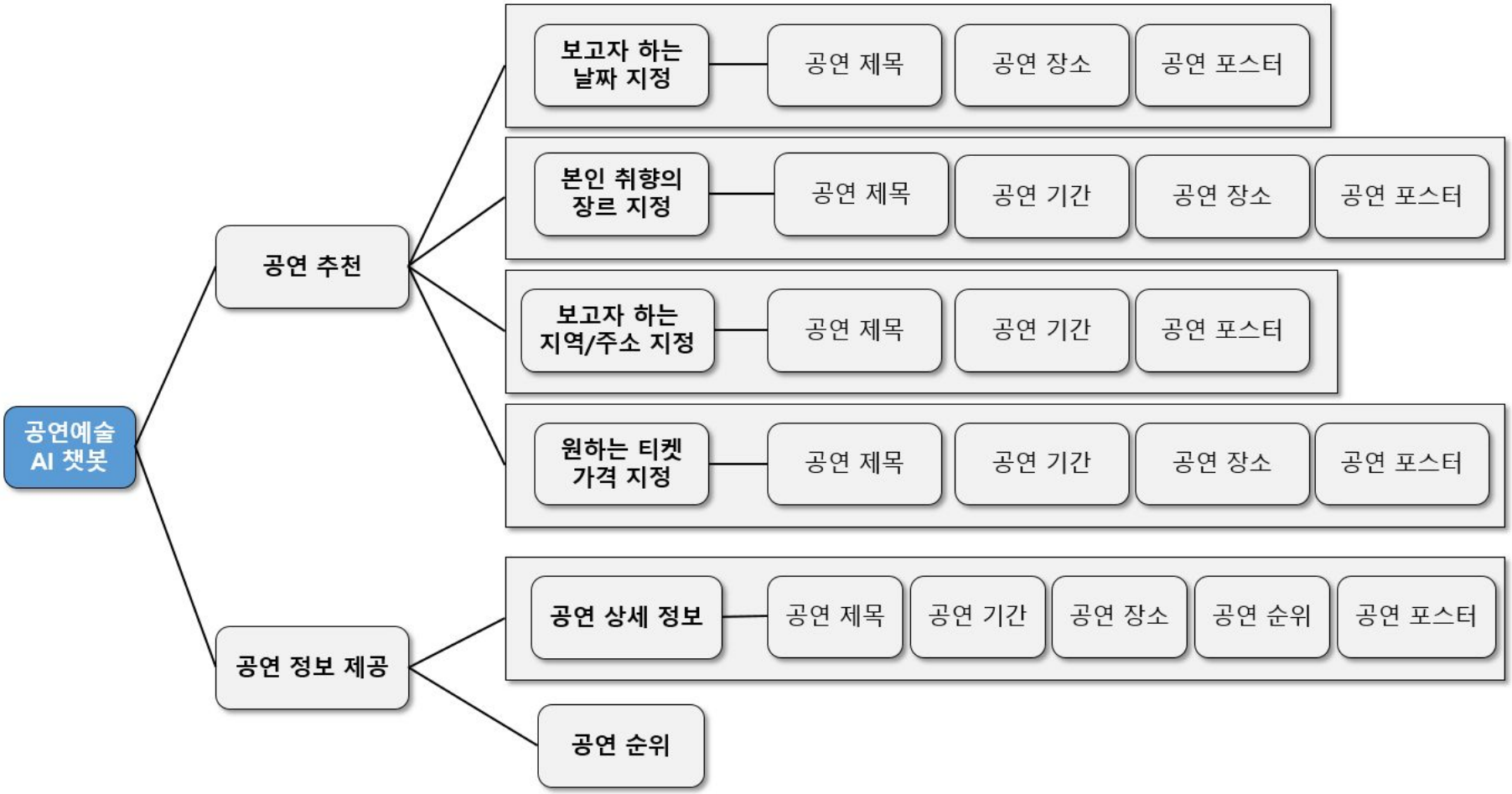
<인기 순위>

1. 사용자가 현재 상영하는 공연의 인기 순위를 알려달라고 챗봇에 입력한다.
2. 현재 박스오피스 순위 1~5위의 공연 명, 공연 장르를 챗봇에 제공한다.
3. 챗봇에서 정보를 출력한다.

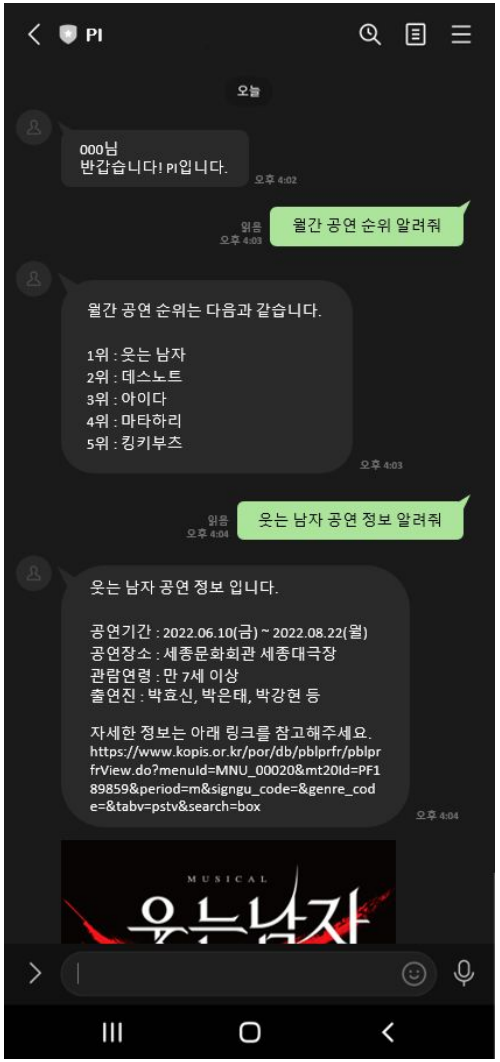
| 메뉴 구성도


LINE (라인)
로그인
회원가입
홈
공연예술
AI 챗봇 대화 창
채팅 입력 바
"~한 공연
추천해줘"
"00 공연 정보
알려줘"
"일간 공연
인기 순위 알려줘"
"XX 공연 가격이
어떻게 돼?"
챗봇 답변/인사
'안녕하세요, 공연예술 AI 챗봇
입니다. 무엇을 도와드릴까요?'
'00 공연 정보 입니다.'
' 주간 공연 순위 1등은
@@입니다.'
'XX 공연의 가격은 ...입니다.'

| 메뉴 구성도


공연예술 AI 챗봇
공연 추천
보고자 하는 날짜 지정
공연 제목
공연 장소
공연 포스터
본인 취향의 장르 지정
공연 제목
공연 기간
공연 장소
공연 포스터
보고자 하는 지역/주소 지정
공연 제목
공연 기간
공연 포스터
원하는 티켓 가격 지정
공연 제목
공연 기간
공연 장소
공연 포스터
공연 정보 제공
공연 상세 정보
공연 제목
공연 기간
공연 장소
공연 순위
공연 포스터
공연 순위

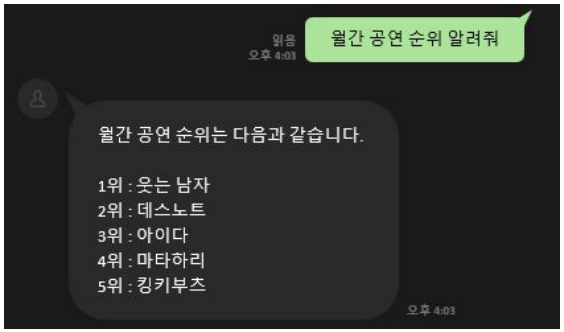
| 화면 설계서



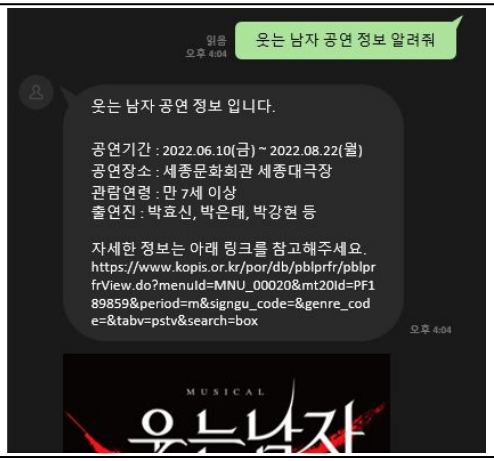
기능 번호	CHAT-01-01
기능 명	공연예술 통합 정보 제공 AI 챗봇과 대화
기능설명	모바일 메신저 어플리케이션 “LINE” 대화 창 “LINE” 로그인/회원가입 후, 챗봇을 친구로 추가하면, 공연예술 통합 정보를 챗봇과의 대화를 통해 수집할 수 있다.
처리내용	▌”LINE” 로그인/회원가입 및 챗봇 친구 추가 본인의 계정으로 로그인 후, 챗봇과의 대화를 시작한다. 이는 “LINE”에서 기본적으로 제공하는 기능이므로 제작 과정은 생략한다. ▌공연예술 정보 얻기 챗봇에게 공연 순위, 공연 정보(기간, 장소, 관람연령, 출연진, 포스터, KOPIS링크 등)를 묻거나, 본인이 원하는 조건을 충족하는 공연을 추천해줄 수 있다. “~ 알려줘”, “~ 부탁해”, “~가 뭐야?”, “~는 어떻게 돼?” 등 사람마다 다양한 문체를 쓰기 때문에, 이를 챗봇이 잘 알아듣도록 만든다.
비고	X
요구사항 명	로그인/회원가입, 챗봇 친구 추가

| 화면 설계서

▶ 사용 예시 정보



설명	사용자가 요즘 공연의 인기 순위를 알고싶어, 챗봇에게 월간 공연 순위를 알려달라고 채팅창에 입력하면, KOPIS 오픈 API에 따라 월간 공연 순위를 1위부터 5위까지 출력한다.
입력 데이터 값	본인이 원하는 기간별(일간, 주간, 월간…) 인기 공연 순위를 알려달라고 입력



설명	사용자가 특정 공연이 요즘 인기가 많아, 이 공연에 대한 정보를 알고 싶어, 챗봇에게 특정 공연 정보를 알려달라고 채팅창에 입력하면, KOPIS 오픈 API에 따라 그 공연의 공연 시간, 장소, 관람연령, 출연진, 자세한 정보가 담긴 링크, 그리고 공연 포스터를 출력해준다.
입력 데이터 값	특정 공연 제목(ex. 웃는 남자)에 대한 정보를 알려달라고 입력

▶ 입출력 데이터 항목

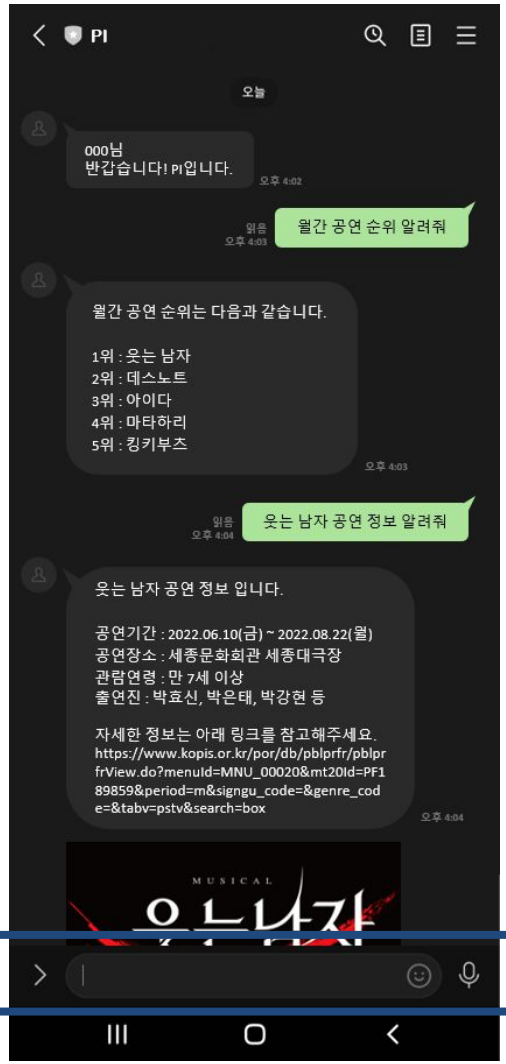
입출력구분	항목	설명
입력	궁금한 공연, 추천받고자 하는 공연 조건	사용자가 알고자 하는 공연 및 추천받고자 하는 공연 조건 입력
출력	공연 정보, 추천 공연	입력한 공연 제목의 공연 정보 출력 및 입력한 조건에 맞는 공연 추천

▶ 선행필수기능 및 예외사항

기능명 / 예외사항	설명
사용자가 입력한 대화가 챗봇이 이해하기 힘듦(오타 등)	알아듣기 힘들다는 대화를 출력하고, 오타나 알아듣기 힘든 언어가 있는지 확인하도록 한다.
없는 공연 정보일 때	챗봇이 알려줄 수 없는 정보(API에 없는 데이터)라는 것을 알려준다.

화면설계서 – 사용자 인터페이스(sw

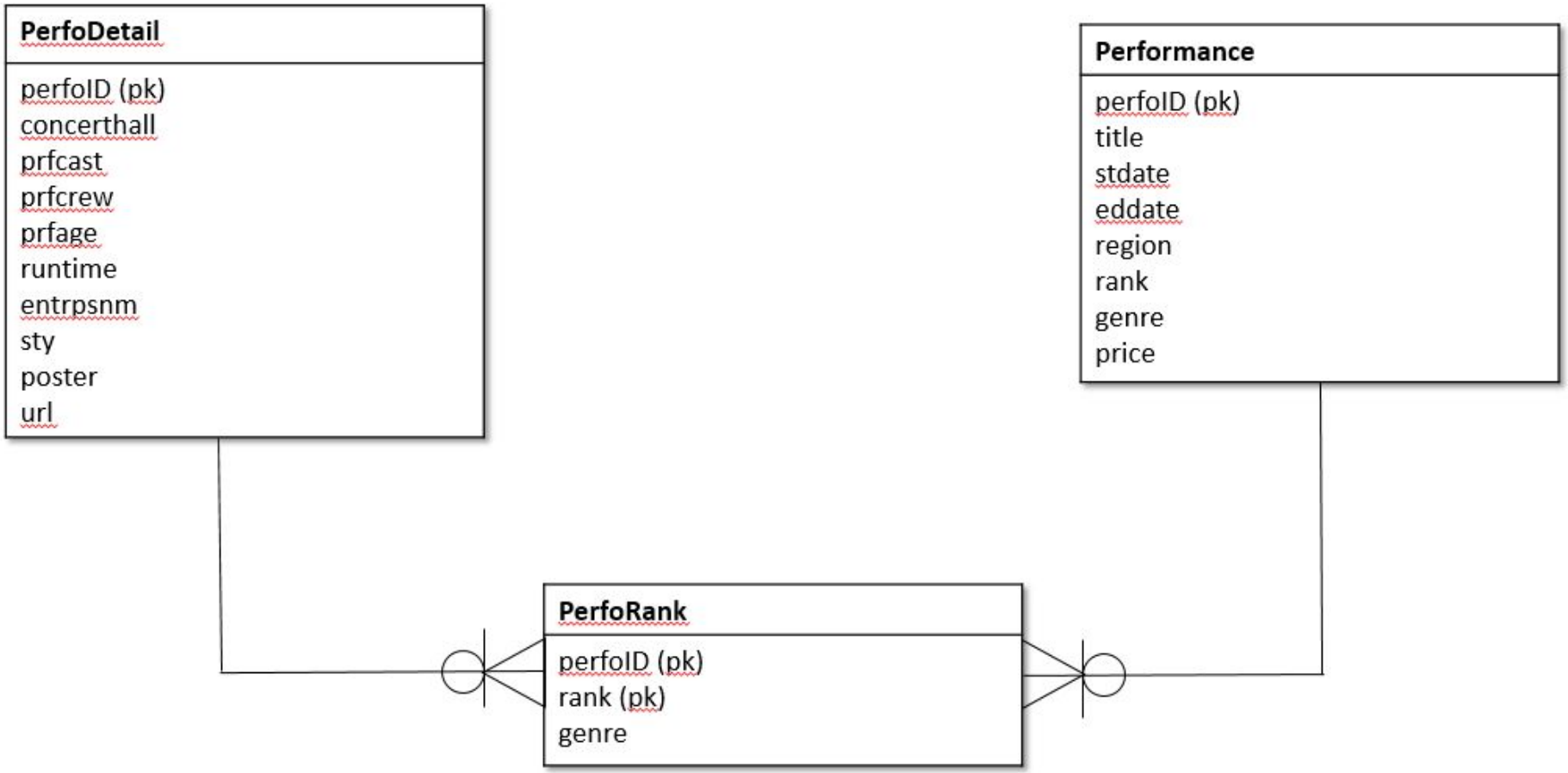
선택 -> 채팅창으로 이동



여기에 채팅 입력하며
AI 챗봇과 대화

| 엔티티관계도 - ERD

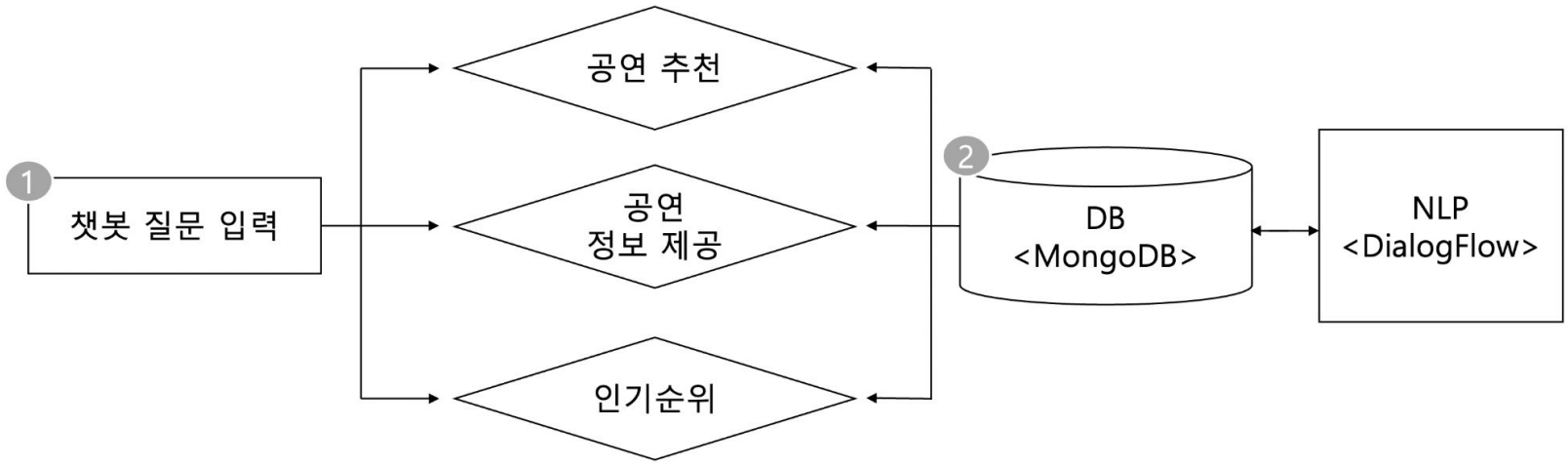
ERD(Entity-Relationship Diagram)



| 기능 처리도(기능 흐름도)

기능 흐름도

< 공연 예술 AI 챗봇 기능 처리도 >

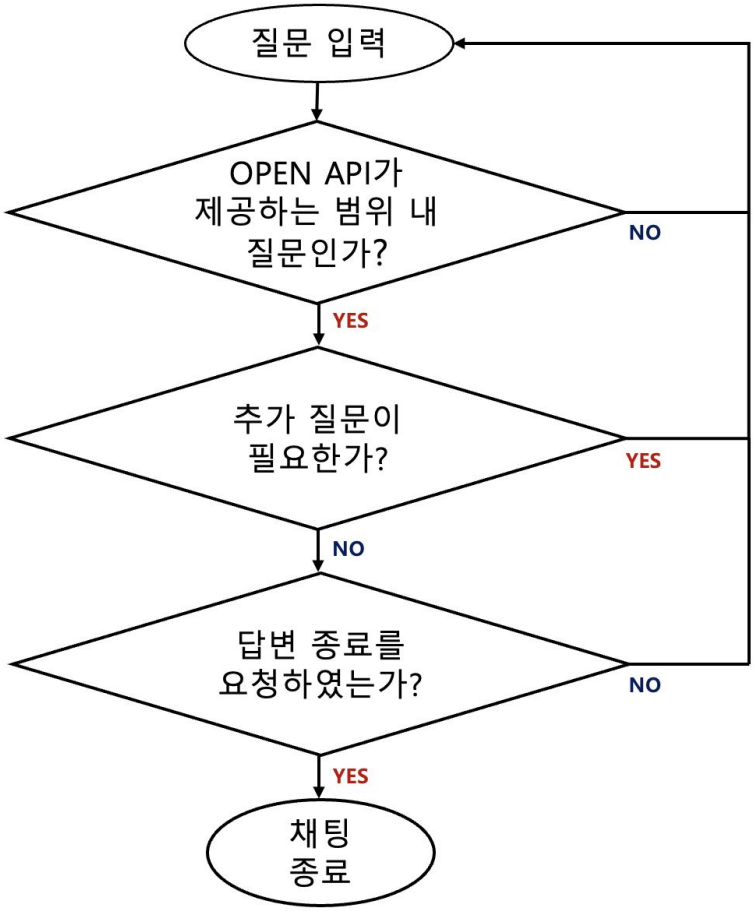


1 챗봇을 실행하여 공연에 관련된 질문을 입력한다.

2 자연어 처리 후 DB에 저장되어있는 데이터를 바탕으로 답변에 응한다.

| 알고리즘 명세서

< 공연 예술 AI 챗봇 알고리즘의 흐름도>



< 알고리즘 시나리오 >

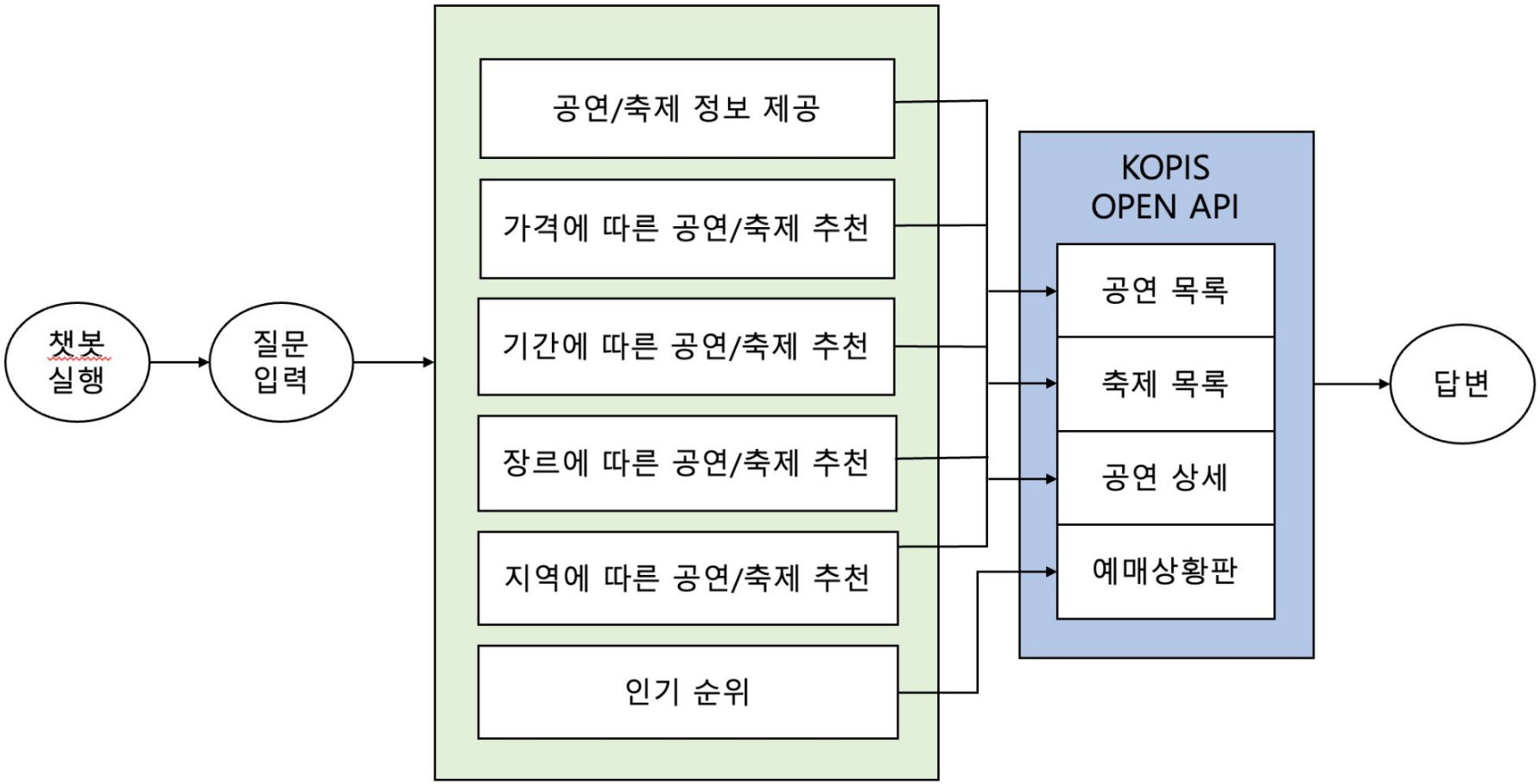
1.채팅창에 질문을 입력한다.

2.질문한 내용이 KOPSI OPEN API에서 제공하는 범위 내의 질문이면 챗봇이 답변을 하고,
아니라면(ex. API에 존재하지 않는 공연/축제에 대한 질문을 한 경우) 질문을 다시 입력해달라는 채팅창이 나오게 된다.

3. 추가 질문이 필요한 경우 질문을 이어 입력하면 된다.

4. 추가 질문이 필요 없어 답변 종료를 요청하였다면 채팅은 종료된다.

| 데이터 수집처리 정의서


챗봇
실행
질문
입력
공연/축제 정보 제공
가격에 따른 공연/축제 추천
기간에 따른 공연/축제 추천
장르에 따른 공연/축제 추천
지역에 따른 공연/축제 추천
인기 순위
KOPIS
OPEN API
공연 목록
축제 목록
공연 상세
예매상황판
답변

| 프로그램 - 목록

기능 분류	기능번호	기능 명
Rank	ranking	인기 공연 순위
Question	SER-title	공연 제목에 따른 검색
	SER-price	가격에 따른 검색
	SER-period	공연 기간에 따른 검색
	SER-region	지역에 따른 검색
	SER-genre	장르에 따른 검색
	SER-concerthall	공연장에 따른 검색
	SEARCH	통합 검색
이하 생략		

| 테이블 정의서 – ERD

- 테이블 구성도

Performance	
pk	perfoID
	title
	stdate
	eddate
	region
	rank
	genre
	price

PerfoDetail	
Pk	perfoID
	concerthall
	prfcast
	prfcrew
	prfage
	runtime
	entrpsnm(제작사)
	sty (줄거리)
	poser
	url

<Performance table>

항목명	Type	필수/선택	값 목록	활성여부	설명
perfoID	Int	필수	34231,34534 ...	활성	pk
title	char	필수	유진과 유진, ...	활성	공연제목
price	int	필수	110,000, ...	활성	공연가격
startdate	date	필수	2022-01-01, ...	활성	공연시작일
enddate	date	필수		활성	마지막공연일
rank	int	선택	1,2,3 ...	활성	인기 순위
region	char	필수	서울, 광주, ...	활성	지역
genre	char	필수	뮤지컬, 연극, ...	활성	장르

| 핵심소스코드(1)

api를 mongodb에 저장

```python
from flask import Flask
from pymongo import MongoClient
import requests
import json
import xmltodict

client = MongoClient('localhost', 27017)
db = client.chatbot
collection = db.perfromance


url = 'http://www.kopis.or.kr/openApi/restful/pblprfr'
service_key="66122ee6118e42288de23913ed3f24fb"

params = {
    'service' : service_key,
        'stdate' : 20220101,
        'eddate' : 202201231,
        'rows' : 100,
        'cpage': 1
}
```

```python
res = requests.get(url=url,params=params)
data = res.text
jsonString = json.dumps(xmltodict.parse(data), indent=4, ensure_ascii = False)


perfList = json.loads(jsonString)
perfos = perfList["dbs"]
perfosarray = perfos['db']

for perf in perfosarray:
    perfoID = perf["mt20id"]
    title = perf["prfnm"]
    stdate = perf["prfpdfrom"]
    eddate = perf["prfpdto"]
    genre = perf["genrenm"]
    concerthall = perf["fcltynm"]

    perfo = {'perfoID':perfoID,'title':title, 'stdate':stdate ,'eddate':eddate ,
        'genre':genre, 'concerthall':concerthall}
    print(perfo)
    db.performance.insert_one(perfo)
```

| 핵심소스코드(2)

flask와 dialogflow연결

```python
from flask import Flask, request,jsonify,make_response
from pymongo import MongoClient
import json
from MakeApiRequests import MakeApiRequests

app = Flask(__name__)

client = MongoClient('localhost', 27017)
db = client.chatbot
collection = db.perfromance


@app.rout('/')
def hello():
    return 'hello world!'


@app.route('/webhook',methods=['GET','POST'])
def webhook():
    req = request.get_json(force=True)
    res = processRequest(req)
    res = json.dumps(res, indent=4)
    print(res)
    responce=make_response(res)
    responce.headers['Content-Type'] = 'application/json'
```

```python
def processRequest(req):
  result = req.get("queryResult")
  intent = result.get("intent").get('displayName')
  parameters = result.get("parameters")
  db = configureDataBase()

  if intent == 'perfomance_title':
        pf_title = parameters.get("title")

        fulfillmentText = makeAPIRequest(pf_title) #공연정보 받아오기
        webhookresponse = "***performance information*** \n\n" + " title :" + \
             + str( fulfillmentText.get('title')) + "\n" + "start date : " + str(
             fulfillmentText.get('stdate')) + "\n" + " end date : " + str(fulfillmentText.get('eddate')) + \
             "\n" + " genre : " + str(fulfillmentText.get('genre')) + \
             "\n" + " concerthall : " + str(fulfillmentText.get('concerthall')) + "\n"
        print(webhookresponse)

        return {
        "fulfillmentText": webhookresponse
        }

def makeAPIRequest(query):
  api = MakeApiRequests.Api()
  if query == "title":
    return api.makeApiRequestForTitle(query)


if __name__ =='__main__':
    #port = int(os.getenv('PORT',80))
    app.run(host='0.0.0.0',port=5000)
```

Thank you