

AI 기반 비대면 의료 서비스

MEDITACT

강재현 김성일 김찬호 여종현 이현훈



CONTENTS

01 프로젝트 및 팀원 소개

02 본 프로젝트의 목적

03 프로젝트의 방향성

사용자 중심의 챗봇

진료과 분류 AI

비대면 상담 서비스

온라인 예약 기능

04 개발 과정

딥러닝

인프라

백엔드

프론트엔드

05 Agile 개발

Agile 개발 프로세스

개발 계획 수립

개발 진행

자가 평가

06 Meditact의 미래

라이센스

공개 SW

챗봇 상용화

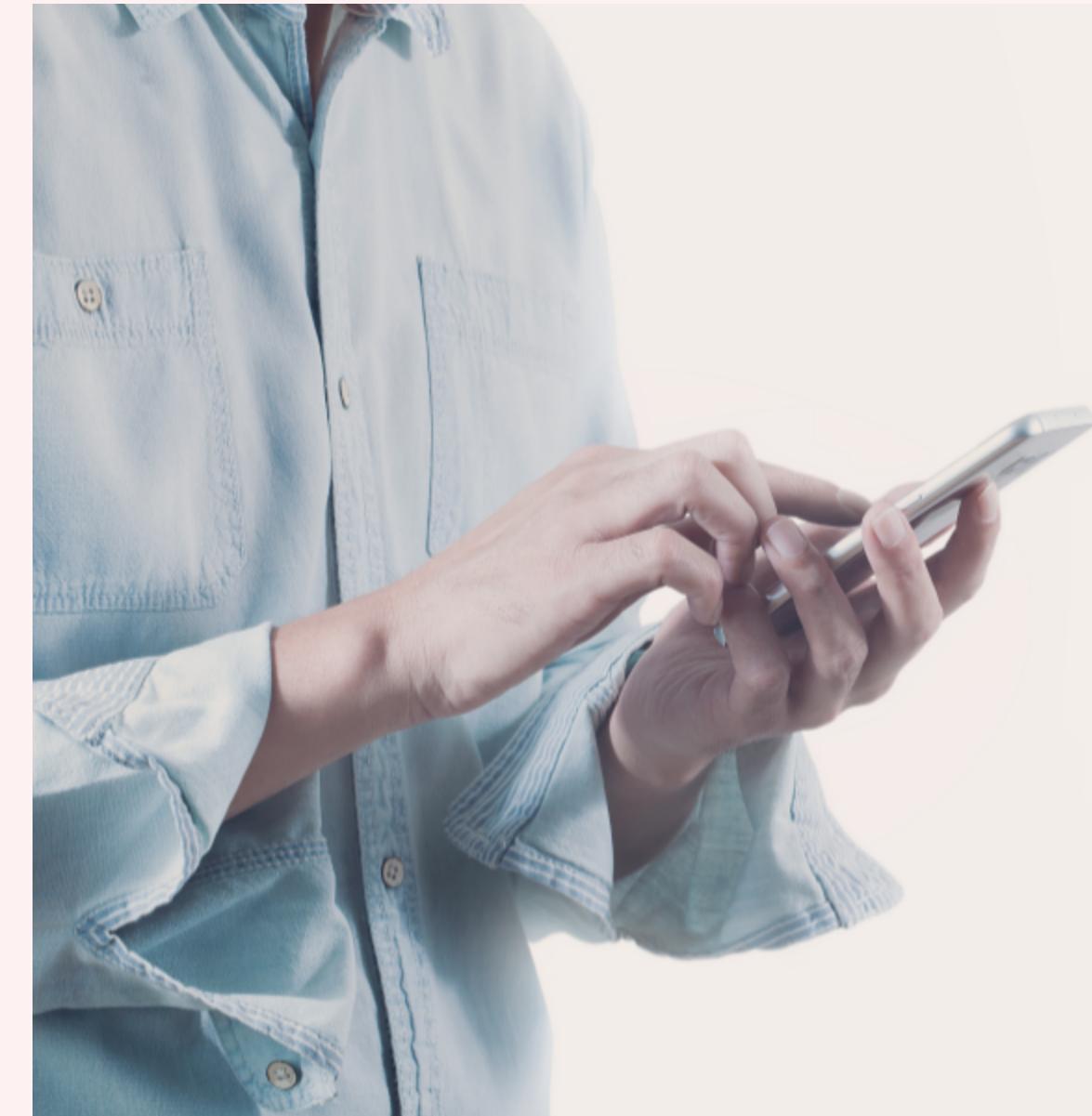
데이터셋 공개

07 해커톤을 마치며

팀 프로젝트

국방 개발 환경

Open your code, Open our future



01

프로젝트 및 팀원 소개

프로젝트 및 팀원 소개

프로젝트 소개: Meditact란?



Medicine

+



Untact

Meditact은 Medicine과 Untact를 합친 합성어로, 국군 의료 서비스의 편리함을 향상시키기 위한 프로젝트입니다.

프로젝트 및 팀원 소개

프로젝트 소개: Meditact 로고

- 육각형
 - 굳건한 국방과 안정적인 사회를 의미
- 맞잡은 두 손
 - 언택트 사회에서도 이어지는 의사와 환자의 신뢰를 의미

- 심장 (하트)
 - 건강한 삶을 의미



Meditact

프로젝트 및 팀원 소개

팀원 소개

*가나다 순

강재현

상병
Deep Learning
Github : Ashhyun

김성일(팀장)

상병
Web-Backend
Github : Kshired

김찬호

일병
Web-Frontend
Github : Chanhoo

여종현

상병
Infra
Github : Mindgitwx

이현훈

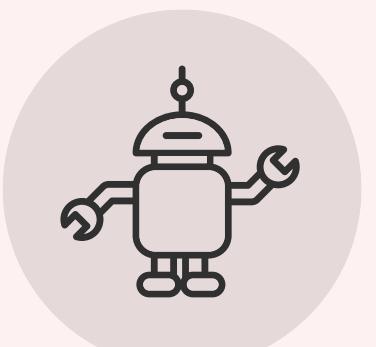
대위 (군의관)
Deep Learning
Github : Hyeonhoonlee

02

본 프로젝트의 목적

본 프로젝트의 목적

프로젝트 목적 4가지



사용자 중심 챗봇

트리형 구조의 직관적인 UI
환자가 본인에게 필요한 서비스를
주도적으로 이용
꼭 필요한 부분에만 딥러닝 접목



진료과 추천 AI

딥러닝에 기반한 증상 분석
어느 진료과에서
최상의 치료를 받을 수 있는지 추천



비대면 상담

GOP/GP 근무와 같은 지리적 한계 극복
COVID-19와 같은 재난상황 극복
군장병의 건강을 보장하기 위한 상담 서비스



온라인 예약

모든 장병들에게 진료 예약 서비스 제공
진료 대기 시간 최소화
최상의 의료 서비스 경험

03

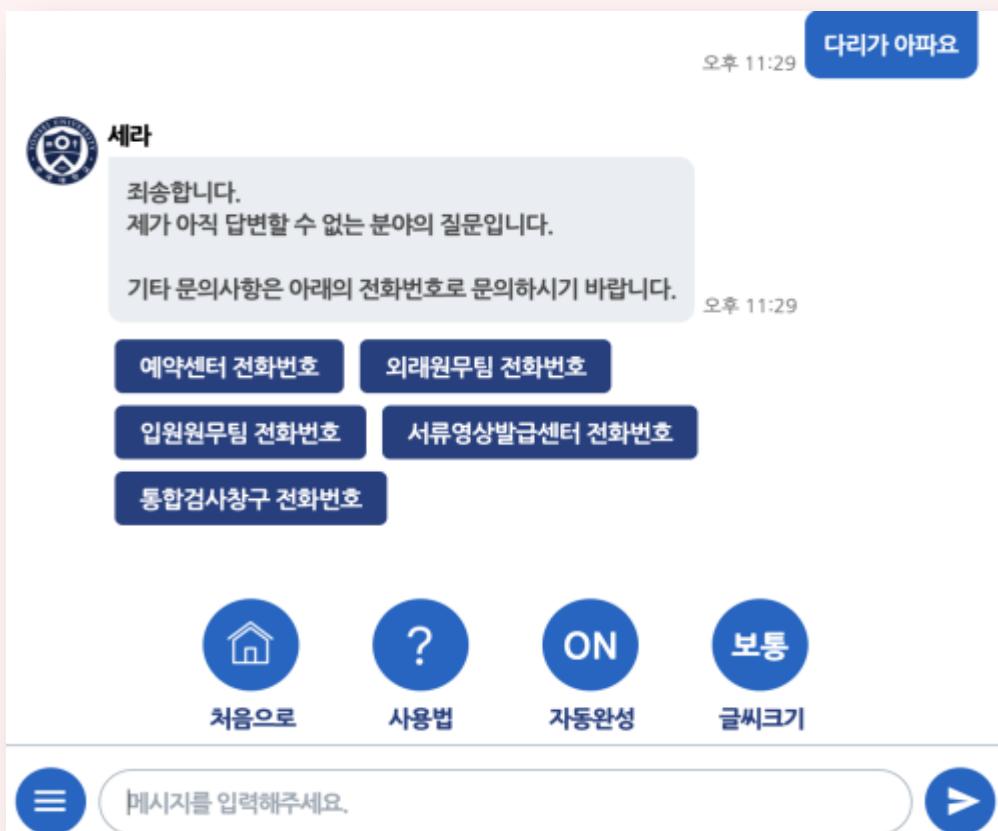
프로젝트의 방향성

사용자 중심의 챗봇
진료과 추천 AI
비대면 상담 서비스
온라인 예약 기능

프로젝트의 방향성

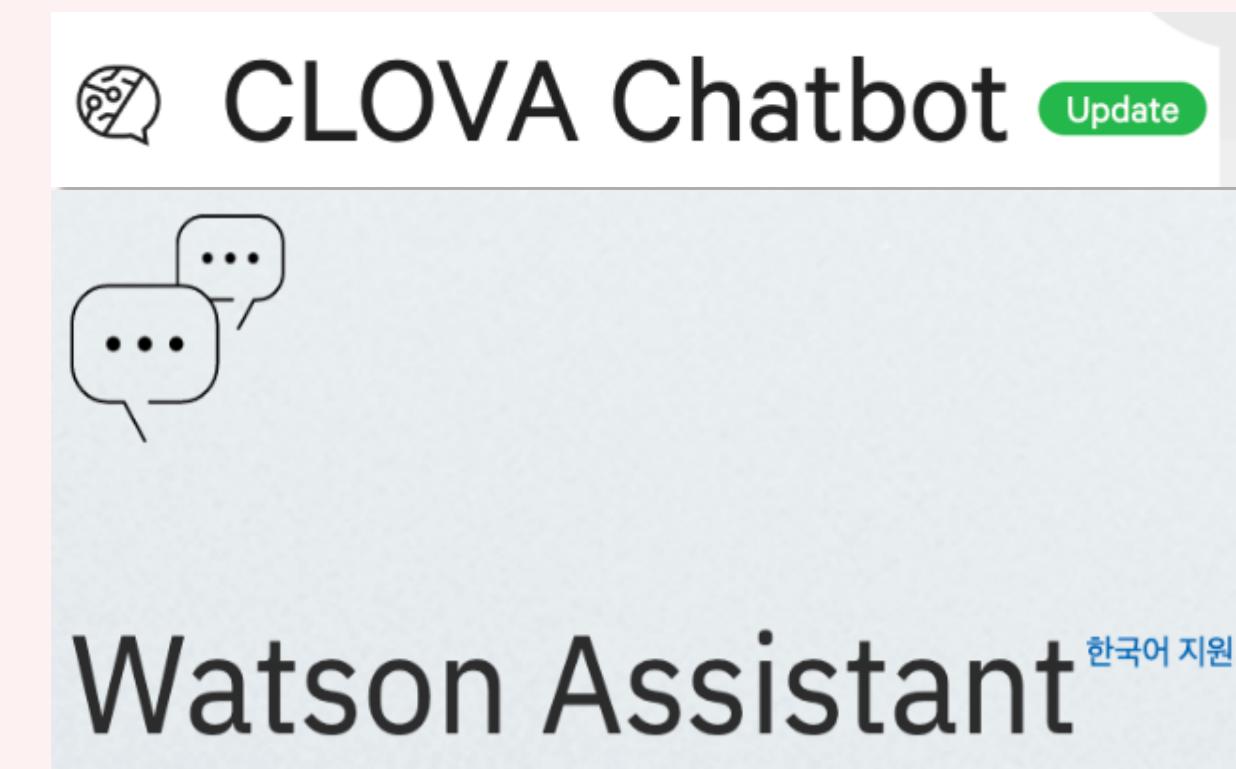
1. 사용자 중심의 챗봇: 기존 챗봇의 한계점

1) 대화형 챗봇의 한계



- 구어체 입력으로 인한 낮은 직관성
- 높은 입력 자유도에 따른 부족한 정확도

2) API로 만들어진 챗봇의 한계

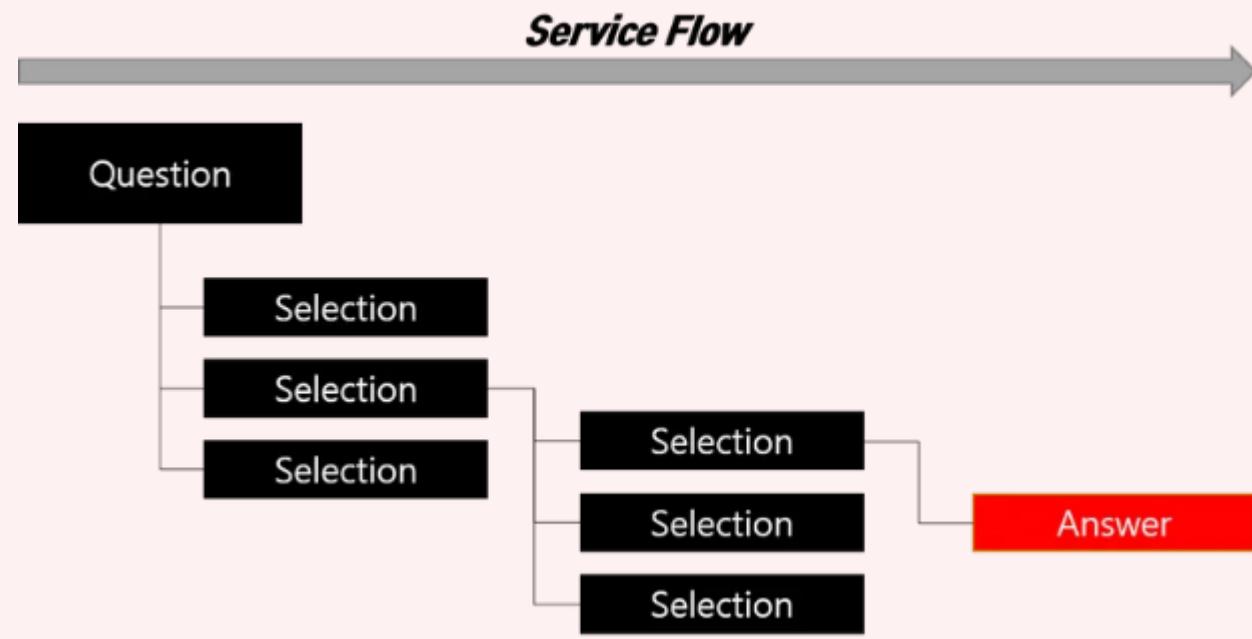


- 제한적인 형태의 AI 접근성
- 분류 정확도 개선 불가능
- 사용량에 따른 경제적 비용 발생

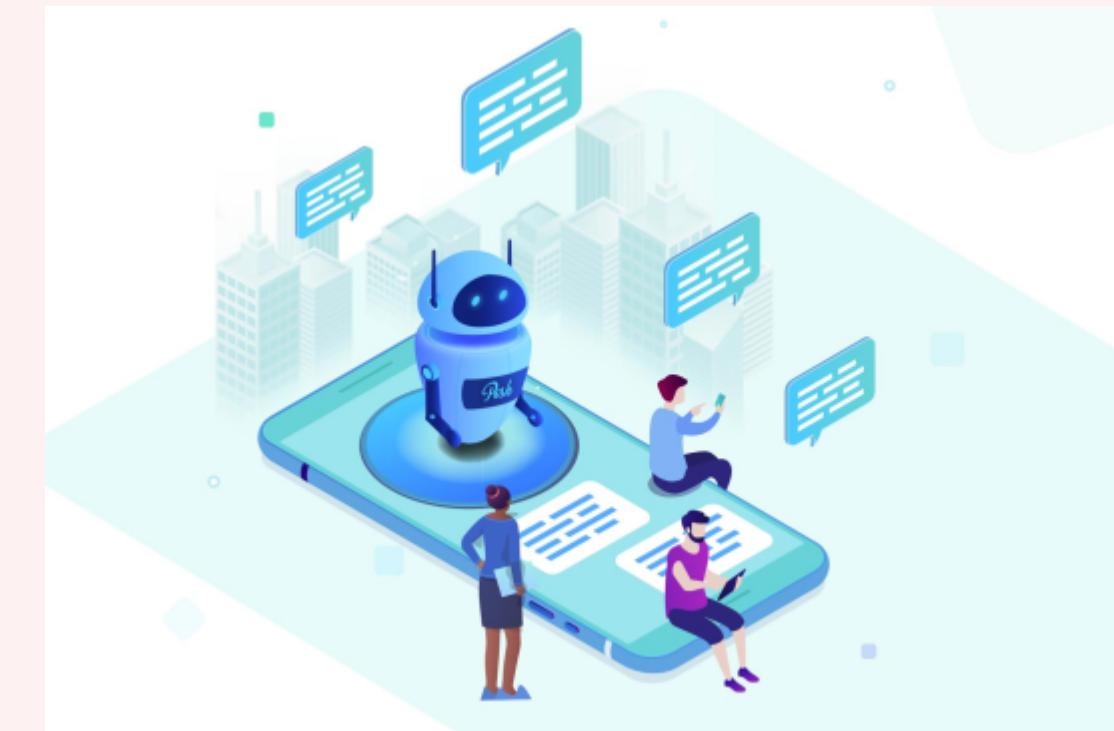
프로젝트의 방향성

1. 사용자 중심의 챗봇: 직접 개발한 챗봇의 필요성

1) 트리형 챗봇 구조



2) 직접 개발한 챗봇의 유연성



- 직관적인 UI로 정보에 신속하게 접근 가능
- 사용자가 원하는 정보를 정확히 제공
- 사용자의 의사결정 과정을 데이터로 축적 가능

- 사용자에게 딱 맞는 AI 모델 구축 가능
- 입출력 형식의 높은 자유도
- 사용자 요구에 따른 모델이기에
수정/보완/추가의 agile한 대처가 가능

프로젝트의 방향성

2. 진료과 추천 AI: 사용자 친화적이지 못한 기존 체계

The image displays two screenshots of hospital websites. The left screenshot shows the '군 특수질환 맞춤형 병원' website, featuring a grid of medical specialties such as Internal Medicine, Cardiology, Pulmonology, etc. The right screenshot shows the '세브란스 병원' website, which includes a step-by-step guide for online appointment booking.

- 병원 사이트에는 진료과에 대한 링크들만 존재
- 환자가 자신의 증상에 맞는 진료과를 **스스로 찾아서 방문**
- 세브란스 병원은 환자가 “진단명”을 알 경우, 적절한 진료과를 안내해주는 서비스를 제공
- 하지만 일반인은 본인의 증상과 진단명을 연결시키는 능력 부족

프로젝트의 방향성

2. 진료과 추천 AI: 헷갈리는 진료과



- 환자들이 처음 새로운 증상을 겪으면 어느 진료과를 가야 할지 혼란
- 불필요한 의사와 환자의 대면은 시간 낭비, 경제적 손실을 유발
- 자신의 증상에 대한 설득론 판단은 **부정확한 진단과 의미없는 치료를 야기**

프로젝트의 방향성

2. 진료과 추천 AI: 진료과 분류 연구 현황

Weng et al. BMC Medical Informatics and Decision Making (2017) 17:155
DOI 10.1186/s12911-017-0556-8

RESEARCH ARTICLE **Open Access**

BMC Medical Informatics and Decision Making

 CrossMark

Medical subdomain classification of clinical notes using a machine learning-based natural language processing approach

Wei-Hung Weng^{1,2,3*} , Kavishwar B. Wagholarikar^{2,4}, Alexa T. McCray¹, Peter Szolovits³ and Henry C. Chueh^{2,4}

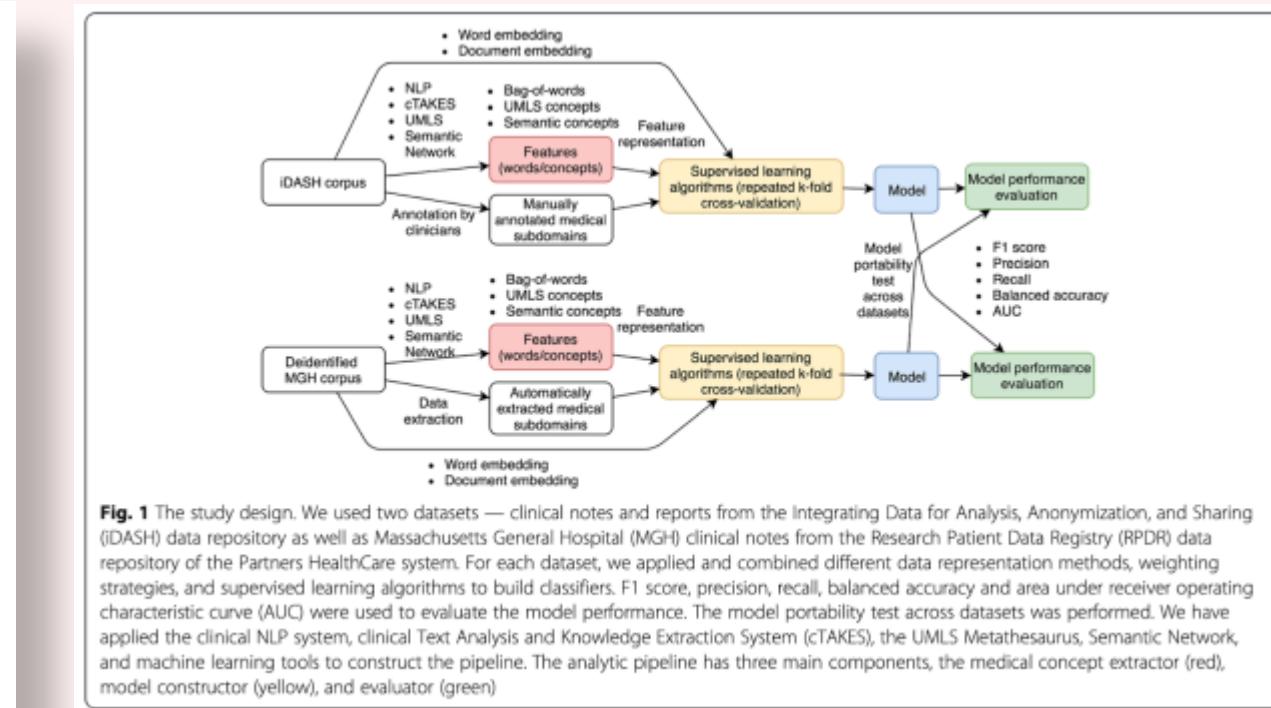


Fig. 1 The study design. We used two datasets — clinical notes and reports from the Integrating Data for Analysis, Anonymization, and Sharing (IDASH) data repository as well as Massachusetts General Hospital (MGH) clinical notes from the Research Patient Data Registry (RPDR) data repository of the Partners HealthCare system. For each dataset, we applied and combined different data representation methods, weighting strategies, and supervised learning algorithms to build classifiers. F1 score, precision, recall, balanced accuracy and area under receiver operating characteristic curve (AUC) were used to evaluate the model performance. The model portability test across datasets was performed. We have applied the clinical NLP system, clinical Text Analysis and Knowledge Extraction System (cTAKES), the UMLS Metathesaurus, Semantic Network, and machine learning tools to construct the pipeline. The analytic pipeline has three main components, the medical concept extractor (red), model constructor (yellow), and evaluator (green).

- 해외 사례로는 하버드 의과대학에서 2017년에 발표한 논문 **단 1편**
- 한국어 기반 논문 또는 특히 **없음**
- 실제 서비스화된 사례 **없음**

프로젝트의 방향성

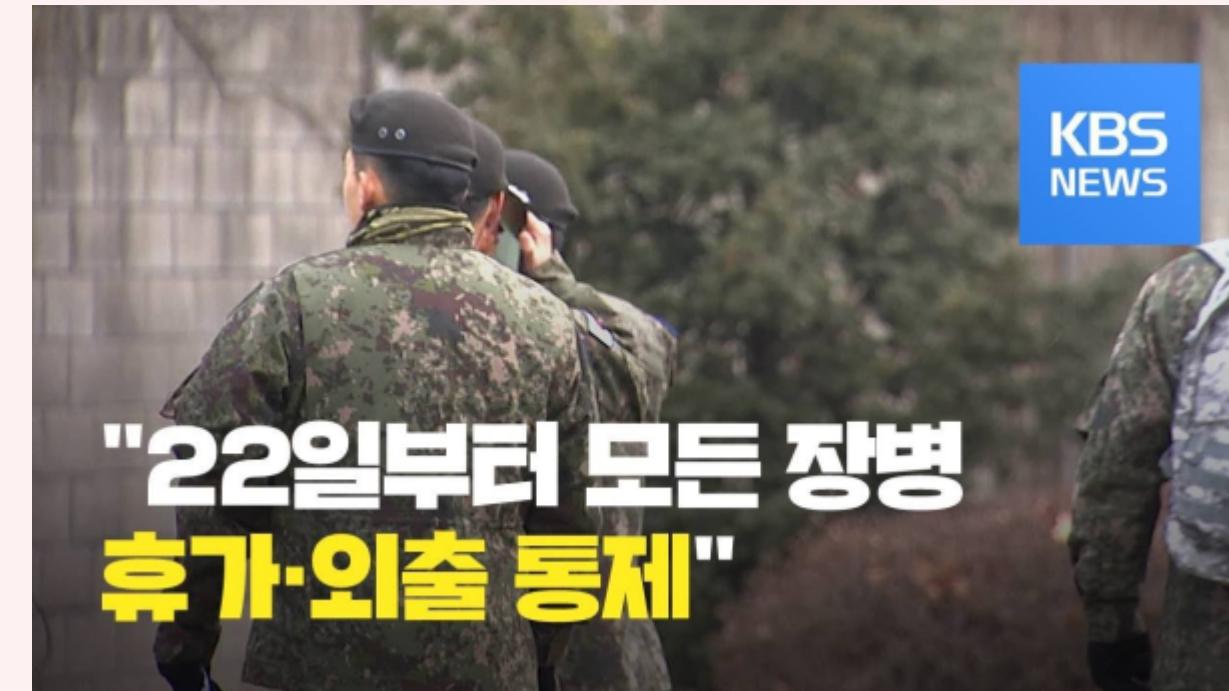
2. 진료과 추천 AI: 진료과 분류 모델의 기대 효과



- 의학적 지식이 없는 일반인도 자신의 증상에 맞는 진료과 방문 가능
- 기존 진료과 안내 서비스의 패러다임 전환
- **국내 최초**의 진료과 분류 모델 개발 및 서비스화

프로젝트의 방향성

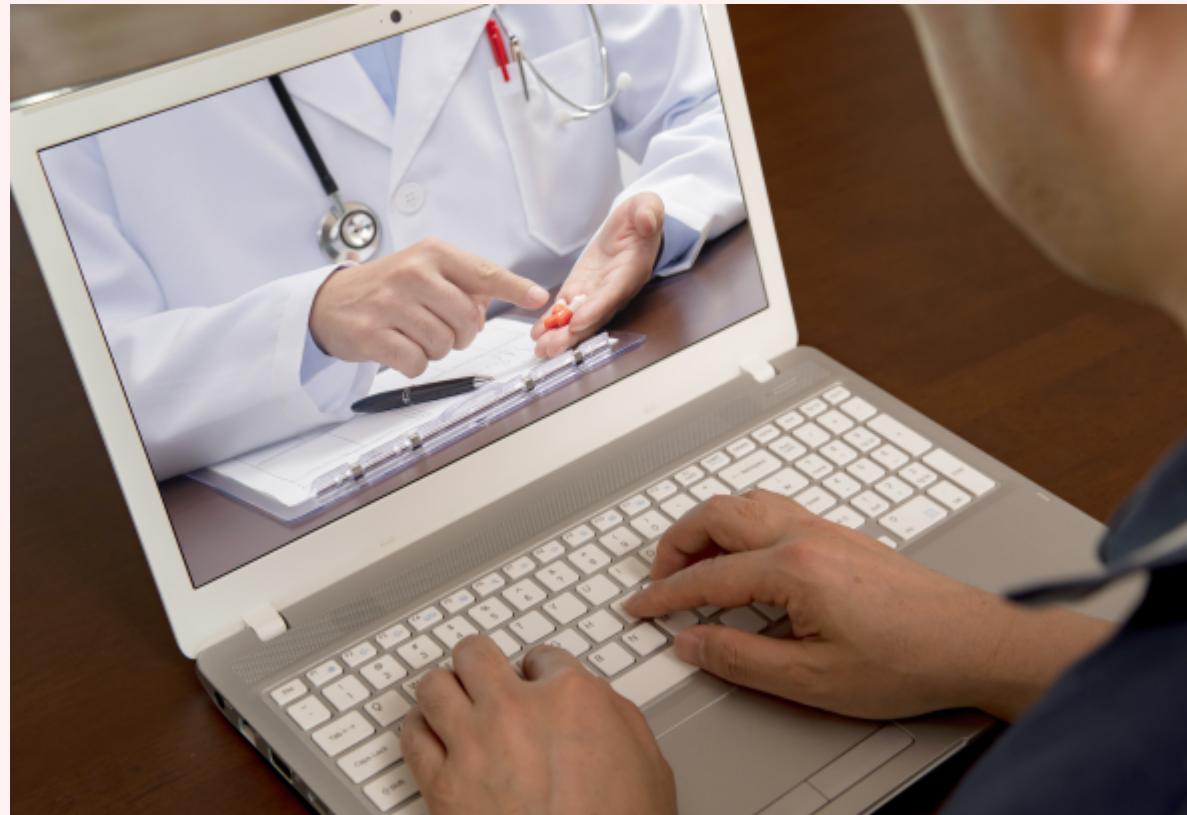
3. 비대면 상담 서비스: 군내 비대면 건강 상담 서비스의 필요성



- GOP/GP와 같은 야전 부대는 비교적 열악한 의료 시스템 보유
- 각 부대와 사단 의무대/군병원 간의 거리 등의 지리적 한계로 충분한 의료서비스 제공 불가
- 최근 COVID-19 사태에 따른 군내 감염으로 인해 사단의무대 혹은 군병원이 폐쇄되는 상황 발생
- 외출 및 휴가 통제에 따른 민간 병원 진료 일정이 취소/연기 되는 사례가 다수 발생
- 병사들의 건강 유지가 걱정되는 상황 초래

프로젝트의 방향성

3. 비대면 상담 서비스: 비대면 의료 상담 서비스의 국가적 지원



- COVID-19 사태에 대응하여, 국민의 안전을 확보하고 감염원으로부터 의료기관과 의료인을 보호하기 위해 정부는 ‘감염 예방 추진 방안’을 한시적으로 다음과 같이 시행
 - 1) 가벼운 감기 환자, 만성질환자 등에 대해 전화상담, 전화처방, 대리처방, 화상진료 등 비대면 의료시행을 적극 활용
 - 2) 환자가 의료 기관을 방문한 경우, 의료기관 내 별도 공간에서 의료인 보조 하에 의료기관 본 건물 내 의사와 화상으로 진료

프로젝트의 방향성

3. 비대면 상담 서비스: 비대면 의료 상담 서비스의 국가적 지원

| | | | |
|---|----------------------------|------|--------------|
|  보건복지부 <small>행복이 되는 평생 친구</small> | 보 도 자 료 배포 즉시 보도 | | |
| 배 포 일 | 10월 29일 / (총 4매) | 담당부서 | 보건의료정책과 |
| 과 장 | 이 창 준 | 전 화 | 02-2023-7305 |
| 팀 장 | 최 경 일 | | 02-2023-7320 |

의료기관을 직접 방문하기 어려운 지역 주민이나 환자를 위해
“동네의원 중심의 의사-환자간 원격진료”추진

- 만성질환 상시 관리, 의료취약지 및 거동불편 주민 의료접근성 제고,
1차 의료 활성화 등 기대 -

복지부는 동네의원 중심으로 의사와 환자간의 원격진료를 허용하는
의료법 개정(안)을 마련하여 10월 29일(화) 입법예고 하였다.

최근 국민편의 증진과 의료기술 발전 등 보건의료 환경변화로
의사-환자간 원격의료를 허용하자는 의견이 대두됨에 따라,

- 최근 보건복지부는 동네의원을 중심으로 비대면 의료 제공을 허용하는 의료법 개정(안)을 마련하여 2020년 10월 29일 입법예고

프로젝트의 방향성

3. 비대면 상담 서비스: 비대면 의료 상담 서비스의 해외 사례

미국



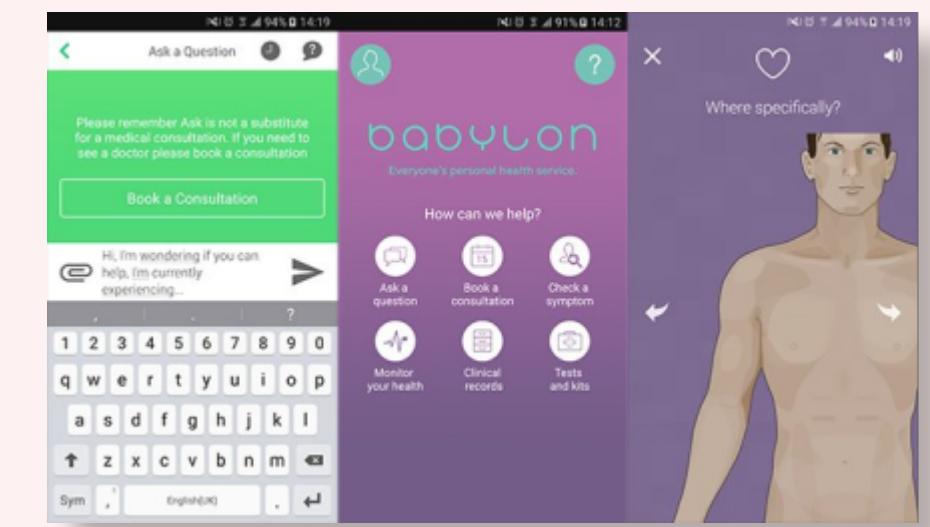
· 보건부는 비대면 의료서비스 제공 시 적용하고 있던 Health Insurance Portability and Accountability Act 규정을 일시적으로 완화하여 페이스 타임, Skype 등의 SNS 애플리케이션을 비대면 의료서비스에 사용 가능하도록 허용

독일



· 2019년 의사가 당뇨병 모니터링 및 관리 등 건강관리 기능 헬스케어 애플리케이션을 이용하여 처방을 허용하는 법안 (digital healthcare act [Digitale–Versorgung–Gesetz])을 통과

영국



· 국가보건서비스인 NHS와 협력하고 있는 바빌론 헬스(Babylon health)는 스마트폰 기반 AI 비대면 의료 서비스를 제공하는데, 채팅을 통한 상담은 물론 의사와의 예약 및 진료 등을 이용할 수 있는 서비스를 제공하고, 진료 이후에는 처방 의약품의 배송서비스까지 제공

프로젝트의 방향성

3. 비대면 상담 서비스: 비대면 의료 상담 서비스의 필요성



- COVID 19 사태의 장기화에 따라, 장병들의 건강 유지를 위해 **비대면 의료 상담 서비스 도입이 시급**
- 성공적인 K방역으로 재평가되는 세계 최고의 의료서비스를 군 장병들에게도 제공
- 군의관을 통한 비대면 의료 상담 서비스를 적극 활용 필요

프로젝트의 방향성

4. 온라인 예약 기능: 온라인 예약 서비스 도입



- 국군수도병원은 우리 군 최고의 의료서비스를 제공하는 곳인 만큼 많은 환자들이 몰리는 현상이 발생
- 정형외과, 정신건강의학과 등 인기과의 경우 어렵게 병원을 방문하더라도 하루종일 대기하거나, 심지어는 아예 진료를 받지 못하는 경우 존재
- 기존 국군수도병원 홈페이지를 통한 진료 예약은 불가능
- 장시간 대기나 환자 누락 없이 모든 환자들에게 의료 서비스를 충분히 제공할 수 있는 환경 조성 필요

04

개발 과정

딥러닝
인프라
백엔드
프론트엔드

개발 과정

1. 딥러닝: 개발 타임라인

| | |
|-----|---|
| 1주차 | 데이터 크롤링 자연어 처리 딥러닝 모델 검토 |
| 2주차 | 데이터 정리 및 라벨링 시작 LSTM 모델 구현 완성 : 남은 기간동안 성능 튜닝 진행 |
| 3주차 | 데이터 정리 및 라벨링 진행 (50%) FastText 모델 구현 완성 |
| 4주차 | 데이터 정리 및 라벨링 진행 (85%) BERT 모델 구현 완성 : 남은 기간동안 성능 튜닝 진행 |
| 5주차 | 데이터 정리 및 라벨링 완료 데이터 최종 리뷰 및 공유 딥러닝 모델 최종 선정 및 배포 |

개발 과정

1. 딥러닝: 데이터 수집(1)



의료 관련 구어체 데이터 필요

- 증상 분류 모델은 대표적인 지도학습에 해당
 - 데이터의 퀄리티가 챗봇의 퀄리티에 직결

개발 과정

1. 딥러닝: 데이터 수집(2)

하이닥(Hi-Doc)



```

import requests
from bs4 import BeautifulSoup
import pandas as pd

partname = ['가정의학과', '아동인후과', '소화기내과',
            '호흡기내과', '감염내과', '알레르기 내과',
            '내분비내과', '순환기내과', '신경내과',
            '증마티스내과', '혈액증양내과', '성형외과',
            '정형외과', '신경외과', '대장항문',
            '복부외과', '외과', '피부과',
            '안과', '치과', '비뇨외과',
            '신부인과', '한방과', '신경과',
            '재활의학과', '중성건강리학과',
            '응급의학과', '마취통증외과학과', '발사선종합학과',
            '영상의학과', '진단검사의학과', '직업환경의학과']

partcode = [ 'PF0000', 'PE0000', 'PM0000',
            'PV0000', 'PH0000', 'PN0000',
            'PMF000', 'PHC000', 'PNA000',
            'PVH000', 'PA0000', 'PG0000',
            'PO0000', 'PS0000', 'PG1000',
            'PC0000', 'PG0000', 'PS0000',
            'PH0000', 'PV0000', 'PL0000',
            'PY0000', 'PL0000', 'PN0000',
            'PR0000', 'PF0000',
            'PJ0000', 'PT0000', 'PX0000',
            'PK0000', 'PQ0000', 'PQ0000' ]
inf = 99999999

qlist = []

for i, part in enumerate(partcode):
    url = "https://www.hidco.co.kr/healthqna/part/list?code={code}&page=".format(code=part)
    try:
        for num in range(1, inf):
            cur_url = url + str(num)
            res = requests.get(cur_url)
            soup = BeautifulSoup(res.content, 'html.parser')

            qnas = soup.find_all('div', {'class': 'box_type1 qna_main'})

            for qna in qnas:
                qlist.append(qna.find('strong', {'class': 'tit_qna'}).text)

    except:
        pass
tempset = set(qlist)
result = list(tempset)
df = pd.DataFrame(result)
df.to_csv(partname[1] + ".csv")

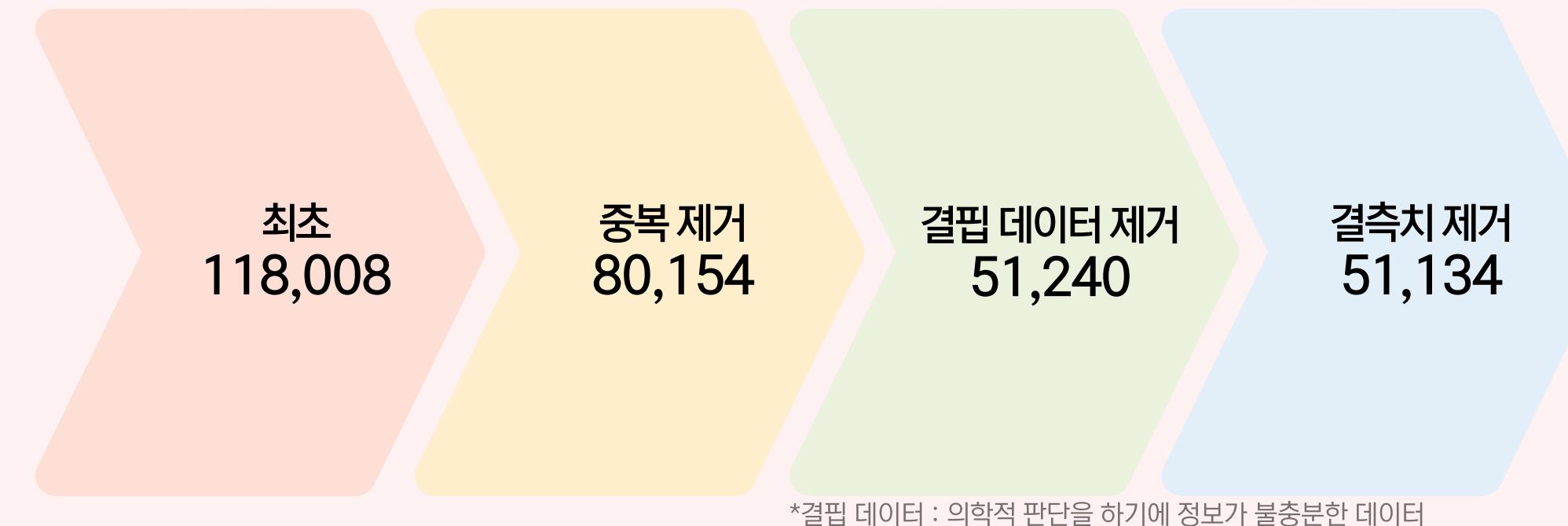
```

- 환자의 질문에 대해 의학 전문의가 직접 답변을 달아주는 서비스를 제공
 - 환자의 궁금증들이 각 진료과로 카테고리화되어 존재

- Python에 있는 requests 모듈 이용
 - 약 11만건의 데이터를 csv형태로 확보

개발 과정

1. 딥러닝: 데이터 정리(1)



- 처음에는 수집한 데이터들이 전혀 정돈되지 않은 상태
- 의학 분야 전문 지식을 가진 군의관이 약 12만개의 데이터를 직접 검수하며 라벨링
- 최종적으로 51,134개의 정돈된 데이터 획득
- 라벨링이 잘못되어 있던 부분은 직접 이현훈 군의관이 진료과를 수정



실제로 환자들은 본인에게 해당하는 진료과를 찾는데 어려움을 느낀다는 결론

개발 과정

1. 딥러닝: 데이터 정리(2)

| 증상명 | 진료과 |
|---|-----|
| 두통약을 먹어도 두통이 안없어짐 | FM |
| 후두에 염증이 생겼어요 | FM |
| 변 보고나서 생기는 복통? | FM |
| 어깨 목 통증이 심합니다 | FM |
| 출산한지 4개월정도되었는데 아직도 복통이 있어요 | FM |
| 건강검진을받았는데요 신장질환의심이랑 단백뇨라고 적혀있어요 어떻게해야할까요? | FM |
| 한쪽 코 안이 헐고 피가 나서 딱지가 생기기도 합니다. | FM |
| 설사 변비 반복되고 소화가 자주 안되는 증상 | FM |
| 담배때문에 그런지 기침을 심하게 합니다. | FM |
| 지속되는 미열 36.9~37.5도 | FM |
| 오래 걸으면 발목이랑 발바닥이 많이 아픕니다. | FM |
| 목안 이물질 어느 진료과로 가야할지 궁금해요 | FM |
| 매일운동하는데 빈혈이 않았다 일어날때 마다 오네요.... | FM |



| 증상명 | 진료과 |
|---|------|
| 두통약을 먹어도 두통이 안없어짐 | NR |
| 후두에 염증이 생겼어요 | ENT |
| 변 보고나서 생기는 복통? | GI |
| 어깨 목 통증이 심합니다 | NS |
| 출산한지 4개월정도되었는데 아직도 복통이 있어요 | OBGY |
| 건강검진을받았는데요 신장질환의심이랑 단백뇨라고 적혀있어요 어떻게해야할까요? | NPH |
| 한쪽 코 안이 헐고 피가 나서 딱지가 생기기도 합니다. | ENT |
| 설사 변비 반복되고 소화가 자주 안되는 증상 | GI |
| 담배때문에 그런지 기침을 심하게 합니다. | IP |
| 지속되는 미열 36.9~37.5도 | INFC |
| 오래 걸으면 발목이랑 발바닥이 많이 아픕니다. | OS |
| 목안 이물질 어느 진료과로 가야할지 궁금해요 | ENT |
| 매일운동하는데 빈혈이 않았다 일어날때 마다 오네요.... | HEON |

· 데이터 정리 전

· 데이터 정리 후

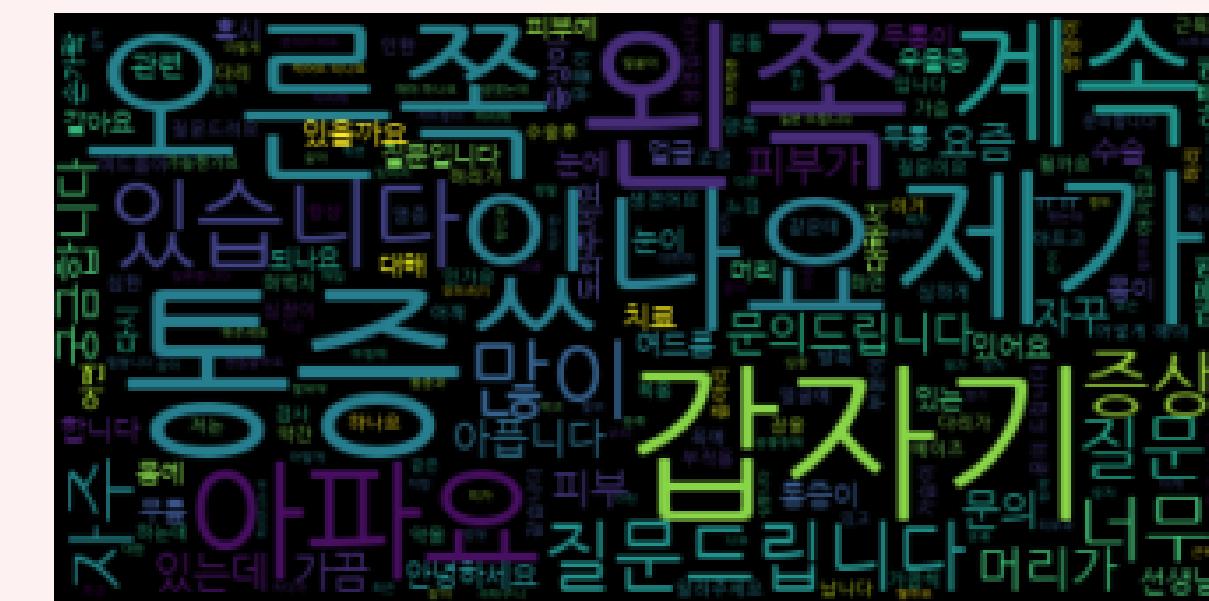
38~43% ---> 71~75%

데이터 정리 전후로 모델 성능 약 1.7배 이상 향상

개발 과정

1. 딥러닝: EDA(1)

| | 학습 데이터 (Train Dataset) | 검증 데이터 (Validation Dataset) |
|-------|------------------------|-----------------------------|
| 데이터 수 | 40,907 | 10,227 |
| 비율 | 80% | 20% |



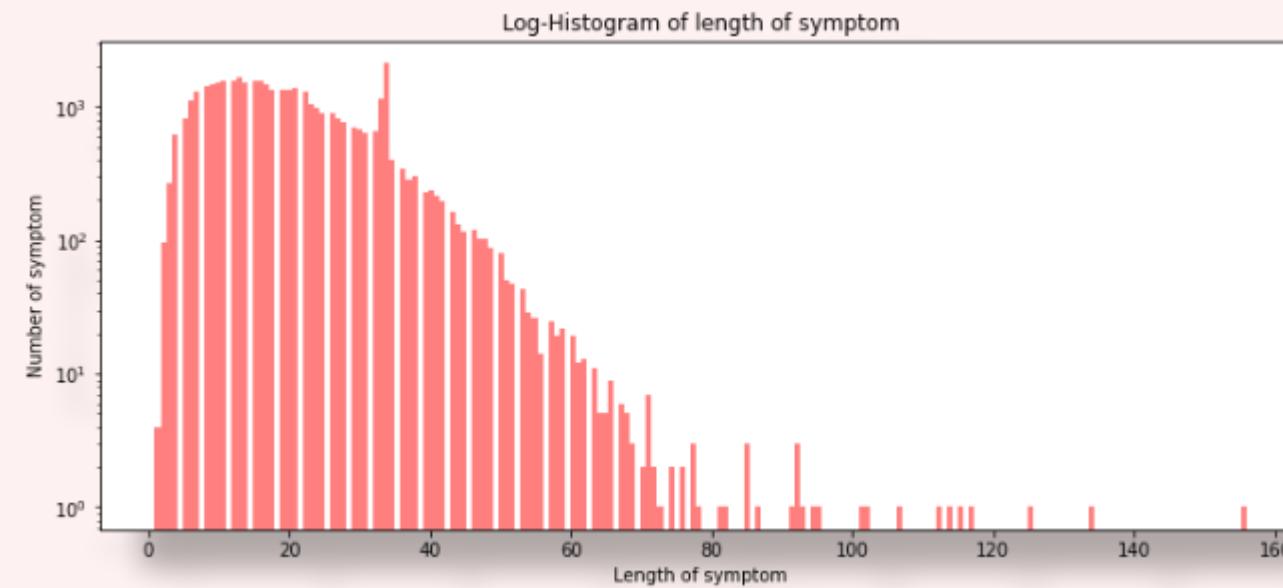
- 학습 데이터와 검증 데이터로 나누어 EDA*를 시작
 - 워드클라우드를 활용한 다용 어휘 시각화
 - 해부학적 위치 표현인 ‘오른쪽’, ‘왼쪽’ 식별
 - 임상 증상 묘사인 ‘통증’, ‘갑자기’ 등의 어휘 확인

*EDA

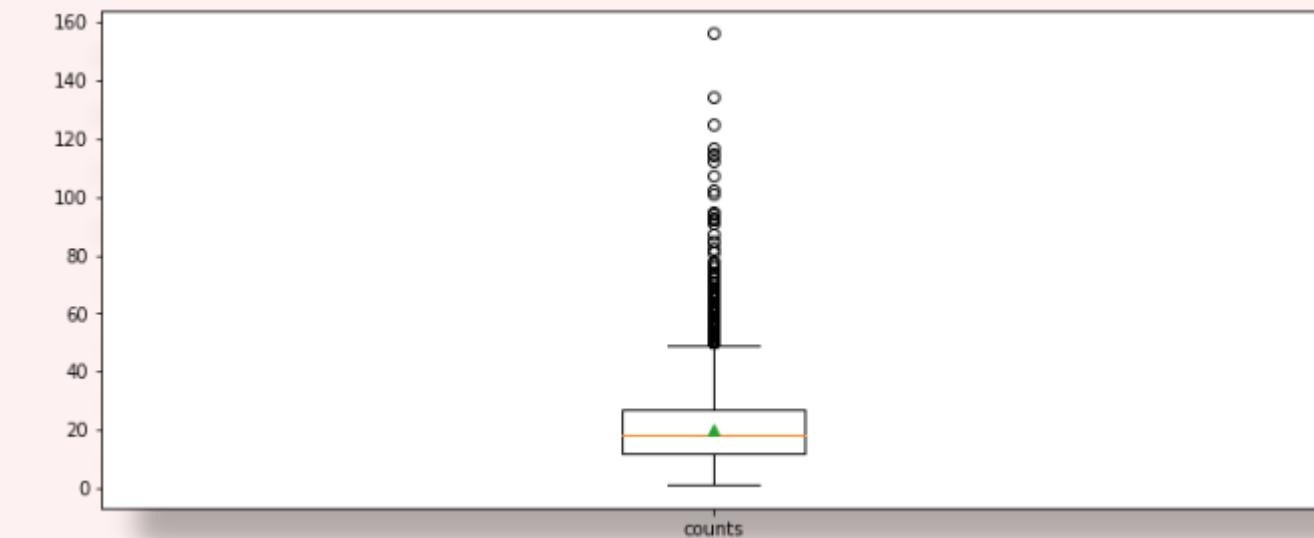
수집한 데이터를 다양한 각도에서 관찰하는 과정

개발 과정

1. 딥러닝: EDA(2)



- 각 문장(증상)별 길이 분석
- 매우 짧은 길이부터 약 35자까지가 대부분
- 60자 이후로는 데이터가 급격히 감소
- 35자 부근의 불규칙은 하이닥 사이트의 한줄 입력이 35자이기 때문

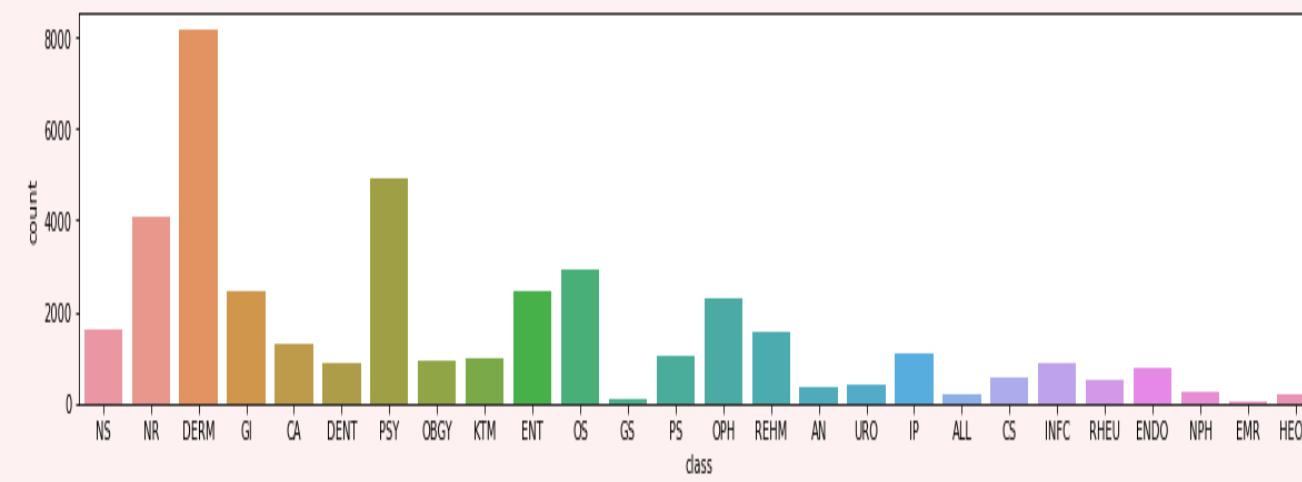


- 증상 길이 최대 값 : 156
- 증상 길이 최소 값: 1
- 증상 길이 평균 값: 20.14
- 증상 길이 표준편차: 11.03
- 증상 길이 중간 값: 18.0
- 증상 길이 제 1 사분위: 12.0
- 증상 길이 제 3 사분위: 27.0

개발 과정

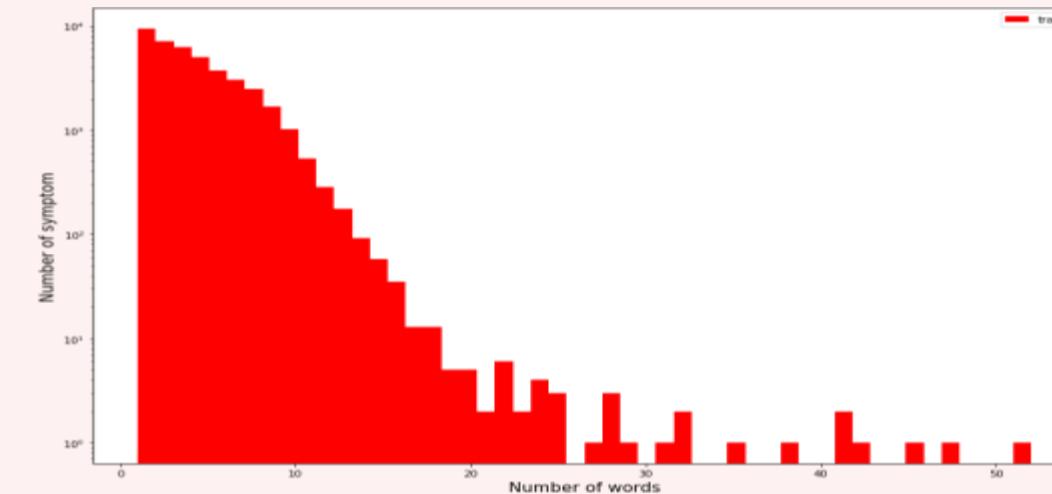
1. 딥러닝: EDA(3)

진료과별 데이터 수



- 심한 데이터 불균형이 존재
- 피부과(DERM), 정신건강의학과(PSY) 순으로 많은 데이터 수

각 문장 내 단어 개수 분석



- 증상 단어 개수 최대 값: 52
- 증상 단어 개수 최소 값: 1
- 증상 단어 개수 평균 값: 4.68
- 증상 단어 개수 표준편차: 2.78
- 증상 단어 개수 중간 값: 4.0

개발 과정

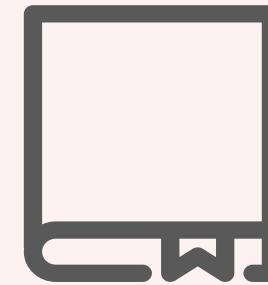
1. 딥러닝: EDA(4)

EDA를 통해 얻은 개발 방향성



불용어 사전 제작

“있나요”, “안녕하세요” 등 불필요한 단어 존재
불용어가 딥러닝에 영향을 끼치지 못하도록 전처리



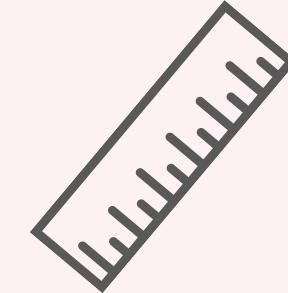
명사 사전 제작

증상 표현에 주로 사용되는 의학 용어가 한정적



진료과별 가중치

진료과별 데이터 불균형이 심각

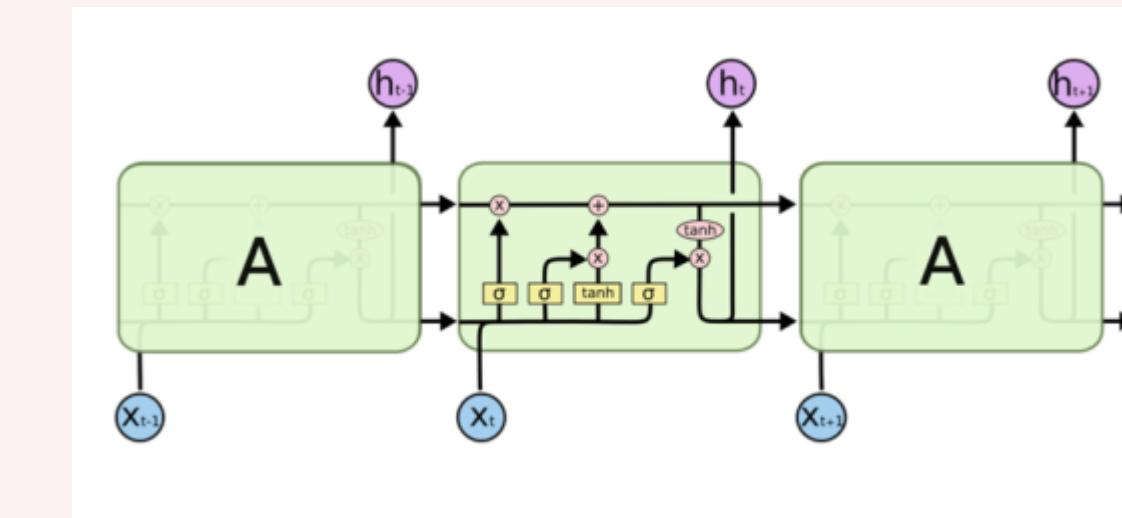
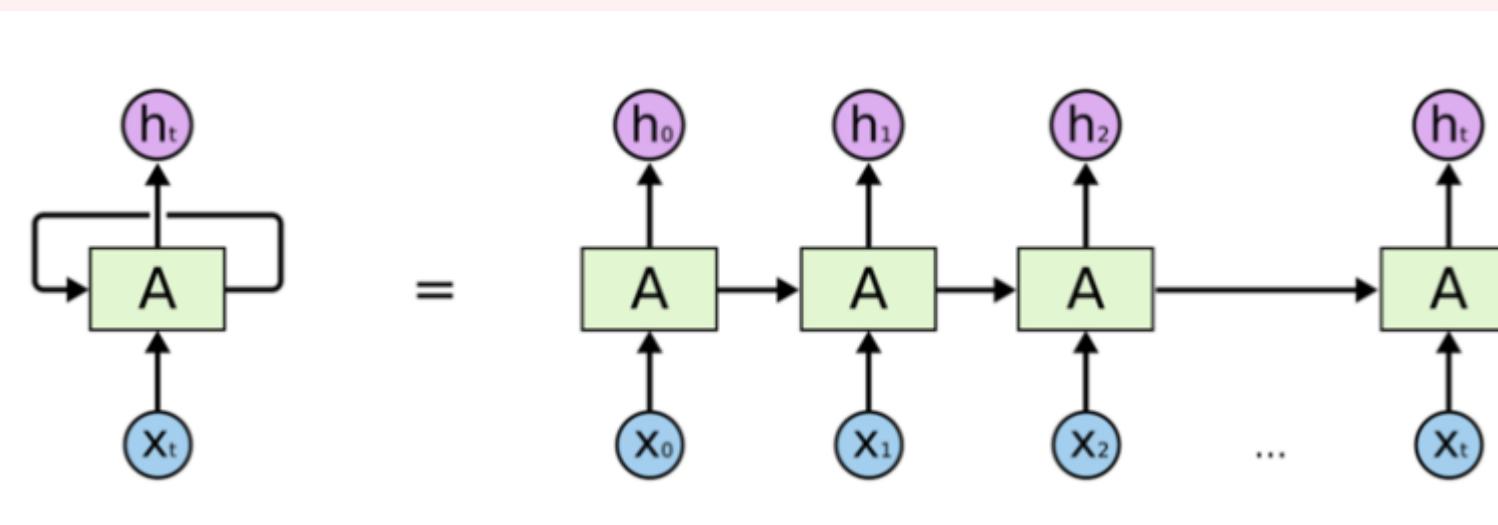


증상 길이 파악

모델이 유효하게 인식할 증상 길이 파악

개발 과정

1. 딥러닝: LSTM - 소개



- 음성, 문자 등 **순차적으로 등장하는 데이터** 처리에 적합
- 필요에 따라 다양하고 유연하게 구조를 만들 수 있다는 장점
- 관련 정보와 그 정보를 사용하는 지점 사이 거리가 멀 경우 학습능력이 크게 저하되는 기존 모델을 개선

개발 과정

1. 딥러닝: LSTM - 데이터 전처리

“축구를 하다가 발목을 빠끗했어요”



“축구”, “발목”, “빠끗”



[172, 34, 71]

- 증상 관련 문장이 챗봇에 입력으로 전달
- 문장을 python의 re와 konlpy 패키지를 이용해 명사들로만 분해
- EDA 워드클라우드에서 필요없는 단어들이 많았던 것을 기억하여 불용어는 추출하지 않음
- 각 명사에 고유 번호 부여 (빈도수 기준 상위 15000개 단어)
- 1~15000까지의 번호는 이후 임베딩*단계에서 각 단어의 의미와 관련된 벡터로 변환

*임베딩

자연어를 기계가 이해할 수 있는 숫자형태인
vector로 바꾸는 과정

개발 과정

1. 딥러닝: LSTM - 단어 임베딩

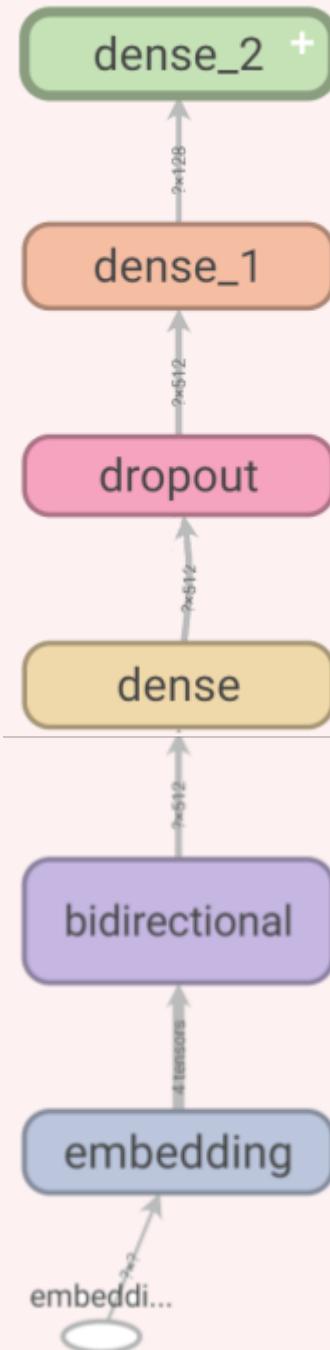
```
1 from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional, Dropout
2
3 embedding_dim = 2048
4
5 model = keras.models.Sequential([
6     Embedding(vocab_size, embedding_dim),
7     Bidirectional(LSTM(256)),
8     Dense(512, activation='relu'),
9     Dropout(0.25),
10    Dense(128, activation='relu'),
11    Dense(26, activation='softmax')
12 ])
Model: "sequential_2"

Layer (type)          Output Shape         Param #
embedding_2 (Embedding) (None, None, 2048)      40960000
bidirectional_2 (Bidirection (None, 512)           4720640
dense_6 (Dense)        (None, 512)            262656
dropout_2 (Dropout)    (None, 512)            0
dense_7 (Dense)        (None, 128)             65664
dense_8 (Dense)        (None, 26)              3354
=====
Total params: 46,012,314
Trainable params: 46,012,314
Non-trainable params: 0
```

- tensorflow.keras의 Embedding 레이어 사용
- 임베딩 차원은 2048, 1024로 바꿔가며 성능을 비교하며 진행

개발 과정

1. 딥러닝: LSTM - 모델링

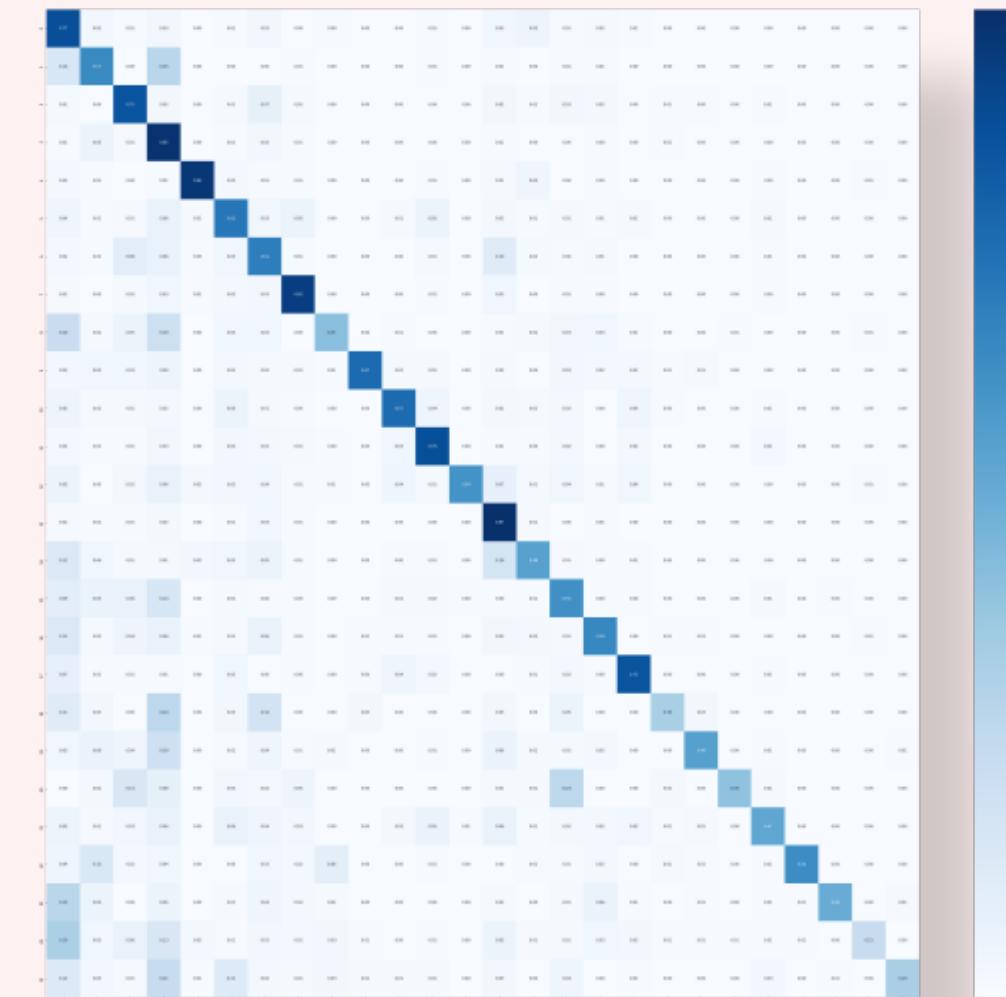


- 모델에 들어가기 전에, 입력된 문장은 이미 **숫자의 배열로** 전처리된 상태
- 각 숫자는 벡터로 임베딩되어 Embedding 층으로 입력
- 벡터화된 문장이 Bidirectional(LSTM) 층에 입력으로 전달
- Bidirectional 층을 통과한 출력은 **문장의 전체 맥락 정보를 가지는 벡터**
- Dense , Dense_1, Dense_2 층은 벡터를 마지막 26개 라벨로 분류하도록 돋는 역할
- Dense, Dense _1에서는 활성함수 relu 사용, Dense_2에서는 softmax 사용
- Dropout 층은 모델의 과대적합을 막는 역할
- EDA중 라벨별 불균형이 관찰되었기 때문에 훈련 진행 시 각 라벨별 데이터 수에 맞게 가중치를 설정

개발 과정

1. 딥러닝: LSTM - 하이퍼파라미터 튜닝 및 평가

| LSTM 층 크기 | 완전 연결 층 크기 | 임베딩 차원 | 최대 단어 수 | 정확도 |
|-----------|-----------------|--------|---------|--------|
| 512 | 512 - 128 - 26 | 1024 | 5000 | 67.42% |
| 512 | 256 - 128 - 26 | 1024 | 10000 | 71.38% |
| 512 | 512 - 256 - 26 | 1024 | 15000 | 73.21% |
| 512 | 1024 - 256 - 26 | 1024 | 15000 | 71.84% |
| 256 | 512 - 128 - 26 | 1024 | 15000 | 72.27% |
| 256 | 512 - 256 - 26 | 1024 | 15000 | 73.05% |
| 256 | 256 - 128 - 26 | 1024 | 15000 | 73.42% |
| 256 | 256 - 128 - 26 | 2048 | 15000 | 73.52% |
| 256 | 256 - 128 - 26 | 2048 | 20000 | 72.28% |
| 128 | 256 - 128 - 26 | 2048 | 15000 | 72.38% |



· 이 표 이외에도 더 추가적인 튜닝을 진행했으나 발표 맥락상 위 표만 수록

- 얻은 모델로 Confusion Matrix 도출
- 26 * 26 표의 대각선이 진할수록 해당 라벨을 정확하게 판단한 것
- 대각선 위주로 짙은 파란색이 나타나는 것을 확인 가능

개발 과정

1. 딥러닝: LSTM - 향후 발전성



- 최근 인기있는 NLP 모델은 주로 이미 주어진 Pretrained 임베딩 벡터를 활용하는 방식
- Pretrained는 일반적인 언어에 대한 정확도는 월등하지만, 의료 분야와 같은 전문적인 언어에는 취약
- **LSTM**은 pretrained 임베딩 벡터를 활용하지 않으므로 가벼운 크기로 모델링 가능
- 원하는 데이터에 관해서만 훈련하여 배포할 수 있으므로 전문적인 분야에 적용이 용이

개발 과정

1. 딥러닝: FastText - 소개

The screenshot shows the fastText website's homepage. At the top, there is a dark header with the fastText logo and links for Docs, Resources, Blog, and GitHub. Below the header, there is a section titled "Resources" with a table of word vectors for various languages. The table has four columns: Indonesian, Interlingua, Irish, Italian, Japanese, Javanese, Kannada, Kapampangan, Kazakh, Khmer, Kirghiz, and Korean. Each row contains a link to download the word vectors in bin and text formats.

| | | | |
|--|--|---|--|
| | Indonesian: bin , text | Interlingua: bin , text | Irish: bin , text |
| | Italian: bin , text | Japanese: bin , text | Javanese: bin , text |
| | Kannada: bin , text | Kapampangan: bin , text | Kazakh: bin , text |
| | Khmer: bin , text | Kirghiz: bin , text | Korean: bin , text |



- FastText는 2017년 페이스북에서 개발해 공개한 단어 임베딩 기법
- 기존의 Word2Vec* 방식에서 각 단어를 문자단위 N-gram**으로 표현한 것이 특징
- 한국어 단어 벡터도 제공

*Word2Vec

단어를 벡터로 표현하는 기법

**N-gram

인접한 단어끼리 묶어서

단어가 가진 의미를 분석하는 방법

개발 과정

1. 딥러닝: FastText - 데이터 전처리

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

t = Tokenizer()
t.fit_on_texts(train_sentence)
# t.fit_on_texts(test_sentence)
vocab_size = len(t.word_index)+1
vocab_size

54219

X_encoded = t.texts_to_sequences(train_sentence)
X_encoded

[[2597, 1471, 2115, 201, 5264, 197, 7, 23, 58, 252, 198, 2324],
 [6, 156, 327, 230, 24],
 [272, 1472, 405, 223],
 [15445, 3279, 301, 15446],
 [760, 9346, 13],
 [48, 116, 3280, 638, 2598, 1225, 2599],
 [15447, 218, 258],]

X_train = pad_sequences(X_encoded, maxlen=max_len, padding='post')
Y_train = np.array(train_label)
print(X_train.shape)
Y_train.shape

(40907, 28)
(40907,)
```

- #### • Tensorflow 와 Keras 를 이용한 토크나이징* 수행

*토크나이징

띄어쓰기 기준으로 문장을 쪼갬

개발 과정

1. 딥러닝: FastText - 단어 임베딩

```
import fasttext.util
fasttext.util.download_model('ko', if_exists='ignore') # Korean
ko_model = fasttext.load_model('cc.ko.300.bin')

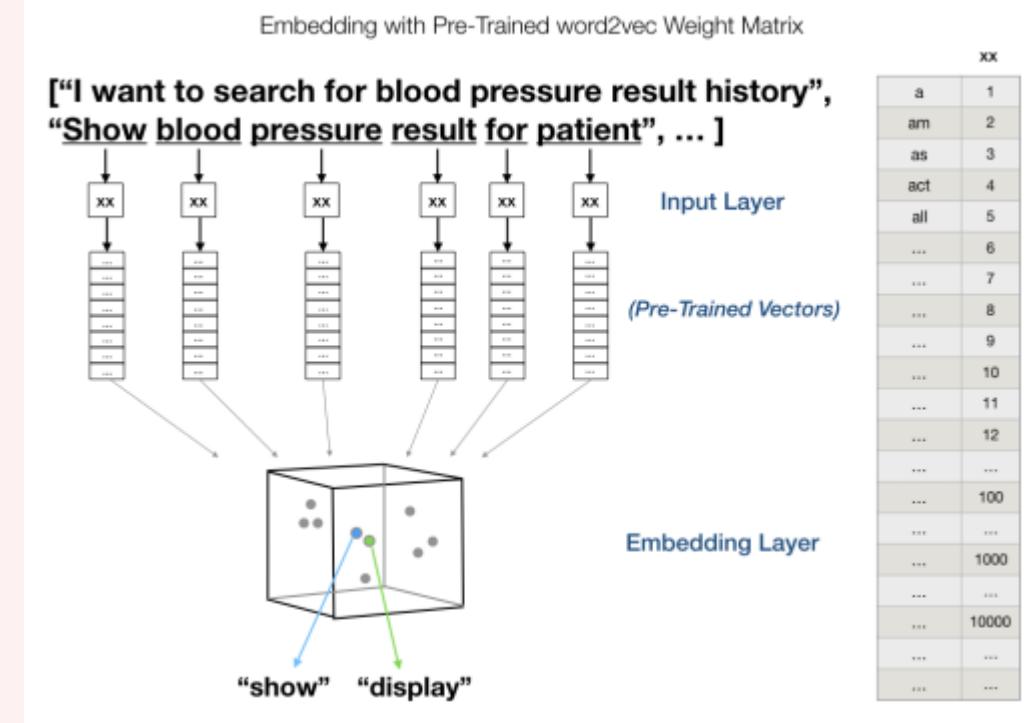
# To make embedding matrix
N = ko_model.get_dimension()
embedding_matrix = np.zeros((vocab_size, N))
# 단어 집합 크기의 행과 N개 열을 가지는 행렬 생성. 값은 전부 0으로 채워진다.
embedding_matrix.shape

(54219, 300)

ko_model.get_words

<bound method _FastText.get_words of <fasttext.FastText object at 0x7f0aa6fc2780>>

for word, i in t.word_index.items(): #훈련 데이터의 단어 집합에서 단어와 정수 인덱스로 1개씩 꺼내온다.
    temp = ko_model.get_word_vector(word) #단어(key) 해당되는 임베딩 벡터의 300개의 값(value)를 임시 변수에 저장
    if temp is not None:
        embedding_matrix[i] = temp
```

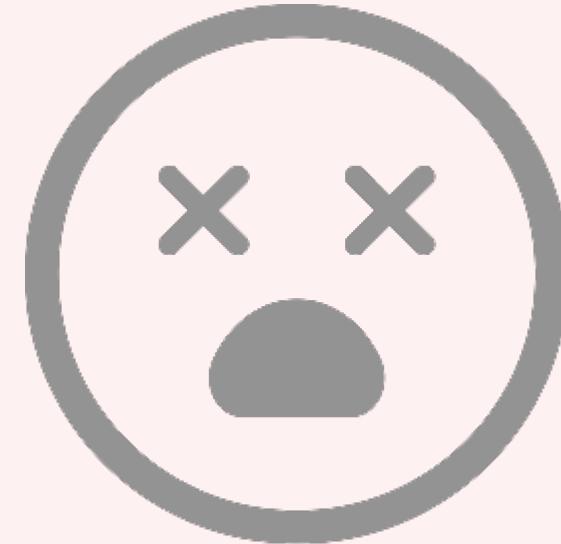


- 사전훈련(Pre-Trained)된 단어 벡터를 이용한 Embedding matrix 생성

- 단어 벡터를 이용한 임베딩의 예시

개발 과정

1. 딥러닝: FastText - 모델 및 평가



- 모델의 틀은 이전 LSTM model과 동일
- 훈련 결과 정확도 37.4%로 기본 LSTM 모델보다 현저히 낮은 성능
- 팀 회의 결과, 추가적인 성능 개선을 진행하지 않고 다른 모델에 집중하기로 결정

개발 과정

1. 딥러닝: BERT – 모델 소개

| SQuAD1.1 Leaderboard | | | | | |
|----------------------|--|--------|--------|--|--|
| Rank | Model | EM | F1 | | |
| | Human Performance Stanford University (Rajpurkar et al. '16) | 82.304 | 91.221 | | |
| 1 Oct 05, 2018 | BERT (ensemble) Google AI Language https://arxiv.org/abs/1810.04805 | 87.433 | 93.160 | | |
| 2 Sep 09, 2018 | nlnet (ensemble) Microsoft Research Asia | 85.356 | 91.202 | | |
| 3 Jul 11, 2018 | QANet (ensemble) Google Brain & CMU | 84.454 | 90.490 | | |

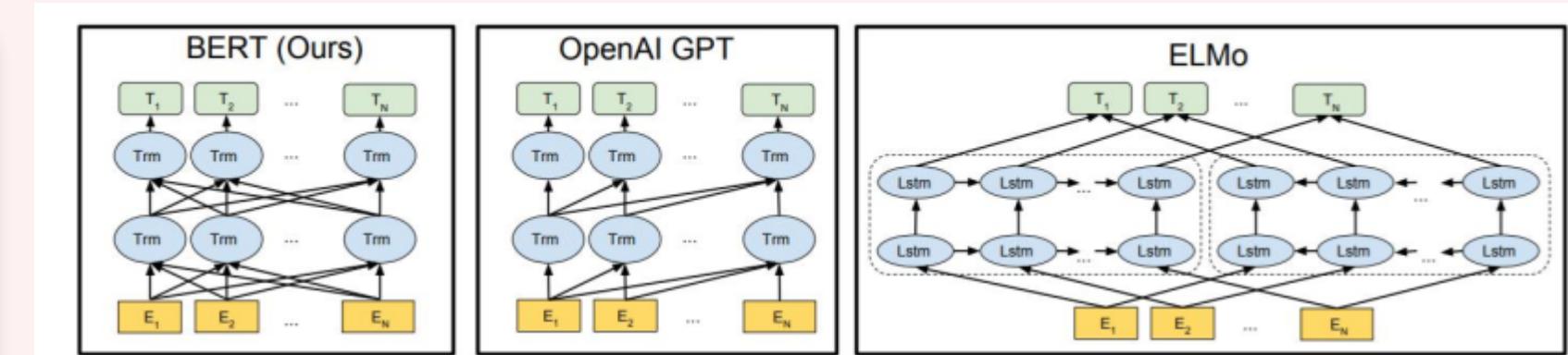


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

- BERT는 2018년 구글에서 공개한 딥러닝 모델로서 총 11개의 자연어 처리 문제에서 **최고의 성능** 과시
- 자연언어 처리 과제(NLP)를 교육 없이 양방향으로 사전학습하는 첫 시스템
- 이전 모델(GPT, ELMo)의 한계를 보완하고 자연어 처리 분야의 최신 기술을 모두 적용

개발 과정

1. 딥러닝: BERT – 데이터 전처리

```
# Bert Tokenizer
# 참조: https://huggingface.co/transformers/main_classes/tokenizer.html?highlight=encode_plus#transformers.PreTrainedTokenizer.encode_plus

def bert_tokenizer(sent, MAX_LEN, truncation=True):

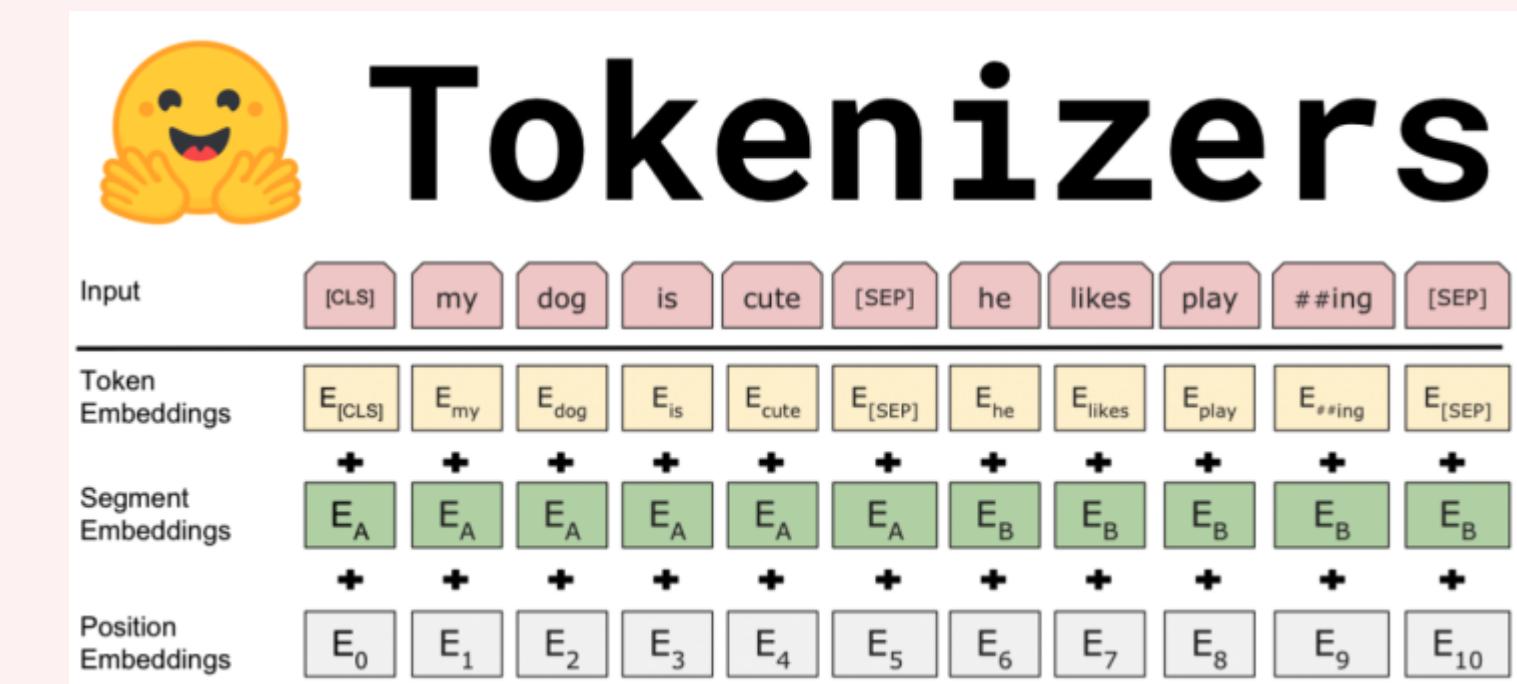
    encoded_dict = tokenizer.encode_plus(
        text = sent,
        add_special_tokens = True, # Add '[CLS]' and '[SEP]'
        max_length = MAX_LEN, # Pad & truncate all sentences.
        pad_to_max_length = True,
        return_attention_mask = True,
        truncation = True # Construct attn. masks.

    )

    input_id = encoded_dict['input_ids']
    attention_mask = encoded_dict['attention_mask'] # And its attention mask (simply differentiates padding from non-padding).
    token_type_id = encoded_dict['token_type_ids'] # Differentiate two sentences

    return input_id, attention_mask, token_type_id

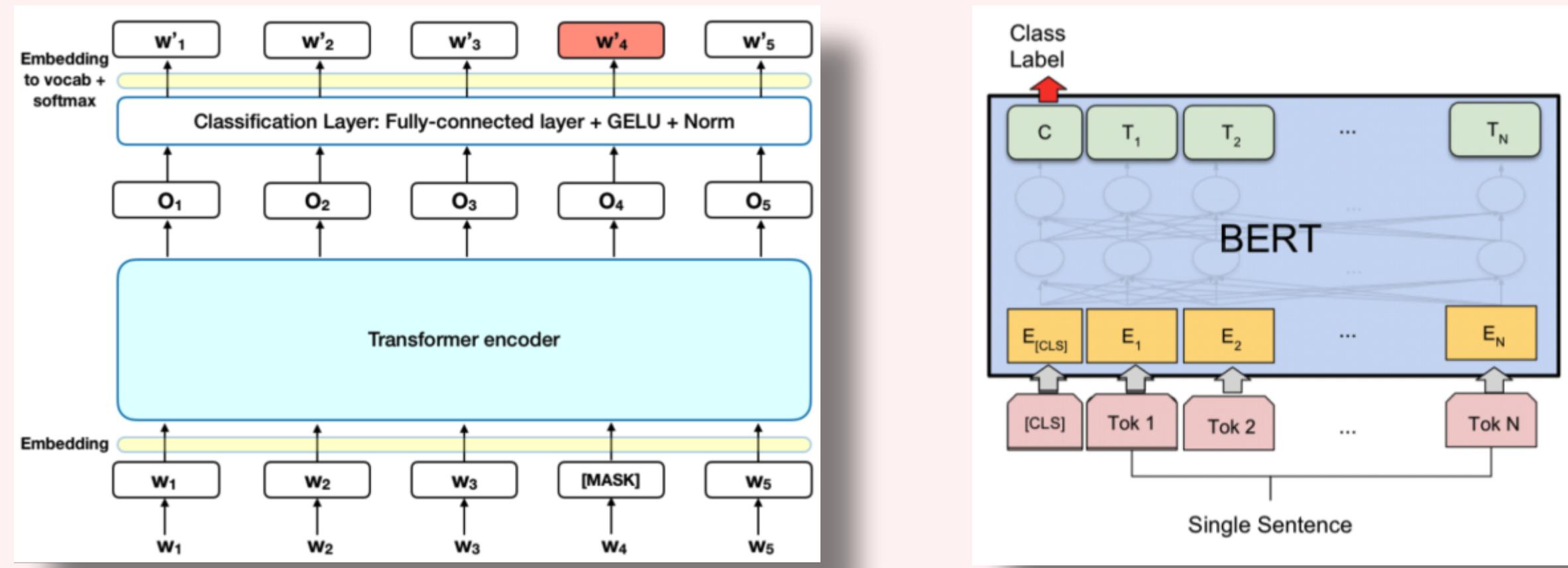
[ 101 9598 22028 9904 9879 119230 11882 9663 67527 9705
  14871 102 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0
[1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[CLS] 원쪽 팔 통증과 저림 증상 [SEP] [PAD] [PAD]
[PAD] [PAD]
```



- Huggingface의 토크나이저 활용

개발 과정

1. 딥러닝: BERT - 모델링



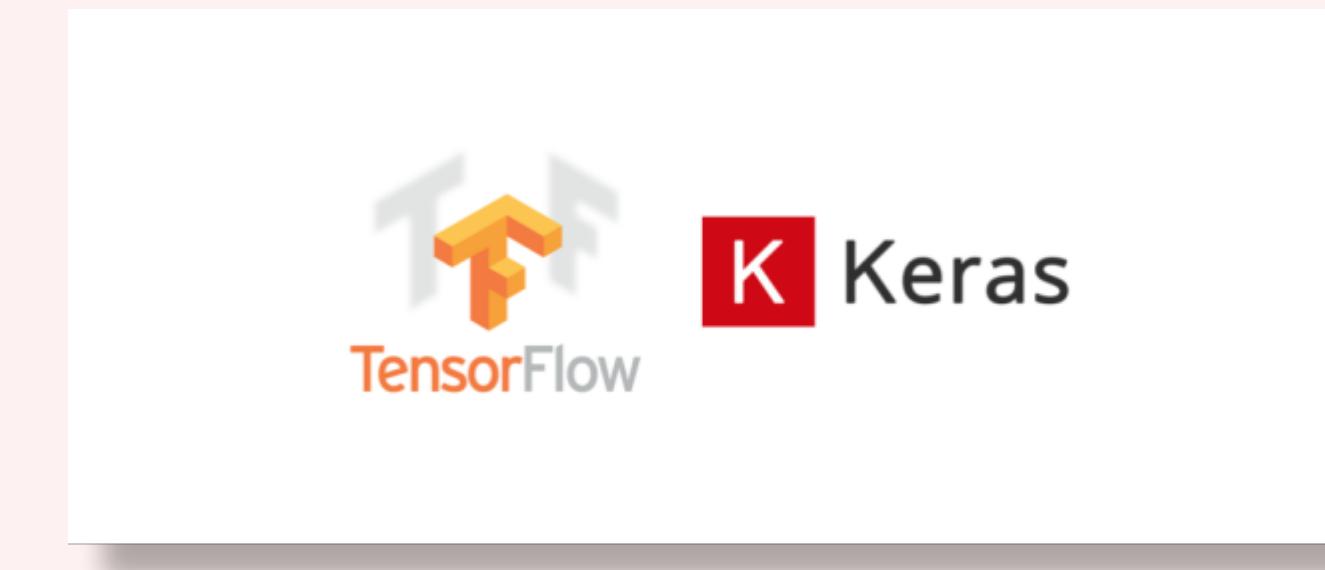
- Masked Language Model(MLM)에 기반을 둔 모델 형태
- 입력에서 무작위하게 몇개의 token을 mask시킨 후, Transformer 인코더로 전달
- Transformer 출력값을 받는 Classification layer(진료과 수=26)를 통해 진료과 라벨을 분류

개발 과정

1. 딥러닝: BERT - 하이퍼파라미터 튜닝

하이퍼파라미터 튜닝 : 모델의 설정값을 바꿔가며 성능을 최대치로 끌어올리는 개발 단계

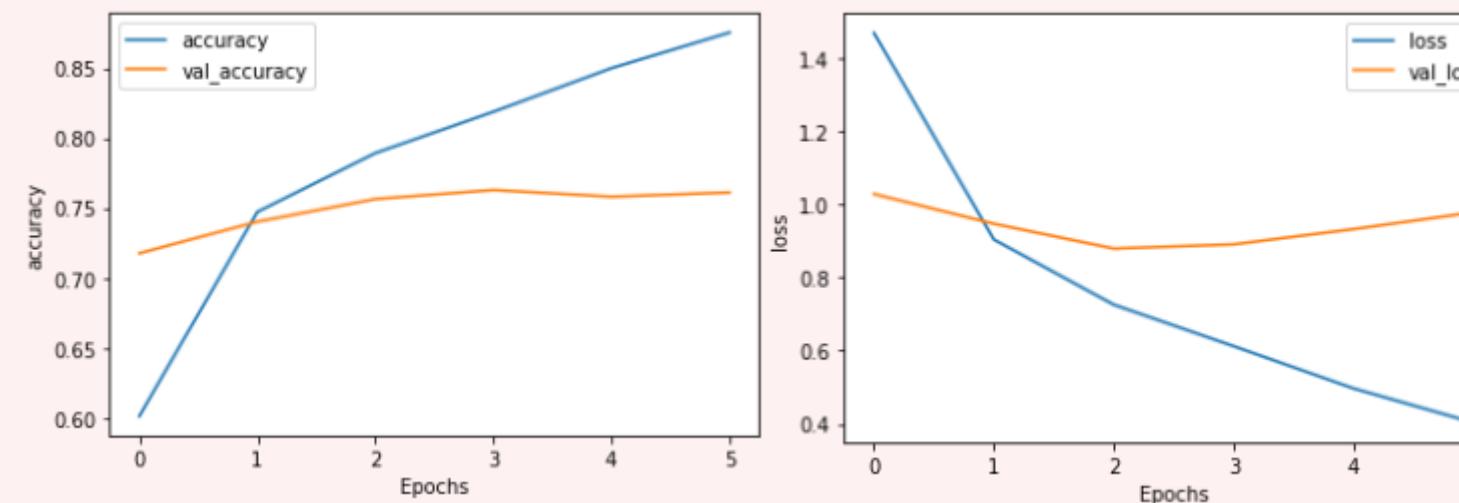
| N | 배치 사이즈 | 학습률 | 에포크 | 정확도 |
|---|--------|------|-----|--------|
| 1 | 32 | 3e-5 | 4 | 76.29% |
| 2 | 16 | 3e-5 | 4 | 76.14% |
| 3 | 32 | 5e-5 | 4 | 75.18% |
| 4 | 16 | 5e-5 | 5 | 75.15% |



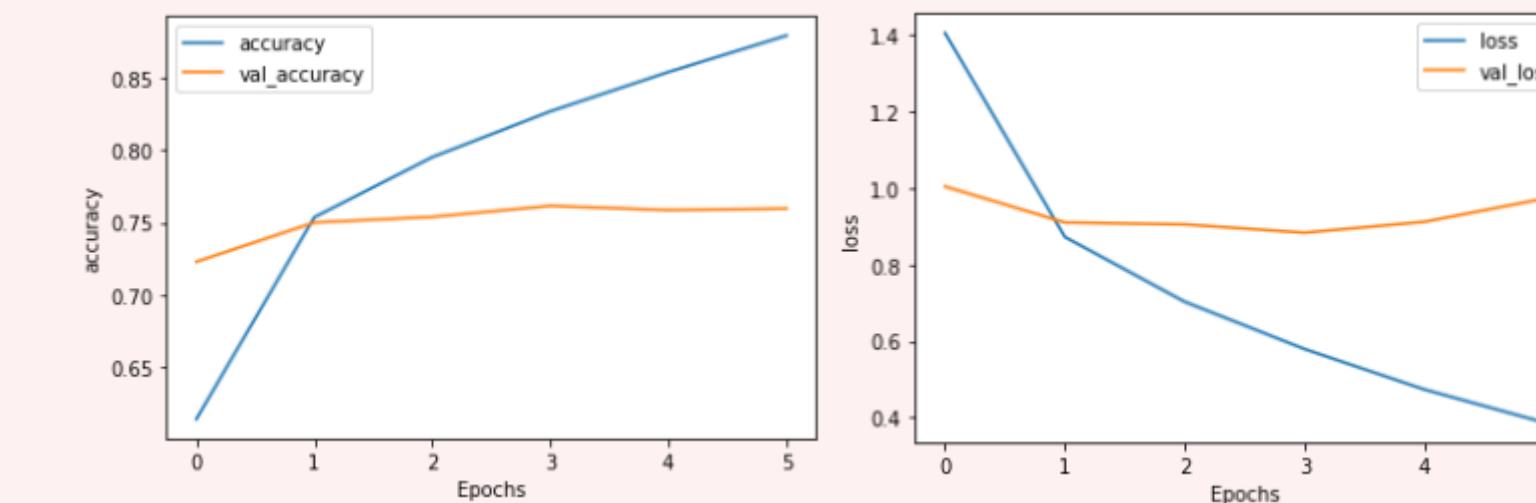
- Optimizer는 Adam으로 고정
- Epochs는 Tensorflow의 Earlystopping(monitor = val_accuracy) 기능을 사용
- Batch size, Learning rate를 계속 수정해 가며 성능 변화 관찰

개발 과정

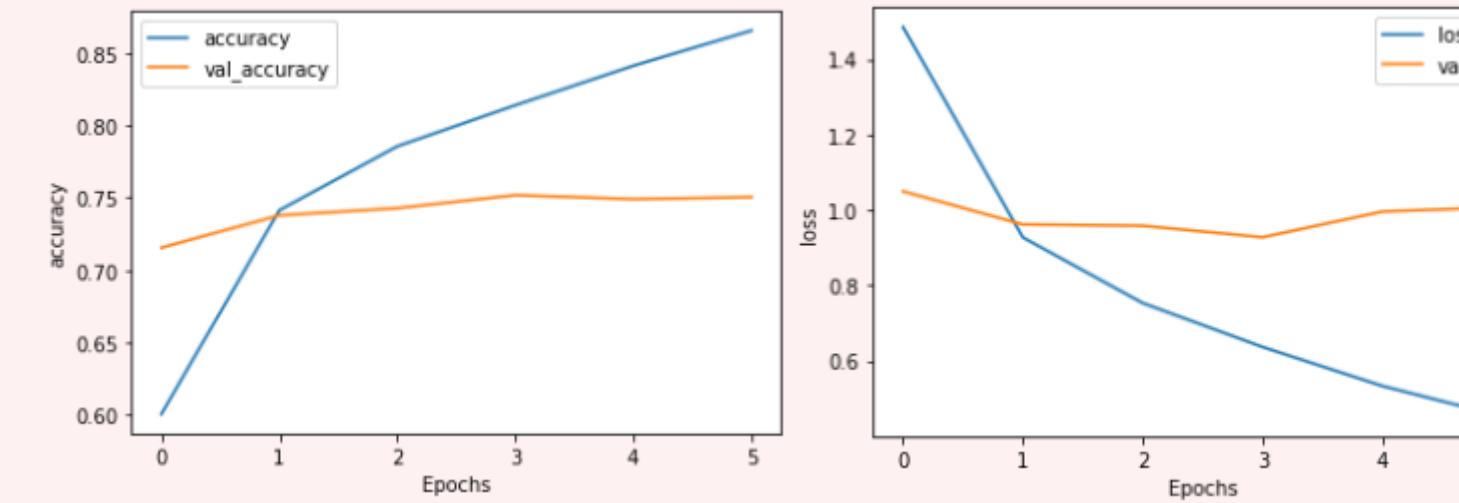
1. 딥러닝: BERT - 모델 평가



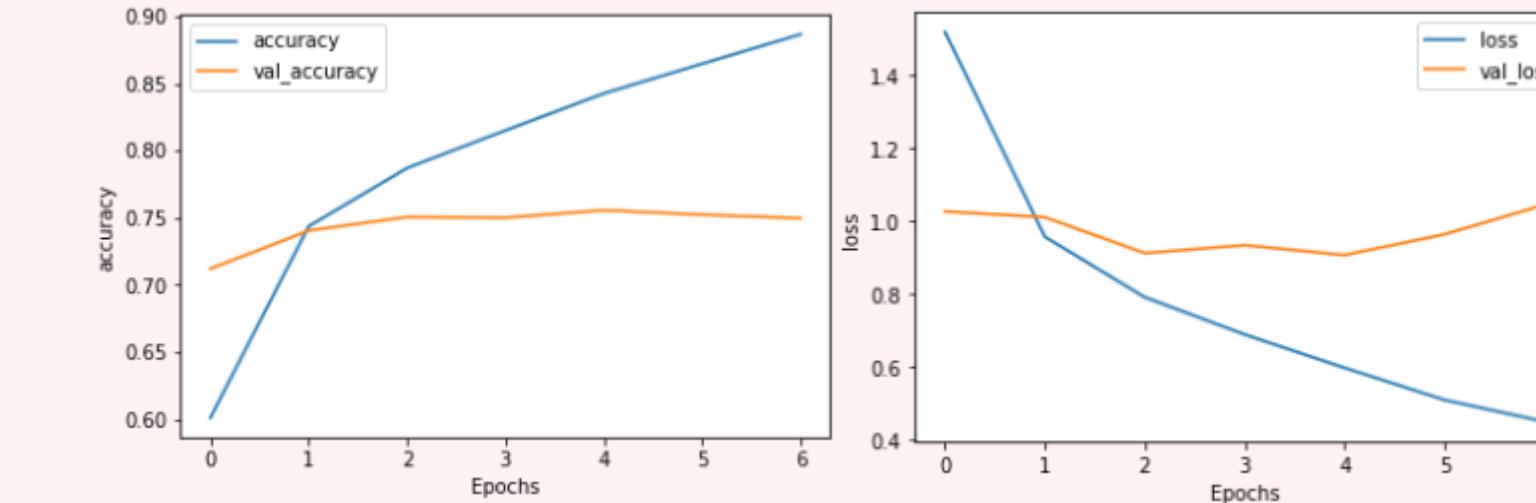
· 모델 1의 정확도, 손실 그래프



· 모델 2의 정확도, 손실 그래프



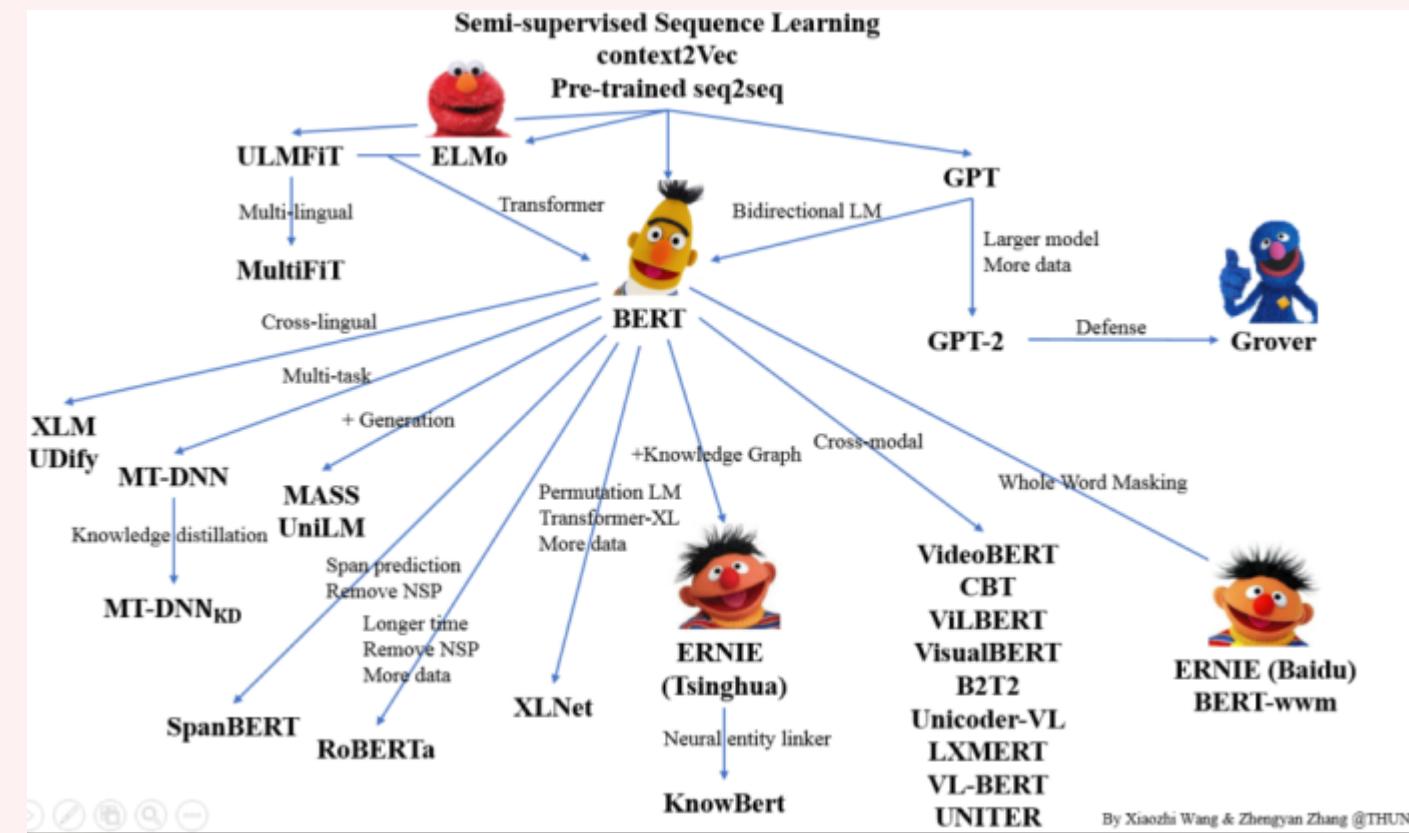
· 모델 3의 정확도, 손실 그래프



· 모델 4의 정확도, 손실 그래프

개발 과정

1. 딥러닝: BERT – 향후 발전성



- BERT는 본 프로젝트의 텍스트 분류 뿐만 아니라 자연어 추론, 개체명 인식, 기계 독해 등 다양한 자연어 분야에서 활용 가능
- 동영상 분석이나 이미지 분류 등 컴퓨터 비전을 포함한 다양한 분야로도 발전 가능
- 오픈 소스로서의 높은 활용가치

개발 과정

1. 딥러닝: 최종 배포용 모델 선택



LSTM
73.52%



FastText
37.44%

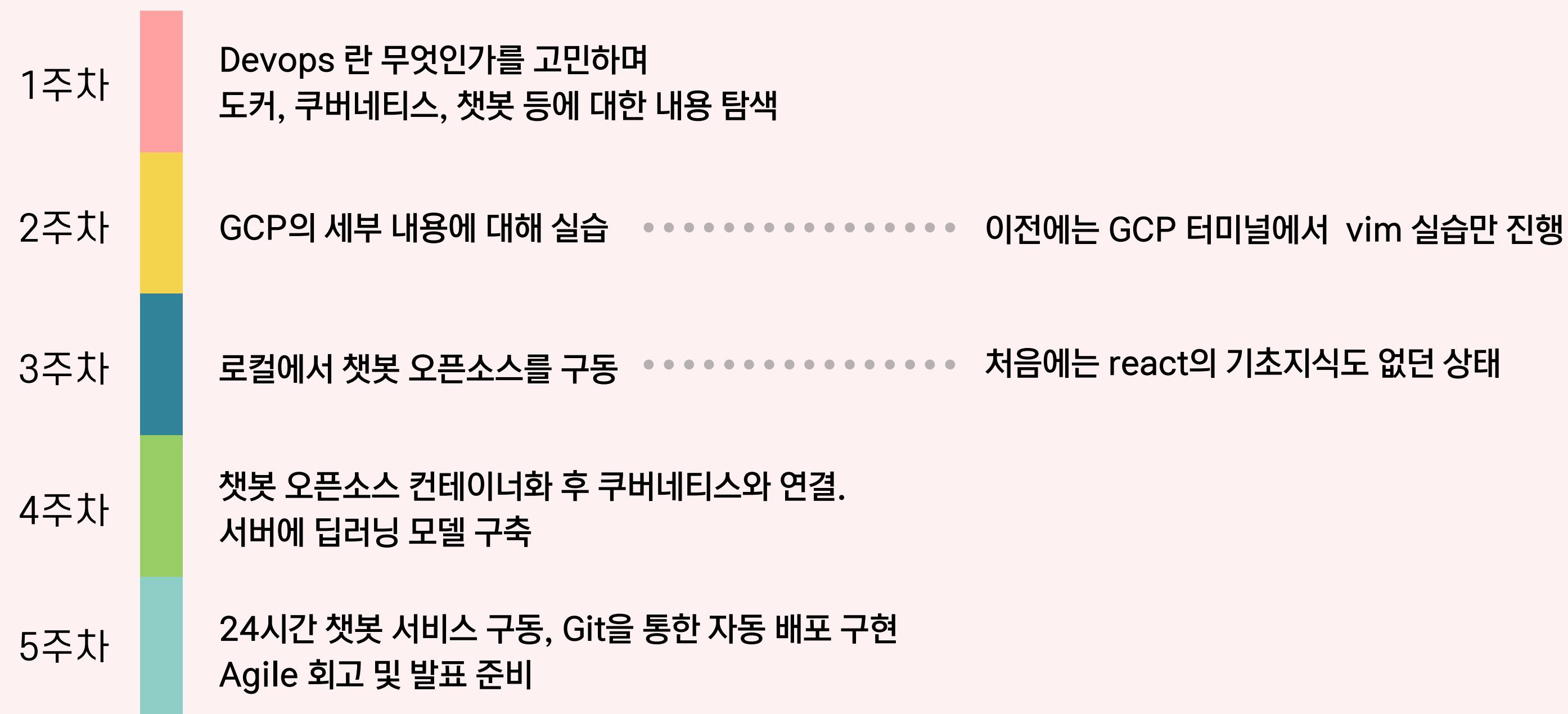


BERT
76.29%

- **BERT 모델**이 최종적으로 약 2.7% 더 좋은 성능 기록
- 하지만 기본 사이즈의 Google cloud vm instance를 사용하는 입장에서 **큰 용량의 모델은 부담**
- 따라서, 성능이 유사하면서도 가볍고 전처리가 간편한 **LSTM 모델을 최종 선택**

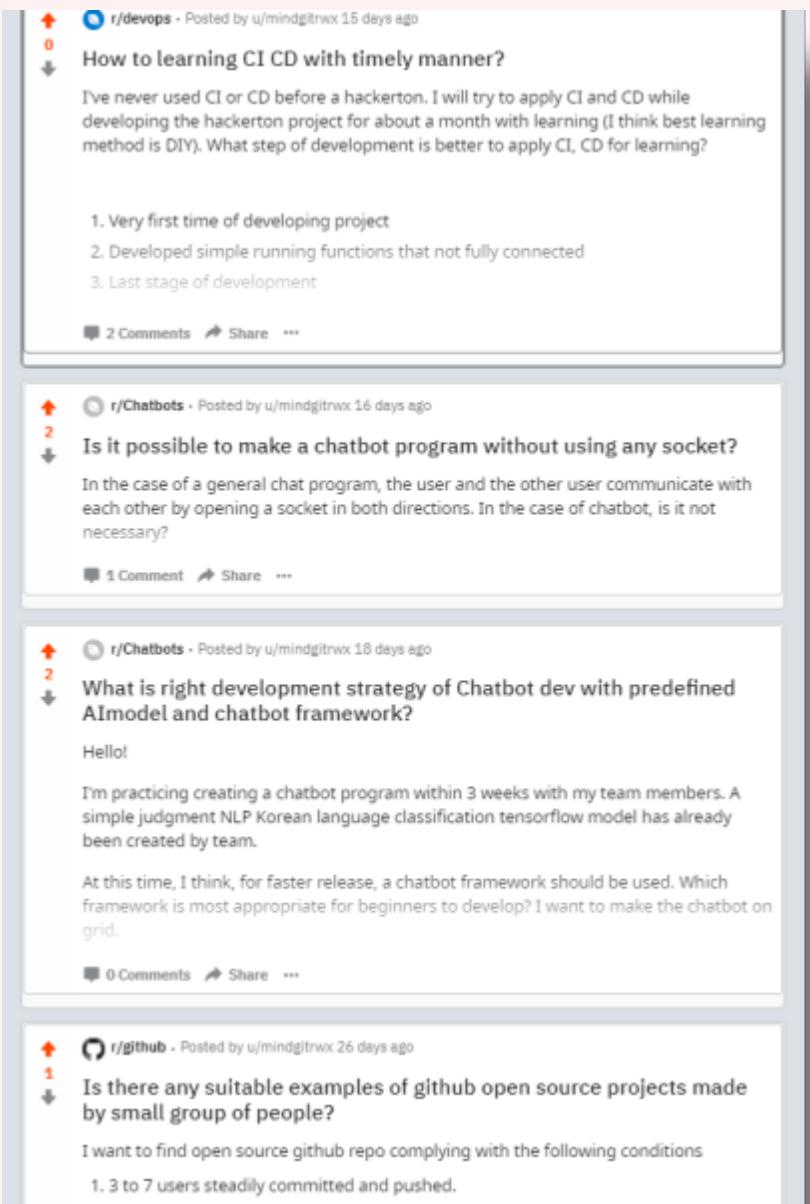
개발 과정

2. 인프라: 개발 타임라인



개발 과정

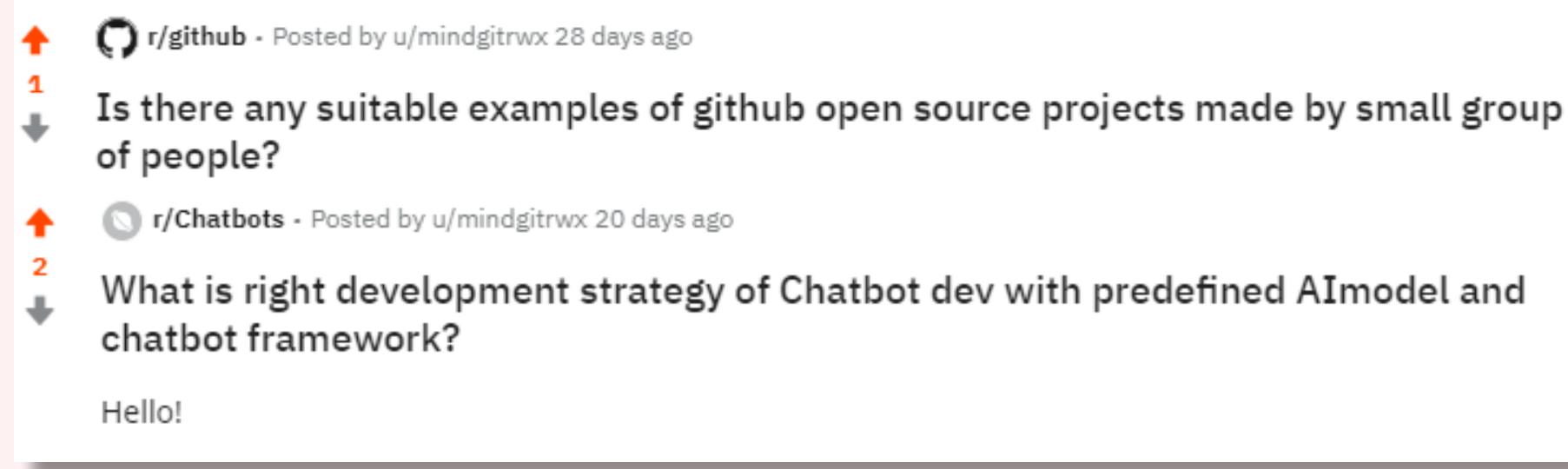
2. 인프라: 개발 범위 정하기



- 챗봇 개발과 devops를 처음 수행
- 군대라는 특수한 상황의 특성상 **가용 시간이 불투명**
- 여러 질문과 토의를 통해서 **개발 범위**를 잡는 것이 중요하다고 판단
- 구글링과 Reddit을 통해 개발자들의 경험과 조언 참고

개발 과정

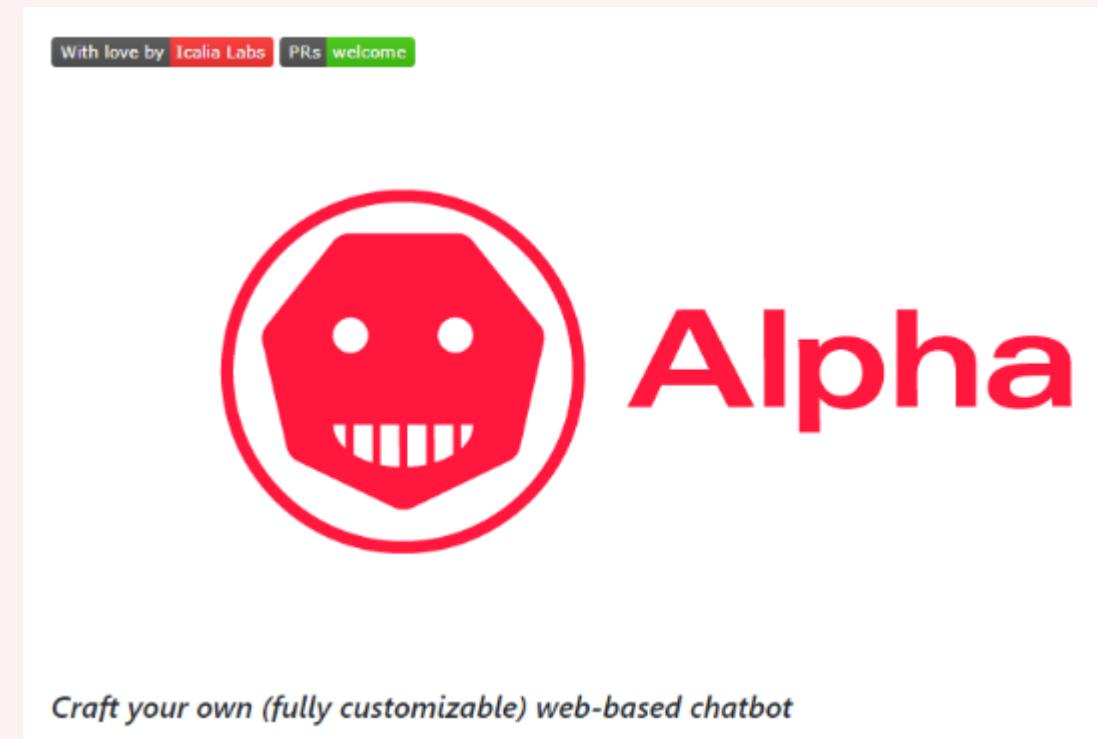
2. 인프라: 오픈소스 선택



- 개발 초기 계획 수립때 기업형 채팅 솔루션은 사용하지 않기로 결정
- 빠른 개발을 위해서 다른 오픈소스 저장소들을 참고
- 참고할 만한 오픈소스를 찾기 위해 여러 커뮤니티를 돌아다니며 질문

개발 과정

2. 인프라: 챗봇 오픈소스 소개 및 선정 이유

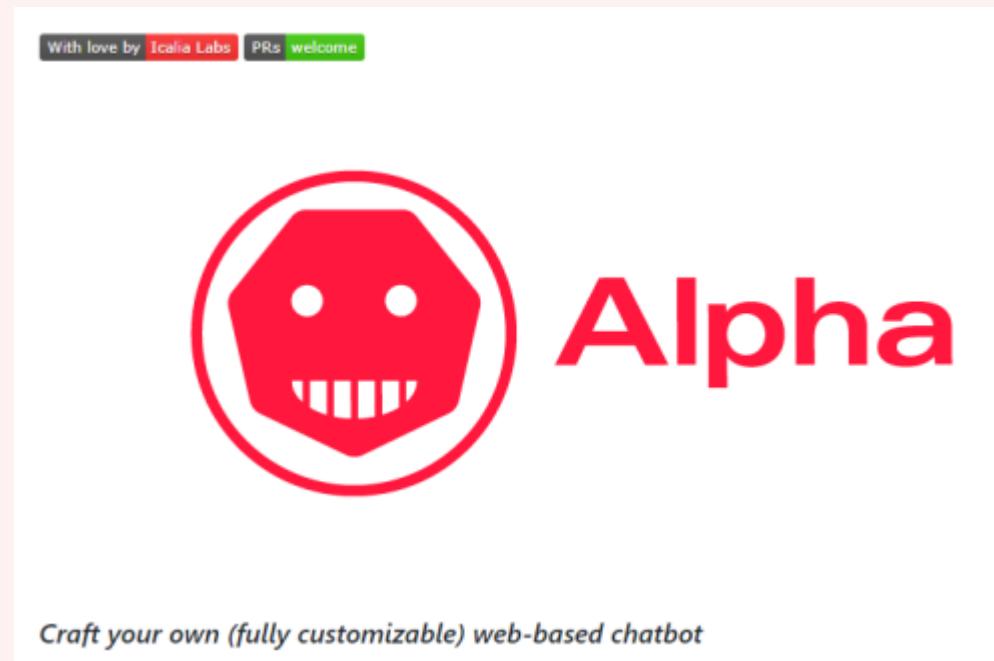


- 24시간 AI챗봇 프로젝트를 하면서 바닥부터 chatbot UI를 만든다는 것은 불가능하다고 판단
- 위 오픈소스 프로젝트의 구조 자체가 결합성이 낮고, 기능을 추가하기에 적합
- Docker 구성이 잘 되어있으며, react와 nodejs로 구성되어 있어 우리 팀의 스택과 일치
- 무엇보다도 챗봇 자체에서 React와 Webpack으로만 구성

개발 과정

2. 인프라: 참고했던 오픈소스

IcaliaLabs/alpha



- 한 명의 슈퍼개발자가 첫 커밋으로 모든 기능을 개발
- 하지만 GIT flow는 부족
- 해커톤이 끝나고 해당 오픈소스의 한글화에 참여 예정

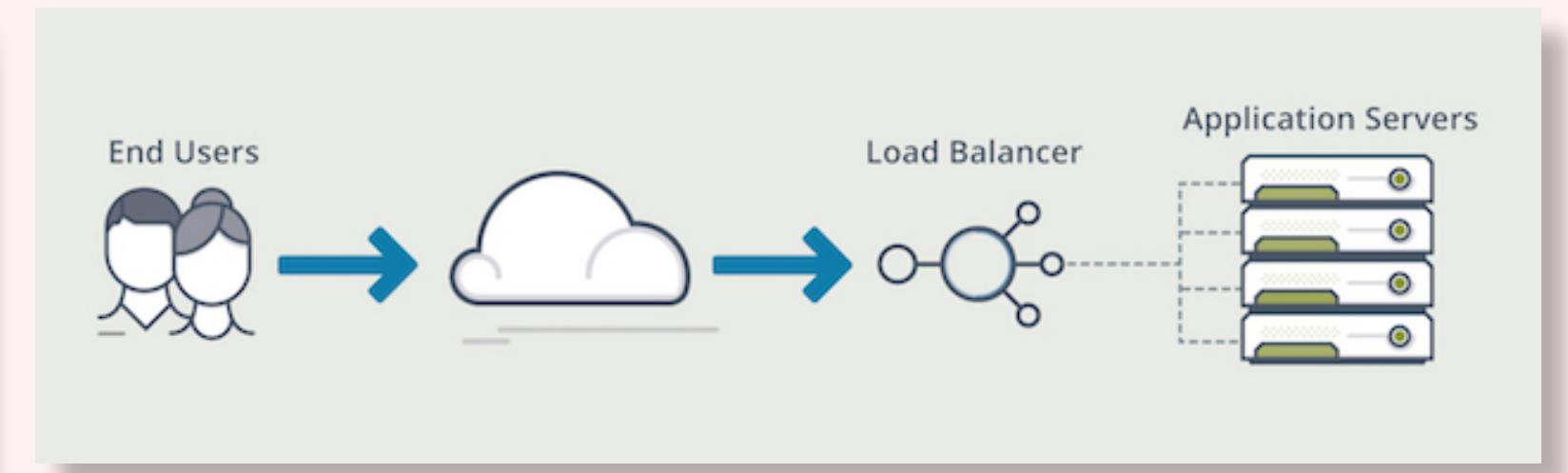
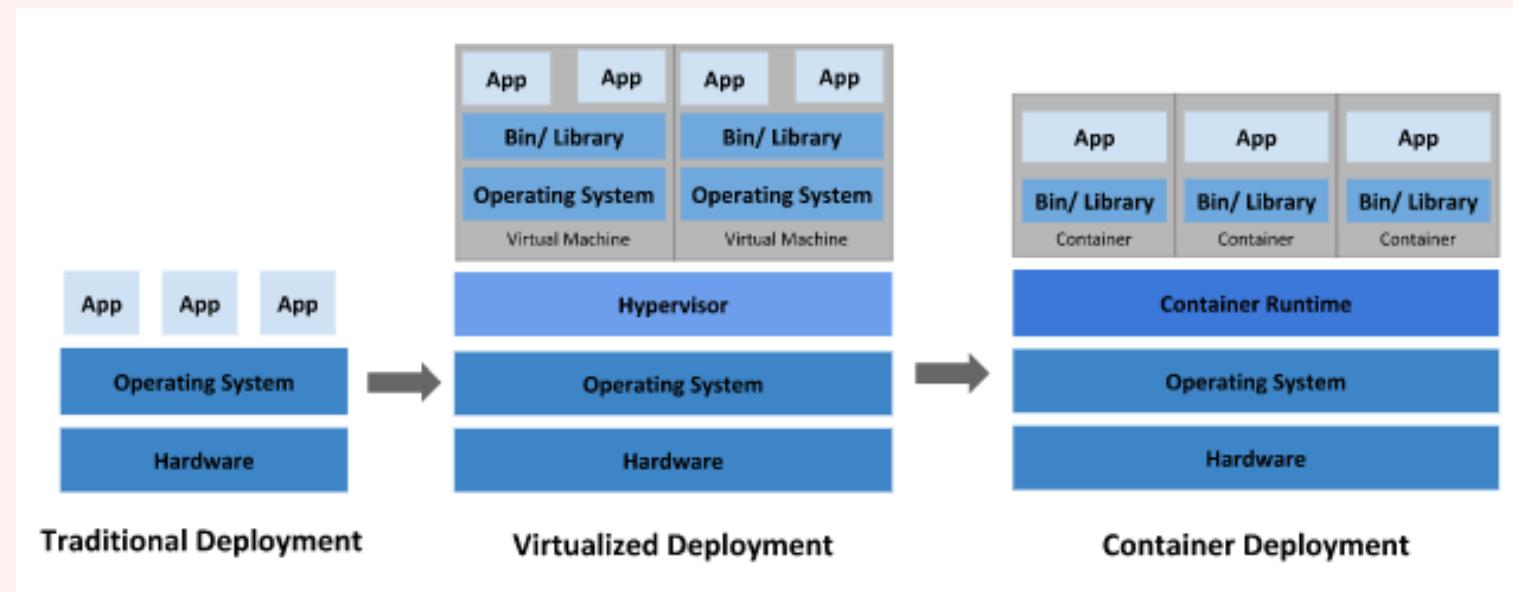
Analysis-pipelines



- Git 개발 플로우가 아주 잘 구성
- Meditact와 같은 NLP 프로젝트이기에 참고
- 하지만 눈에 보이는 채팅 기능이 부족

개발 과정

2. 인프라: 세팅 계획



- 쿠버네티스와 **로드 밸런서***를 이용한 24시간 무중단 챗봇 쿠버네티스 환경 구축 계획

*로드 밸런서

- 유저가 보내는 메세지를 다양한 서버로 분배
- 한 서버에 과부하가 걸리지 않도록 트래픽을 자동으로 분배해주는 역할

개발 과정

2. 인프라: 서버 구축

구축 완료된 쿠버네티스 서버

The screenshot shows the Kubernetes Engine dashboard under the 'Services & Ingress' tab. It displays a list of services and their corresponding endpoints. Services listed include meditact-djnd8, meditact-8xfqs, and meditact-1-1-1-service. Each service entry shows its status (OK), port, and IP address.

GIT을 이용한 지속적 배포

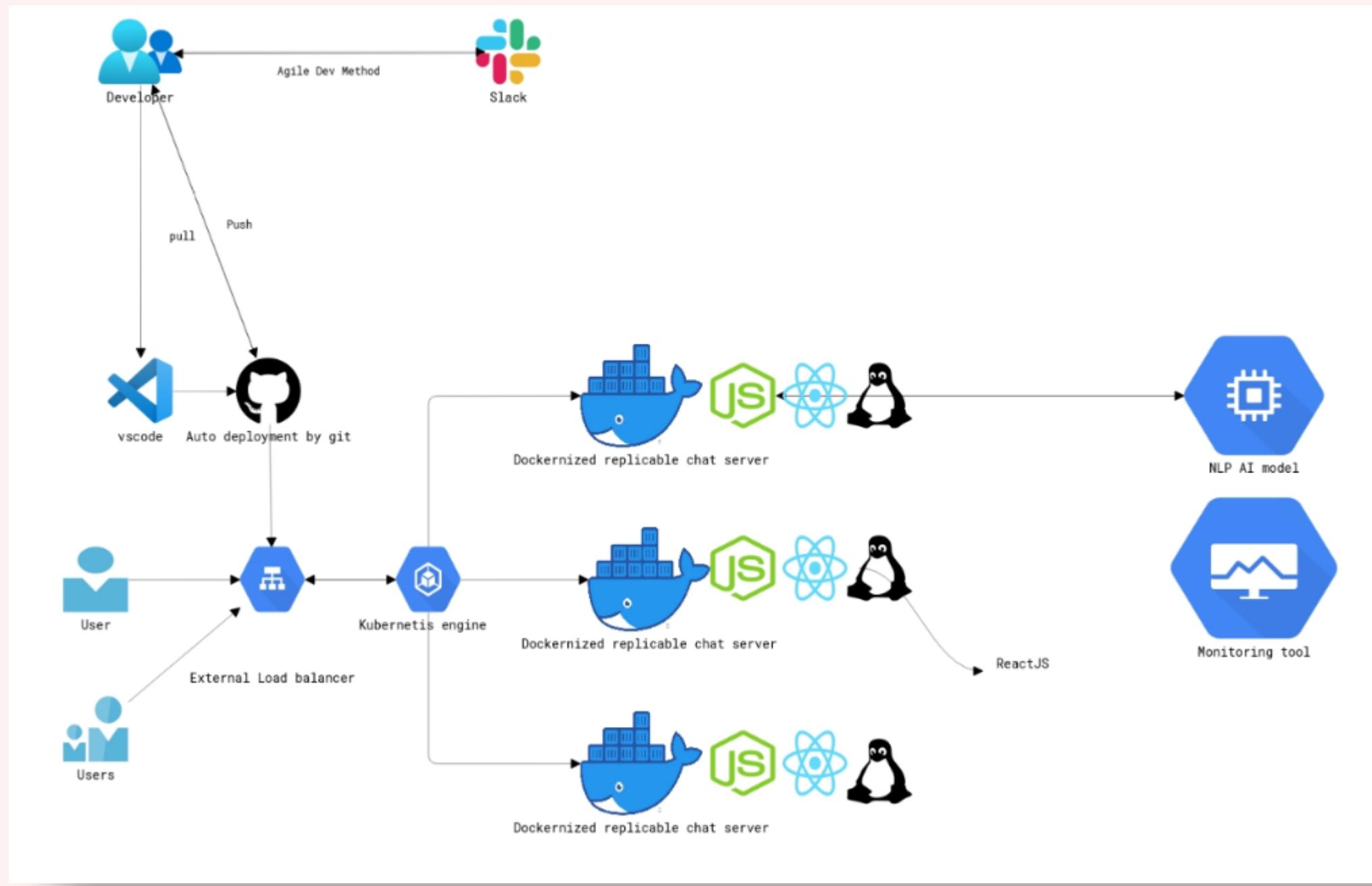
The screenshot shows the Google Cloud Build console. The main area displays a list of build logs for the project 'meditact'. Below this, there are two detailed views of specific builds: one for a successful deployment and one for a failed deployment. Each detailed view includes information such as build ID, start time, duration, and status.

- 외부 부하 분산기를 사용
- Meditact 챗봇을 외부에서 이용 가능

- 챗봇 소스를 저장해 둔 repo에 push를 하면 자동으로 빌드후 웹 페이지에 업데이트되도록 GCP에서 셋팅

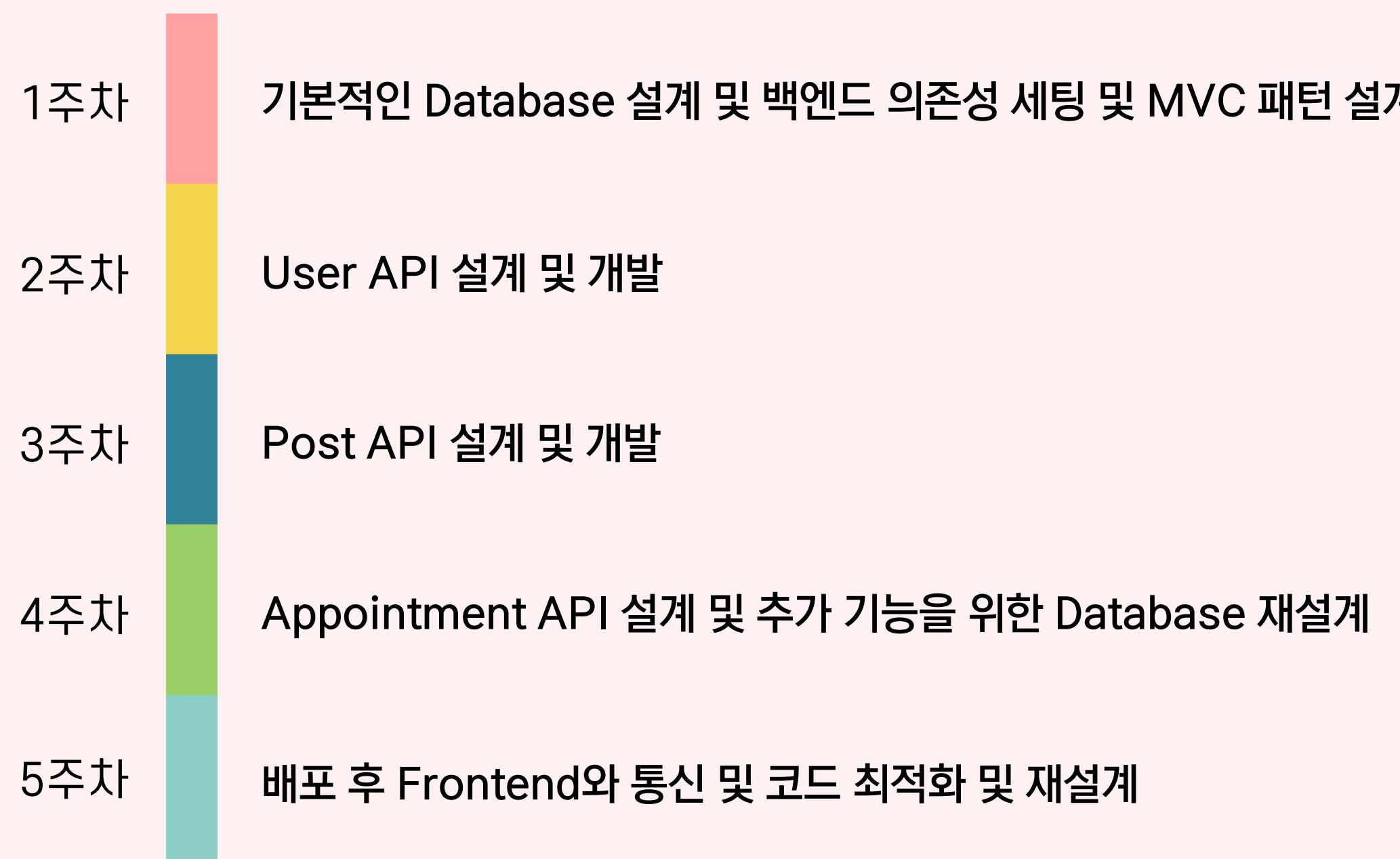
개발 과정

2. 인프라: 최종 구조



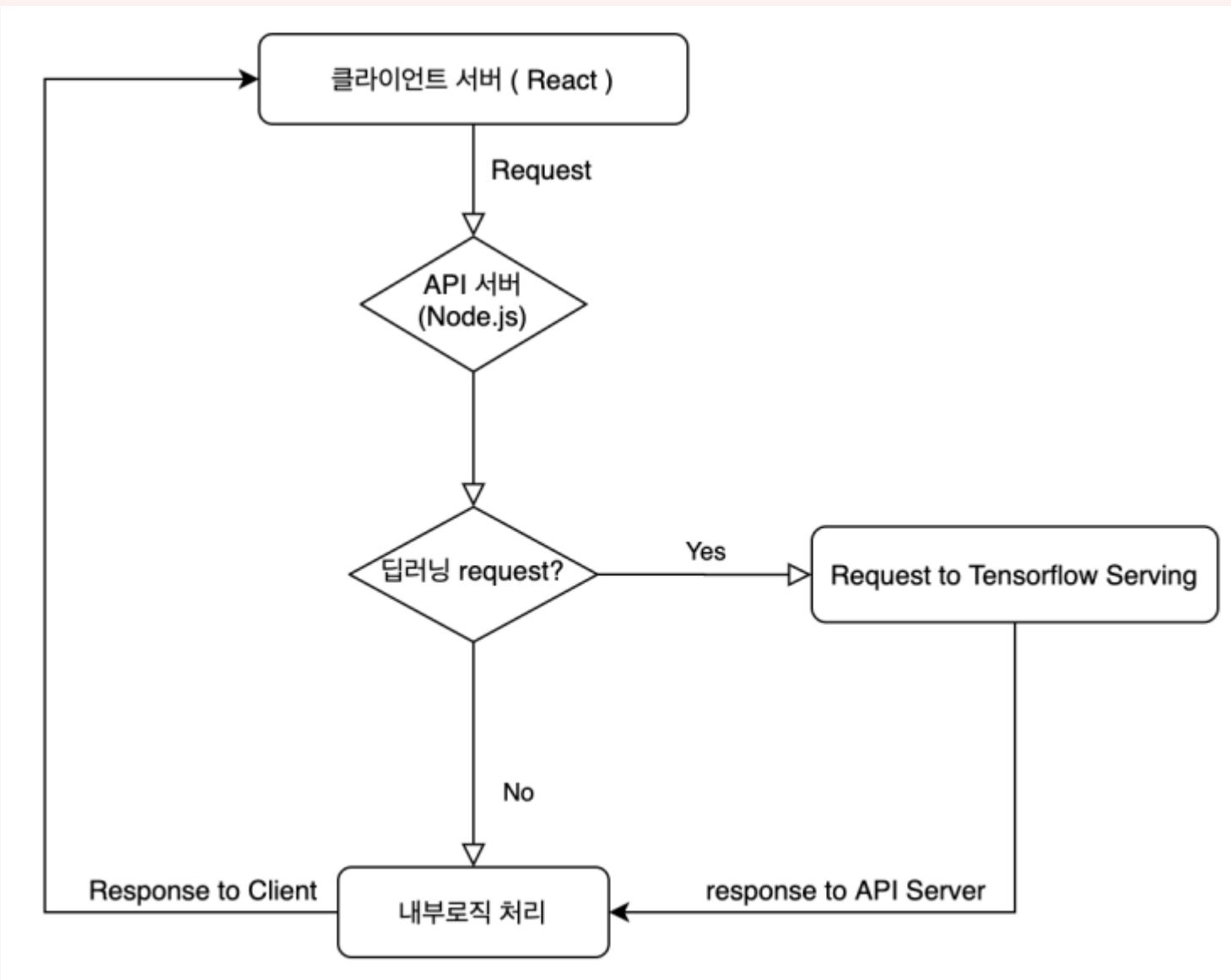
개발 과정

3. 백엔드: 개발 타임라인



개발 과정

3. 백엔드: 로직 설계 과정



기존 계획

- 클라이언트와 백엔드 API 서버, tensorflow 딥러닝 서버 두 개를 사용
- 딥러닝 - 백엔드 - 챗봇 구조로 연결

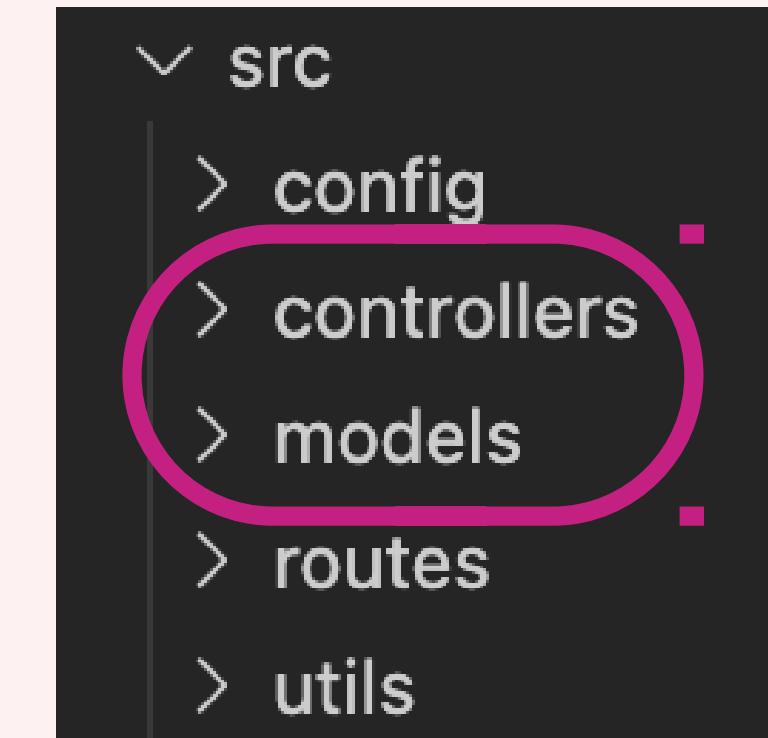
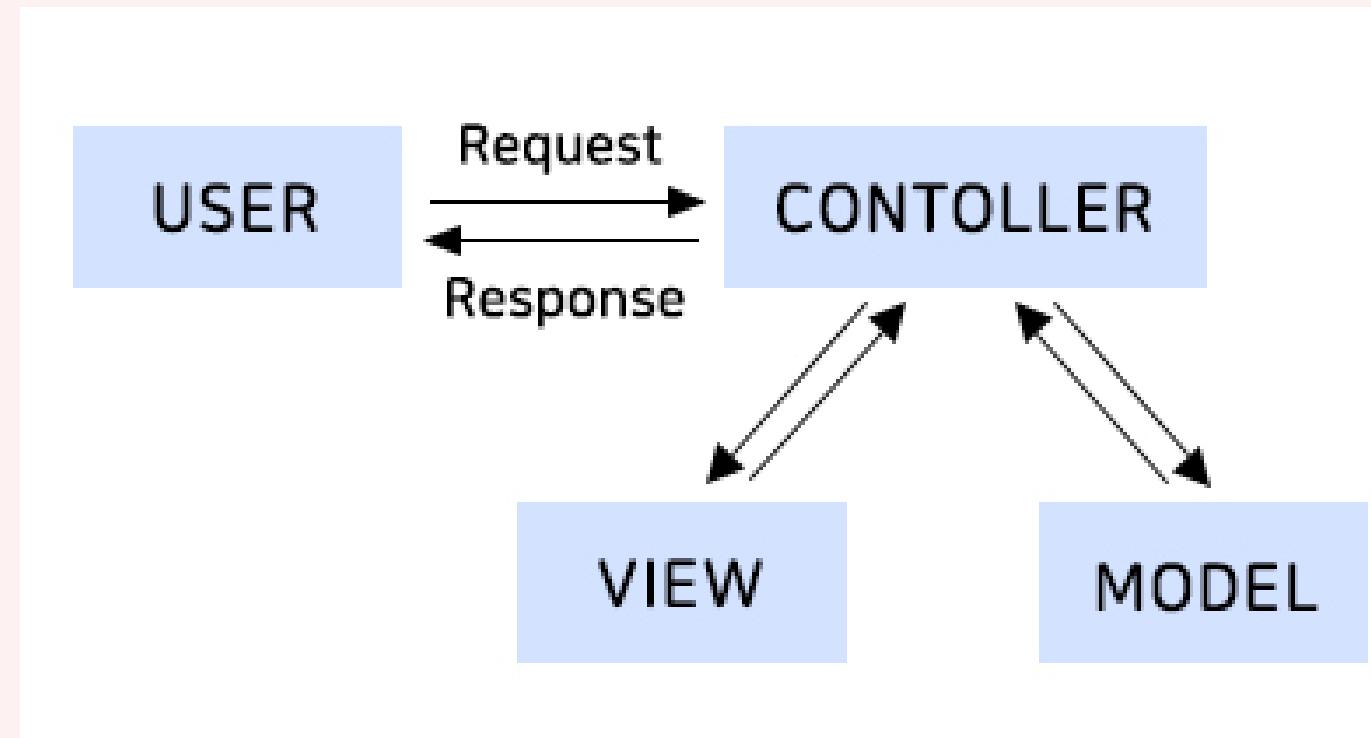


수정 계획

- 상담과 예약을 위한 웹페이지는 백엔드 API를 사용
- 챗봇은 딥러닝 서버와 직접 통신하는 것으로 변경

개발 과정

3. 백엔드: MVC 패턴



- **MVC 패턴**을 지키기 위해 Model과 Controller를 나누어서 개발
- View는 프론트엔드가 존재하므로 그쪽과 통신하는 방향으로 REST API를 설계
- 비즈니스 처리 로직과 사용자 인터페이스 요소들을 분리시켜 **동시에 개발 가능**

개발 과정

3. 백엔드: User API 설계

| END POINT | 설명 |
|------------------|---|
| /api/user | User의 CRUD를 구성하는 End point POST, UPDATE, DELETE, PUT, GET 모두 동작하는 API |
| /api/users | Authorization token을 통해 Role을 검증하여 User의 정보를 조회 할 수 있는 Admin을 위한 API |
| /api/users/login | POST를 이용하여 User의 Login 기능을 구현한 API |
| API docs | https://github.com/osamhack2020/WEB_Meditact_Meditact/blob/main/server/APIdocs/UserAPI.md |

- User 모델의 기본적인 필드*를 정의하여
그에 맞는 REST API를 설계
- 보안을 위하여 **Crypto.js** 를 이용
- 비밀번호는 hash와 salt를 이용한 암호화를 진행
- Authorize token을 이용하여 보안을 강구

➤ *필드 구성
이름, 이메일, 나이, 신장, 체중, 전화번호

개발 과정

3. 백엔드: Post API 설계

| END POINT | 설명 |
|-----------|---|
| /api/post | Post의 CRUD를 구성하는 End point POST, UPDATE, DELETE, PUT, GET 모두 동작 User의 Role을 검증하기 위해 POST, GET, DELETE 동작 시 Auhorize 미들웨어를 한 번 거치도록 설계 |
| /upload | Post의 사진, 영상을 Upload하기 위해 오픈소스인 multer 모듈을 이용하여 개발한 API |
| API docs | https://github.com/osamhack2020/WEB_Meditact_Meditact/blob/main/server/APIdocs/PostAPI.md |

- 필드*를 정의한 후 그에 따른 RESTful 설계를 지향
- User의 Role권한에 따라 API 접근 권한 설정

➤ *필드 구성
Title, Content, Category

개발 과정

3. 백엔드: Appointment API 설계

| END POINT | 설명 |
|-------------------|--|
| /api/appt | Appointment(병원 예약)의 생성 및 조회(POST, GET)를 위한 API 예약 시간과 의사 환자의 정보가 필요 confirm은 아래 End Point의 API를 이용하여 진행 |
| /api/appt/confirm | 위에서 생성 된 Appointment를 승인 및 취소를 할 때 사용되는 API POST를 이용하여 사용 Authorization token을 이용하여 Role을 검증 |
| API docs | https://github.com/osamhack2020/WEB_Meditact_Meditact/ blob/main/server/APIdocs/APptAPI.md |

- 병원 예약을 위한 API
- 환자와 의사는 User Model과 Reference를 설정 시
authorization token을 이용한 Role 검증

▶ 필드 구성
예약 시간, 환자, 의사

개발 과정

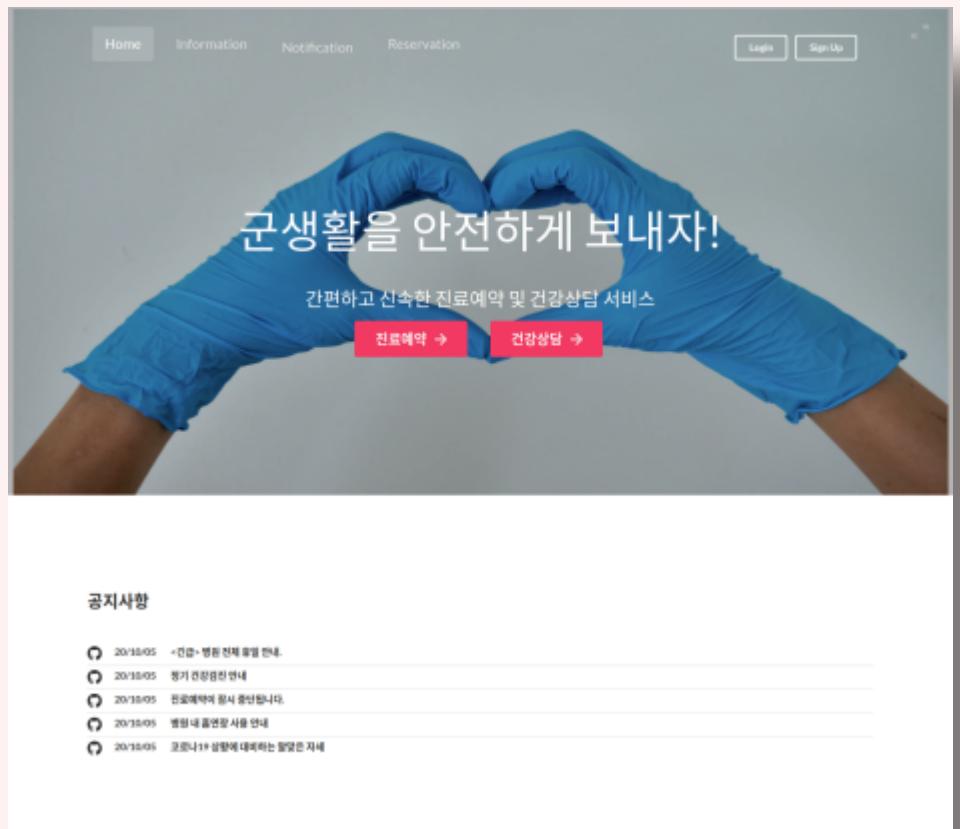
4. 프론트엔드: 개발 타임라인

| | |
|-----|--|
| 1주차 | vscodespace 환경 셋팅 사용할 라이브러리, dev tools, package version 선택 페이지별 필요한 기능 정리 |
| 2주차 | User : 메인 페이지, 공지사항, 로그인, 회원가입 디자인(80%) 예약페이지 디자인(50%) |
| 3주차 | User : 메인 페이지, 공지사항, 예약페이지, 로그인, 회원가입 디자인 구성 완료 Data : api를 사용하여 DB 데이터 사용 테스트 |
| 4주차 | User : 정보출력 페이지, 건강관리 페이지 디자인 구성 완료 Data : 프론트 측면에서 DB데이터 사용 |
| 5주차 | User : 마이페이지 데이터 연결완료(상담 기능 포함) Admin : 예약관리 페이지 디자인 완료 Medic : 자신의 스케줄 관리 페이지 디자인 및 개발 완료 |

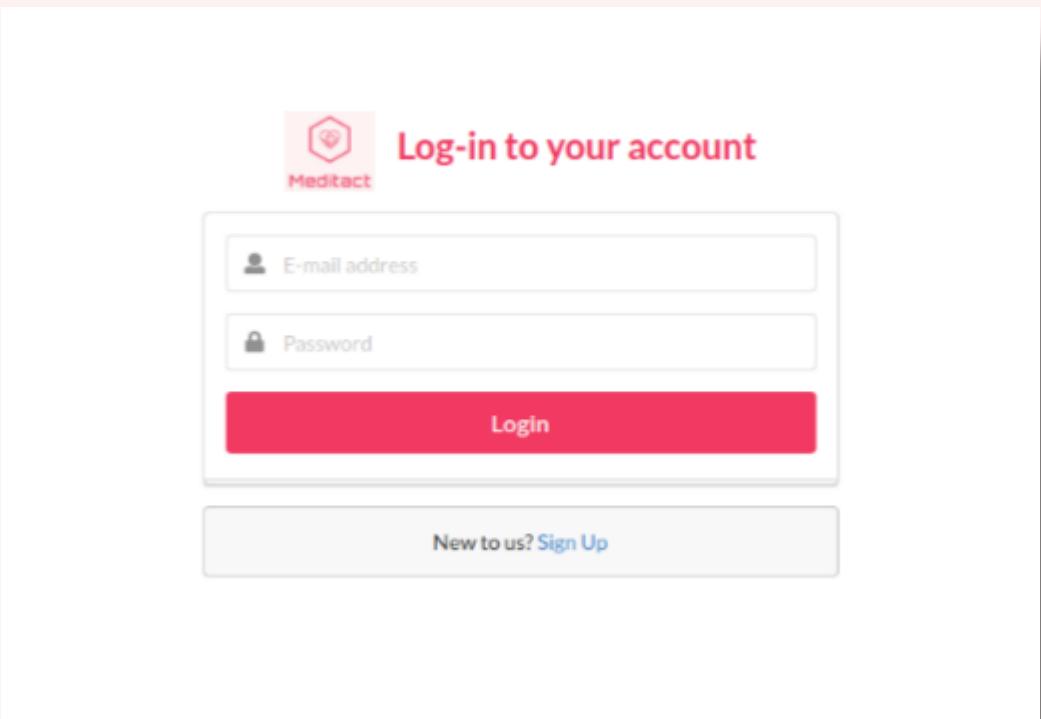
개발 과정

4. 프론트엔드: 페이지 소개(1)

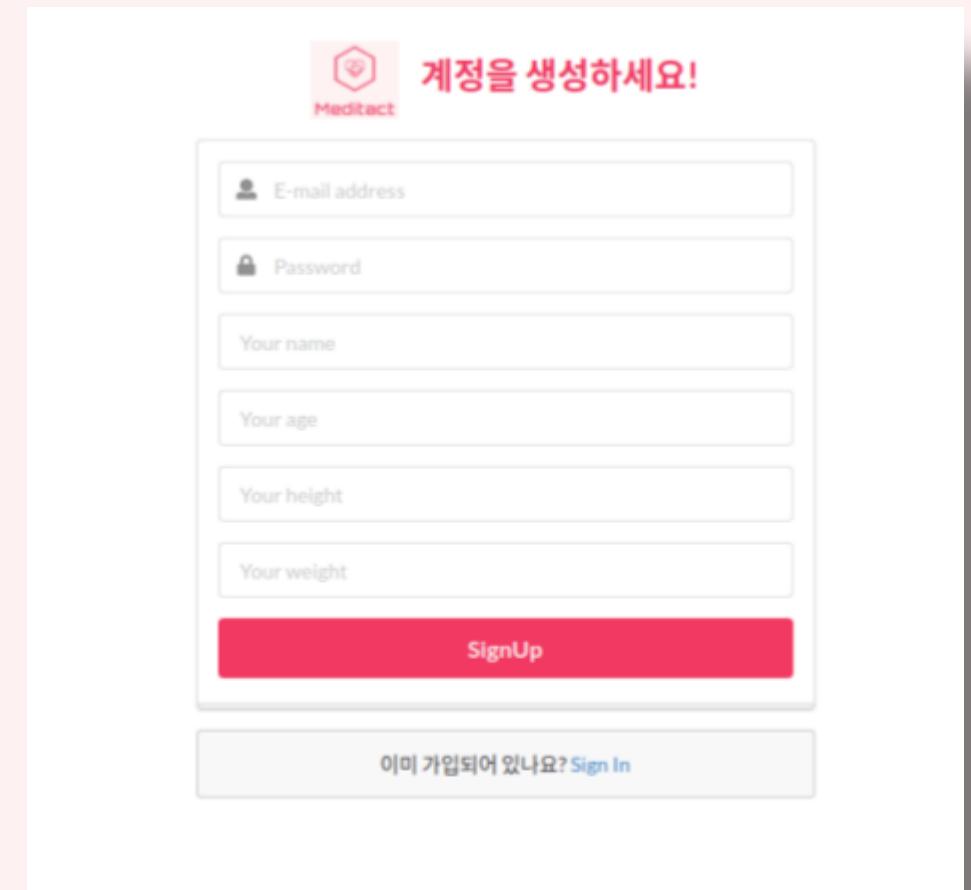
메인 페이지



로그인 페이지



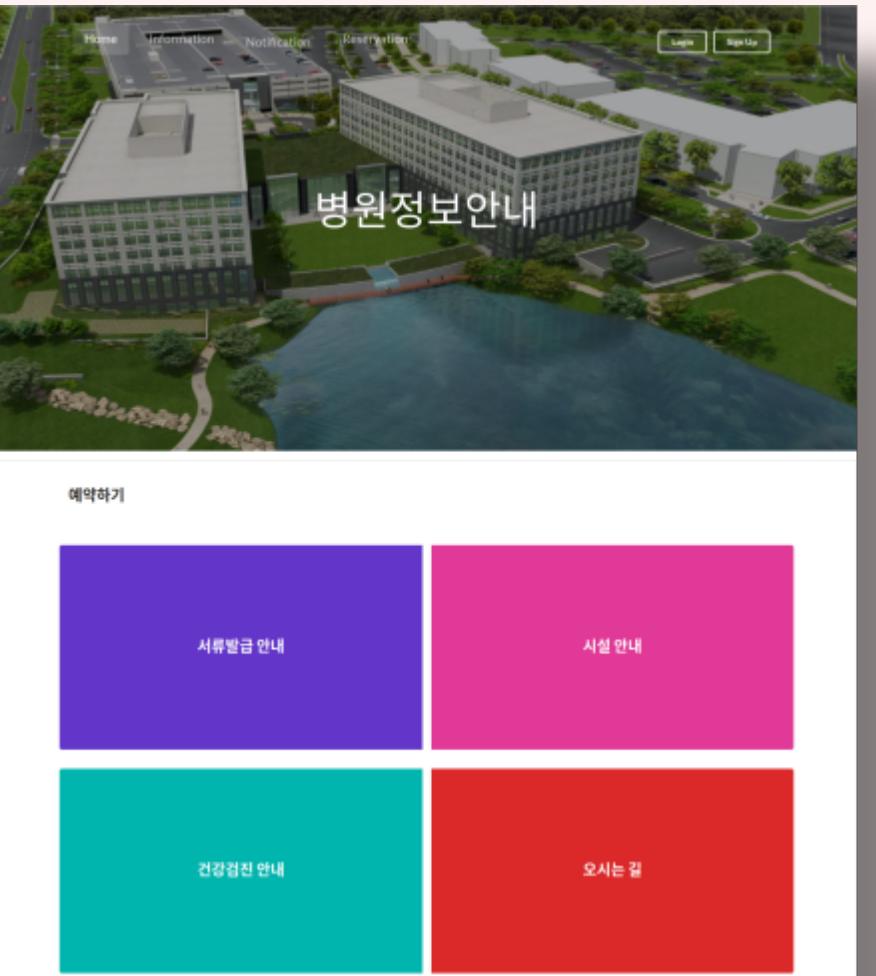
회원가입 페이지



개발 과정

4. 프론트엔드: 페이지 소개(2)

병원 정보 확인 페이지



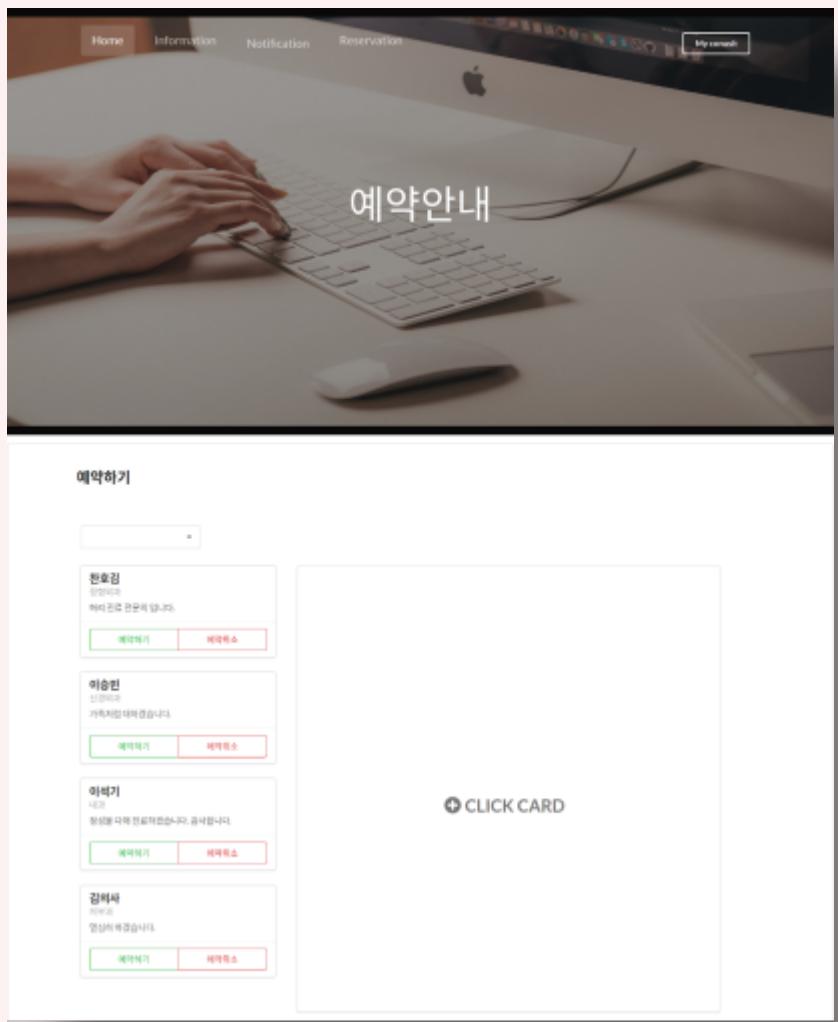
건강 관리 페이지



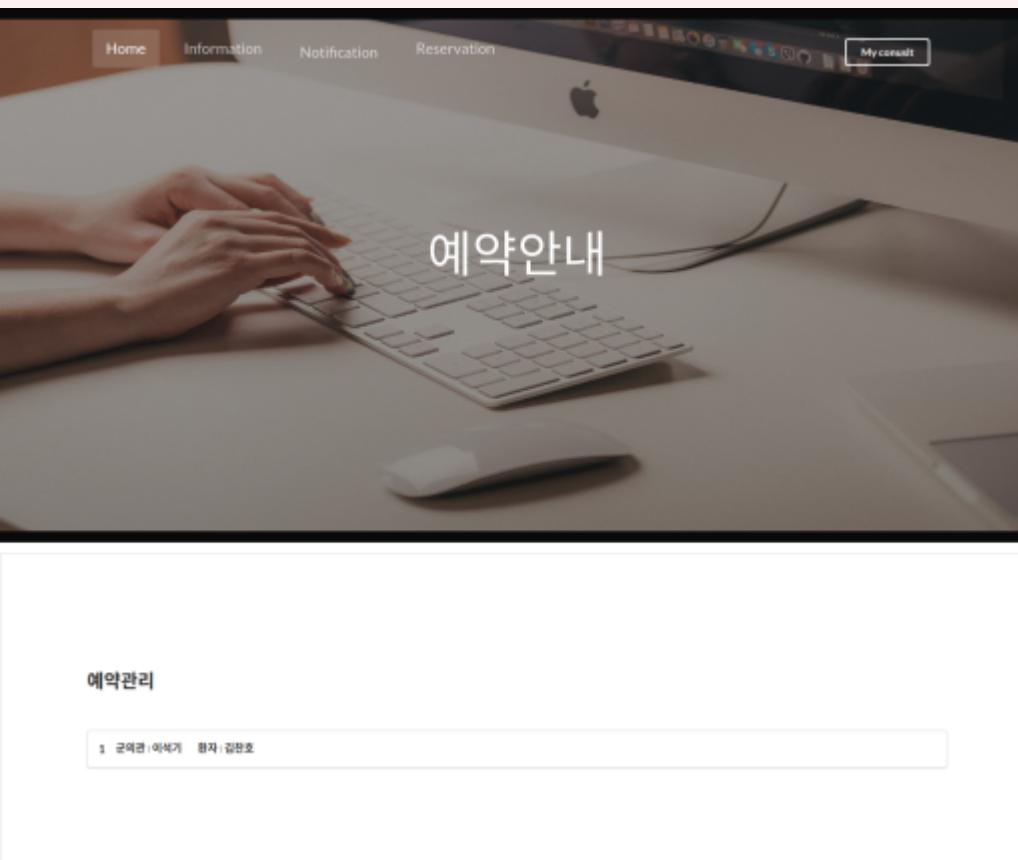
개발 과정

4. 프론트엔드: 페이지 소개(3)

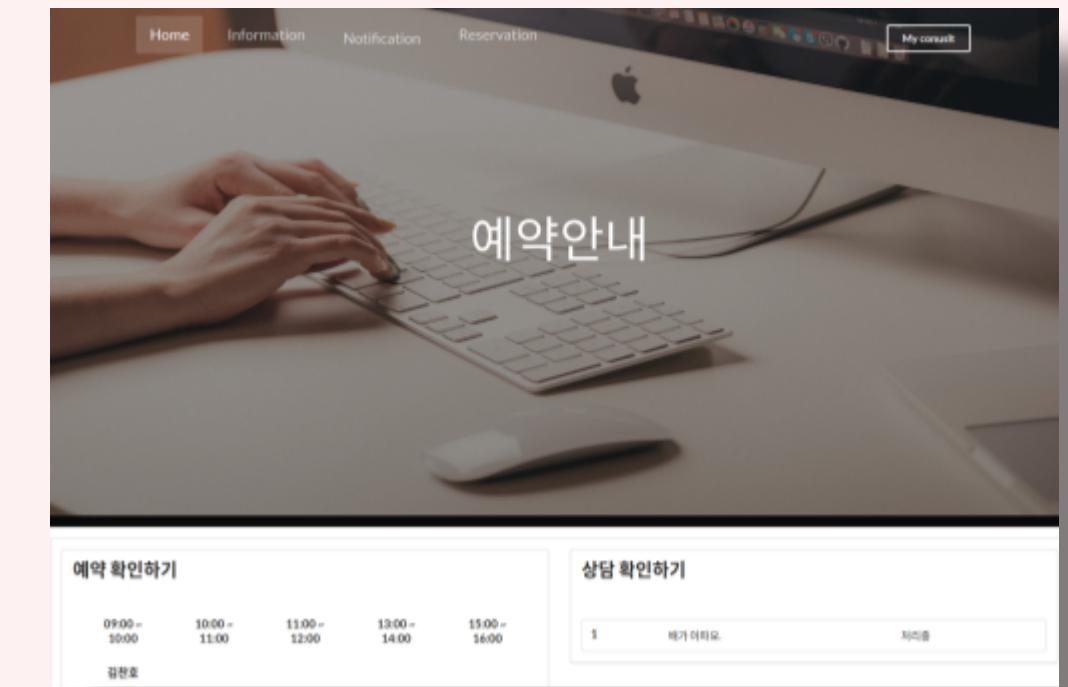
예약 페이지 (사용자)



예약 페이지 (관리자)



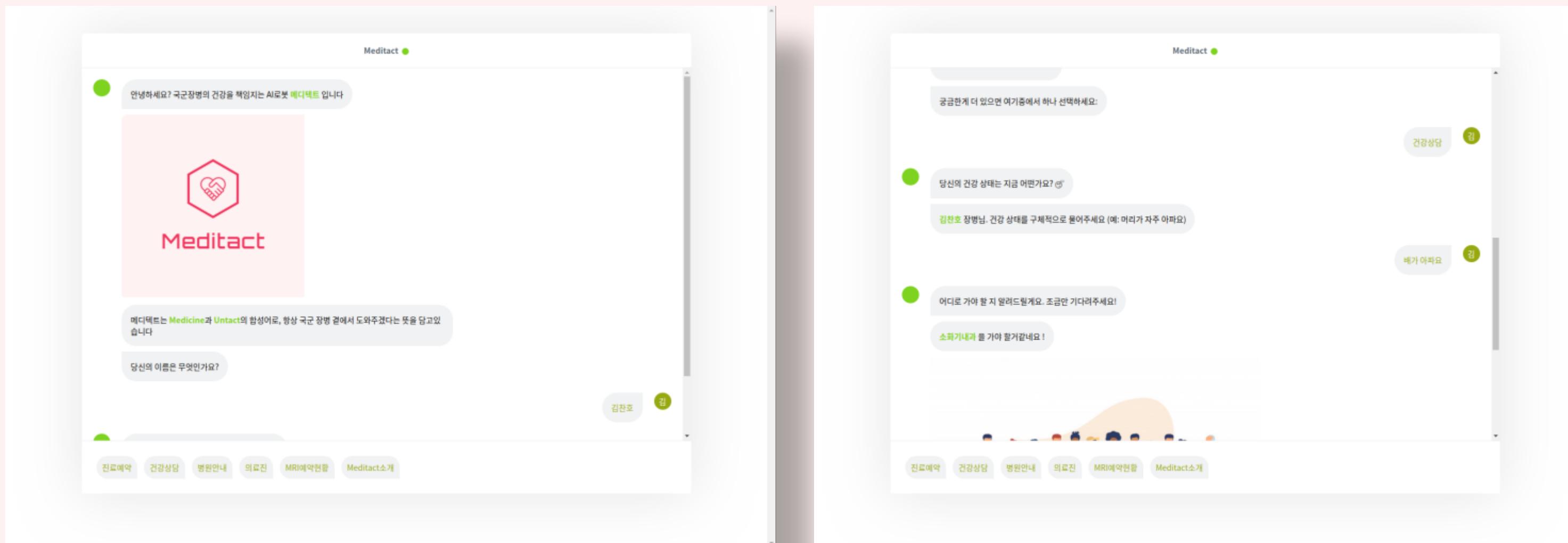
예약 페이지 (군의관)



개발 과정

4. 프론트엔드: 페이지 소개(4)

챗봇 구동 스크린샷

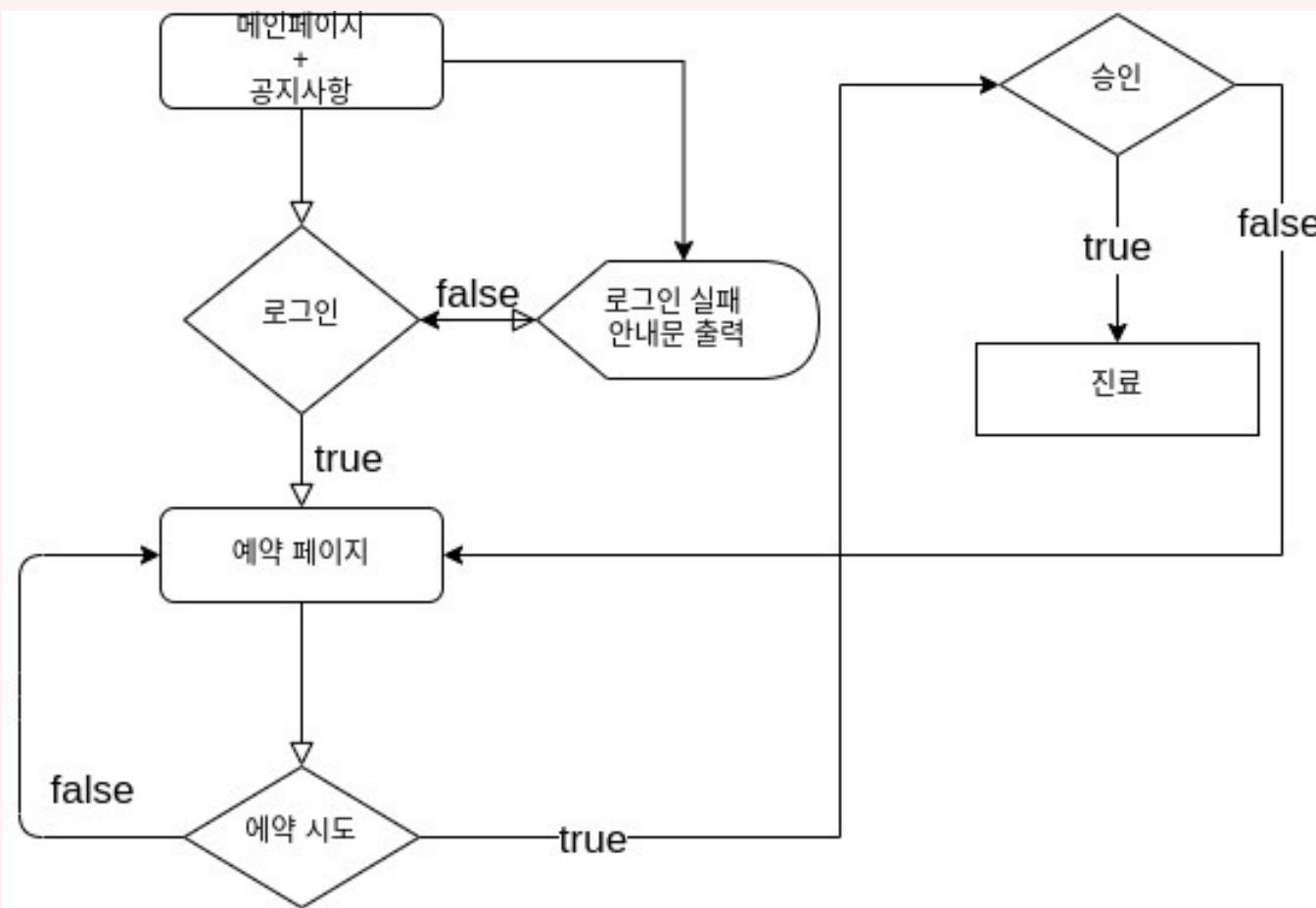


개발 과정

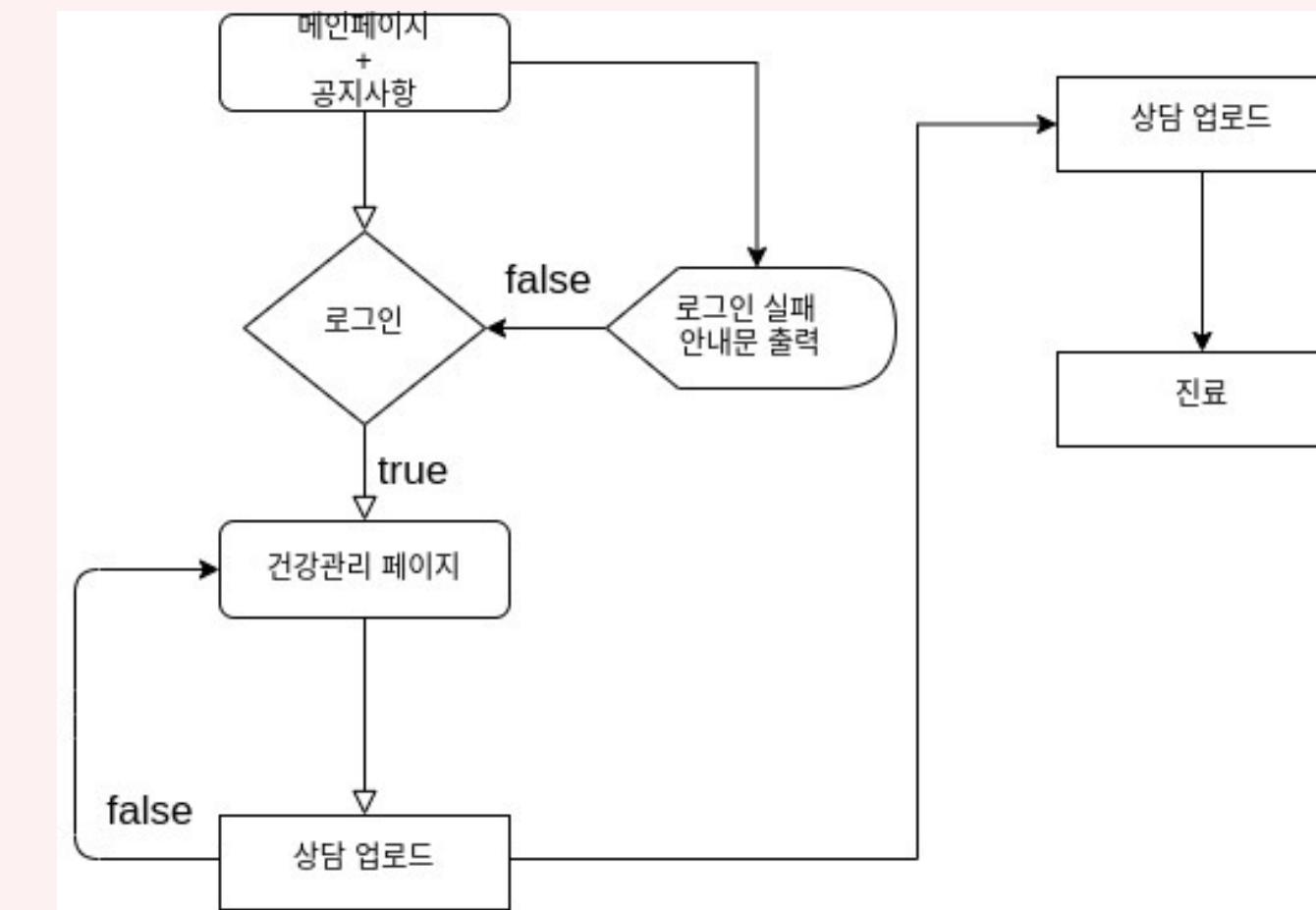
4. 프론트엔드: 플로우 설계

회원가입은 이미 되어 있는 상태라고 가정

예약 플로우

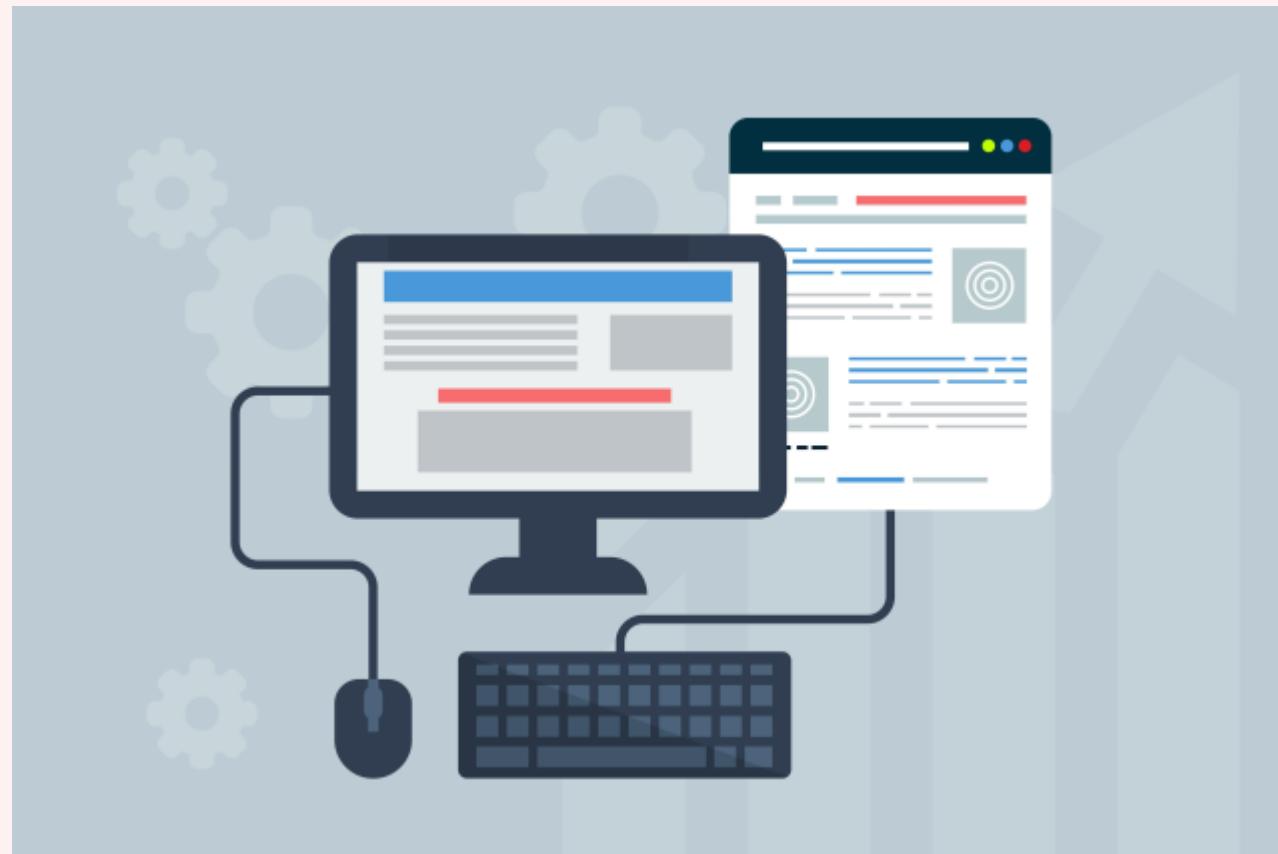


상담 플로우



개발 과정

4. 프론트엔드: 디자인 구성 간 고려요소



반응형 웹 구현

- 사용자의 디스플레이 환경에 따라 디자인이 유동적으로 변하는 반응형 웹 필요

예약 기능 사용성 증가

- 처음 사용하는 유저도 사용하는 데 지장이 없도록 디자인 구성

공지사항 배치

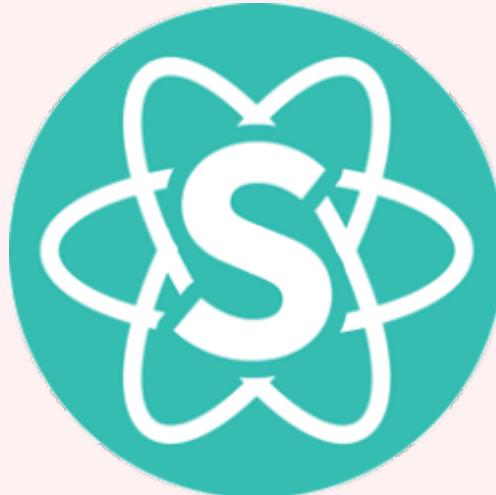
- 병원 이용을 편리하게 할 수 있는 서비스를 제공
- 공지사항은 사용자들이 가장 많이 보는 곳에 배치

건강관리 페이지

- 내 정보와 상담 기능 필요

개발 과정

4. 프론트엔드: 반응형 웹



Semantic UI

The screenshots show a medical appointment booking interface. On the left, the PC version displays four doctor profiles: '찬호김' (정형외과), '이승빈' (신경외과), '이석기' (내과), and '김의사' (피부과). Each profile includes a '예약하기' (Green) and '예약취소' (Red) button. The middle screenshot shows a detailed appointment form for '이석기' (내과), featuring a doctor's illustration, a text area for symptoms, and a grid of time slots from 09:00 to 16:00. The right screenshot shows the mobile version of the same interface, which is vertically oriented and scaled down.

PC 화면

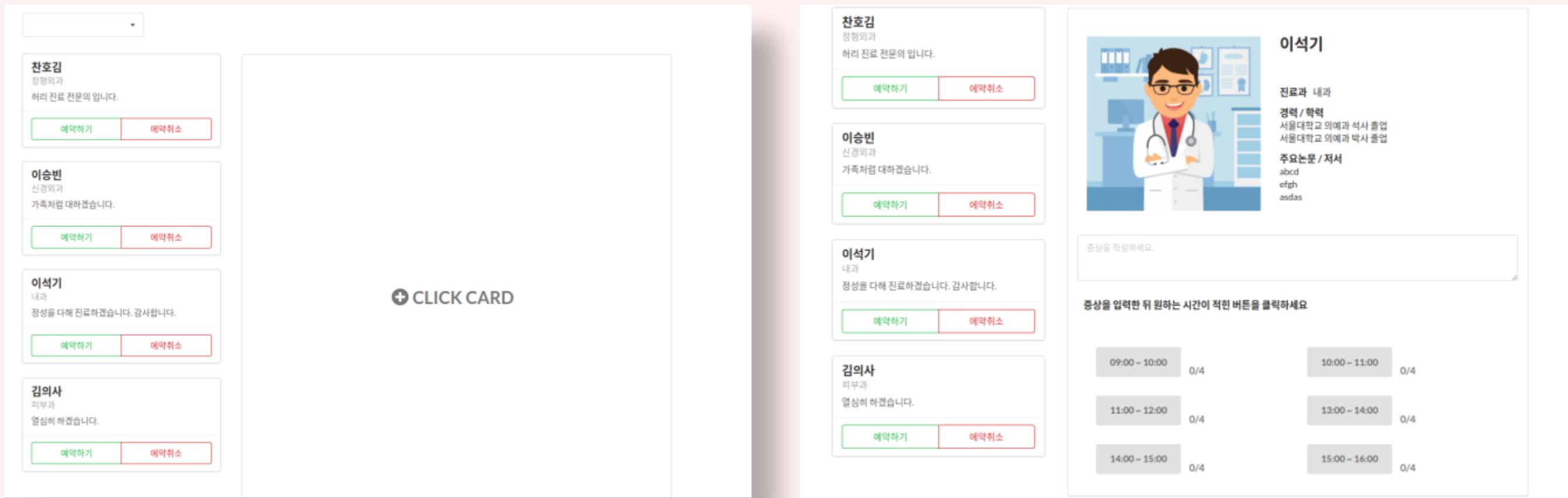
모바일 화면

Semantic UI

- 특별한 설정없이 반응형 웹 제작 가능 (모바일 환경에서도 사용 가능)
- Grid Layout을 사용하여 영역(div)들이 차지하는 공간을 디스플레이에 따라 유동적으로 배치
- 단위는 “em”으로 통일

개발 과정

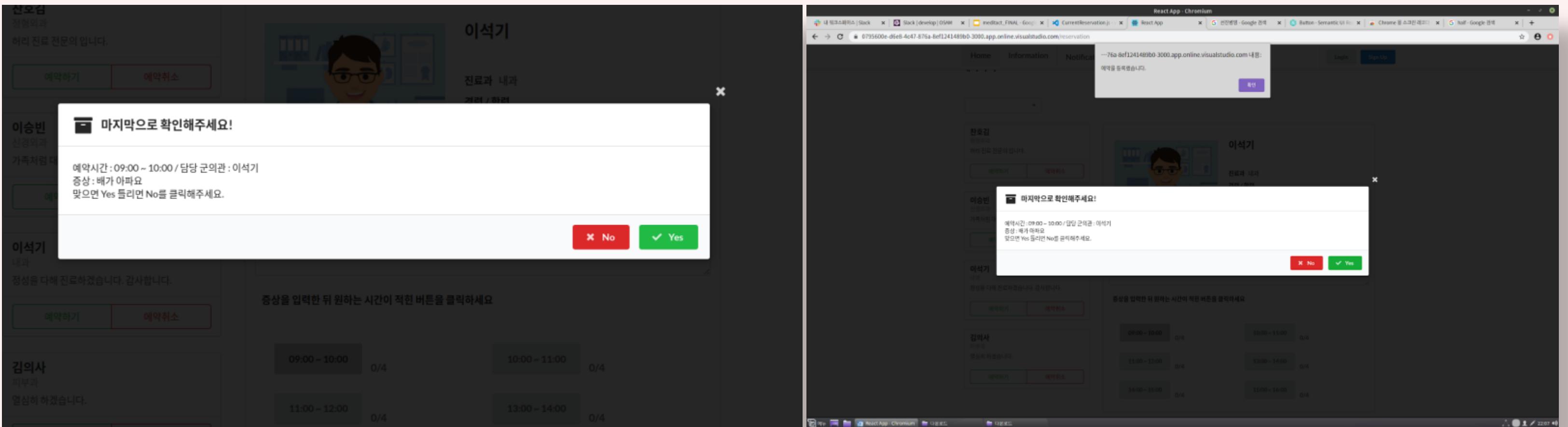
4. 프론트엔드: 예약기능 사용성 증가(1)



- 페이지 이동 없이 군의관 정보를 한눈에 볼 수 있는 **카드형 디자인 구성**
- “예약하기” 클릭 시 “Click card” 영역에 군의관 상세 정보, 증상입력 칸, 예약 가능시간 노출
- 증상 입력부터 예약버튼, 최종확인으로 이어지는 **원스톱 예약 시스템**

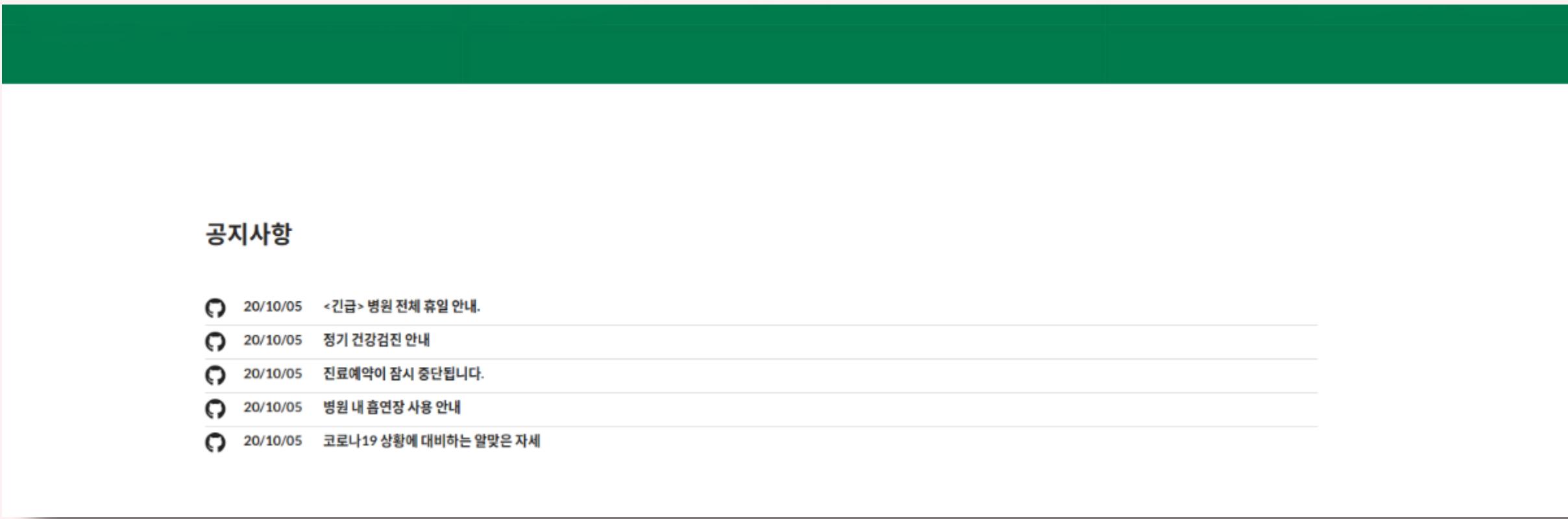
개발 과정

4. 프론트엔드: 예약기능 사용성 증가(2)



개발 과정

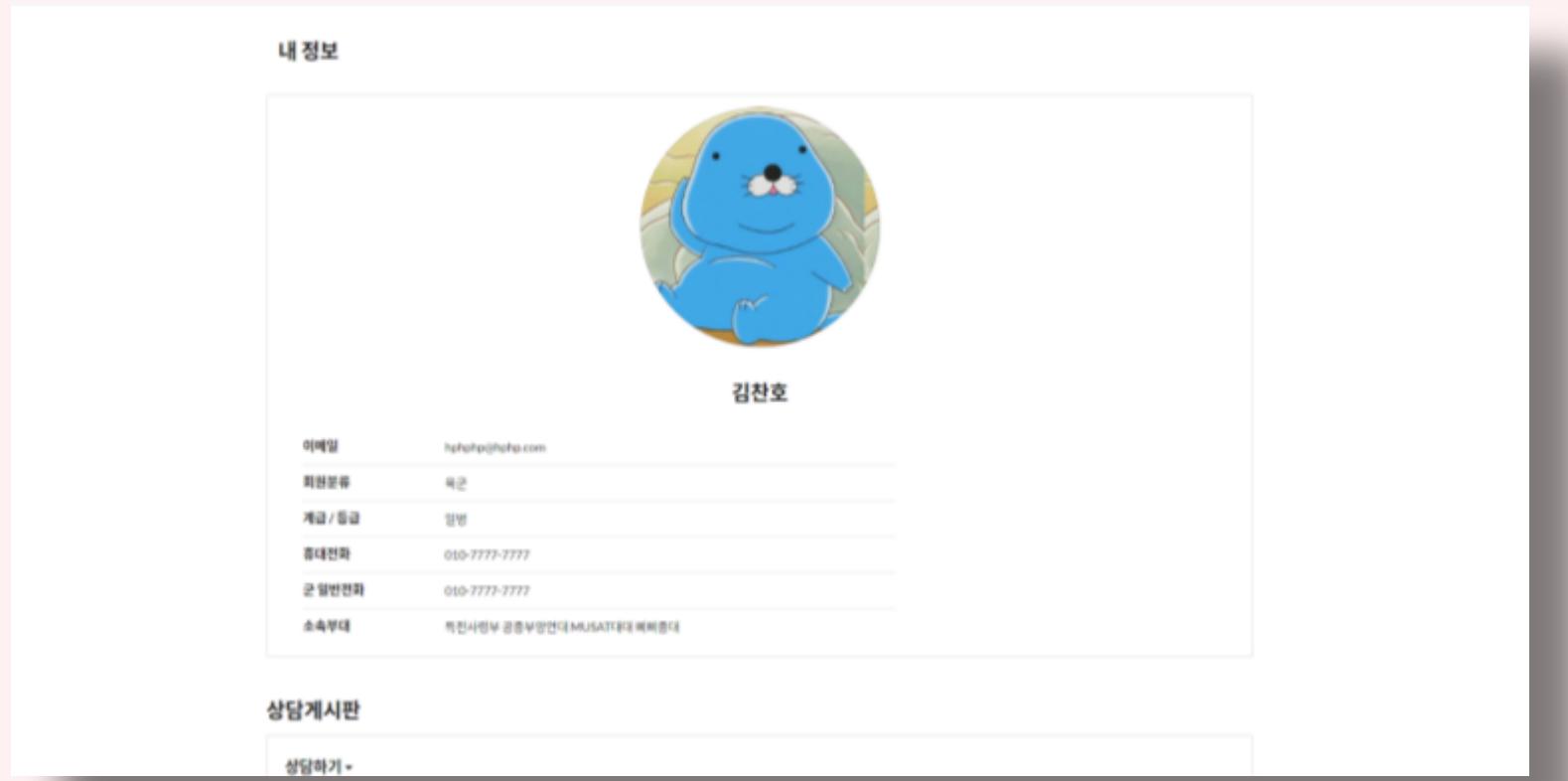
4. 프론트엔드: 공지사항 배치



- 사용자들이 가장 많이 사용하는 페이지는 메인 페이지
- **공지사항을 메인페이지 아랫부분에 배치**

개발 과정

4. 프론트엔드: 건강관리 페이지 구성(1)



- 유저의 정보를 노출함과 동시에 핵심기능 중 하나인 **상담 기능**이 적용

개발 과정

4. 프론트엔드: 건강관리 페이지 구성(2)

The diagram illustrates the development of a health management page through two versions of a user interface.

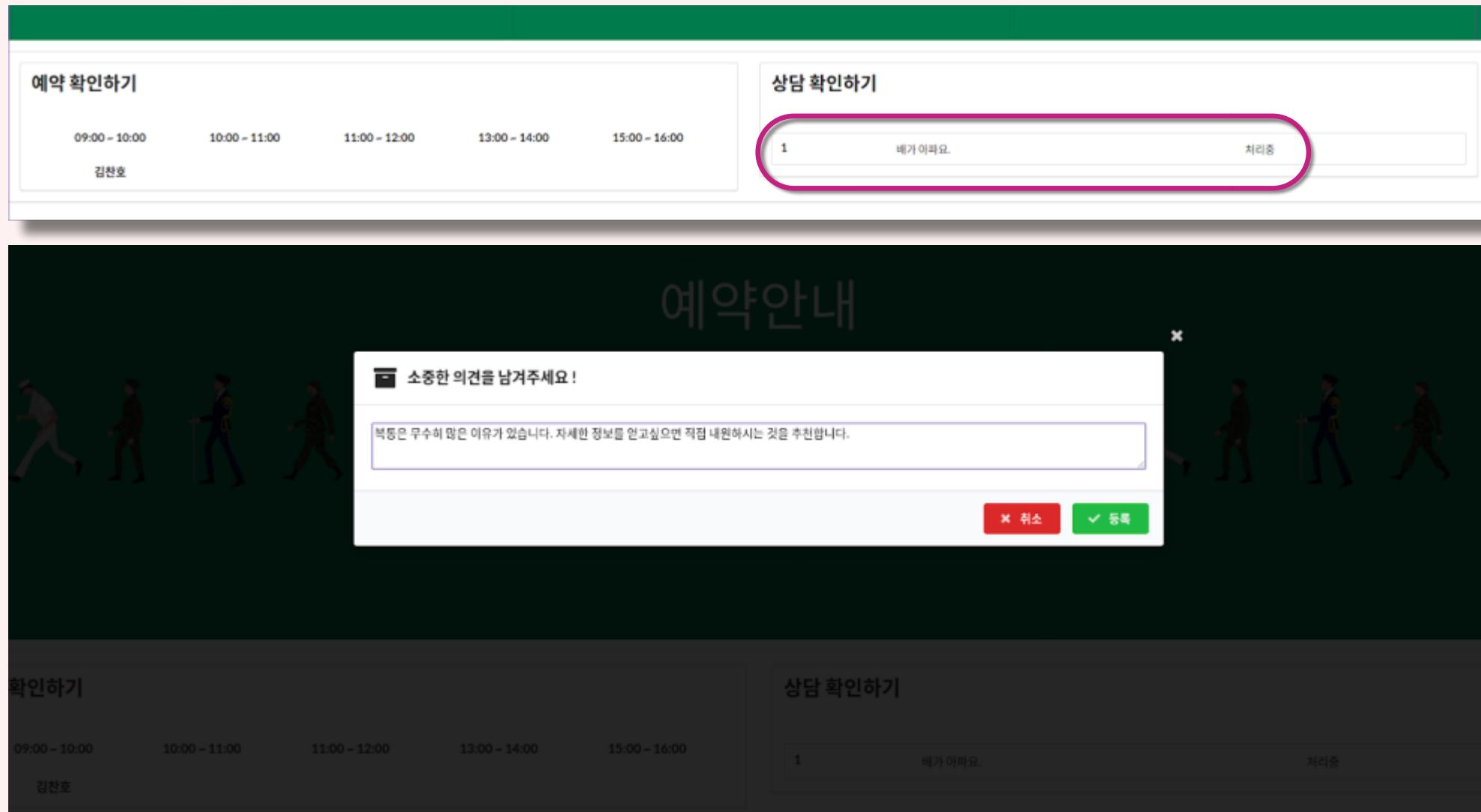
Initial Version: On the left, a screenshot shows a simple form titled "상담하기~". It includes a header with "담당 군의관: 이석기" and a button bar with "찬호김", "이승빈", "이석기" (highlighted in green), and "김의사". Below this is a text area containing "배가 아파요." and "배가 너무 아픕니다.". At the bottom is a file selection field "파일 선택 선택된 파일 없음" and a green "상담하기" button.

Final Version: An arrow points to the right, leading to a more advanced version of the same page. This version has a larger text area for "상담글을 작성하세요." and a file selection field "파일 선택 선택된 파일 없음". A green "상담하기" button is also present. Below the form, a summary table shows a single entry: "1 배가 아파요 처리중".

- 상담기능을 사용자 친화적으로 개발하는 것이 목표

개발 과정

4. 프론트엔드: 건강관리 페이지 구성(3)



- 군의관의 스케줄 관리 및 상담 답변
- 한 페이지 내에서 관리할 수 있게 구현

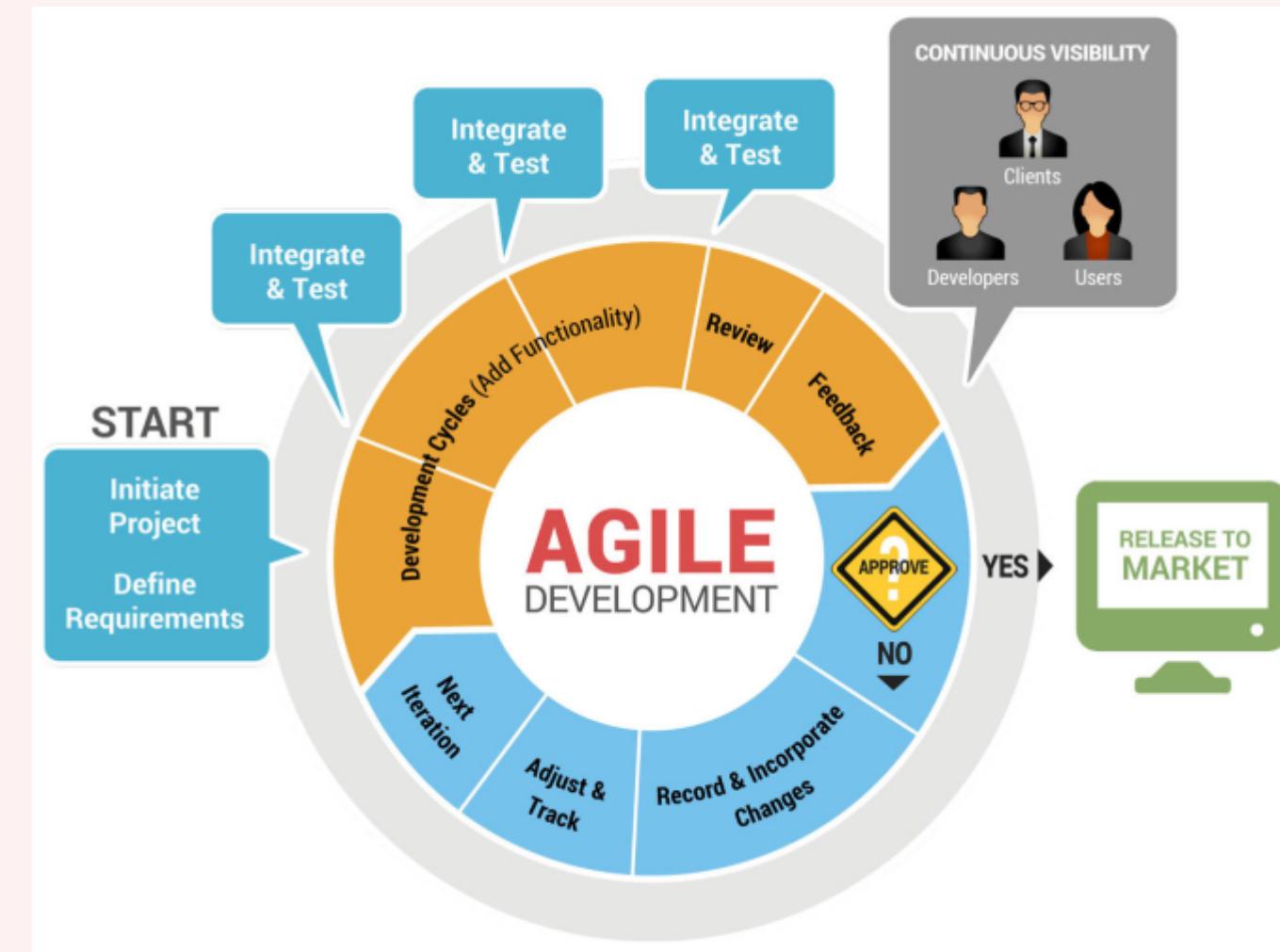
05

Agile 개발

Agile 개발 프로세스
개발 계획 수립
개발 진행
자가 평가

Agile 개발

1. Agile 개발 프로세스



1. 개발 계획 수립
2. 개발 진행
3. 피드백 (자가 평가)

Agile 개발

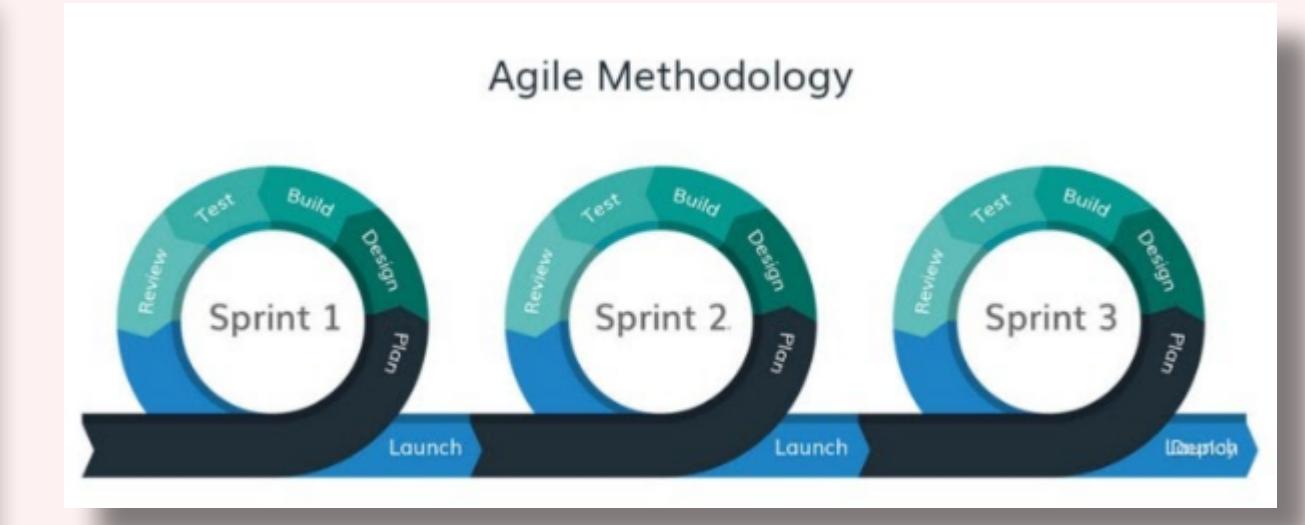
2. 개발 계획 수립

김성일 11:41 PM
! 9월 24일 공지입니다!
아래 것들을 진행하셔서 편한 시간대에 슬랙에 올려주시면 됩니다

1. 팀명 생각하기
2. 개발 파트에서 진행 계획 생각하기
3. 홈페이지를 만들 시에 필요 기능

Hyeonhoon Lee 10:38 PM
1. 제가 올린 기획안에 대해 먼저 얘기해보고,
2. 각 파트별 진행정도를 확인한 후
3. 병원안내, 서류발급안내 등 사용자 인풋이 필요없는 페이지는 수도
병원페이지로 연결시켜줄지 아니면 저희 페이지에서 새로 만들지: 우선, 수도병원페이지 연결 정도만 생각하고 나중에 개발할수 있는 시간
이 남으면 그 때 정하는걸로.
4. 각 기능별 세부 항목은 무엇으로 할지 정하기: "메딕봇_구체화_수정.pptx" 에 있는 내용 외에는 세부 항목 추가 없음.

4 5+



- 슬랙을 이용하여 팀원 전원이 매주 1시간가량 프로젝트 목표 및 개발 계획 회의
- 'AI 모델 자동화', '챗봇 인프라 구축', '리액트 예약 시스템' 등으로 구글링 및 검토
- 첫번째 Sprint 단계에서는 '실행 가능한 시연'을 우선순위로 두고 설계

Agile 개발

3. 개발 진행: 멘토링



Part 4 질문사항

- <https://github.com/lcalialabs/alpha> 를 이용, 채팅플랫폼의 UI 틀을 잡을려고 합니다. 이때 라이센스 관련 특별히 주의해야 할 사항이 있습니까?
- 사실 지금까지 git을 통한 협업 기반의 프로젝트 진행이 잘 되지는 않고 있습니다. 하지만 여러 오픈소스들을 보면, 초반에 많은 양의 코드를 PUSH를 해 놓고 후반에 천천히 유지보수하는 경우가 있습니다. 커다란 코드베이스들을 통합할때 특별히 유용한 노하우가 따로 있을까요?
- 이번 해커톤에서 진행되는 기본 플랫폼 스택으로 Microsoft 의 제품이 많습니다. 하지만 이미 Google Cloud Platform을 처음부터 고려하고 해커톤을 진행해서, Microsoft의 제품을 사용하는데 학습비용이 있습니다. 혹시 이와 관련해서 불이익이 있을까요?

20 Meditact 게시물 파일

lanyndhei (캐스트) 이제 오후 9:19
몇 가지 궁금한 게 있습니다:

1. Continuous Deployment은 어떤 것으로 구성할까요? (Jenkins, Azure DevOps, GitHub Actions, Travis, Netlify, ...?)
2. 우리의 GITHUB* 가 osamhack2020 에 있는 저장소를 말씀하시는 건가요?
3. 지금 infra와 web이 위에 링크 공유한 저장소에 같이 있는 건가요?

더 보기

yoo Jonghyeon (캐스트) 이제 오후 9:50

Google Cloud Platform - Node.js

구글 Cloud Build 라는 툴을 사용했습니다.

우리의 GITHUB은 osamhack2020에 있는 저장소를 말하는 것 같습니다.

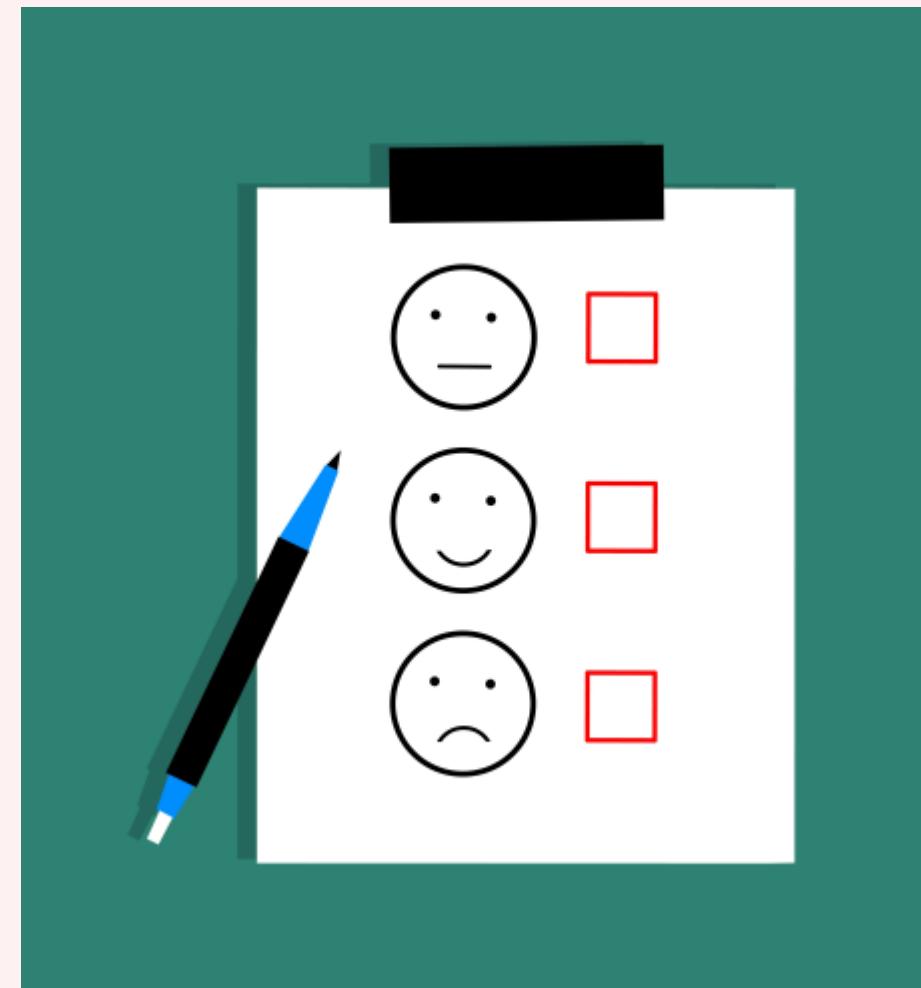
현재 osamhack2020에서 인프라쪽 코드는 AI model을 구동시키는 코드입니다. Web코드는 채팅/웹과는 거의 loosely coupled 한 별도 게시판 사이트에 가깝습니다.

체임 코드, 시코드, 웹코드를 이렇게 결합성이 낮게 분리되어 개발이 진행된 것 같은데, 해커톤 특성상 반드시 chat 앱을 미단으로 push

- 매주 멘토링을 위해 현재 개발 상황을 ppt로 상세하게 보고
- 정확한 의미전달을 위해 질문사항을 면밀히 검토하여 작성
- Microsoft Teams를 활용해 멘토님과 소통하고, 해결 방안을 적극적으로 개발에 적용

Agile 개발

4. 자가 평가



- Github을 통해 현재까지의 산출물과 개발 과정에 대해 스스로 평가
- 다음 Sprint를 위한 목표 수립

Agile 개발

4. 자가 평가 – 산출물 평가

| Deep Learning | Infra |
|---|---|
| 약 75%의 좋은 성능을 보이는 진료과 분류 모델 개발 데이터셋과 딥러닝 모델 공개 | 실행 가능한 live-demo 완성 동작하는 AI 모델 자체를 챗봇 사용자가 사용 가능 쿠버네티스, 도커, Git 자동화를 성공적으로 구축 |
| Web-Backend | Web-Frontend |
| MVC 디자인 패턴을 통한 개발 유연성 증가 Git 관리 및 배포를 위한 설계를 통해 빠른 배포 가능 | 각 페이지의 깔끔하고 효율적인 기능 배치 |

Agile 개발

4. 자가 평가 – 개발 한계

| Deep Learning | Infra |
|--|--|
| <p>증상–진료과 분류 관련 데이터셋이 없어서 직접 만들어내는데 많은 시간 소요</p> <p>모델 성능 향상을 위해 더 많은 데이터가 필요했지만 시간적인 한계</p> <p>챗봇에 탑재된 모델보다 더 좋은 모델을 만들었지만 환경 상 포기</p> <p>명사만으로 문맥을 파악하기에는 최대 정확도에 한계</p> <p>모듈화를 계획했으나 시간상의 이유로 중단</p> | <p>코드 자체의 퀄리티에 집중하지 못하고 구현에만 집중</p> <p>처음부터 Git push 자동화를 적용하지 못하고 학습에 치중하다가 시간 지체</p> <p>군대 사지방의 경우 로컬에서 구축할 수 있는 오픈소스가 제한적</p> |
| Web-Backend | Web-Frontend |
| <p>API를 구성하면서, 프로젝트 시작 단계에서 제대로 된 DB설계 부족</p> <p>일부 API의 RESTful 설계 미흡</p> | <p>시간 배분을 잘못한 결과 일부 페이지의 디자인이 구상과 다르게 구현</p> |

Agile 개발

4. 자가 평가 – 개발 과정을 통해 학습한 내용

| Deep Learning | Infra |
|---|--|
| <p>딥러닝 모델 자체의 개발보다는 데이터셋 구축에 훨씬 많은 인력이 필요 최신 모델이라고 해서 반드시 좋은 성능을 나타내지 않음 데이터셋에 맞는 자연어 처리 모델 설계가 중요 데이터 정리 및 전처리가 모델 성능 향상의 핵심 포인트</p> | <p>우수한 React–Nodejs 기반 오픈소스를 통한 빠른 학습 Git의 add commit push 뿐만 아니라 다른 명령어들의 유용성</p> |
| Web-Backend | Web-Frontend |
| <p>RESTful API 설계 방법 및 구조화 협업을 위한 API docs 작성 통신을 위한 response 설계</p> | <p>프론트 라우팅 기법 문서화를 통한 프로젝트 소개 방법</p> |

Agile 개발

4. 자가 평가 – 다음 Sprint 계획

| Deep Learning | Infra |
|--|---|
| <p>네이버 지식인 API 등을 활용한 추가적인 데이터셋 구축 방안 마련 국군수도병원 군의관들의 데이터셋 라벨링 검토 독려 군의관들이 실무 중 신규 데이터를 쉽게 데이터셋에 추가할 수 있는 방법 모색 치과 전문분야(보존과, 치주과, 교정과 등)도 분류 가능하도록 프로젝트 확장 진료과 예측 모델을 패키지화 후 모듈 배포</p> | <p>실무에서 거대한 크기의 AI모델을 개발할 시 이를 관리하기 위한 방안 모색 AI 기반 챗봇의 성능 이슈 측정 및 해결 방안 검토 테스트 서버와 배포 서버를 구분해서 개발 환경과 서비스 환경을 분리 모듈화가 덜된 코드를 기능단위로 분해해서 테스트 코드 적용</p> |
| Web-Backend | Web-Frontend |
| <p>Controller의 코드 최적화 방안 마련 DB 재설계 및 그에 따른 코드 수정</p> | <p>하드코딩된 부분을 간결화 대용량의 리스트 처리를 고려하여 디자인 개발 리모트와 배포 리모트를 나누어 작업</p> |

06

Meditact의 미래

라이센스
공개 SW
챗봇 상용화
데이터셋 공개

Meditact의 미래

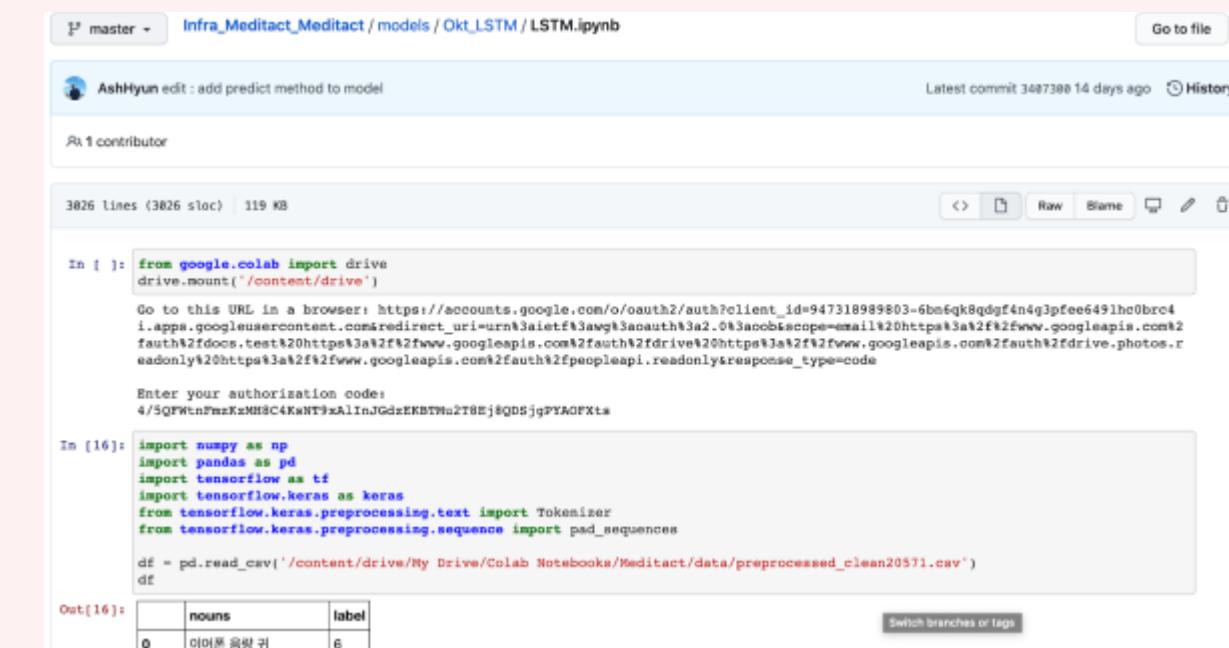
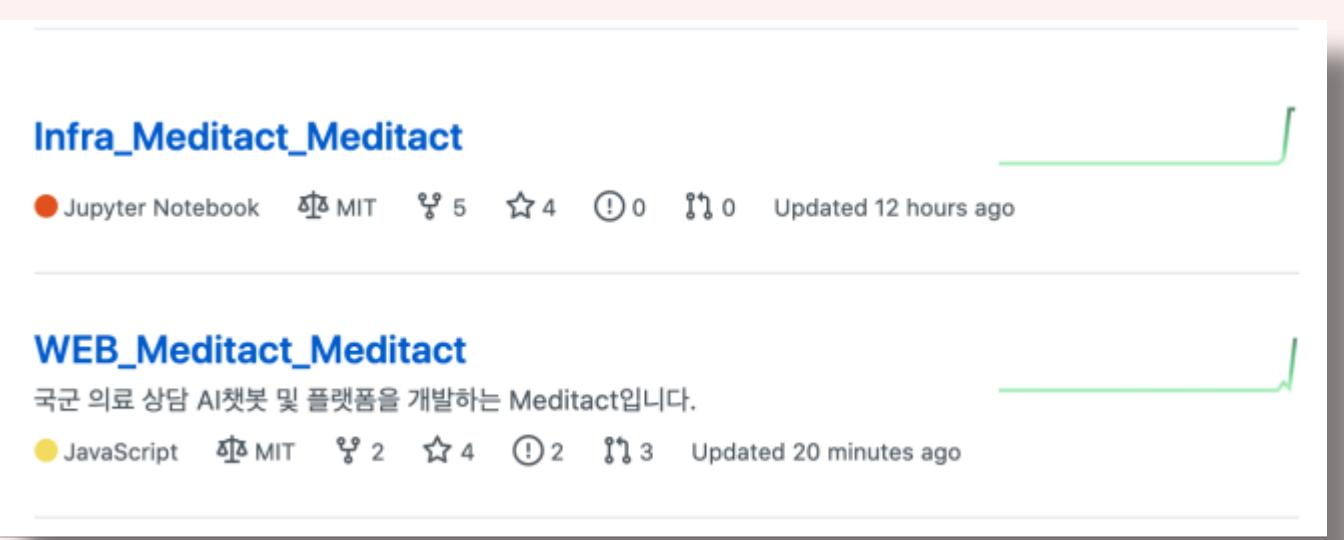
1. 라이센스



- MIT 라이센스는 가장 널리 쓰이는 오픈소스 라이센스 중 하나
- 재배포시 오픈소스로 배포해야 할 의무가 없어 2차 사용자의 자유로운 사용이 가능
- 국방 오픈소스 아카데미 해커톤의 취지에 가장 잘 부합
- 라이센스 문서는 github repository에 업로드

Meditact의 미래

1. 공개 SW



The image shows the GitHub organization 'Meditact' with two main repositories:

- Infra_Meditact_Meditact**: A Jupyter Notebook repository with 5 stars, 4 forks, and 0 issues. It was updated 12 hours ago.
- WEB_Meditact_Meditact**: A JavaScript repository with 2 stars, 4 forks, and 2 issues. It was updated 20 minutes ago.

The right side of the image shows a Jupyter Notebook titled 'LSTM.ipynb' in the 'master' branch. The notebook has 3826 lines of code (3826 sloc) and is 119 KB in size. The code imports 'google.colab' and 'drive' from 'content/drive'. It also imports numpy, pandas, tensorflow, and tensorflow.keras. A CSV file is read from 'content/drive/My Drive/Colab Notebooks/Meditact/data/preprocessed_clean20571.csv'. The output shows a single row with '이어폰' as the noun and '6' as the label.

- Github을 Web과 Infra로 나누어 사용
- 개발자들의 분야에 따라 원하는 소스 코드에 쉽게 접근 가능

- 본 프로젝트에서 개발한 다양한 자연어 처리 모델을 누구나 쉽게 따라할 수 있도록 ipynb 형식으로 제공
- ipynb 파일에 목차와 주석을 자세히 작성

Meditact의 미래

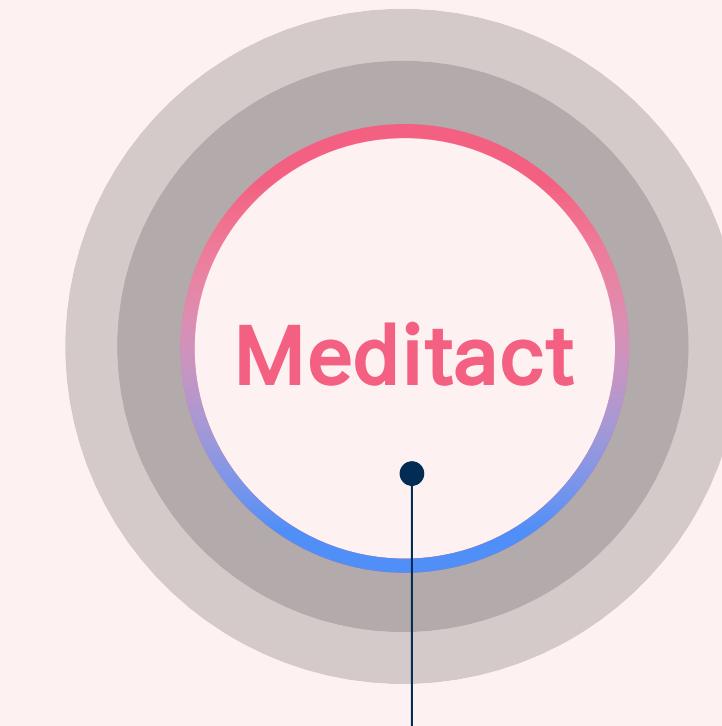
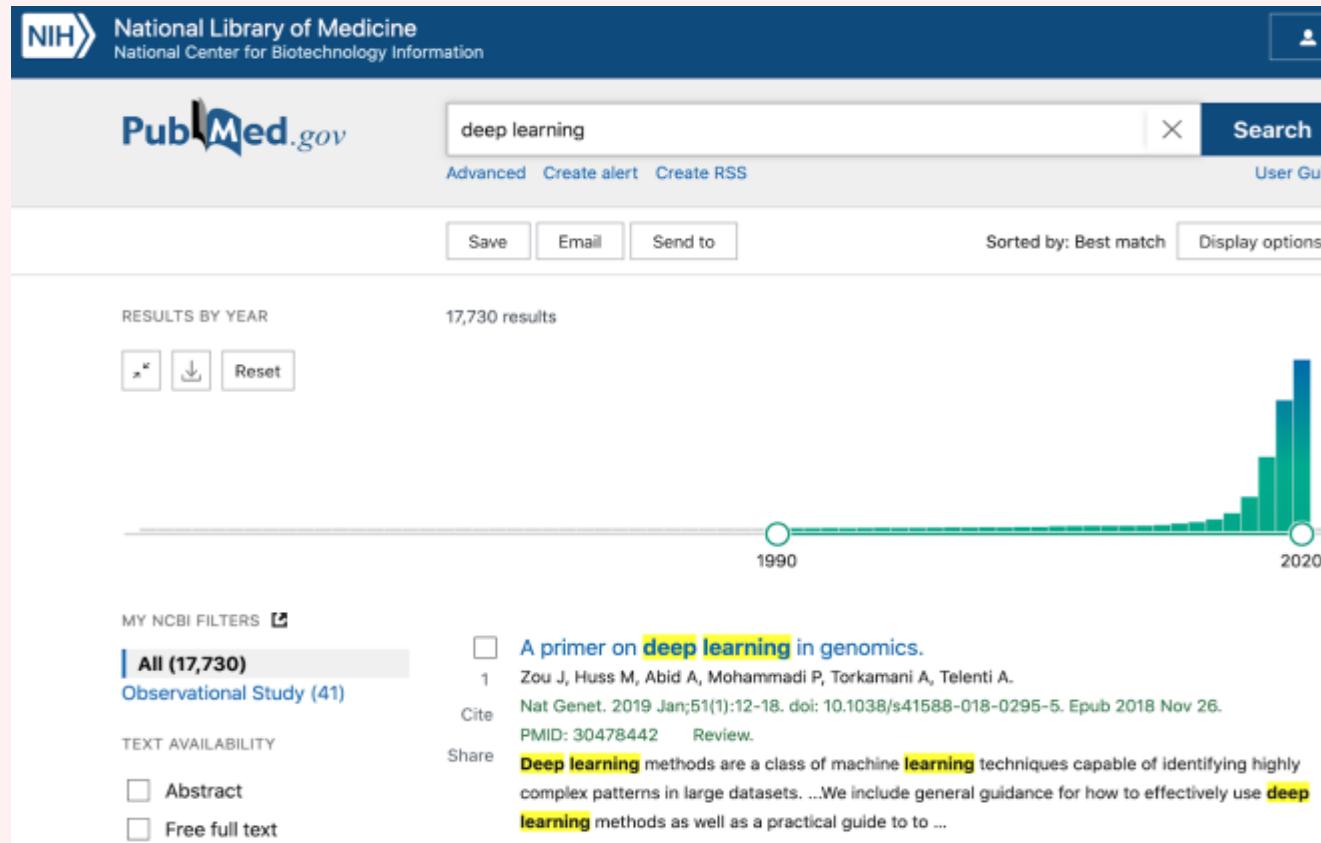
2. 챗봇 활용 방안: 군병원 및 민간병원 상용화



- 현재 만들어진 데모의 안정성 확인 후 의무사령부 등 유관기관과 논의 예정
- 국내 종합병원으로 확장: 본 챗봇 서비스에 대해 이미 구연 발표 완료 (이현훈 군의관)
- 해당 병원의 환경(진료과, 홈페이지와의 호환성 등)에 맞게 수정/보완 후 추진 예정
- 사용자 피드백 수립: 서비스 상용화 후 만족도 평가를 통한 피드백 반영

Meditact의 미래

2. 챗봇 활용 방안: 해외 우수 저널에 논문 출판

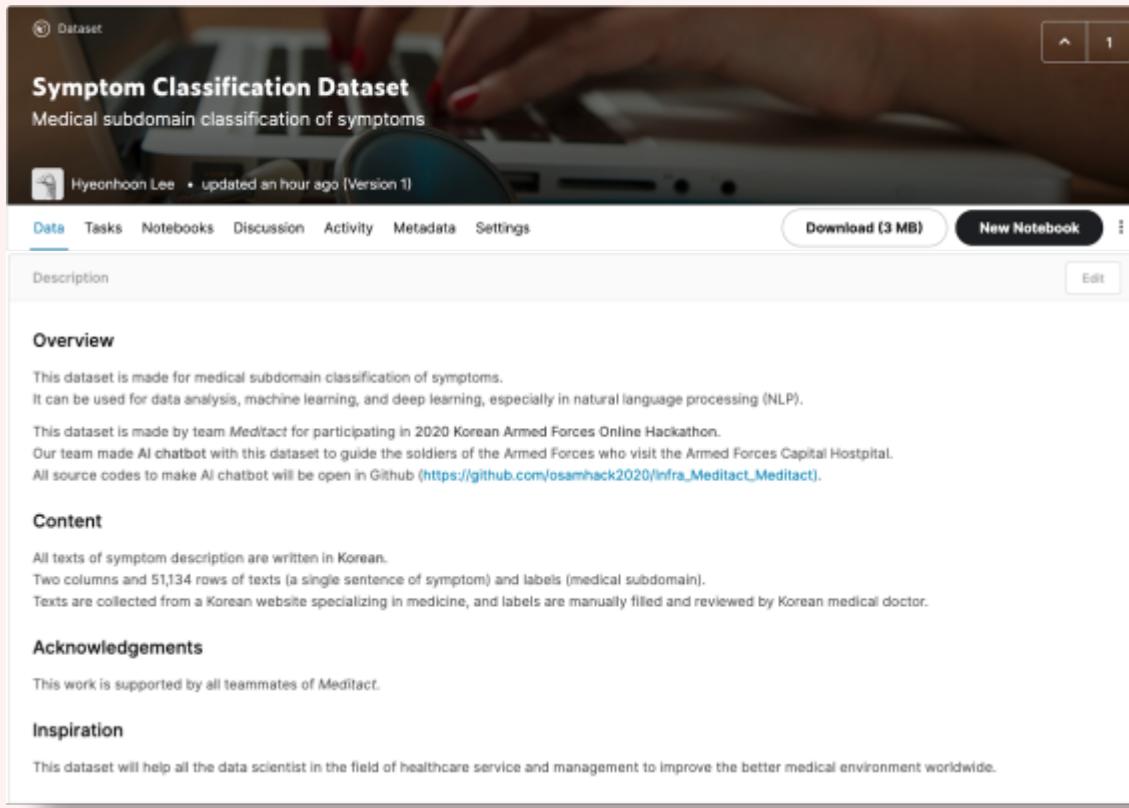


글로벌 비대면 의료 서비스

- 최근 딥러닝을 의료 분야에 적용하는 사례 관련 논문이 **기하급수적으로 증가** (현재 의학 논문 DB에서 1만 7천건 이상)
- SCI(E)급 해외 저널에 5편 이상의 논문을 출판한 경험을 살려, 본 프로젝트도 **논문으로 작성하여 해외저널에 발표** 예정(2021년 6월경)
- 데이터셋만 한국어가 아닌 타 언어로 바꾸면 충분히 응용 가능 : **글로벌 서비스**로서의 가능성

Meditact의 미래

3. 데이터셋 공개 – Kaggle에 배포



- Kaggle은 개발자들 사이에서 널리 알려진 구글의 빅데이터 플랫폼
- 데이터 관리, 코드 공유, 토론 게시판 운영 등 데이터셋을 공유할 수 있는 최적의 환경 제공
- 기업 또는 단체에서 제공하는 데이터를 활용한 다양한 경진대회 개최
- 현재 데이터셋을 누구나 활용할 수 있도록 Github와 Kaggle에 업로드한 상태
- 앞으로 다양한 source를 이용한 데이터셋 확장 및 quality control 시행 예정

Meditact의 미래

3. 데이터셋 공개 – 추가적인 수집 방안



- 환자와 소통하는 의료 데이터가 더 높은 품질의 데이터로 평가되는 최신 의료 트렌드 반영
- 진료과에 잘못 찾아오는 환자들이 있을 경우, 전문의 군의관들이 '증상'과 '올바른 진료과' 정보를 데이터셋에 추가
- 환자들이 **공통적으로 판단하기 어려워했던 증상들**에 대한 데이터를 점차 누적 가능
- 치과 군의관이 추가로 참여하면 전문분야(보존과, 치주과, 교정과, 구강외과 등)에 대한 데이터셋 세분화 가능

07

해커톤을 마치며

팀 프로젝트
국방 개발 환경
오픈 소스 생태계

해커톤을 마치며

1. 팀 프로젝트



- 처음 진행해본 형태의 프로젝트이기에 각 분야에 대한 요구사항이 계속 수정
- 프로젝트에 요구되는 선행지식이 많아 학습과 개발을 병행하며 진행
- 팀원들의 분야별 가용 시간 예측과 러닝 커브 고려에 한계
- 하지만, 팀이 아니라 개개인이었다면 절대 프로젝트를 성공적으로 마무리하지 못했을 것

해커톤을 마치며

2. 국방 개발 환경: 한계



- Docker를 설치하려면 재부팅이 필수지만, 사지방 특성상 재부팅이 불가능
- 사지방에서는 Docker-compose up 명령어를 통해 계속 로컬호스트에서 build된 앱을 실행하며 개발하는것이 불가능
- Online VS CODE은 확장프로그램이 잘 적용되지 않고, 서버를 열 때 브라우저에서 열리지 않고 서버를 한 번 거치기 때문에 속도 저하

해커톤을 마치며

2. 국방 개발 환경: 극복



- Git으로 자동 CD 파이프라인을 구축하여 테스트서버로 계속 코드를 전송하며 개발
- 가능하면 프로젝트를 웹에서 작업하여 PC에 문제가 생겨도 작업한 내용이 없어지지 않게 노력

해커톤을 마치며

3. Open your code, Open our future

2020 온라인 군장병 해커톤은

제한적인 환경 속에서도 개발 결과물을 창출해내는 하나의 거대한 **실험실**이었습니다.

저희 팀 뿐만 아니라 이번 해커톤에 참여한 **모든 팀**들의 피와 땀이

Github 저장소에 담겨 바깥 세상에 나올 것입니다.

이러한 결실이 **오픈 소스 생태계 활성화**의 좋은 예가 되어,

대한민국이 4차 산업 미래의 선두주자로 한 걸음 더 나아갈 수 있길 바랍니다.



Meditact

Meditact

THANK YOU

국방오픈소스아카데미