해커톤 1조 스터디

챗봇 기능이 포함된 채팅 웹사이트 구현

프로젝트 참가자 목록

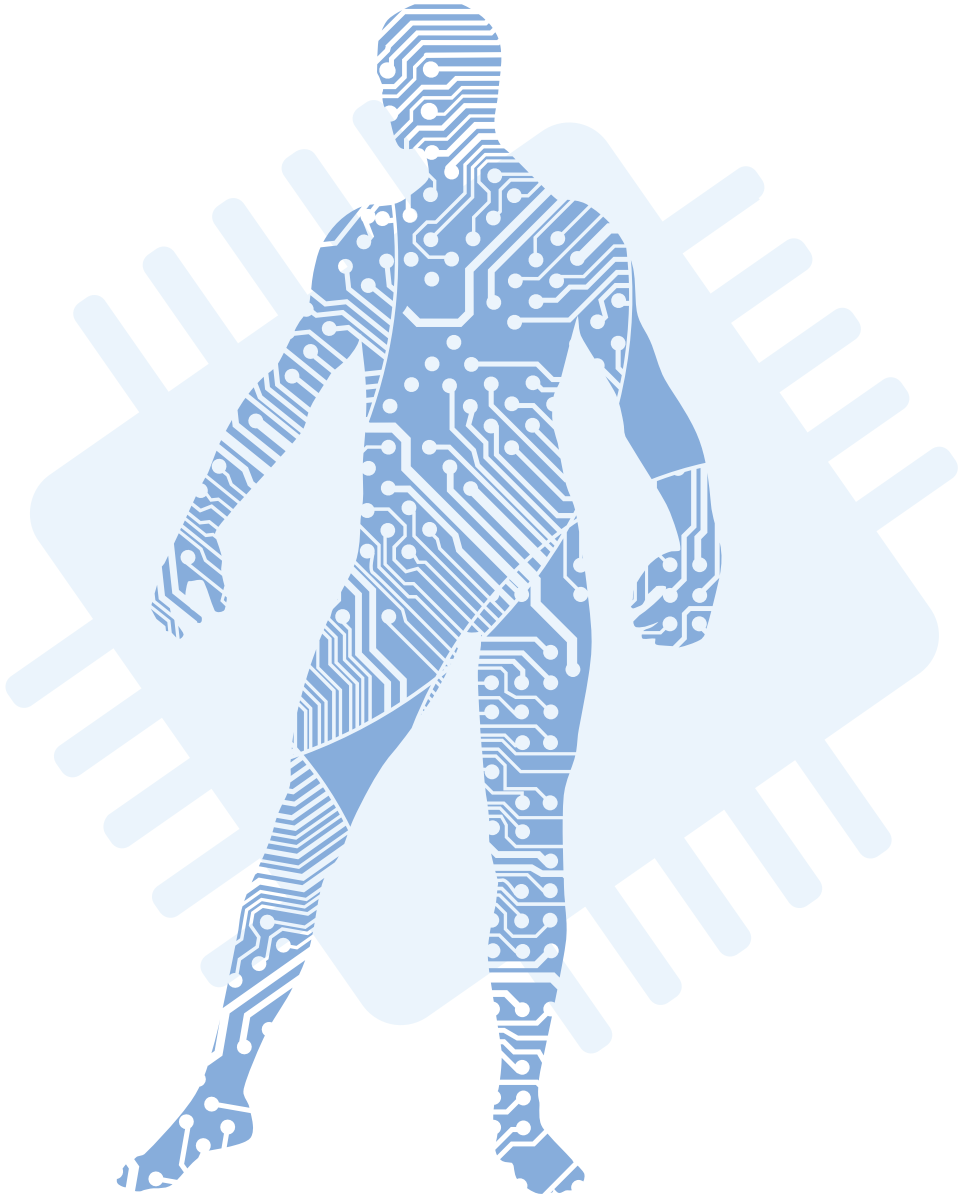
Front-End : 서은영(201910672)
김해인(202010399)

Back-End : 박현호(201810057)
안호빈(202110102)

AI Engineer : 강예성(202010050)

<깃허브 주소> https://github.com/cheomuk/studyNodeJs

프리젠테이션 목차



2021.12.31

프로젝트 첫 회의

자기소개 및 주 전공분야 소개

2022.01.04

프로젝트 주제 선정

서로의 전공을 살려 AI 챗봇 기능을 넣은 채팅 웹사이트 구상

2022.01.30

프론트 & 백엔드 연결

서로 작업했던 내용물을 Socket client를 통해 연결, 채팅 송수신 및 DB 저장 모두 이상 없이 구현됐다.

2022.02.08

백엔드 & 딥러닝 연결

학습된 AI를 백엔드에 연결해 프론트 단에서 정상 작동하도록 구현했다.

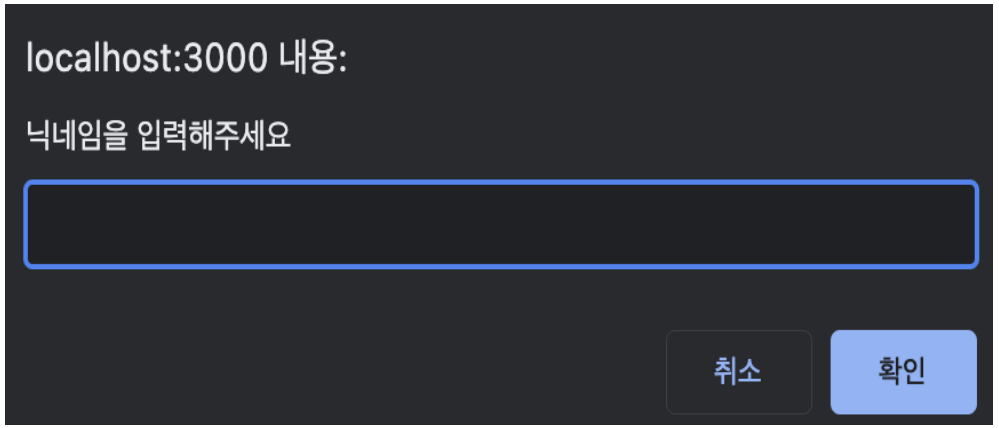
2022.02.11

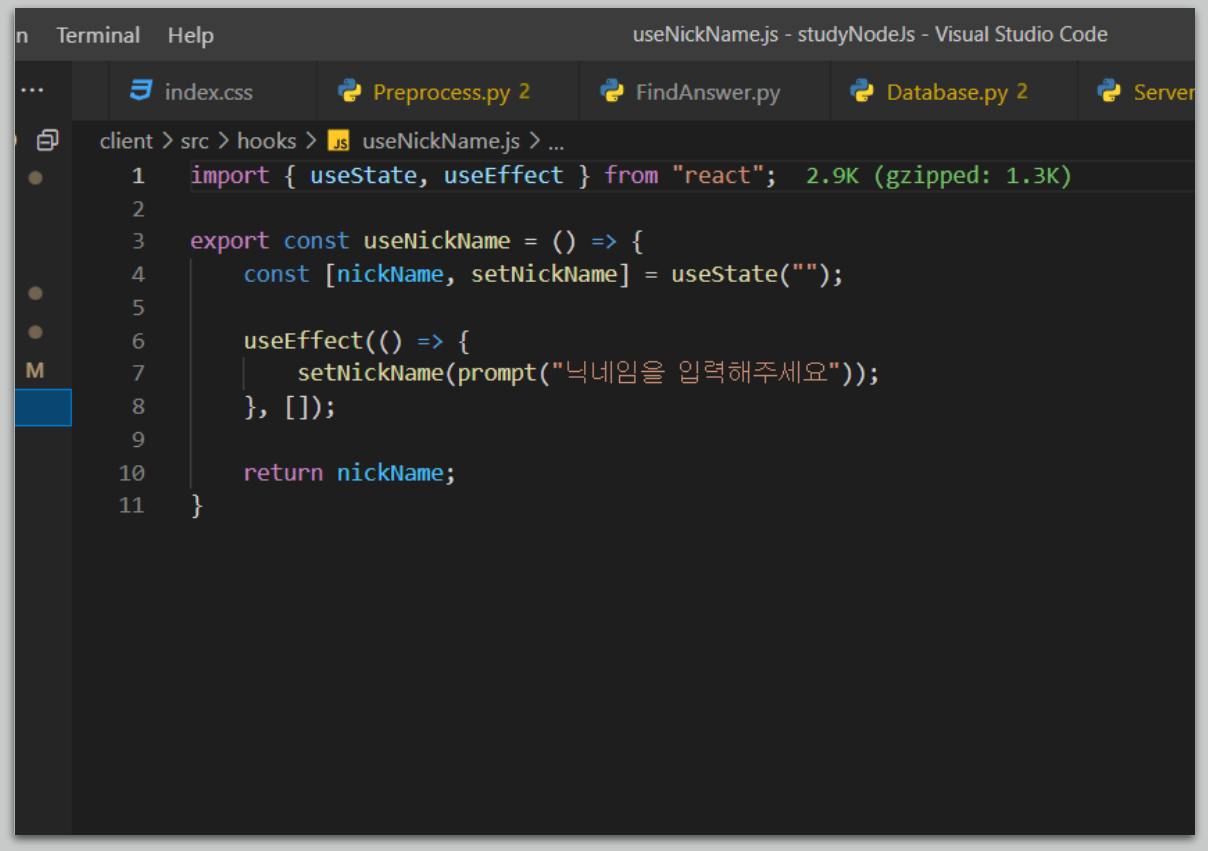
프로젝트 완성 및 발표

useNickName.js
기능 설명

- 아래 프롬프트 창을 처음 접속할 때 띄워 닉네임 값을 받고 그 값을 저장합니다.

- 이 코드는 MySQL 내에서 메시지를 탐색 및 정렬하거나 프론트에서 글 쓴 유저의 닉네임을 띄우는 등 중요한 역할을 합니다.





```jsx
const TopBar = () => {
  const askExit = () =>
    Swal.fire({
      icon: "warning",
      html: "정말로 대화를 종료하시겠습니까?",
      showCancelButton: true,
      showCloseButton: true,
      focusConfirm: false,
      cancelButtonText: "cancel",
      confirmButtonText: "ok",
    }).then((result) => {
      if (result.value) {
        window.close();
      }
    });
  return (
    <>
      <div className="topBar">
        <AiOutlineLeft className="btnExit" onClick={askExit} />
        <p className="chatbotName">CHATBOT</p>
      </div>
    </>
  );
};
```

TopBar

TopBar & UnderBar 기능 설명

- TopBar 코드는 뒤로 가기 버튼을 눌렀을 때 채팅을 종료할 것인지 물어보는 것과 이미지 파일 추가 기능을 구현했습니다.

```jsx
import React from "react";  7.2K (gzipped: 2.9K)
import { AiOutlineArrowUp } from "react-icons/ai";  2.3K (gzipped: 1.1K)
import "./UnderBar.css";

const UnderBar = (props) => {
  return (
    <>
      {/* 언더바 레이아웃 및 이벤트 등록 */}
      <div className="underBar">
        <form onSubmit={props.onSubmit}>
          <input
            className="chat"
            type="text"
            value={props.message}
            placeholder="채팅을 입력하세요"
            onChange={props.onChange}
            onSubmit={props.onSubmit}
          />
        </form>
        <AiOutlineArrowUp className="btnEnter" onClick={props.onSubmit} />
      </div>
    </>
  );
};

export default UnderBar;
```

Under Bar

- UnderBar 코드는 하단바를 구성하며 채팅 입력칸과 전송 버튼이 구현되어 있다. 편의성을 위해 엔터키와 전송 버튼 둘다 송신 가능하도록 구현했습니다.

Front & Back-End Connect

★ React와 Node.js를 연결하기 위해 Proxy 설정을 통해 웹페이지와 직접적인 연결이 아닌 백엔드를 거쳐가는 간접적인 방법으로 연결하였습니다.

☆ Socket client 파일을 하나 만들어 메시지 전송, 삭제, 이미지 전송 등의 기능들이 작동 가능하게끔 서버로 송신하도록 만들었습니다.

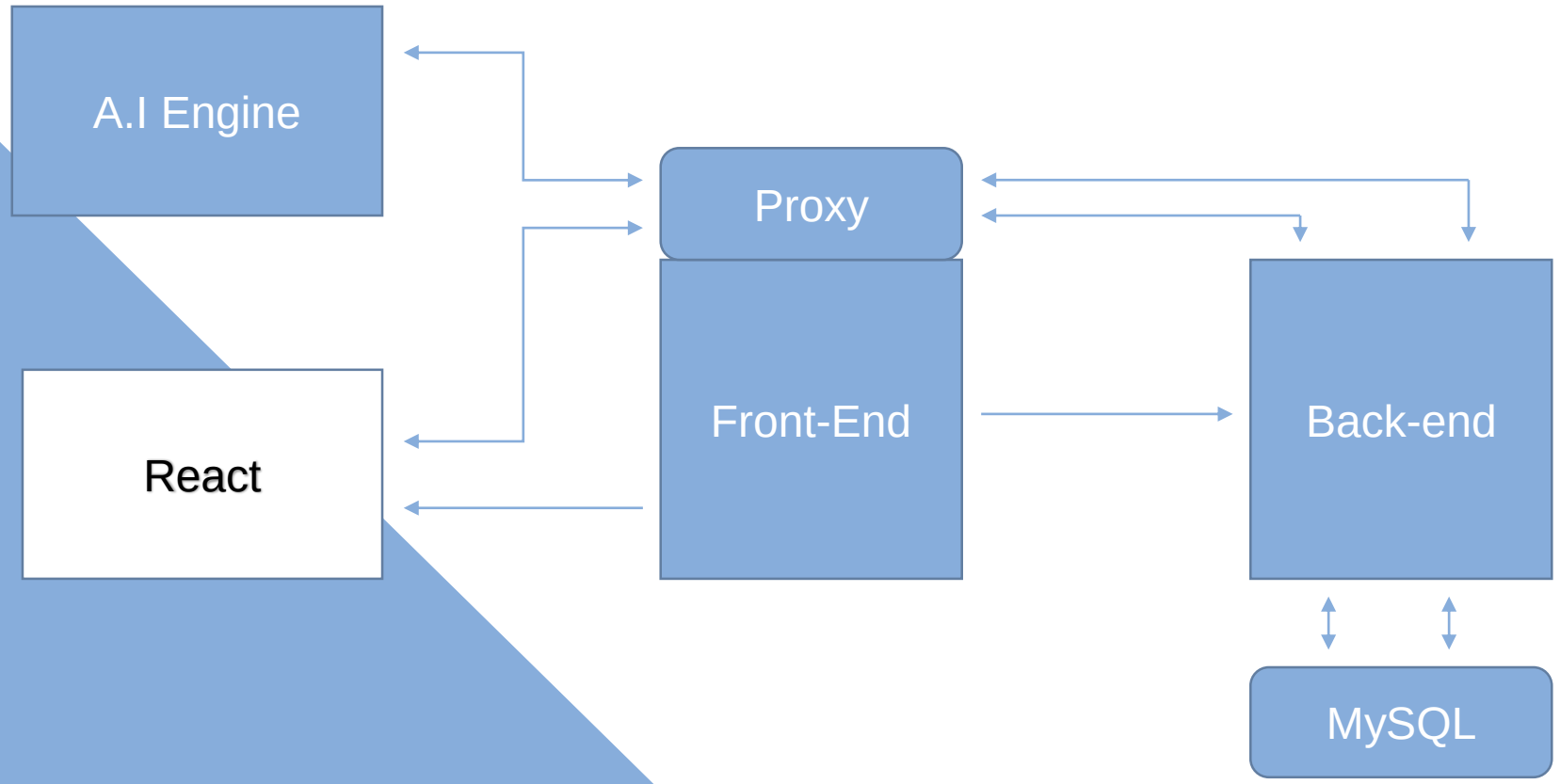
Proxy 설정

```
const { createProxyMiddleware } = require('http-proxy-middleware');

module.exports = function (app) {
  app.use(
    createProxyMiddleware('/socket.io', {
      target: 'http://localhost:4000',
      changeOrigin: true,
      ws: true,
    })
  );
};
```

Socket Client

```
useEffect(() => {
  socket.on("send", (id, sender, message, date) => {
    setChats((value) =>
      value.concat({ id, sender, message, date: getDatetime(date) })
    );
  });

  socket.on("remove", (id) => {
    setChats((value) => value.filter((chat) => chat.id !== id));
  });
}, [socket]);

const send = (sender, message) => {
  socket.emit("send", sender, message);
};

const remove = (id) => {
  socket.emit("remove", id);
};

const sendImage = (sender, url) => {
  setChats((value) =>
    value.concat({
      id: 0,
      sender,
      message: url,
      date: getDatetime(Date.now()),
      isImage: true,
    })
  );
};

return { chats, error, send, remove, sendImage };
};
```

프 로 젝 트 개 형 도



```
http.listen(4000, () => {
  console.log('Connect at 4000');
});

io.on('connection', (socket) => {
  socket.on('disconnect', () => {
    console.log('user disconnected');
  });

  /**
   * 채팅 수신
   */
  socket.on('send', async (sender, message) => {
    const chat = await db.chatlist.create({
      type: 'text',
      sender: sender,
      data: message,
    });

    io.emit('send', chat.id, chat.sender, chat.data, chat.createdAt);
  });

  /**
   * 채팅 삭제
   */
  socket.on('remove', async (id) => {
    await db.chatlist.destroy({
      where: {
        id
      }
    });

    io.emit('remove', id);
  });
});
```

Back-End Web Socket API

- HTTP 통신으로 먼저 4000번 포트로 서버를 연 다음 Socket을 실행하여 프론트와 연결했습니다.

- 채팅을 수신했을 때 DB에 메시지를 저장해야 AI가 학습할 수 있으므로 쿼리를 작성하여 DB에 값을 넣었습니다.

- 채팅 삭제 시 프론트 뿐만 아니라 DB 내의 데이터도 지워져야 하기 때문에 destroy를 통해 해당 id를 가지고 있는 컬럼 내 정보를 삭제하도록 구현했습니다.

db.js

```
const path = require('path');
var Sequelize = require('sequelize');
var sequelize;

sequelize = new Sequelize('chatbot', '', '', {
  host: '',
  port: 3306,
  dialect: 'mysql', // 데이터베이스 종류
  timezone: '+09:00', // 시간대 설정
  define: {
    // 안의 정보를 미리 안내함
    charset: 'utf8',
    collate: 'utf8_general_ci',
    timestamps: true, // 언제 몇시에 만들었는지 표시함
    freezeTableName: true, // 기본적으로 복수명사 이름을 부여하는데 그 기능을 끈다.
  },
});

var db = {};
// 객체식으로 데이터베이스 테이블을 구성하고, 직접 접속할 수 있다.
// 따라서 이런 ORM 방식을 통해, 직접 쿼리를 날리지 않아도
// 테이블 정보가 담긴 객체를 이용하여 데이터 조회,생성,변경,삭제가 가능하다.
db.chatlist = require(path.join(__dirname + '/chatlist.js'))(
  sequelize,
  Sequelize.DataTypes
);

db.sequelize = sequelize;
db.Sequelize = Sequelize;

module.exports = db;
```

chatlist.js

```
module.exports = function(sequelize, DataTypes){
    return sequelize.define('chatlist',{
        id: { // 채팅의 ID
            type: DataTypes.INTEGER,
            autoIncrement: true,
            primaryKey: true,
            allwNull: false
        },
        sender: { // 채팅을 보낸 사람의 닉네임 / 'bot'은 특별 취급
            type: DataTypes.STRING(32),
            allwNull: false
        },
        type: { // 'text' or 'image'
            type: DataTypes.STRING(32),
            allwNull: false
        },
        data: { // type의 따른 데이터
            type: DataTypes.TEXT
        }
    })
}
```

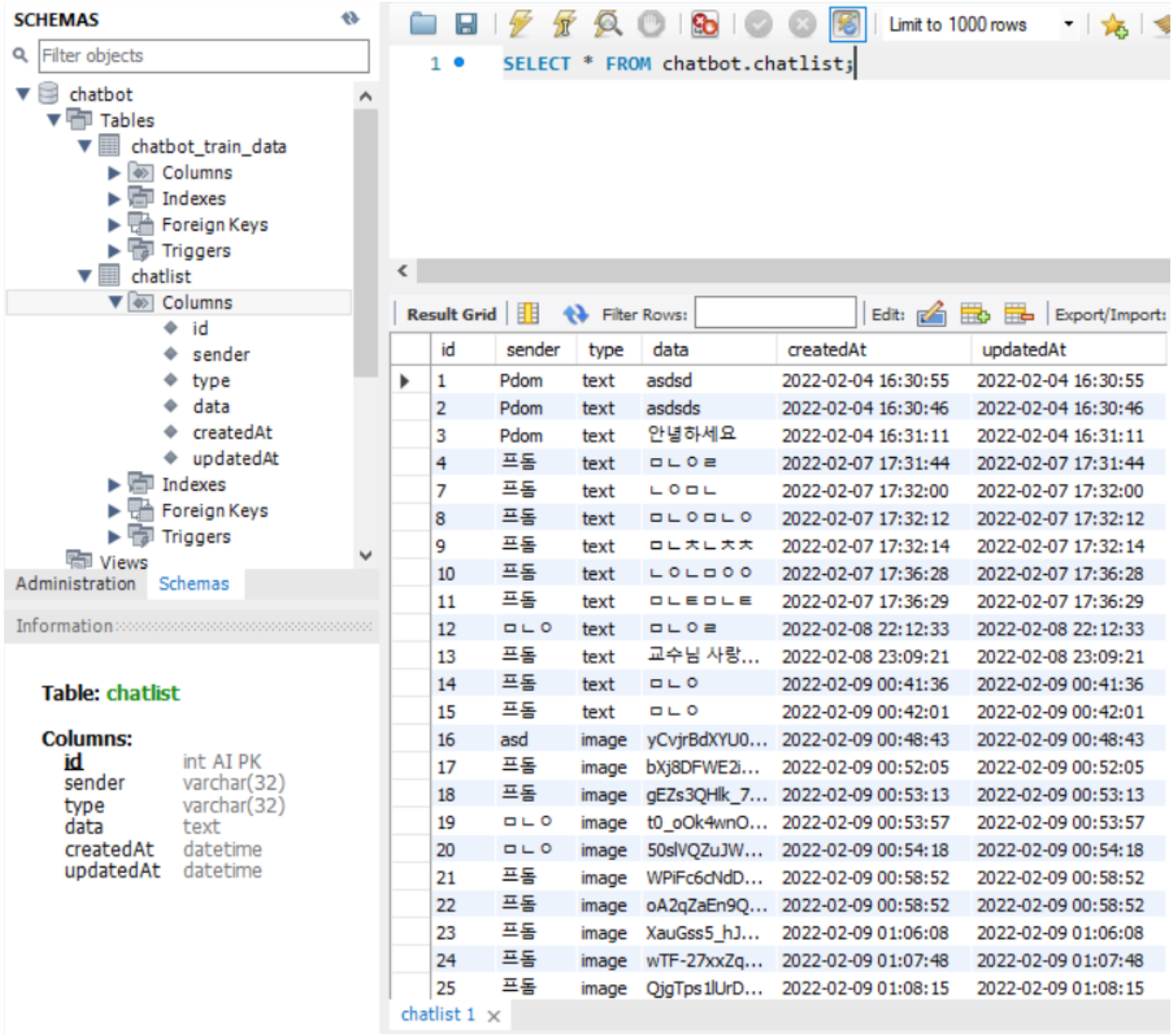
Back-End MySQL

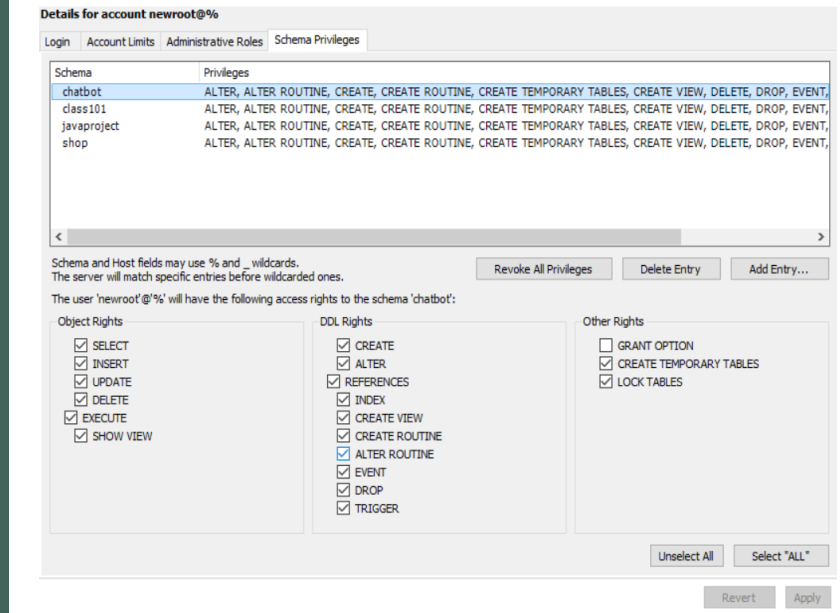
- 프로젝트를 진행하며 팀원들이 DB를 자유롭게 테스트할 수 있도록 포트 포워딩을 하여 DB를 개방했습니다.

- chatlist.js 파일의 define 내 첫 번째 값으로 만들고 싶은 테이블 명을 쓰고 두 번째 값으로 만들고 싶은 컬럼값들을 정의했습니다.

- autoincrement = 값이 저장될 때 자동으로 1씩 증가
- primary key = 고유 값
- allwNull = Not Null로 true면 값이 항상 존재해야 한다.

- chatlist.js에서 테이블을 만든 뒤 db.js에서 Sequelize 라이브러리를 통해 Node.js와 MySQL 내 chatlist 테이블을 연결해주었습니다.


SCHEMAS
Filter objects
chatbot
Tables
chatbot_train_data
Columns
Indexes
Foreign Keys
Triggers
chatlist
Columns
id
sender
type
data
createdAt
updatedAt
Indexes
Foreign Keys
Triggers
Views
Administration
Schemas
Information
Table: chatlist
Columns:
id int AI PK
sender varchar(32)
type varchar(32)
data text
createdAt datetime
updatedAt datetime
Limit to 1000 rows
1 SELECT * FROM chatbot.chatlist;
Result Grid
Filter Rows:
Edit:
Export/Import:
id sender type data createdAt updatedAt
1 Pdom text asdsd 2022-02-04 16:30:55 2022-02-04 16:30:55
2 Pdom text asdsds 2022-02-04 16:30:46 2022-02-04 16:30:46
3 Pdom text 안녕하세요 2022-02-04 16:31:11 2022-02-04 16:31:11
4 프돔 text ㅁㄴㅇㄹ 2022-02-07 17:31:44 2022-02-07 17:31:44
7 프돔 text ㄴㅇㅁㄴ 2022-02-07 17:32:00 2022-02-07 17:32:00
8 프돔 text ㅁㄴㅇㅁㄴㅇ 2022-02-07 17:32:12 2022-02-07 17:32:12
9 프돔 text ㅁㄴㅊㄴㅊㅊ 2022-02-07 17:32:14 2022-02-07 17:32:14
10 프돔 text ㄴㅇㄴㅁㅇㅇ 2022-02-07 17:36:28 2022-02-07 17:36:28
11 프돔 text ㅁㄴㅌㅁㄴㅌ 2022-02-07 17:36:29 2022-02-07 17:36:29
12 ㅁㄴㅇ text ㅁㄴㅇㄹ 2022-02-08 22:12:33 2022-02-08 22:12:33
13 프돔 text 교수님 사랑... 2022-02-08 23:09:21 2022-02-08 23:09:21
14 프돔 text ㅁㄴㅇ 2022-02-09 00:41:36 2022-02-09 00:41:36
15 프돔 text ㅁㄴㅇ 2022-02-09 00:42:01 2022-02-09 00:42:01
16 asd image yCvjrBdXYU0... 2022-02-09 00:48:43 2022-02-09 00:48:43
17 프돔 image bXj8DFWE2i... 2022-02-09 00:52:05 2022-02-09 00:52:05
18 프돔 image gEZs3QHlk_7... 2022-02-09 00:53:13 2022-02-09 00:53:13
19 ㅁㄴㅇ image t0_oOk4wnO... 2022-02-09 00:53:57 2022-02-09 00:53:57
20 ㅁㄴㅇ image 50slVQZuJW... 2022-02-09 00:54:18 2022-02-09 00:54:18
21 프돔 image WPiFc6cNdD... 2022-02-09 00:58:52 2022-02-09 00:58:52
22 프돔 image oA2qZaEn9Q... 2022-02-09 00:58:52 2022-02-09 00:58:52
23 프돔 image XauGss5_hJ... 2022-02-09 01:06:08 2022-02-09 01:06:08
24 프돔 image wTF-27xxZq... 2022-02-09 01:07:48 2022-02-09 01:07:48
25 프돔 image QjgTps1lUrD... 2022-02-09 01:08:15 2022-02-09 01:08:15
chatlist 1


Details for account newroot@%
Login
Account Limits
Administrative Roles
Schema Privileges
Schema
Privileges
chatbot ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT,
class101 ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT,
javaproject ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT,
shop ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT,
Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.
Revoke All Privileges
Delete Entry
Add Entry...
The user 'newroot'@'%' will have the following access rights to the schema 'chatbot':
Object Rights
SELECT
INSERT
UPDATE
DELETE
EXECUTE
SHOW VIEW
DDL Rights
CREATE
ALTER
REFERENCES
INDEX
CREATE VIEW
CREATE ROUTINE
ALTER ROUTINE
EVENT
DROP
TRIGGER
Other Rights
GRANT OPTION
CREATE TEMPORARY TABLES
LOCK TABLES
Unselect All
Select "ALL"
Revert
Apply

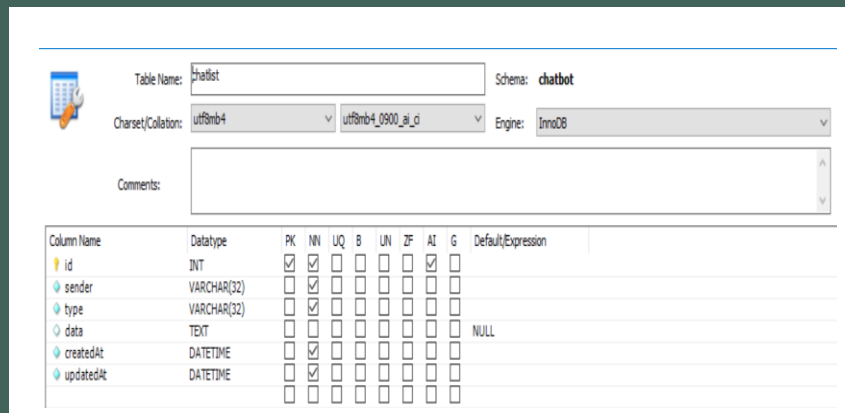


Table Name: chatlist
Schema: chatbot
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:
Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression
id INT
sender VARCHAR(32)
type VARCHAR(32)
data TEXT NULL
createdAt DATETIME
updatedAt DATETIME

```python
import os
import tensorflow as tf
from tensorflow.keras.models import Model, load_model
from tensorflow.keras import preprocessing
from deep.config.GlobalParams import MAX_SEQ_LEN
from deep.Preprocess import Preprocess
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'


# 의도 분류 모델 모듈
class IntentModel:
    def __init__(self, model_name, prep):
        # 의도 클래스별 레이블
        self.labels = {0: "인사", 1: "욕설",
                       2: "주문", 3: "예약", 4: "기타"}

        # 의도 분류 모델 불러오기
        self.model = load_model(model_name)

        # 챗봇 Preprocess 객체
        self.p = prep

    # 의도 클래스 예측
    def predict_class(self, query: str):
        # 형태소 분석
        pos = self.p.pos(query)

        # 문장내 키워드 추출(불용어 제거)
        keywords = self.p.get_keywords(pos, without_tag=True)
        sequences = [self.p.get_wordidx_sequence(keywords)]

        # 패딩처리
        padded_seqs = preprocessing.sequence.pad_sequences\
            (sequences, maxlen=MAX_SEQ_LEN, padding='post')

        predict = self.model.predict(padded_seqs)
        predict_class = tf.math.argmax(predict, axis=1)
        return predict_class.numpy()[0]
```

의도 분류 모델 모듈

- 문장 뒤에 있는 라벨을 보고 의도를 파악하는 모델입니다.

- 라벨을 인식하기 위해 문장 뒤에 있는 라벨이 0이면 인사를 하는 문장이고 1이면 욕설 등의 문장, 2면 주문... 등등 라벨에 따라 의도가 달라집니다.

- 의도 예측 클래스는 Preprocess 내에 있는 형태소 분석기에 들어온 문장을 넣어 불용어 제거 등의 전처리를 한 뒤 케라스 내 패딩처리 메소드로 문장의 길이를 일정하게 맞추고 병렬처리를 하도록 합니다.

```
    # 문장 내 키워드 추출(불용어 제거)
    keywords = self.p.get_keywords(pos, without_tag=True)
    sequences = [self.p.get_wordidx_sequence(keywords)]

    # 패딩 처리
    max_len = 40
    padded_seqs = preprocessing.sequence.pad_sequences(sequences, pad

    # 키워드별 개체명 예측
    predict = self.model.predict(np.array([padded_seqs[0]]))
    predict_class = tf.math.argmax(predict, axis=-1)

    tags = [self.index_to_ner[i] for i in predict_class.numpy()[0]]
    return list(zip(keywords, tags))

def predict_tags(self, query):
    # 형태소 분석
    pos = self.p.pos(query)

    # 문장 내 키워드 추출(불용어 제거)
    keywords = self.p.get_keywords(pos, without_tag=True)
    sequences = [self.p.get_wordidx_sequence(keywords)]

    # 패딩 처리
    max_len = 40
    padded_seqs = preprocessing.sequence.pad_sequences(sequences, pad
    predict = self.model.predict(np.array([padded_seqs[0]]))
    predict_class = tf.math.argmax(predict, axis=-1)

    tags = []
    for tag_idx in predict_class.numpy()[0]:
        if tag_idx == 1: continue
        tags.append(self.index_to_ner[tag_idx])
        if len(tags) == 0:
            return None
        return tags
```

NER 모델 모듈

- 문장 내 불용어를 제외한 키워드에 태그가 달려있는데 그 태그를 보고 단어가 어떤 레이블 안에 포함되는지 학습하는 모듈입니다.

- 자연어 처리 : 주어, 목적어, 서술어처럼 의도를 파악하기 위해 형태소를 분석하는데 예를 들어
- *나 5시에 아이스티 주문할께* 라는 문장에서
- 나, 아이스티, 5시, 주문, 같이 전처리를 한 뒤 문장의 의도를 분석하고 키워드가 어떤 레이블에 속해 있는지 모델을 통해 분석을 해서 이에 맞는 대답을 출력합니다.

```python
# 전처리 객체 생성
p = Preprocess(word2index_dic='../train_tools/dict/chatbot_dict.bin',
               userdic='../user_dic.tsv')

# 의도 파악 모델
intent = IntentModel(model_name='../models/intent/intent_model.h5', prep=p)

# 개체명 인식 모델
ner = NerModel(model_name='../models/ner/ner_model.h5', prep=p)


def to_client(conn, addr, params):
    db = params['db']
    try:
        db.connect()  # DB 연결

        # 데이터 수신
        read = conn.recv(1024)  # 수신 데이터가 있을 때까지 블로킹
        print("============================")
        print("Connection from: %s" % str(addr))

        if read is None or not read:
            # 클라이언트 연결이 끊어지거나 오류가 있는 경우
            print("클라이언트 연결 끊어짐")
            exit(0)  # 스레드 강제 종료

        # json 데이터로 변환
        recv_json_data = json.loads(read.decode())
        print("데이터 수신 : ", recv_json_data)
        query = recv_json_data['Query']

        # 의도 파악
        intent_predict = intent.predict_class(query)
        intent_name = intent.labels[intent_predict]

        # 개체명 파악
        ner_predicts = ner.predict(query)
        ner_tags = ner.predict_tags(query)

        # 답변 검색
        try:
            f = FindAnswer(db)
```

AI Engine

- 앞에 만든 모델 모듈들을 사용해서 입력된 문장들을 분석하고 요구하는 답변을 반환하는 기능을 합니다.

- 앞서 설명한 DB에 연결을 한 뒤 DB에서 수신한 데이터들을 decode한 뒤 의도와 개체명을 파악하고 그것을 토대로 답변을 검색합니다.

- 도출된 답변은 JSON으로 인코딩한 후 클라이언트한테 전달합니다.

프로젝트 시연

시연에 앞서 시연에 사용된 기술에 대해 간략히 설명하겠습니다.

안호빈 팀원이 Synology nas를 이용해서 시연 환경을 구성했습니다.

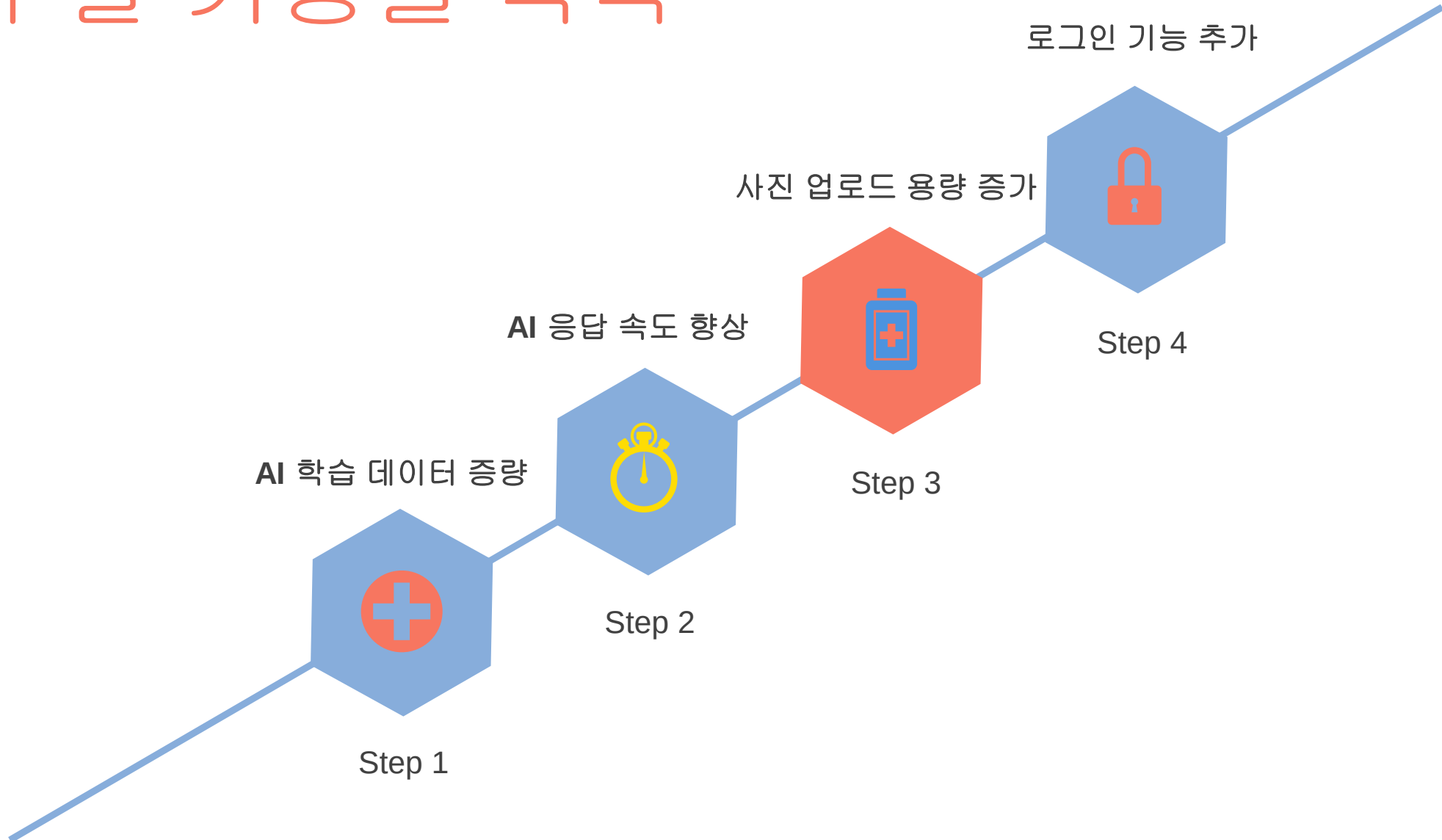
Synology에서 제공하는 mariaDB와 docker 를 이용해서
프론트엔드와 백엔드 서버를 구동하고,

새로운 도메인으로 역방향 프록시를 구성해서 기존 nas의 다른 서비스 80/443 포트와 겹치지 않도록 했습니다.

프로젝트 시연은 현장에서
직접 참여 형태로 진행하고자
합니다.
위 **QR** 코드를 스캔하셔서
저희 채팅 웹사이트를
즐겨보세요!!

QR 코드 스캔 후 닉네임을
입력하고 자유롭게
사용하면 됩니다!

개선해야 할 기능들 목록


로그인 기능 추가
사진 업로드 용량 증가
AI 응답 속도 향상
AI 학습 데이터 증량
Step 4
Step 3
Step 2
Step 1

미구현된 기능들 목록


AI 실시간 학습
Step 1
AI 이미지 학습 추가
Step 2
관리자 기능 추가
Step 3
소셜 로그인 기능 추가
Step 4

참고 문헌

<한빛미디어> 처음배우는 딥러닝 챗봇 (조경래 지음)

- 챗봇 제작의 기초를 탄탄히 다지기 위해 참고하였습니다.

<영진닷컴> 파이썬으로 챗봇 만들기 (저자 Sumit Raj 역자 Daniel Lee)

- -

<한빛미디어> 밑바닥부터 시작하는 딥러닝 (저자 사이토 고키 옮긴이 개앞맵시)

- CNN 구조에 대해서 좀 더 심도 있게 다루기 위해서, 또 CNN의 장단점에 대해 알기 위해서 참고했습니다.

https://derekahndev.github.io/machine%20learning/chatbot-1/

- -

https://youngq.tistory.com/40

- CNN 구조를 이해하기 쉽게 풀어준 덕분에 많이 참고를 했습니다.

https://wikidocs.net/book/2155

- 자연어 처리를 익히기 위해 어떻게 자연어 텍스트 전처리를 하는지 등을 익혔습니다.

https://wegonnamakeit.tistory.com/7

- LSTM에 대해서 더 구체적으로 이해하기 위해 참고하였습니다.

참고 문헌

- https://nowonbun.tistory.com/674 — 파이썬에서 웹소켓을 어떻게 사용하는지 알기 위해 참고했습니다.
- https://igotit.tistory.com/entry/파이썬-웹소켓-WbeSocket-구현 -
- https://85chong.tistory.com/79 -
- https://konlpy.org/ko/latest/index.html 한국어 KoNLPy 전처리 사용을 위해 공식 홈페이지에서 사용 방법을 참조했습니다.
- https://datascienceschool.net/03%20machine%20learning/03.01.02%20KoNLPy%20한국어%20처리%20패키지.html -
- https://velog.io/@soo-im/konlpy-설치-에러-해결책-아나콘다-JPYPE KoNLPy를 사용하기 위해 필요한 JPYPE 설치를 위해서 참고했습니다.


Thank You