KAKAO TALK 대화 분석

심현섭

https://github.com/SimHyunSub/Project/

목차

- 개발 환경
- 개발 동기
- 화면 구현
- 요구 사항
- 분석 설계
- 개발기간
- 구조
- 시행착오

개발 환경

Language

etc

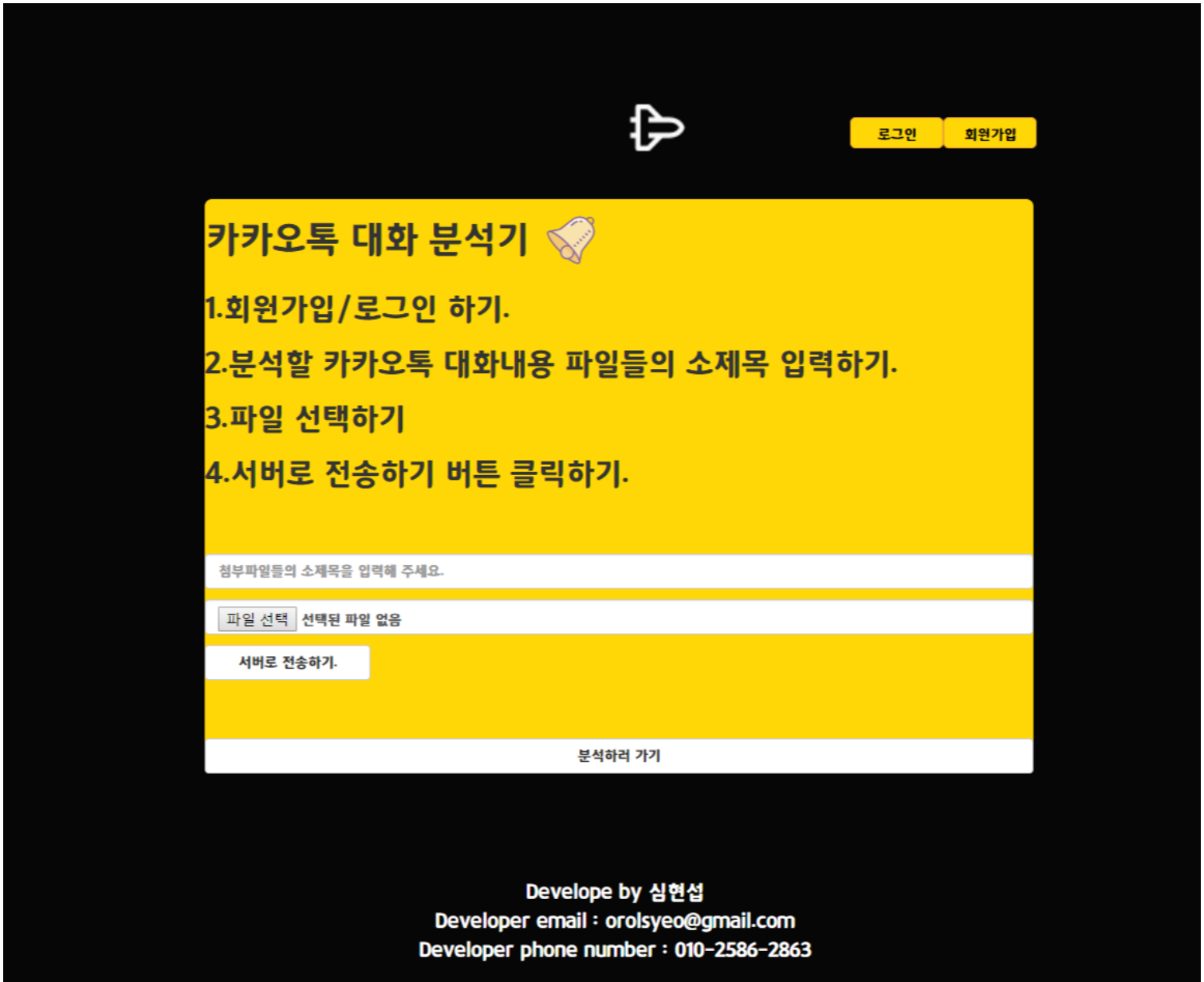

개발 동기

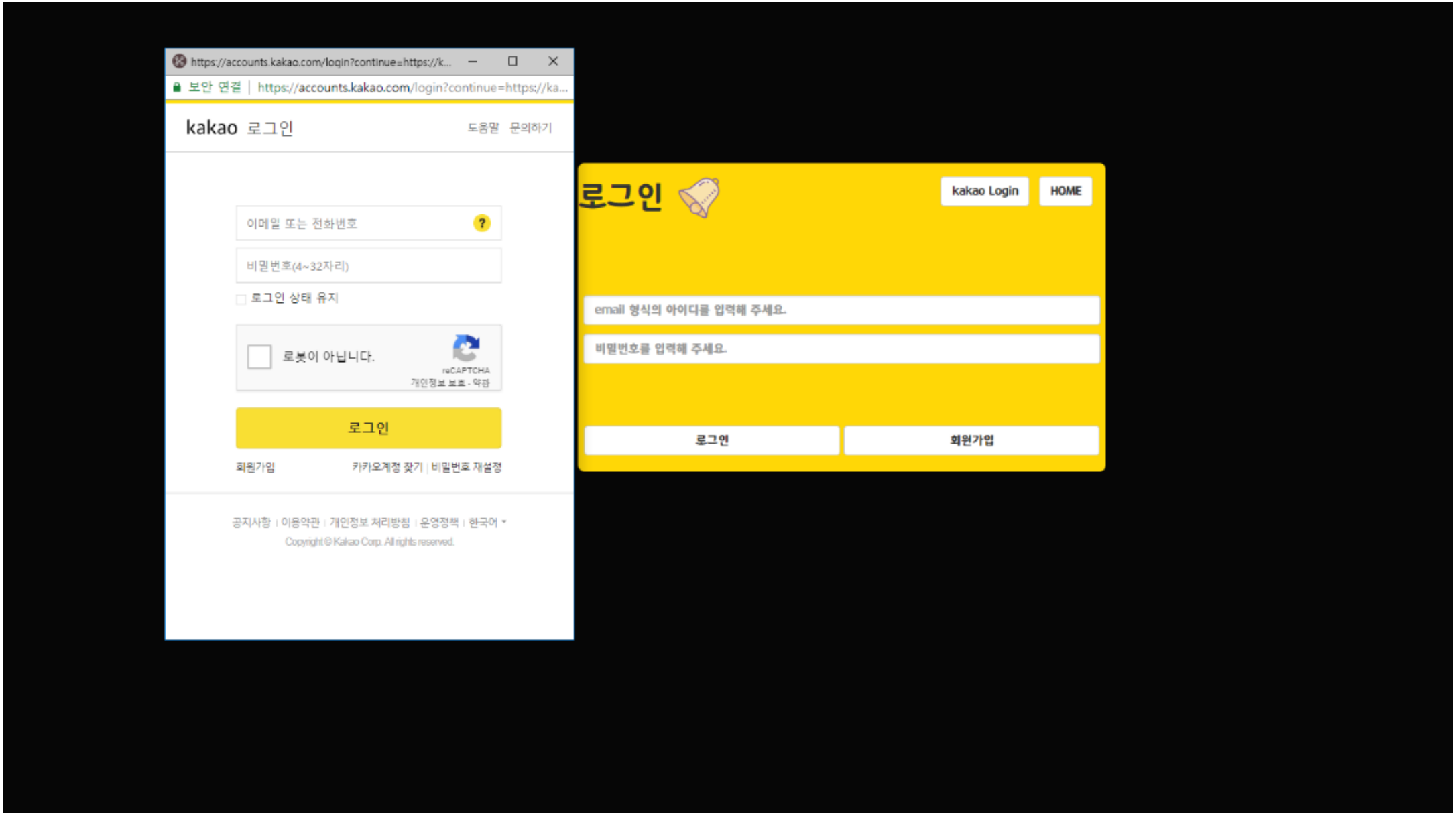
- 지인들에게 빅데이터에 대해서 공부한다고 했더니 카카오톡 대화방 분석을 할 수 있게 해 달라고 해서 만들게 되었다.

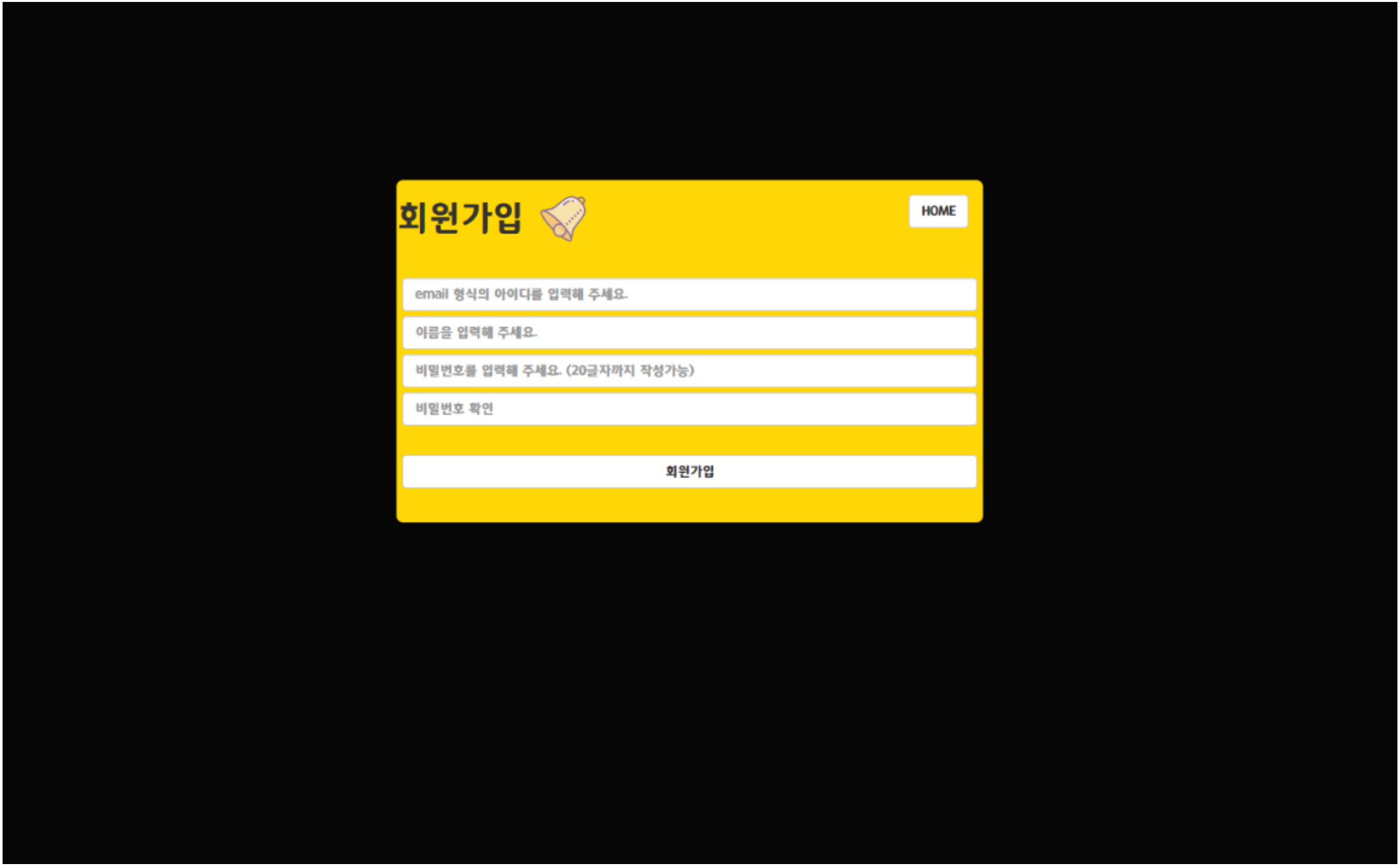
요구 사항

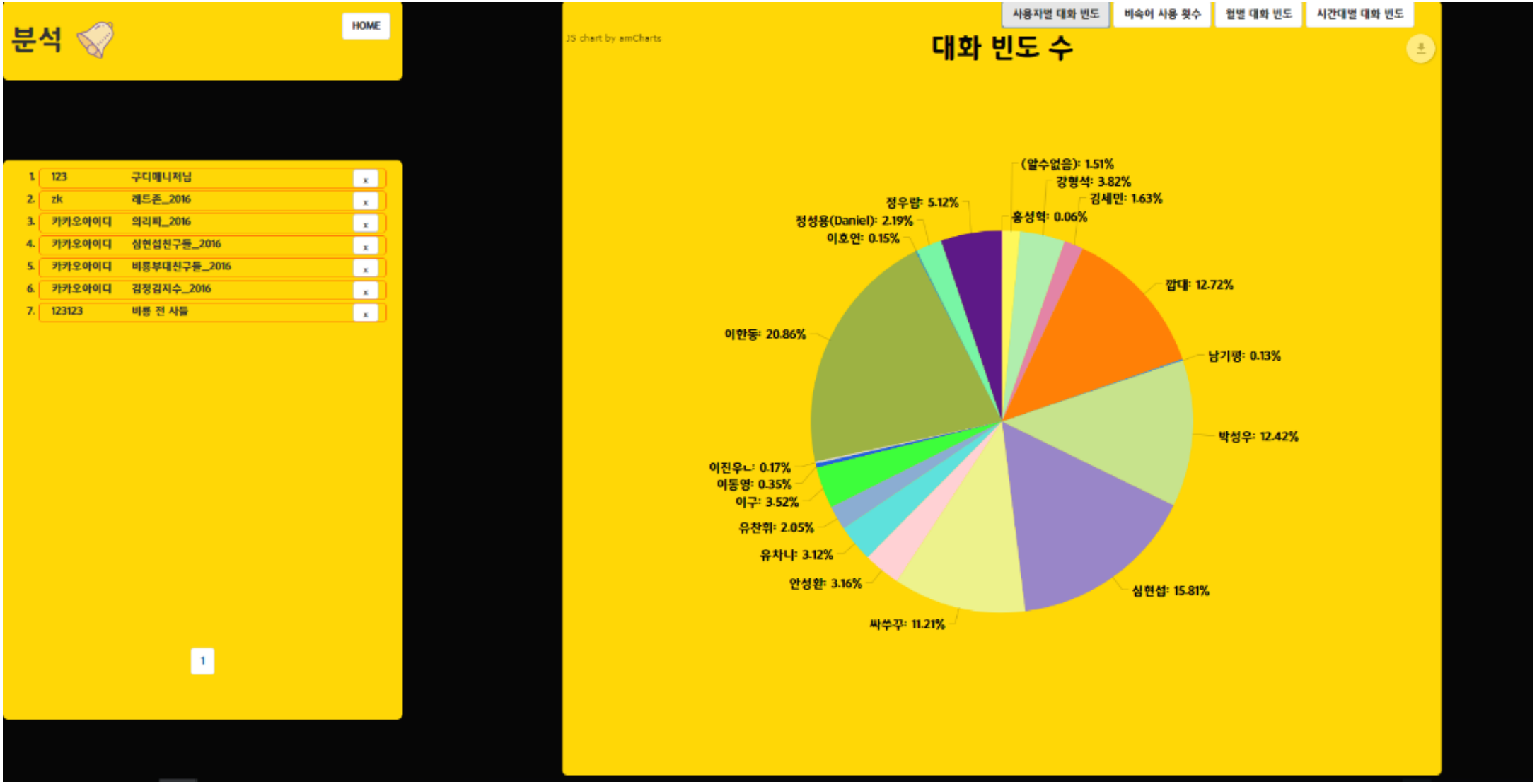
1. Spring을 사용할 것.
2. Hadoop을 사용할 것.
3. 분석 데이터 1000건이상.
4. 분석 데이터를 시각화 할 것.

화면 구현


로그인
회원가입
카카오톡 대화 분석기
1.회원가입/로그인 하기.
2.분석할 카카오톡 대화내용 파일들의 소제목 입력하기.
3.파일 선택하기
4.서버로 전송하기 버튼 클릭하기.
첨부파일들의 소제목을 입력해 주세요.
파일 선택
선택된 파일 없음
서버로 전송하기.
분석하러 가기
Develope by 심현섭
Developer email : orolsyeo@gmail.com
Developer phone number : 010-2586-2863


https://accounts.kakao.com/login?continue=https://k...
보안 연결 | https://accounts.kakao.com/login?continue=https://ka...
kakao 로그인
도움말 문의하기
이메일 또는 전화번호
비밀번호(4~32자리)
로그인 상태 유지
로봇이 아닙니다.
reCAPTCHA
개인정보 보호 - 약관
로그인
회원가입
카카오계정 찾기 | 비밀번호 재설정
공지사항 | 이용약관 | 개인정보 처리방침 | 운영정책 | 한국어
Copyright © Kakao Corp. All rights reserved.
로그인
kakao Login
HOME
email 형식의 아이디를 입력해 주세요.
비밀번호를 입력해 주세요.
로그인
회원가입


회원가입
HOME
email 형식의 아이디를 입력해 주세요.
이름을 입력해 주세요.
비밀번호를 입력해 주세요. (20글자까지 작성가능)
비밀번호 확인
회원가입


분석
HOME
1. 123 구디매니저님 x
2. zk 레드존_2016 x
3. 카카오아이디 의리파_2016 x
4. 카카오아이디 심현섭친구들_2016 x
5. 카카오아이디 비룡부대친구들_2016 x
6. 카카오아이디 김정김지수_2016 x
7. 123123 비룡 전 사들 x
1
사용자별 대화 빈도
비속어 사용 횟수
월별 대화 빈도
시간대별 대화 빈도
JS chart by amCharts
대화 빈도 수
(알수없음): 1.51%
강형석: 3.82%
김세민: 1.63%
홍성혁: 0.06%
정우람: 5.12%
정성용(Daniel): 2.19%
이호연: 0.15%
깝대: 12.72%
이한동: 20.86%
남기행: 0.13%
박성우: 12.42%
이진우ㄴ: 0.17%
이동영: 0.35%
이구: 3.52%
유찬휘: 2.05%
유차니: 3.12%
안성환: 3.16%
심현섭: 15.81%
싸쑤꾸: 11.21%

핵심코드

```
public class KaKaoMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

	@Override
	protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

		KaKaoBean bean = new KaKaoBean(value);

		Text outputKey = new Text();
		IntWritable outputValue = new IntWritable(1);

//		누가 제일 말을 많이 했는가
		outputKey.set(bean.getName());

		if(bean.getContents().length() > 0 ) {
			if(bean.getName().indexOf("\uFFFD") == -1) {
				context.write(outputKey, outputValue);
			}
		}
	}
}
```

사용자별 대화 빈도에 대한
Mapper Class

```
public class KaKaoTermMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

	@Override
	protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

		KaKaoBean bean = new KaKaoBean(value);

		Text outputKey = new Text();
		IntWritable outputValue = new IntWritable(1);

		//몇년도 몇월이 가장 활발하였는가.
		outputKey.set(bean.getDate());
		if(bean.getContents().length() > 0 ) {
			context.write(outputKey, outputValue);
		}
	}
}
```

월별 대화 빈도에 대한
Mapper Class

핵심코드

```
        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

                KaKaoBean bean = new KaKaoBean(value);

                Text outputKey = new Text();
                IntWritable outputValue = new IntWritable(1);

                String tt = bean.getTimeSlot();
                int tl = tt.length();

                String timeSlot = "";
                if(tl >= 6) {
                        timeSlot = tt.substring(0, tt.lastIndexOf(":")) + "시";
                }


                //시간대별 분석
                outputKey.set(timeSlot);

                if(bean.getContents().length() > 10 ) {
                        if(timeSlot.indexOf("") == -1) {
                                context.write(outputKey, outputValue);
                        }
                }
        }
    }
public class KaKaoReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

      @Override
      protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {


                IntWritable result = new IntWritable();

                int sum = 0;
                for(IntWritable value : values) {
                        sum += value.get();
                }
                result.set(sum);
                context.write(key, result);
      }
}
```

시간대별 대화 빈도에 대한
Mapper Class

Reducer Class

핵심코드

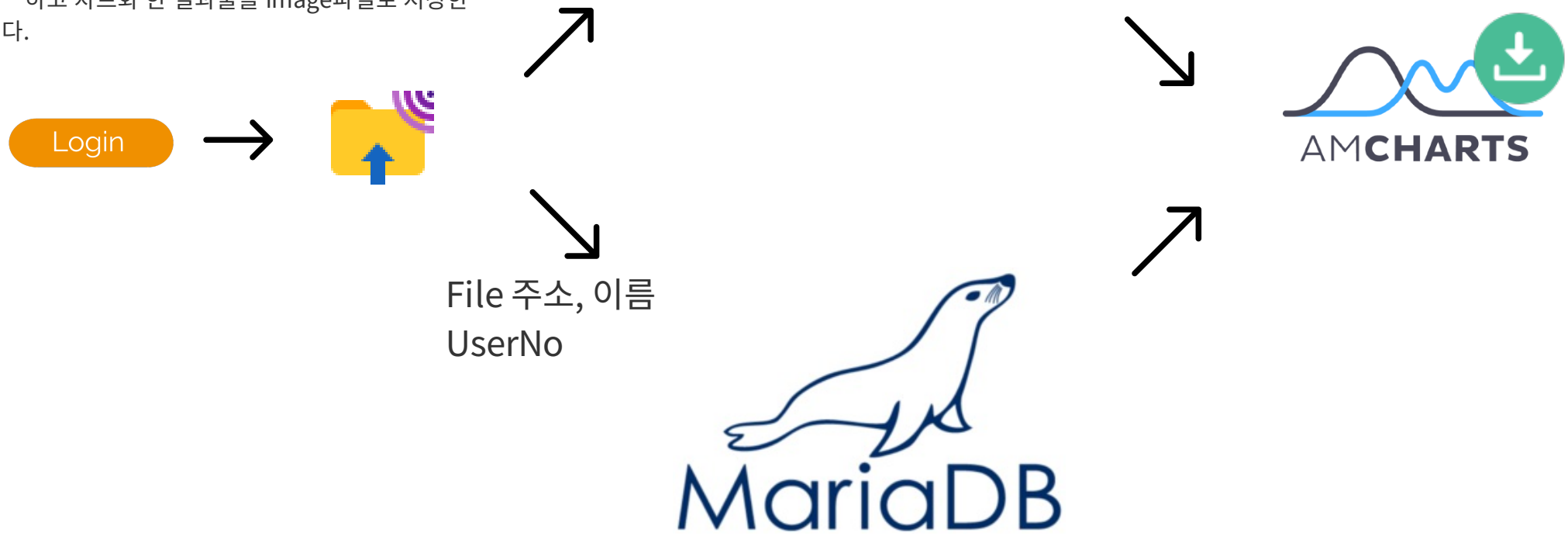
```
public HashMap<String, Object> chartData(String part_r_URL) throws IOException {

        System.out.println("======================시작======================");

        URI uri = URI.create(part_r_URL);
 Path path = new Path(uri);
        FileSystem file = FileSystem.get(uri, conf);
        FSDataInputStream fsis = file.open(path);

        byte[] buffer = new byte[50000];
        int byteRead = 0;
        String result = "";

        while((byteRead = fsis.read(buffer)) > 0) {
           result = new String(buffer, 0, byteRead);
        }
        List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

        HashMap<String, Object> resultMap = new HashMap<String,Object>();

        String[] rows = result.split("\n");
        for (int j = 0; j < rows.length; j++) {
                String row = rows[j];
                String[] cols = row.split("\t");
                HashMap<String, Object> map = new HashMap<String, Object>(); //map을 새롭게

                 for(int c = 0; c < cols.length; c++) {
                    map.put(c + "", cols[c]); //int 더하기 string은 스트링으로 변한다.
             }
        list.add(map);
        }
        resultMap.put("result", list);
        System.out.println("======================끝======================");
        return resultMap;
}
```

part-r-00000
파일의 데이터를 화면으로 보내는 코드

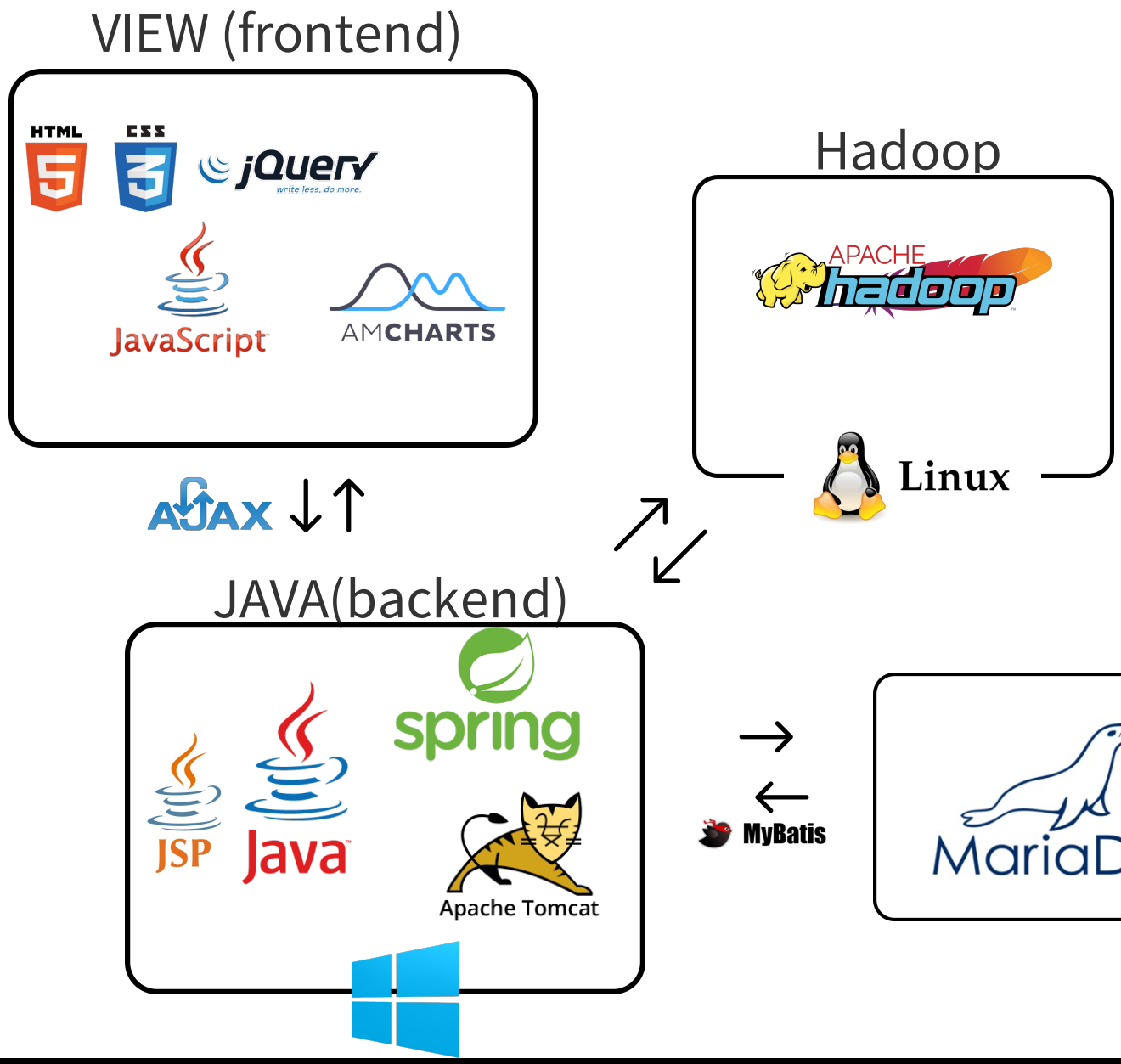
설계

1. 로그인을 하여야 한다.
2. file 업로드를 한다 :
   2-1 : file 주소와 이름을 DB에 저장한다.
   2-2 : 누가 업로드 했는지 알기 위해 user의 번호를 DB에 저장한다.
   2-3: file의 내용을 hadoop에 저장한다.

3. DB에서 파일의 이름, 주소 , user의 번호를 가지고 실제 파일을 분석하고 분석한 내용을 차트화 하고 차트화 한 결과물을 image파일로 저장한다.



Architecture



개발기간

UI 구현 : 32시간
데이터분석 및 개발 : 80시간

시행착오

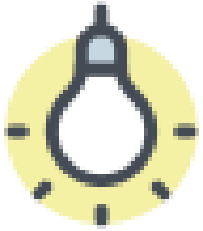
- 카카오톡 텍스트파일을 분석하기좋게 가공하는 과정.

- 카카오톡계정으로 로그인을 하게되면서 본래의 User 데이터와 구분지어 예외처리를 하는 과정.

- Amchart library를 사용하는 과정.

우수 프로젝트 선정


GOODEE ACADEMY
Goodee Academy
빅데이터 분석 및 시각화 처리를 위한 자바 스프링 개발자 과정
소인성
강지안
심현섭
이현경
조한서
구디아카데미
프로젝트 시상식
우수 프로젝트상
♥축하합니다♥
카카오톡 대화 분석기 구현
&빅데이터 분석 및 시각화
쇼핑몰 구현
&빅데이터 분석 및 시각화



감사합니다.