
컴퓨터과학부 종합설계 4조 사용자가 참여하는 **키패드 주차존**

2015920021 박인규
2017920011 김은해
2017920036 양다은
2018920059 허정우

CONTENT

01 프로젝트 개요

02 진행 상황

03 향후 계획

CONTENT

01 프로젝트 개요

02 진행 상황

03 향후 계획

❖ 문제 상황

- 전동 킥보드 방치 문제
 - 교통의 불편함을 초래
- 관련 법안 개정 및 방치된 킥보드 단속
 - 여전히 문제 해결 X

❖ 원인

- 전동 킥보드를 주차할 장소를 찾기 어려움
- 지정된 위치로 이동하여 전동 킥보드를 반납하기 불편함

❖ 과제 목표

- 주차장 정보 제공
 - 어디에 전동 킥보드를 주차해야 하는가?
 - 주차장 위치를 지도에 표시하여 제공
- 주차장 정보 수집
 - 어디를 주차장으로 사용할 수 있을까?
 - 사용자로부터 주차장으로 적합한 지역을 추천받음
- 주차장 정보 공유
 - 우리가 가진 정보를 효과적으로 사용할 수 있을까?
 - API 형태로 수집한 데이터 제공

❖ 기대 효과

- 전동 킥보드 사용자
 - 사용한 전동 킥보드를 어디에 주차해야 할지 쉽게 알 수 있음
 - 가까운 전동 킥보드의 위치를 쉽게 파악할 수 있음
- 일반 시민
 - 방치된 전동 킥보드로 인해 발생하는 교통의 불편함을 줄일 수 있음
- 지자체
 - 전동 킥보드 관리를 위한 피로 완화
(지정된 영역에 전동 킥보드가 밀집해 있으므로)
- 전동 킥보드 업체
 - 자사 플랫폼에 활용 가능



CONTENT

01

프로젝트 개요

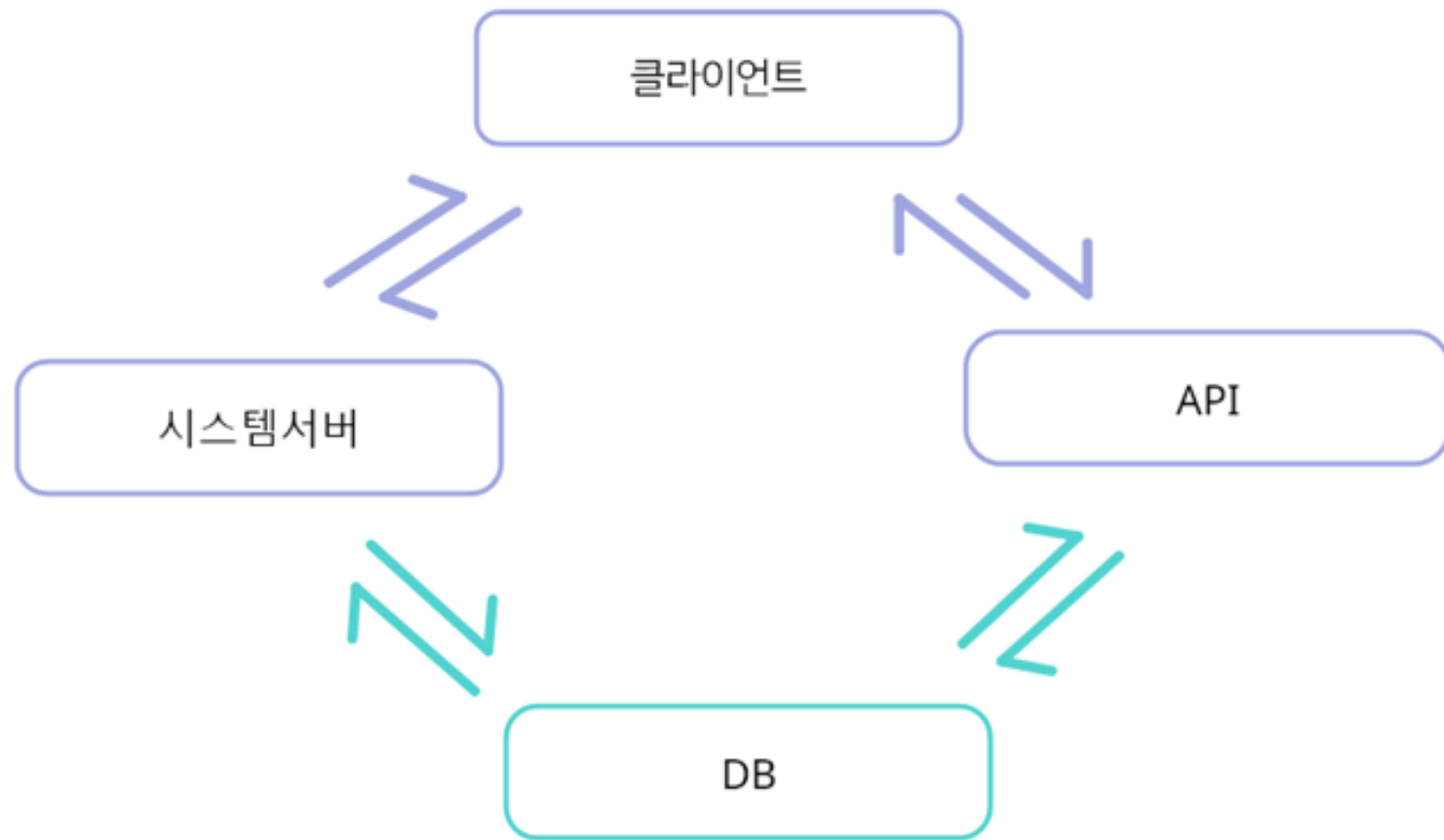
02

진행 상황

03

향후 계획

❖ 시스템 설계



❖ 클라이언트

- 개요
 - 스마트폰 어플리케이션
 - 아이폰, 안드로이드 지원
- 기능
 - 로그인
 - 주차장 정보 제공
 - 유저 데이터 수집

❖ 클라이언트

- 멀티 플랫폼 지원
 - 패키지 설치, 지도 api 파라미터 관리, 제스처 핸들링, 디바이스 권한 등 여러 부분에서 멀티 플랫폼을 고려한 설계 필요
 - OS를 분리하여 CSS와 함수 설정
 - 페이지 단위 개발
 - 스타일 부분에서 중복적인 코드 사용이 있음
- But)
- 직관적으로 확인할 수 있는 파일 구조
 - Component 사용보다 react-native에서 제공하는 기능 위주로 개발

```
ToastAndroid,  
Platform,  
AlertIOS,  
from 'react-native';
```

```
function checkexception(msg){  
  if (Platform.OS === 'android') {  
    ToastAndroid.show(msg, ToastAndroid.SHORT)  
  } else if(Platform.OS === 'ios') {  
    AlertIOS.alert(msg);  
  }  
}
```

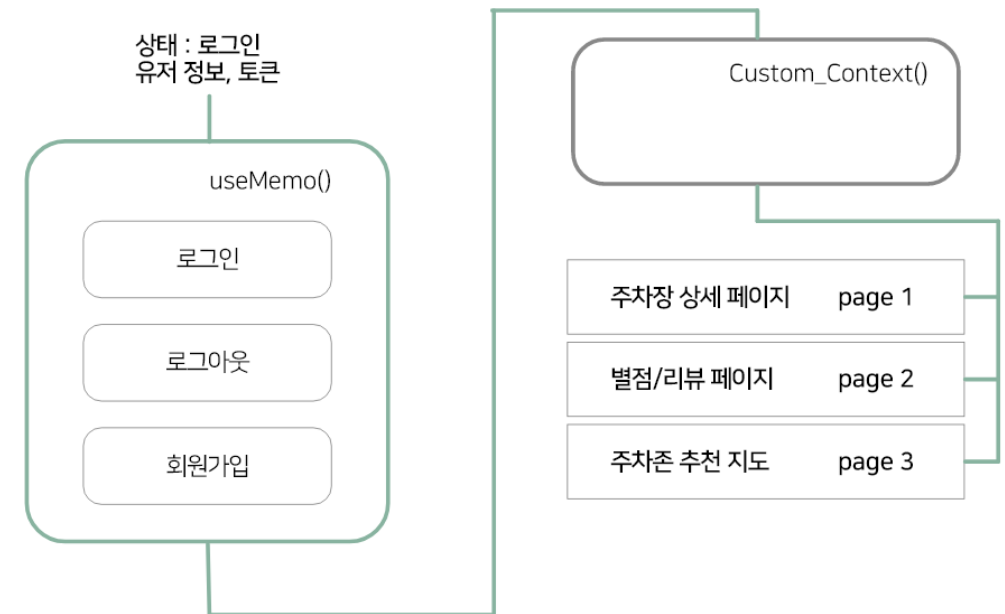
❖ 클라이언트

- 로그인
 - 사용자 인증 시스템
 - 단순 주차장 정보 조회에는 불필요
 - 주차장 추천, 평가 기능 사용을 위해 필요
 - 이미지와 같이, 인터페이스 제작
 - Client-level validation
 - 비밀번호 확인
 - 아이디, 이메일 형식

The image displays two mobile application screens for the PARKICK system. The left screen, titled 'PARKICK', is the login interface, featuring input fields for 'ID' and 'PASSWORD', and a button labeled '로그인 | 회원가입'. The right screen, also titled 'PARKICK', is the registration interface, featuring input fields for '아이디입력' (ID input), '비밀번호입력' (Password input), '비밀번호확인' (Confirm Password), '이름' (Name), '닉네임' (Nickname), and '이메일' (Email), followed by a '회원가입 완료' (Registration complete) message.

❖ 클라이언트

- 로그인
 - 사용자 정보 입력
 - 시스템 서버와 TCP 통신
 - 일회용 토큰 발급
 - 발급 받은 토큰 활용, 로그인 정보 유지



❖ 클라이언트

- 정보 갱신
 - 기존 주차장의 별점, 리뷰 전달
 - 신규 주차장 추천지 전달
 - API로 주차장 정보 수신
 - React native에서 제공하는 Fetch API 사용
 - 네트워크 요청에 관한 Request와 Response 객체를 제공하는 API
 - 서버에서의 응답을 response로 받고, 데이터를 request 객체로 보내는 형식
 - Js 의 promise와 함께 쓰면 네트워크에서 정보를 받고 그 정보로 페이지를 새로 렌더링 후 출력

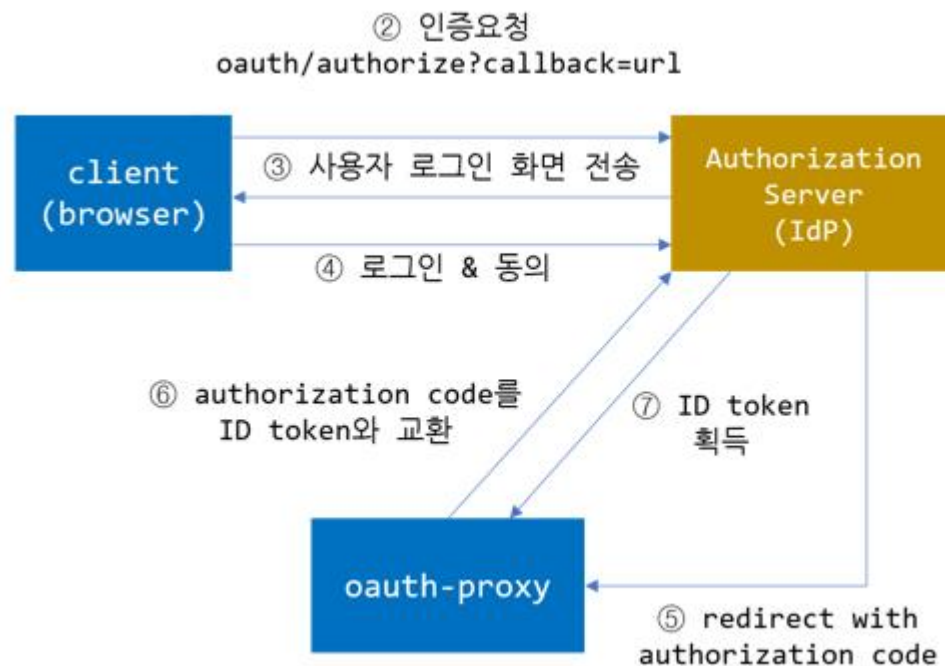
```
function send_zone(lat, lon, name){
  fetch('http://118.67.131.50/zones', {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      zoneid: name,
      latitude: lat,
      longitude: lon
    })))
  .then((response) => response.json())
  .then((data) => {
    console.log(data);
  });
}
```

❖ 시스템 서버

- 개요
 - JAVA를 활용하여 제작
 - 클라이언트가 원활한 서비스를 제공할 수 있도록 보조
- 기능
 - 로그인
 - 사용자 정보 수집
 - 주차장 정보 가공

❖ 시스템 서버

- 로그인
 - TCP통신 활용
 - 사용자 인증을 위한 1회용 토큰 발급
 - 일정 기간 내 갱신하지 않으면 토큰 폐기

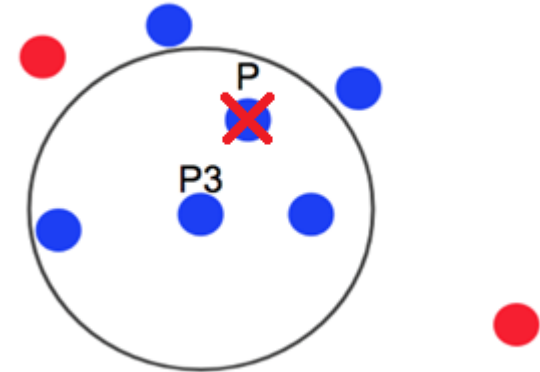
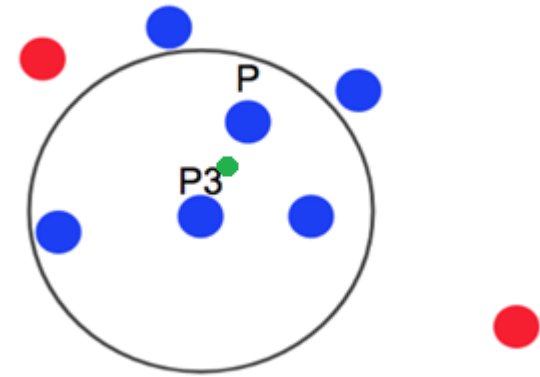
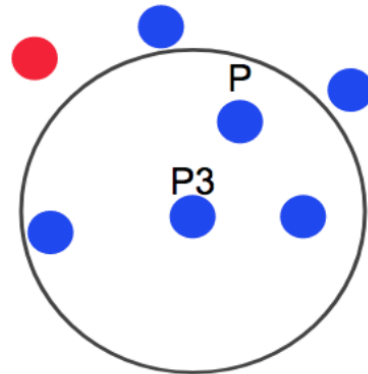
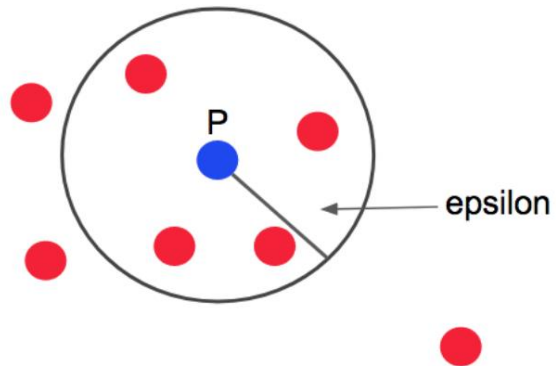


❖ 시스템 서버

- 사용자 정보 수집
 - 사용자로부터 주차장 추천지 수집
 - 수집한 정보를 바탕으로 신규 주차장 선정
 - 발급받은 토큰과 추천지의 위도, 경도를 서버로 전송
 - DB에 추천받은 위치 저장 요청
 - 이후 주차장 정보 가공 과정에 활용되어 신규 주차장 선정

❖ 시스템 서버

- 신규 주차장 선정
 - DBSCAN(밀도기반 클러스터링)기반으로 개선한 알고리즘(자체 개발)
 - Competitive DBSCAN



❖ API 서버

- 개요

- 누구나 주차장 정보를 조회할 수 있도록 정보 공개
- Node.js기반 express 모듈을 이용하여 REST API 구현
 - Express 모듈은 간단하게 URI를 지정하고 request와 response의 처리를 전담하면서 개발 부담을 줄여줌
 - 원하는 API의 동작에 집중해서 REST API를 개발할 수 있어 더욱 효율적이다.
- 보다 발전된 서비스 등장의 기반

- 기능

- 주차장 정보 제공
- 신규 주차장 데이터 수집
- 사용자 코멘트 수집

❖ API 서버

• 주차장 정보 제공

▪ 누구나 정보 접근 가능

- GET http://118.67.131.50/parklots/

▪ 주차장의 위도, 경도와 평가 정보 제공

Parklot

API 기본정보 : Parklot 조회

Verb	Action	Path	Used for
GET	read All	/parklots/	주차장 목록 조회
GET	read	/parklots/no/:no	lotid가 no인 주차장 조회
GET	read	/parklots/id/:id	_id가 id인 주차장 조회

1. 요청변수

분류	요청변수	타입	필수여부	기본값	설명
params	no	Number	-	-	조회하려는 주차장의 lotid
params	no	ObjectId	-	-	조회하려는 주차장의 _id

2. 출력결과

필드	타입	설명	비고
_id	ObjectId	조회된 주차장 document 고유의 id	-
lotid	Number	조회된 주차장 고유의 번호(이름)	-
latitude	Number	조회된 주차장의 위도	-
longitude	Number	조회된 주차장의 경도	-
ratelist	[ObjectId]	조회된 주차장에 평가를 남긴 user의 목록	-
rate	Schema	조회된 주차장의 평가 정보	{like, dislike}
comments	[Schema]	조회된 주차장에 작성된 댓글 목록	{user의 _id, comment} 형식으로 기록

```
{
  "_id": "6199ef51843527ce88b5ab6b",
  "lotid": 1,
  "latitude": 37.58421266273993,
  "longitude": 127.05987609922886,
  "ratelist": [
    {
      "nickname": "이루매"
    }
  ],
  "comments": [
    {
      "user": {
        "nickname": "이루매"
      },
      "comment": {
        "comment": "안녕하세요 서울시립대학교 마스코트 이루매입니다."
      }
    },
    {
      "user": {
        "nickname": "Jelly"
      },
      "comment": {
        "comment": "jelly가 쓴 코멘트 : @>-----"
      }
    },
    {
      "user": {
        "nickname": "Ash"
      },
      "comment": {
        "comment": "렌더링 늦게 하는건 내가 아닌데"
      }
    }
  ],
  "rate": {
    "like": 1,
    "dislike": 0
  },
  "createdAt": "2021-11-21T07:03:45.735Z",
  "updatedAt": "2021-11-21T15:46:44.411Z",
  "__v": 4
},
```

❖ API 서버

- 신규 주차장 데이터 수집
 - 누구나 신규 주차장 추천 가능
 - GET <http://118.67.131.50/zones/>

API 기본정보 : Zone 생성/수정

Verb	Action	Path	Used for
POST	create	/zones/	신규 주차포인트 생성
PUT	update	/zones/id/:id	_id가 id인 주차포인트 수정

1. 요청변수

분류	요청변수	타입	필수여부	기본값	설명
params	id	ObjectId	-	-	수정하려는 주차포인트 document의 _id
body	latitude	String	Y	-	생성/수정하려는 주차포인트의 위도
body	longitude	String	Y	-	생성/수정하려는 주차포인트의 경도
body	suggest	Boolean	Y	-	생성/수정하려는 주차포인트 추천/비추천(T/F) 속성

2. 출력결과

필드	타입	설명	비고
----	----	----	----

```
{
  "_id": "619ae6afd24a388c295ca081",
  "latitude": "37.58421266273993",
  "longitude": "127.05987609922886",
  "suggest": true,
  "createdAt": "2021-11-22T00:39:11.201Z",
  "updatedAt": "2021-11-22T00:39:11.201Z",
  "__v": 0
}
```

❖ API 서버

- 사용자 코멘트 수집
 - 누구나 기존 등록된 주차장에 대한 코멘트 작성 가능
 - GET <http://118.67.131.50/parklots/com/>

API 기본정보 : Parklot에 comment 작성/수정

Verb	Action	Path	Used for
POST	write comment	/parklots/com	comment 작성
PUT	update comment	/parklots/com	comment 수정

1. 요청변수

분류	요청변수	타입	필수여부	기본값	설명
body	no	Number	Y	-	comment를 작성하려는 주차장의 lotid
body	user	ObjectId	Y	-	comment를 작성하려는 사용자의 _id
body	comment	String	Y	-	작성하려 하는 comment의 내용

2. 출력결과

필드	타입	설명	비고
----	----	----	----

```
{
  "_id": "6199f034843527ce88b5ab9c",
  "id": "irumae1918",
  "name": "이루매",
  "nickname": "이루매",
  "email": "irumae@uos.ac.kr",
  "mycomments": [
    {
      "comment": "안녕하세요 서울시립대학교 마스코트 이루매입니다."
    }
  ],
  "lot_rate_list": [
    {
      "lot": {
        "lotid": 1
      },
      "myrate": 1
    }
  ],
  "createdAt": "2021-11-21T07:07:32.822Z",
  "updatedAt": "2021-11-21T15:46:44.411Z",
  "__v": 2
},
```

❖ API 서버

- 사용자 평가 수집
 - 누구나 기존 등록된 주차장에 대한 평가 가능
 - GET <http://118.67.131.50/parklots/rate/>

API 기본정보 : Parklot 평가

Verb	Action	Path	Used for
POST	rate parklot	/parklots/rate/	parklot 평가하기

1. 요청변수

분류	요청변수	타입	필수여부	기본값	설명
body	lotid	ObjectId	Y	-	평가하려는 주차장의 _id
body	userid	ObjectId	Y	-	평가를 하는 사용자의 _id
body	pmt	Number	Y	-	like는 1, dislike는 2

2. 출력결과

필드	타입	설명	비고
----	----	----	----

```
{
  "_id": "6199f034843527ce88b5ab9c",
  "id": "irumae1918",
  "name": "이루매",
  "nickname": "이루매",
  "email": "irumae@uos.ac.kr",
  "mycomments": [
    {
      "comment": "안녕하세요 서울시립대학교 마스코트 이루매입니다."
    }
  ],
  "lot_rate_list": [
    {
      "lot": {
        "lotid": 1
      },
      "myrate": 1
    }
  ],
  "createdAt": "2021-11-21T07:07:32.822Z",
  "updatedAt": "2021-11-21T15:46:44.411Z",
  "__v": 2
},
```

❖ DB 서버

- 개요


- 서비스 제공에 필요한 데이터 저장
- 가시성이 뛰어난 GUI환경인 Mogo DB Compass를 활용하여 전체적인 DB 상황 체크
- open shell 접근이 용이하여 DB connect과 질의 작성에 유용한 Robo 3T를 활용

- 기능

- 유저 정보, 주차장 정보 저장
- 주차장 관련 정보(유저 댓글, 평가) 수정
- 시스템서버와 TCP통신으로 데이터 교환
- API와 restAPI통신으로 데이터 교환

❖ DB 서버

• DB 스키마

Schema	Term	Used for & Event
users	유저 정보	<ul style="list-style-type: none"> sign up/sign in한 유저에 대한 데이터 저장 유저별 comments(댓글), rate(평가) 배열 데이터 저장
zones	주차공간 정보	<ul style="list-style-type: none"> 유저가 추천한 주차공간 혹은 주차금지공간 (위도, 경도) 좌표형태 데이터 저장 주차공간/ 주차금지공간 구분 Boolean 데이터 저장
parklots	주차장 정보	<ul style="list-style-type: none"> 유저기반 공간데이터를 기반으로 클러스터링을 진행한 후 생성된 주차장 (위도, 경도) 좌표형태 데이터 저장 UI를 위한 주차장 이름 데이터 저장 주차장별 comments(댓글) 배열, rate(평가) 수 데이터 저장
rates	주차장 평가 정보	<ul style="list-style-type: none"> 제안한 주차장에 대한 유저 평가  데이터 저장 주차장 사용성 검토를 위한 스키마로, 해당 데이터도 추후 업데이트 클러스터링 알고리즘의 데이터로 사용 Boolean 아닌 Number 타입
comments	주차장 댓글 정보	<ul style="list-style-type: none"> 제안한 주차장에 대한 유저에 대한 텍스트 평가 저장 스키마 검증(validate) 옵션으로 텍스트 수 제한

CONTENT

01

프로젝트 개요

02

진행 상황

03

향후 계획

❖ 개선할 점 논의 (릴리즈본 개발에 앞서)

- 서비스 관련
 - 악성 사용자를 어떻게 관리할 것인가
 - 더미 데이터(노이즈)를 어떻게 처리할 것인가
 - 유저 편의성
 - 서비스가 직관적인가
 - 이해하기 어려운 요소가 있는가
 - 필요성이 낮은 기능이 있는가
 - 폴트 톨러런스(Fault tolerant)
 - 서비스 일부에 장애가 발생하여도 정상적으로 동작하도록 함

❖ 개선할 점 논의 (릴리즈본 개발에 앞서)

- 서비스 외적
 - 시스템 로그는 어떻게 관리할 것인가
 - 모니터링 시스템은 어떻게 구축할 것인가
 - 베타 테스트는 어떻게 진행할 것인가
 - 자체 QA
 - 기술적 QA
 - 사용자 QA