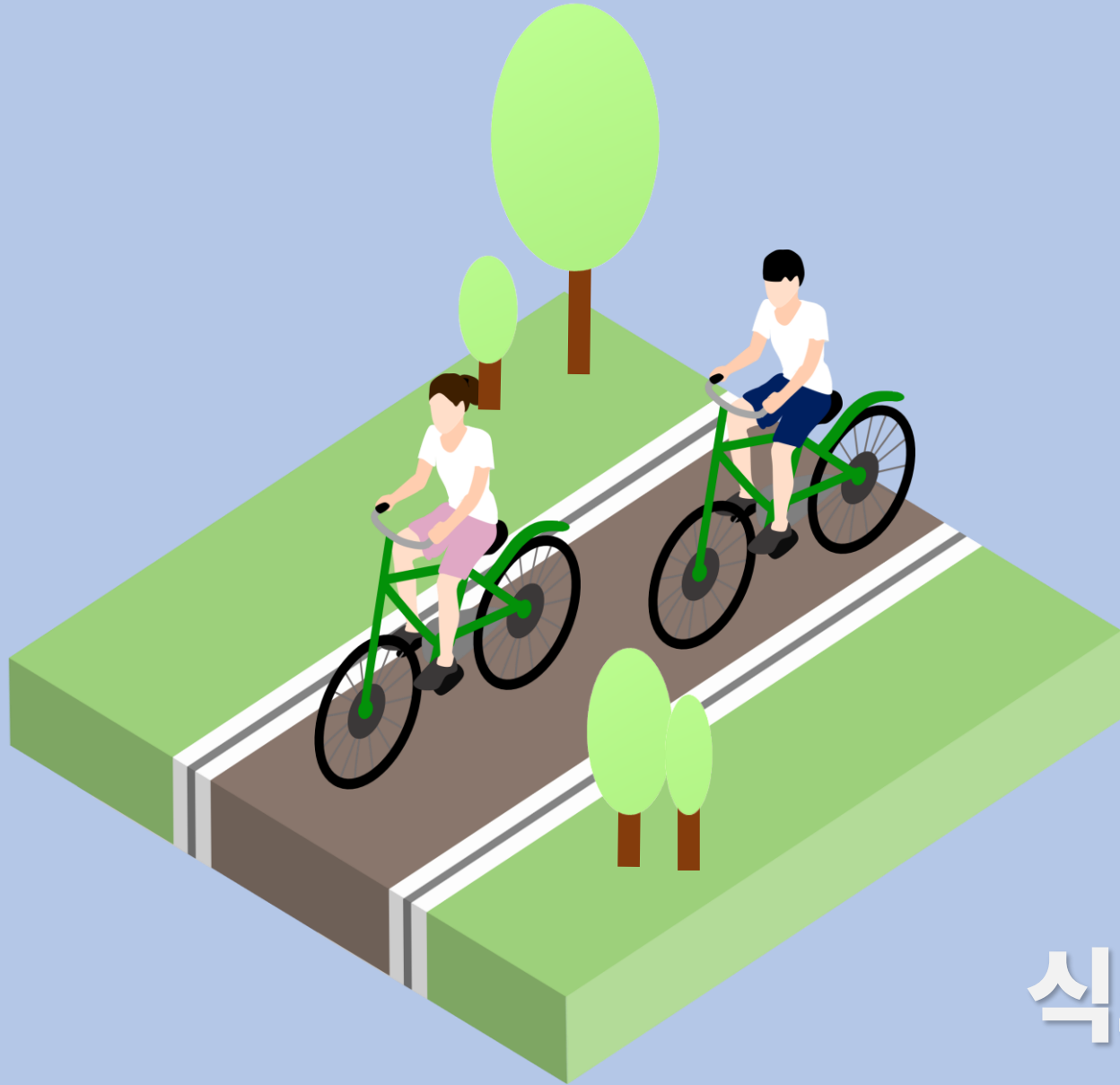


# 바이크 코스 바코

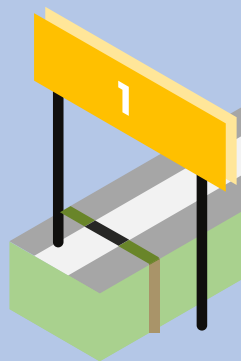


식스센스 팀 권의정 김예나 박혜민  
유영서 이기주 정지윤

# INDEX

01

프로젝트  
목적 및 의도



03

사용한  
기술 스택

3

2

4

02

역할 분담

04

구현 기능  
소개

# 01 프로젝트 목적 및 의도



## 자전거 코스 후기 공유

사용자들의 후기를  
해시태그로 분류해  
코스를 추천



## 경로를 포함한 나만의 후기 저장

경로를 포함한  
후기를 저장하고  
나의 후기를 확인



## 후기에 대한 감정분석

후기 감정 분석으로  
힐링/무난/비추코스로 분류

# 02 역할 분담

## 데이터분석

### 김예나

데이터 전송 및 반환  
단어/감정 분류 모델

### 유영서

데이터 전송 및 반환  
단어/감정 분류 모델

## 프론트엔드

### 권의정

메인 페이지  
후기 작성 페이지  
마이페이지 (회원 정보 수정)  
마이페이지 (나의 후기 보기)  
후기 공유 게시판 페이지

### 이기주

회원가입 페이지  
로그인 페이지  
후기 공유 게시판 페이지

## 백엔드

### 박혜민 (팀장)

후기 저장 API  
경로 표시 API  
후기 상세 조회 API  
경로 좌표 반환 API  
서버 배포

### 정지윤

회원가입 API  
로그인 API  
회원 정보 수정 API  
해시태그 필터링 API  
작성 글 목록 조회API

# 03 사용한 기술 스택

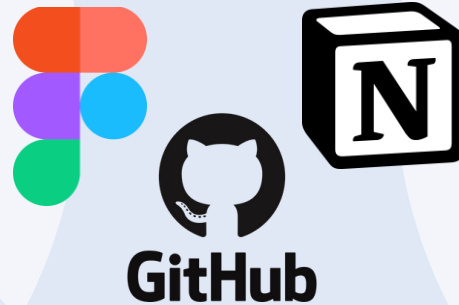
프론트엔드



백엔드



협업 툴

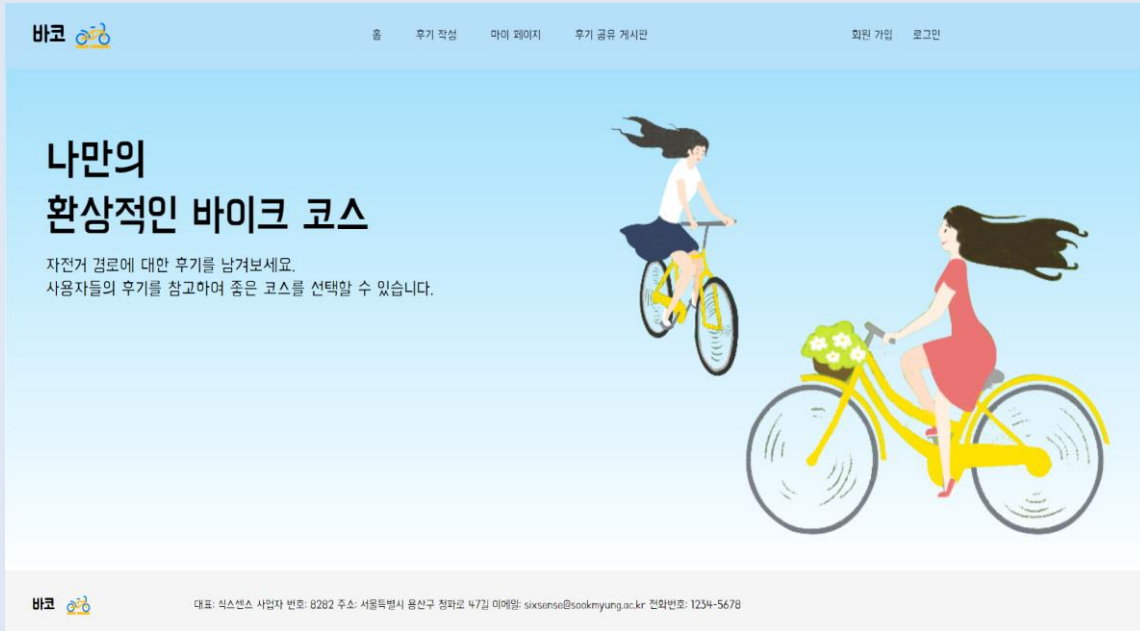


데이터분석



# 04 구현 기능 소개

## 메인페이지



## 로그인 전 바코 기본 홈페이지

## 메인 페이지



## 로그인을 하고나면 로그인한 계정의 닉네임이 우측 상단에 표시됨.

# 04 구현 기능 소개

## 회원가입 페이지

The screenshot shows the Baeco website's membership registration page. The header includes the Baeco logo and navigation links: 홈, 후기 작성, 후기 공유 게시판, 회원 가입, and 로그인. The main content area is titled '회원 가입' and contains a registration form with the following fields: 이메일 (Email) with the value 'test@email.com', 닉네임 (Nickname) with the value '닉네임', 비밀번호 (Password) with masked characters '\*\*\*\*\*', and 비밀번호 확인 (Confirm Password) with masked characters '\*\*\*\*\*'. A yellow '회원가입' button is located at the bottom of the form.

후기작성 및 후기 조회가 가능하도록  
이메일,닉네임,비밀번호 및 비밀번호 확인을  
거쳐 회원가입을 할 수 있도록 함.

## 회원가입 페이지

This screenshot shows the same Baeco membership registration page, but with red error messages indicating that certain fields are required. The errors are: '이메일은 필수 입력입니다.' (Email is required), '닉네임은 필수 입력입니다.' (Nickname is required), '비밀번호는 필수 입력입니다.' (Password is required), and '비밀번호 확인은 필수입니다.' (Confirm password is required). The '회원가입' button remains visible at the bottom.

회원가입 필수 정보를 입력하지 않았을 경우  
필수 입력 조건 안내문구 출력함.

# 04 구현 기능 소개

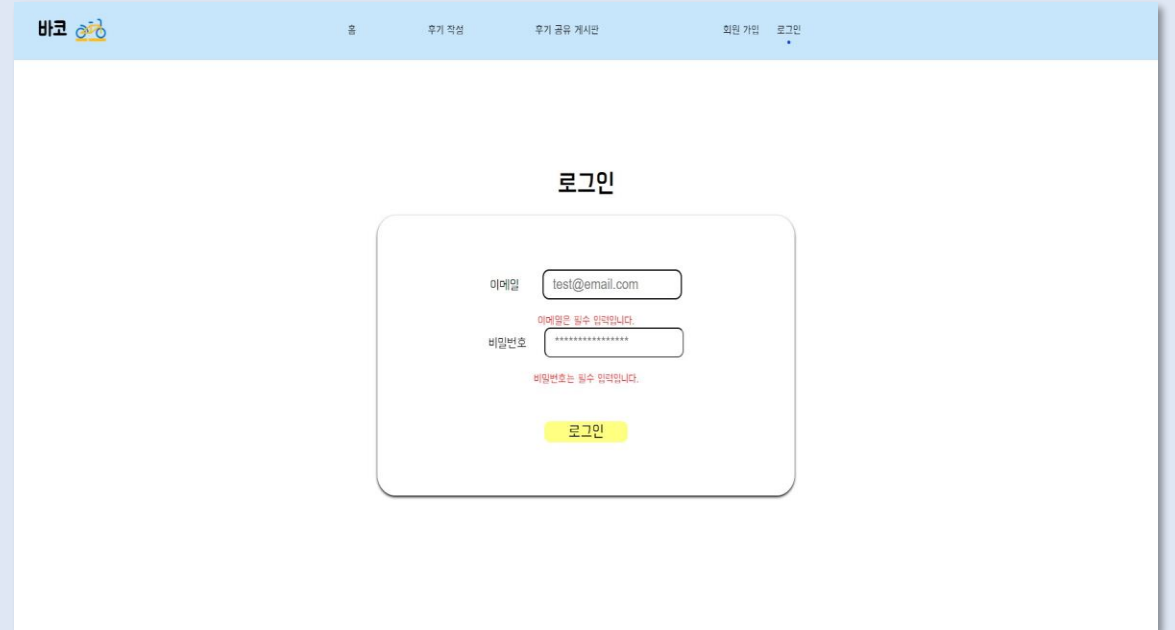
## 로그인 페이지



The screenshot shows the login page of a web application. The header includes the logo '바코' and navigation links: 홈, 후기 작성, 후기 공유 게시판, 회원 가입, and 로그인. The main content area is titled '로그인' and contains a form with two input fields: '이메일' (Email) with the value 'test@email.com' and '비밀번호' (Password) with masked characters. A yellow '로그인' button is at the bottom of the form.

회원가입 시 기입한 이메일과 비밀번호를  
통해 로그인을 할 수 있도록 함.

## 로그인 페이지




The screenshot shows the login page with error messages. The header is the same as the previous image. The main content area is titled '로그인' and contains the same form. Below the password field, there are two red error messages: '이메일은 필수 입력입니다.' (Email is required) and '비밀번호는 필수 입력입니다.' (Password is required). The '로그인' button is still visible at the bottom of the form.

로그인 필수 정보를 입력하지 않았을 경우  
필수 입력 안내문구를 출력함.



# 04 구현 기능 소개

## 마이페이지\_회원정보 수정

바코  홈 후기 작성 후기 공유 게시판 마이 페이지 바코 님

마이페이지

회원 정보 수정하기 나의 후기 보기

닉네임 변경할 닉네임을 입력하세요

비밀번호 변경할 비밀번호를 입력하세요

비밀번호 확인 비밀번호를 한번 더 입력하세요

수정 완료

비밀번호와 닉네임을 수정할 수 있는  
회원 정보 수정 기능.

## 마이페이지\_회원정보 수정

바코  홈 후기 작성 후기 공유 게시판 마이 페이지 식스센스 님

마이페이지

회원 정보 수정하기 나의 후기 보기

닉네임 변경할 닉네임을 입력하세요

비밀번호 변경할 비밀번호를 입력하세요

비밀번호를 입력하세요.

비밀번호 확인 비밀번호를 한번 더 입력하세요

비밀번호 확인은 필수입니다.

수정 완료

회원 정보 수정에 필요한 필수 정보를  
입력하지 않은 경우 안내문구 출력.

# 04 구현 기능 소개

## 후기 작성 페이지

바코  홈 후기 작성 후기 공유 게시판 마이 페이지 바코 님

후기 작성

map loading...

**출발지** 숙명여자대학교

**도착지** 남산타워

**후기**  
날씨가 더워서 많이 힘들어요!

후기 저장

출발지와 도착지 장소 및 후기 내용을  
입력할 수 있는 후기 작성 페이지.

## 후기 작성 페이지

바코  홈 후기 작성 후기 공유 게시판 마이 페이지 바코 님

후기 작성



**출발지** 출발지를 입력하세요.

**도착지** 도착지를 입력하세요.

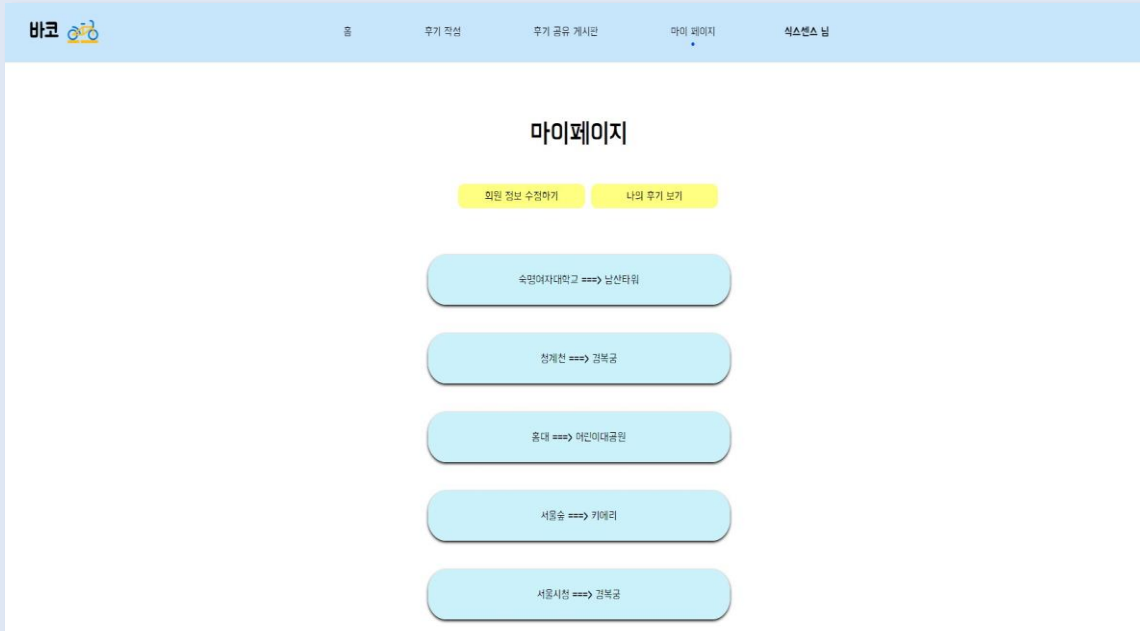
**후기**  
후기를 작성하세요.

후기 저장

후기 저장 이후, 출발지와 도착지까지의  
경로를 확인할 수 있는 지도 표시.

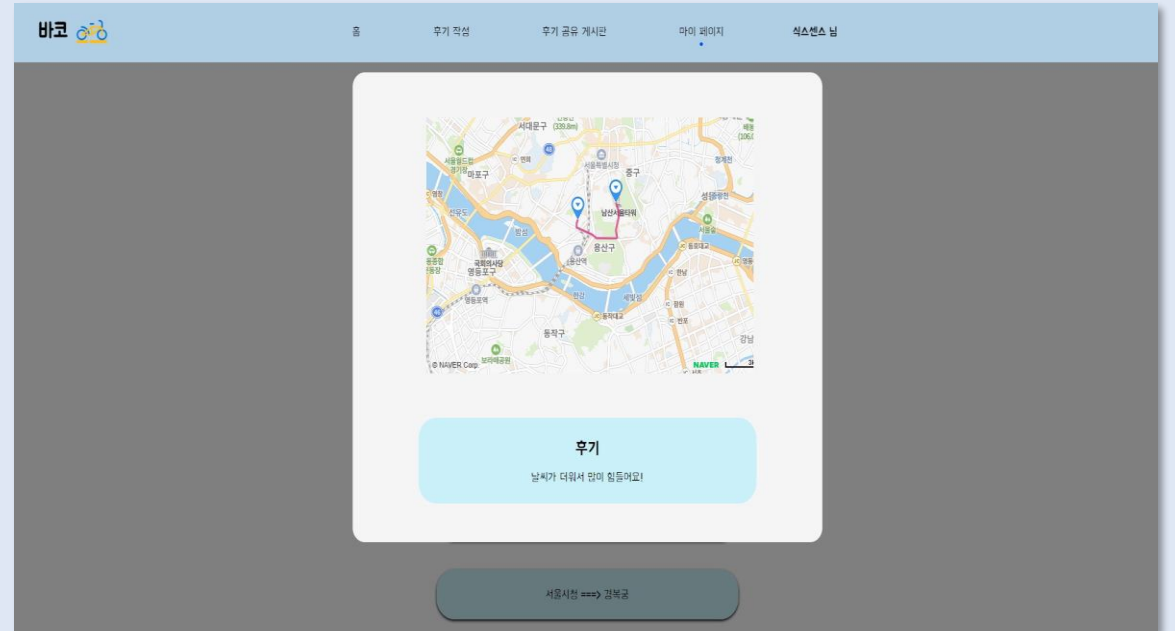
# 04 구현 기능 소개

## 마이페이지\_나의 후기 보기



사용자는 자신이 작성한 후기들을 한번에 모아서 확인할 수 있음.

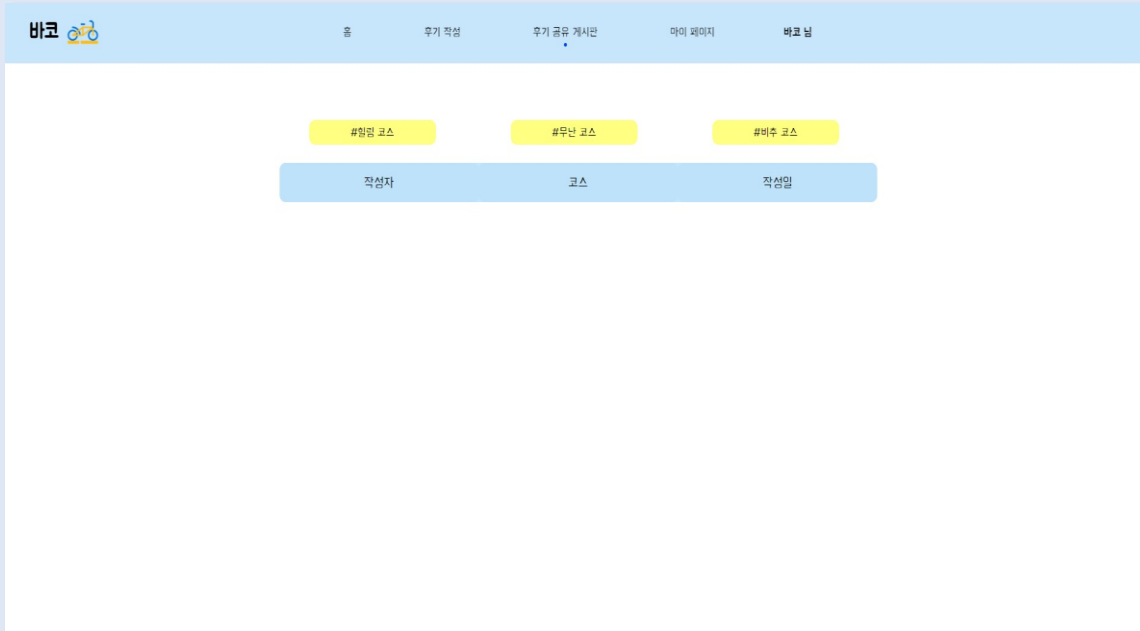
## 마이페이지\_나의 후기 보기



목록에서 후기 하나를 선택하면 후기의 상세 내용 및 경로를 확인할 수 있음.

# 04 구현 기능 소개

## 후기 공유 게시판 페이지



후기 내용을 기반으로 감정 분석 진행 후  
해시태그를 부여하고, 이후 게시판에서는  
해시태그를 기준으로 필터링 할 수 있음.

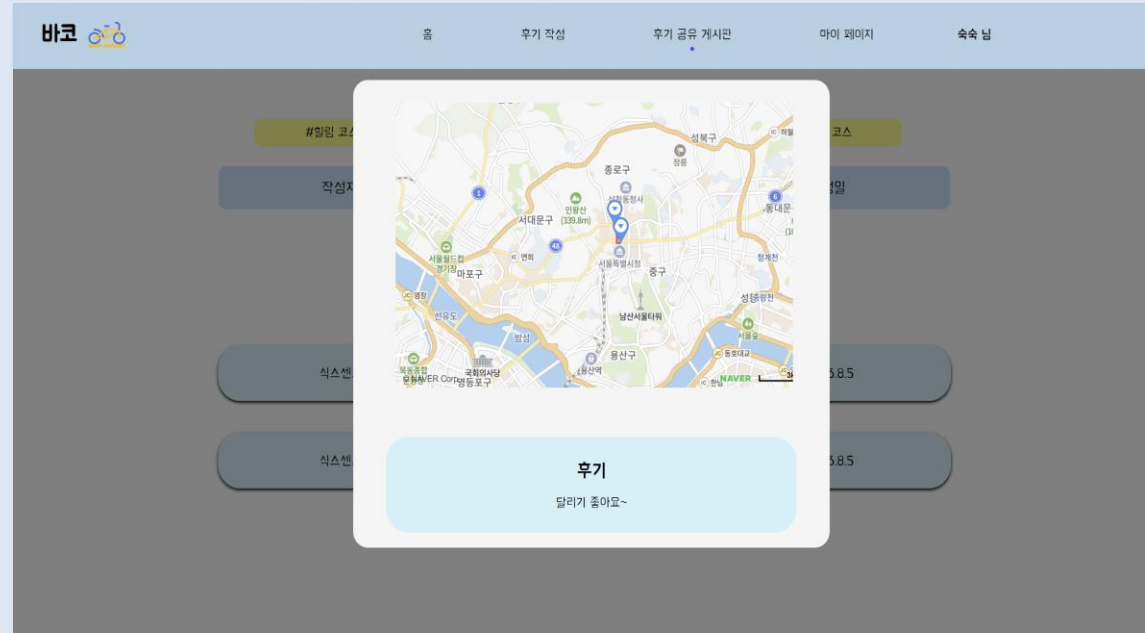
## 후기 공유 게시판 페이지



#힐링코스, #무난코스, #비추코스 중  
하나의 해시태그 선택 시, 해시태그에  
해당하는 후기 목록을 확인 가능.

# 04 구현 기능 소개

## 후기 공유 게시판 페이지



해시태그를 기반으로 한 후기 목록 중 후기 하나를 선택한 경우 상세 조회가 가능.

# 05 주요 코드 설명\_다중 감정 분류 모델 훈련 및 테스트 (데이터분석①)

```
# train : 모델 훈련시키기
for e in range(num_epochs):
    train_acc = 0.0
    test_acc = 0.0
    model.train()
    for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm_notebook(train_dataloader)):
        optimizer.zero_grad()
        token_ids = token_ids.long().to(device)
        segment_ids = segment_ids.long().to(device)
        valid_length = valid_length
        label = label.long().to(device)
        out = model(token_ids, valid_length, segment_ids)
        loss = loss_fn(out, label)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_grad_norm)
        optimizer.step()
        scheduler.step() # Update learning rate schedule
        train_acc += calc_accuracy(out, label)
        if batch_id % log_interval == 0:
            print("epoch {} batch id {} loss {} train acc {}".format(e+1, batch_id+1, loss.data.cpu().numpy(), train_acc / (batch_id+1)))
    print("epoch {} train acc {}".format(e+1, train_acc / (batch_id+1)))

    model.eval()

    for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm_notebook(test_dataloader)):
        token_ids = token_ids.long().to(device)
        segment_ids = segment_ids.long().to(device)
        valid_length = valid_length
        label = label.long().to(device)
        out = model(token_ids, valid_length, segment_ids)
        test_acc += calc_accuracy(out, label)
    print("epoch {} test acc {}".format(e+1, test_acc / (batch_id+1)))
```

# train : 모델 훈련에 대한 코드

사용된 라이브러리

```
# koBERT
from kobert.utils import get_tokenizer
from kobert.pytorch_kobert import get_pytorch_kobert_model
# Transformers
from transformers import AdamW
from transformers.optimization import get_cosine_schedule_with_warmup
# torch, gluonnlp, numpy, pandas
```

# **.eval()** : nn.Module에서 train time과 eval time에서 수행하는 다른 작업을 수행할 수 있도록 switching 하는 함수

즉, model이 Dropout이나 BatNorm2d를 사용하는 경우, train 시에는 사용하지만 evaluation을 할 때에는 사용하지 않도록 설정해주는 함수

## KoBERT

(Korean Bidirectional Encoder Representations from Transformers)  
: 기존 BERT의 한국어 성능 한계를 극복하기 위해 SKT Brain에서 개발한 모델

## 활용 데이터셋

AI Hub내 KAIST 인공지능연구소에서 구축한 감정 분류를 위한 대화 음성 데이터셋에서 '5차년도 2차 CSV'로, 총 7가지 감정 (*happiness, angry, disgust, fear, neutral, sadness, surprise*)에 대해 라벨링이 되어 있음

# 05 주요 코드 설명\_다중 감정 분류 모델 훈련 및 테스트 (데이터분석①)

# predict : 학습 모델을 활용하여 다중 분류된 클래스를 출력해주는 함수  
# 학습된 모델을 활용하여 다중 분류된 클래스를 출력해주는 predict 함수를 구현한 것

def predict(predict\_sentence): # input = 감정분류하고자 하는 sentence

data = [predict\_sentence, '0']  
dataset\_another = [data]

another\_test = BERTDataset(dataset\_another, 0, 1, tok, vocab, max\_len, True, False) # 토큰화한 문장  
test\_dataloader = torch.utils.data.DataLoader(another\_test, batch\_size = batch\_size, num\_workers = 5) # torch 형식 변환

model.eval()

for batch\_id, (token\_ids, valid\_length, segment\_ids, label) in enumerate(test\_dataloader):  
 token\_ids = token\_ids.long().to(device)  
 segment\_ids = segment\_ids.long().to(device)

valid\_length = valid\_length  
label = label.long().to(device)

out = model(token\_ids, valid\_length, segment\_ids)

test\_eval = []

for i in out: # out = model(token\_ids, valid\_length, segment\_ids)  
 logits = i  
 logits = logits.detach().cpu().numpy()

# 결과값이 0이면 비추코스 의미, 차례대로 무난코스, 힐링코스

```
if np.argmax(logits) == 0:  
    test_eval.append("0") #비추코스  
elif np.argmax(logits) == 1:  
    test_eval.append("1") #무난코스  
elif np.argmax(logits) == 2:  
    test_eval.append("2") #힐링코스
```

return test\_eval[0]

# input = 감정분류하고자 하는 sentence

# 토큰화한 문장

# torch 형식 변환

결과 값이 0이면 비추코스  
1이면 무난코스  
2이면 힐링코스

# predict : 학습 모델을 활용하여 다중 분류된 클래스를  
출력해주는 함수

# 학습된 모델을 활용하여 다중 분류된 클래스를 출력해주는  
predict 함수를 구현한 것

# 05 주요 코드 설명\_ 데이터 전송 및 반환(데이터분석②)

```
# 실시간으로 추가되는 데이터를 가져오고 분석하여 저장
def process_realtime_data():
    while True:
        # 새로운 데이터 가져오기
        new_data_query = "SELECT content FROM review WHERE analyzed = 0"
        cursor.execute(new_data_query)
        new_data = cursor.fetchall()

        if new_data:
            for row in new_data:
                text_to_analyze = row[0]
                print("입력 문장:", text_to_analyze)

                # 분석 결과 예측
                analyze_text = predict(text_to_analyze)
                print("분석 결과:", analyze_text)

                # 분석 결과 데이터베이스에 저장
                update_query = f"UPDATE review SET hashtag = '{analyze_text}', analyzed = 1 WHERE content = '{text_to_analyze}'"
                cursor.execute(update_query)
                conn.commit()

            else:
                print("새로운 데이터 없음")
                # 적절한 대기 시간을 두고 반복 수행
                time.sleep(5)

try:
    process_realtime_data()
except KeyboardInterrupt:
    print("실시간 데이터 처리를 종료합니다.")
```

# 실시간으로 추가되는 데이터를 가져오고 분석하여 저장

# 새로운 데이터 가져오기

# 분석 결과 예측

# 분석 결과 데이터베이스에 저장

# 적절한 대기 시간을 두고 반복 수행

실시간으로 데이터를 받아오기 위해 **time** 함수 사용

이미 분석한 데이터의 **analyzed** 컬럼을 1로 변경하여 새로 입력된 데이터들만 분석 가능하게 설정

런타임이 종료되어도 **process\_realtime\_data()** 함수가 무한 루프로 동작하고 있기 때문에 실시간으로 데이터를 가져와서 분석하고 저장하는 작업을 지속적으로 수행 가능



# 05 주요 코드 설명\_후기 작성 페이지(프론트엔드 ①)

```
//...style component 생략...//
function Road() {
  //(1) 사용자로부터 출발지(start), 도착지(end), 후기(review) 값을 입력받음
  const { register, handleSubmit, setValue, getValues, formState: { errors, isDirty }, watch } = useForm<IForm>({
    mode: "onSubmit",
    defaultValues: {
      start: "",
      end: "",
      review: "",
    },
  });
  //(2) "후기 저장" 버튼을 누르면, 입력 받았던 값들이 서버로 전송
  const onValid = ({start, end, review}: IForm) => {

    axios.post('https://port-0-baco-server-eg4e2alkhufq9d.sel4.cloudtype.app/Review/save',
    { email:email ,
      startPlace:start, //(2-1) 서버에 로그인한 이메일, 출발지, 도착지, 후기 내용 전송
      endPlace:end,
      content:review,
    },
    )
    .then((response) => { //(2-2) 성공적으로 전송 되었을 때, data 반환
      setMapUrl(response.data.mapUrl); //(2-3) 출발지, 도착지에 해당하는 map URL을 가져옴 => 화면에 map 바로 출력
      console.log(response.data);
    }).catch(function (error) { //(2-4) 서버에 데이터 전송 실패시, 실행 되는 함수 => 오류메세지 출력
      console.log("서버에 후기 작성 정보를 보내는데 실패했습니다");
    }).then(function() {
      // 항상 실행
    });
  });

  /*완성 된 컴포넌트 화면에 보여주는 코드
  .... (생략)...*/
}
```

## Road.tsx (후기 작성 페이지)의 코드

(1) 사용자로부터 출발지(start), 도착지(end), 후기(review) 값을 입력 받음

(2) "후기 저장" 버튼을 누르면, 입력 받았던 값들이 서버로 전송

(2.1) 서버에 로그인한 이메일, 출발지, 도착지, 후기 내용 전송

(2.2) 성공적으로 전송 되었을 때, data 반환

(2.3) 출발지, 도착지에 해당하는 map URL을 가져옴  
=> 화면에 map 바로 출력

(2.4) 서버에 데이터 전송 실패 시, 실행 되는 함수  
=> 오류 메시지 출력

# 05

## 주요 코드 설명\_ 마이 페이지 → 나의 후기 보기 페이지(프론트엔드 ①)

다음 장에  
계속 (1/2)

//... style component 요소 생략 ...

function MyList(){

//(1) 로그인 한 userID값 가져오기

const userID = useRecoilValue(isLoginAtom);

//(2) 내가 저장한 후기 목록 데이터를 서버로 부터 가져오기

function fetchReview() {

return fetch(`\${BASE\_URL}/mypage/my-reviews/\${userID}`)

.then((response) =>

response.json())

);

}

const { isLoading, data: reviewData } = useQuery<IReview[]>("allReview", fetchReview);

//(3) 내가 저장한 후기 목록의 "상세보기" 데이터를 서버로 부터 가져오기

function fetchReviewInfo(review\_id: string) {

return fetch(`\${BASE\_URL}/Review/detail/\${review\_id}`).then((response) =>

response.json())

);

}

const { isLoading: infoLoading, data: infoData } = useQuery<InfoData>("3-1 서버로부터 받아온 데이터를 infoData 변수로 저장

['info', id],

() => fetchReviewInfo(id)

);

/\* 가능 관련 함수 선언 및 변수 코드 길어서 생략...\*/

return {

<>

<Wrapper>

<AnimatePresence>

{isLoading? <div style = {{color:"black"}}> loading...</div> //(4) 서버에서 데이터를 받아오고 있는 중일 때 "로딩 상태"로 표시

### Mylist.tsx (마이 페이지 → 나의 후기 보기)의 코드(1/2)

(1) 로그인 한 userID값 가져오기

(2) 내가 저장한 후기 목록 데이터를 서버로 부터 가져오기

API 호출이 성공적으로 이뤄졌을 경우, 반환값으로 data를 받음

(2.1) 서버로부터 받아온 데이터를 reviewData 변수로 저장

(3) 내가 저장한 후기 목록의 "상세보기" 데이터를 서버로부터 가져옴

후기 번호를 서버 url에 넣어서 요청하면, 해당 후기의  
"내용"과 "map"을 반환해주어 상세정보를 확인할 수 있게 됨

(3.1) 서버로부터 받아온 데이터를 infoData 변수로 저장

(4) 서버에서 데이터를 받아오고 있는 중일 때 "로딩 상태"로 표시

## 주요 코드 설명\_ 마이 페이지 → 나의 후기 보기 페이지(프론트엔드 ①)

(2/2)

## Mylist.tsx (마이 페이지 → 나의 후기 보기)의 코드(2/2)

(5) 서버에서 받아온 "후기 목록"을 하나씩 꺼내어  
review 데이터에 저장 후, 화면에 보여 주는 코드

### (5.1) 하나의 "후기"를 클릭 했을 때 해당 후기 id를 저장

(6) 후기 목록 중 하나의 후기를 클릭했을 때,  
상세정보(후기 내용, map)를 보여주는 모달창 생성

(6.1) 클릭한 해당 후기 id의 상세정보를 서버에 요청하여 데이터로 받아와서 화면에 보여줌

## map 보여주는 코드후기 내용 보여주는 코드

## 후기 내용 보여주는 코드

```
< >  
    <ul> //(5) 서버에서 받아온 "후기목록"을 하나씩 꺼내어 review 데이터에 저장 후, 화면에 보여 주는 코드  
        {reviewData?.map((review) => (  
            <li>  
                <Box  
                    layoutId={review.review_id+""}  
                    key={review.review_id}  
                    onClick = {{}}=> // (5-1) 하나의 "후기"를 클릭 했을 때 해당 후기 id를 저장  
                        {onBoxClicked(review.review_id+"")  
                            setId(review.review_id+"")}  
                        }  
                >  
                    <h1> {review.startPlace} <Strong>{rarr}</Strong> {review.endPlace} </h1>  
                </Box>  
            </li>  
        )})</ul>  
    </AnimatePresence>  
    <AnimatePresence>  
    {bigRoadMatch ? ( // (6) 후기 목록 중 하나의 후기를 클릭했을때, 상세정보(후기 내용, map)을 보여주는 모달창 생성  
        <>  
            /* 길여서 중간 코드 생략 ....*/  
            {  
                clickedBoxOne && //(6-1) 클릭한 해당 후기 id의 상세정보를 서버에 요청하여 데이터로 받아와서 화면에 보여줌  
                <>  
                    <div >  
                        <iframe title="Naver Map" src= {infoData?.mapUrl} ></Iframe> //map 보여주는 코드  
                    </div>  
                    <fontBox>  
                        <Title > 후기 </Title>  
                        <H1>{infoData?.content}</H1> //후기 내용 보여주는 코드  
                    </FontBox>  
                </>  
            }  
            /* 길여서 중간 코드 생략 ....*/  
        );  
    }
```

# 05 주요 코드 설명\_ 로그인 페이지(프론트엔드 ②)

다음 장에  
계속 (1/2)

## Login.tsx (로그인 페이지)의 코드(1/2)

hook을 사용하여 폼의 상태와 유효성 검사를 관리

이메일 입력 받기

폼에 값이 들어가 있지 않은 경우에 undefined,  
email이 입력 되었지만 양식이 올바르지 않은 경우 aria-invalid가 true,  
email 입력 되었고 양식이 올바른 경우 aria-invalid가 false로 값을 부여

폼이 제출될 때 규칙들을 검사하여 에러 메시지를 표시

이메일 형식 지정

email에 대한 오류 메시지를 조건에 맞게 렌더링,  
이메일 입력에 대한 오류가 있는 경우 오류 메시지가 표시,  
오류가 없는 경우 (또는 아직 값이 들어오지 않았을 경우) 오류 메시지를  
표시하지 않음

```
function Login({
  onSubmit = (data: FormData) => {
    return new Promise<void>((resolve) => {
      setTimeout(() => {
        alert(JSON.stringify(data));
        resolve();
      }, 1000);
    });
  },
}) {
  const {
    register,
    handleSubmit,
    formState: { isSubmitting, isDirty, errors },
  } = useForm<FormData>(); // hook을 사용하여 폼의 상태와 유효성 검사를 관리

  return (
    <Wrapper>
      <Header>
        <Title> 로그인 </Title>
      </Header>
      <Box>
        <Container onSubmit={handleSubmit(onSubmit)}>
          <P>
            <Label htmlFor="email" style={{paddingRight:10}}>이메일</Label>
            <Input // 이메일 입력 받기
              id="email"
              type="text"
              placeholder="test@email.com"
              aria-invalid={!isDirty ? undefined : errors.email ? "true" : "false"}
            />
            {...register("email", { // 폼이 제출될 때 규칙들을 검사하여 에러 메시지를 표시
              required: "이메일은 필수 입력입니다.",
              pattern: {
                value: /^[S+@S+\.S+/, // 이메일 형식 지정
                message: "이메일 형식에 맞지 않습니다.",
              },
            })}
          </P>
          {errors.email && <Span>{errors.email?.message}</Span>}
        </Container>
      </Box>
    </Wrapper>
  );
}
```

## Login.tsx (로그인 페이지)의 코드(2/2)

```

<P>
<Label htmlFor="password">비밀번호</Label>
<Input
  id="password"
  type="password"
  placeholder="*****"
  aria-invalid={
    !isDirty ? undefined : errors.password ? "true" : "false"
  }
/>

{...register("password", { // 폼이 제출될 때 규칙들을 검사하여 에러 메시지를 표시
  required: "      비밀번호는 필수 입력입니다.",
  minLength: {
    value: 8, // 비밀번호 8자리 이상 받도록 설정
    message: "      8자리 이상 비밀번호를 사용하세요.",
  },
})},
)}}
/>

</P>
{errors.password && (
  <Span>{errors.password?.message}</Span>)}
<Button type="submit" disabled={isSubmitting}>
  로그인
</Button>
</Container>
</Box>
</Wrapper>
);
}

export default Login;

```

폼에 값이 들어가 있지 않은 경우에 undefined,  
password가 입력 되었지만 양식이 올바르지 않은 경우 aria-invalid가 true,  
password가 입력 되었고 양식이 올바른 경우 aria-invalid가 false로 값을 부여

폼이 제출될 때 규칙들을 검사하여 에러 메시지를 표시

비밀번호 8자리 이상 받도록 설정

password에 대한 오류 메시지를 조건에 맞게 렌더링,  
이메일 입력에 대한 오류가 있는 경우 오류 메시지가 표시,  
오류가 없는 경우 (또는 아직 값이 들어오지 않았을 경우) 오류 메시지를  
표시하지 않음

폼 제출 버튼, isSubmitting 상태에 따라 폼이 제출 중일 때 버튼을  
클릭할 수 없도록 처리

# 05 주요 코드 설명\_회원가입 API (백엔드 ①)

```
//회원가입
public class MemberService {

    private final MemberRepository memberRepository;

    //join메서드에서 password_check() 실행하여 비밀번호와 비밀번호 확인이 일치하는지 검증 후, DB에 저장
    public Long join(Member member){
        password_check(member.getPassword(), member.getPassword2());
        memberRepository.save(member);
        return member.getMember_id();
    }

    //비밀번호와 비밀번호 확인이 일치하는지 검증하는 메소드
    private void password_check(String pwd, String pwd2) {
        if(!pwd.equals(pwd2)){
            throw new IllegalStateException("비밀번호가 일치하지 않습니다.");
        }
    }
}
```

**MemberService.java** (회원가입)에 대한 코드

join메서드에서 password\_check() 실행하여  
비밀번호와 비밀번호 확인이 일치하는지 검증 후, DB에 저장

'비밀번호'와 '비밀번호 확인'이 일치하는지 검증하는 메소드

```
public class MemberRepository {

    @PersistenceContext
    private EntityManager em;

    //DB에 저장하고자 하는 회원 객체의 DB에서의 존재 유무에 따라 다르게 저장
    public void save(Member member){
        if(member.getMember_id() == null){
            em.persist(member);
        } else { //이미 DB에 등록되어 있을 경우
            em.merge(member);
        }
    }
}
```

**MemberRepository.java** (회원가입)에 대한 코드

DB에 저장하고자 하는 회원 객체가 DB에서의 이미 존재하는 경우와  
존재하지않는 경우에 따라 다르게 저장

이미 DB에 등록되어 있을 경우

# 05 주요 코드 설명\_작성목록 조회API,해시태그 필터링 API (백엔드 ①)

//작성목록 조회

```
public class ReviewRepository {  
  
    private final JdbcTemplate jdbcTemplate;  
  
    @PersistenceContext  
    private EntityManager entityManager;  
  
    public List<Review> findMemberReviews(Long memberId) {  
        return entityManager.createQuery("select m from Review m where m.member.member_id = :memberId", Review.class)  
            .setParameter("memberId", memberId)  
            .getResultList();  
    }  
}
```

**ReviewRepository.java** (작성목록 조회에 대한 코드)

전달받은 memberId와  
DB의 Review테이블의 member\_id 컬럼 값이 같은  
리뷰들을 선택하는 JPQL쿼리 사용  
  
=>리스트 형태로 반환

//해시태그 필터링

```
public class ReviewRepository {  
  
    @PersistenceContext  
    private EntityManager entityManager;  
  
    public List<Review> findHashtagReviews(String hashtag) {  
        return entityManager.createQuery("select m from Review m where m.hashtag = :hashtag", Review.class)  
            .setParameter("hashtag", hashtag)  
            .getResultList();  
    }  
}
```

**ReviewRepository.java** (해시태그 필터링에 대한 코드)

전달받은 hashtag와  
DB의 Review테이블의 hashtag 칼럼의 값이 같은  
리뷰들을 선택하는 JPQL쿼리 사용  
  
=>리스트 형태로 반환

## 05

## 주요 코드 설명\_후기 상세 조회API (백엔드 ②)

다음 장에  
계속 (1/2)

```
/**후기 게시물 상세 조회*/
//ReviewController
```

```
@GetMapping("/detail/{review_id}")
public ReviewDetailDTO reviewDetailController(@PathVariable Long review_id, Model model) {
    String mapUrl;

    //Html에 경로 좌표를 전달하기 위해 API를 호출
    WebClient webClient = WebClient.create();
    String apiUrl = "https://port-0-baco-server-eg4e2alkhufq9d.sel4.cloudtype.app/map";
    UriComponentsBuilder uriBuilder = UriComponentsBuilder.fromUriString(apiUrl)
        .queryParams("review_id", review_id);

    String mapTest = webClient.get()
        .uri(uriBuilder.toUriString())
        .retrieve()
        .bodyToMono(String.class)
        .block();
```

ReviewController.java (후기 상세조회에 대한 코드)

후기 상세 조회API 호출 시 ReviewController의  
reviewDetailController메서드 실행

지도에 경로를 표시하기 위해 MapController 호출

```
//MapController
@GetMapping("/map")
public String showMapPage(@RequestParam String review_id, Model model) {
    Long review_idLong = Long.parseLong(review_id);
    String jsonData = reviewService.getJsonData(review_idLong);
    if (jsonData != null) {
        //jsonData를 파싱해서 이중배열형태로 다시 만들기 위해 service 호출
        Double[][] jsonDataArray = reviewService.makeArray(jsonData);

        //jsonData를 html에 렌더링하기 위해 Thymeleaf 템플릿으로 전달. (=Thymeleaf를 통해 html로 전달)
        model.addAttribute("jsonData", jsonDataArray);
    }
    return "map"; } // HTML 파일 이름 (확장자 제외)을 리턴
}
```

MapController.java (후기 상세조회-경로 표시에 대한 코드)

전달받은 review\_id가 속한 레코드에 저장된  
String 형태의 경로좌표 데이터를 가져옴.String 형태의 경로좌표 데이터를  
Double 타입 이중배열로 만들어내는 과정.지도 API를 활용해 Web Dynamic Map을 나타내는  
map.html로 Thymeleaf 템플릿 엔진을 통해  
이중배열을 넘겨서 경로를 나타냄.



# 05 주요 코드 설명\_후기 상세 조회API (백엔드 ②)

(2/2)

```
@GetMapping("/detail/{review_id}")
public ReviewDetailDTO reviewDetailController(@PathVariable Long review_id, Model model) {
    //이전 코드 생략
    //html에 경로 표시하기 성공하면 지도 경로가 표시되는 html로 접속할 수 있도록 mapUrl 전달
    mapUrl = "https://port-0-baco-server-eg4e2alkhufq9d.sel4.cloudtype.app/map?review_id=" + review_id;
    ReviewDetailDTO reviewDetail = reviewService.reviewDetail(review_id, mapUrl);
    return reviewDetail;
    //... 이후 생략...
```

**ReviewController.java** (후기 상세조회에 대한 코드)

경로좌표가 표시된 map.html의 url과 review\_id를  
ReviewService의 reviewDetail 메서드로 넘김.

```
//후기 상세조회
public ReviewDetailDTO reviewDetail(Long review_id, String mapUrl) {
    ReviewDetailDTO reviewDetailDTO = new ReviewDetailDTO();
    //review_id에 해당하는 저장값 가져오도록 repository 호출
    try {
        Optional<Review> reviewEntity = reviewRepository.detailReview(review_id);

        // Optional이기 때문에 null일 수도 있음.
        if (reviewEntity.isPresent()) {
            Review review = reviewEntity.get();
            String content = review.getContent();

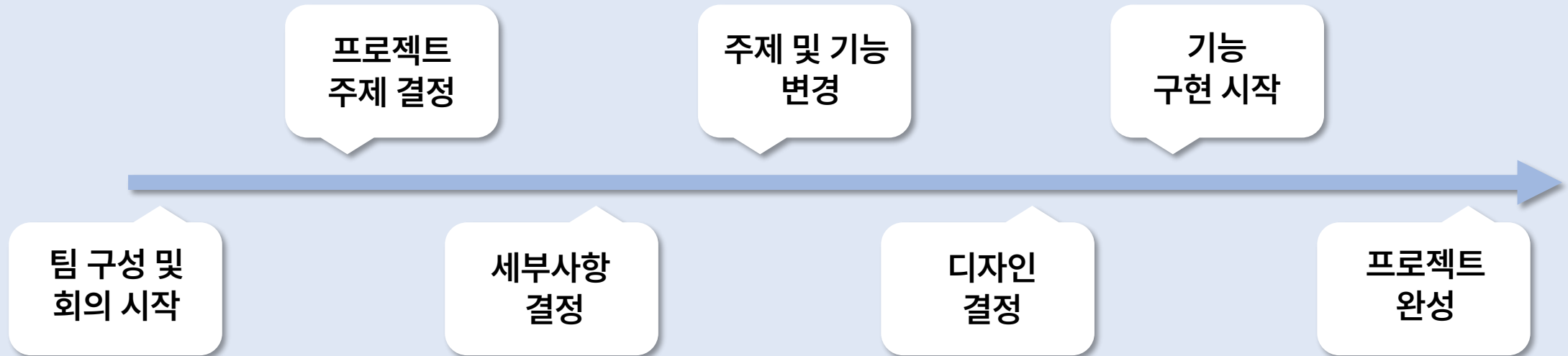
            reviewDetailDTO.setContent(content);
            reviewDetailDTO.setMapUrl(mapUrl);
        }
        // ...이후 생략...
        return reviewDetailDTO;
    }
}
```

**ReviewService.java** (후기 상세조회에 대한 코드)

review\_id를 통해 조회하고자 하는 후기에 해당하는 레코드를  
ReviewRepository를 통해 DB에서 불러옴.

조회하고자 하는 게시글의 정보가 담긴 레코드를  
ReviewRepository를 통해 가져온 뒤, API 반환 값으로 넘길  
데이터를 모아서 객체형태로 넘김.

# 06 회의 진행 흐름



## 07 아쉬웠던 점

경로가 표시되는 기능 이외에 이동거리 및 이동시간도 같이 표시되도록 했다면 좋았을 것 같다

후기 공유게시판이지만 후기 확인 이외에 댓글이나 후기 사진 첨부 등 부가적인 기능도 있었으면 좋았을 것 같다

후기에 대한 추천 기능을 통해 해시태그 필터링 목록 내에서 추천 수에 따른 정렬도 가능했다면 좋았을 것 같다

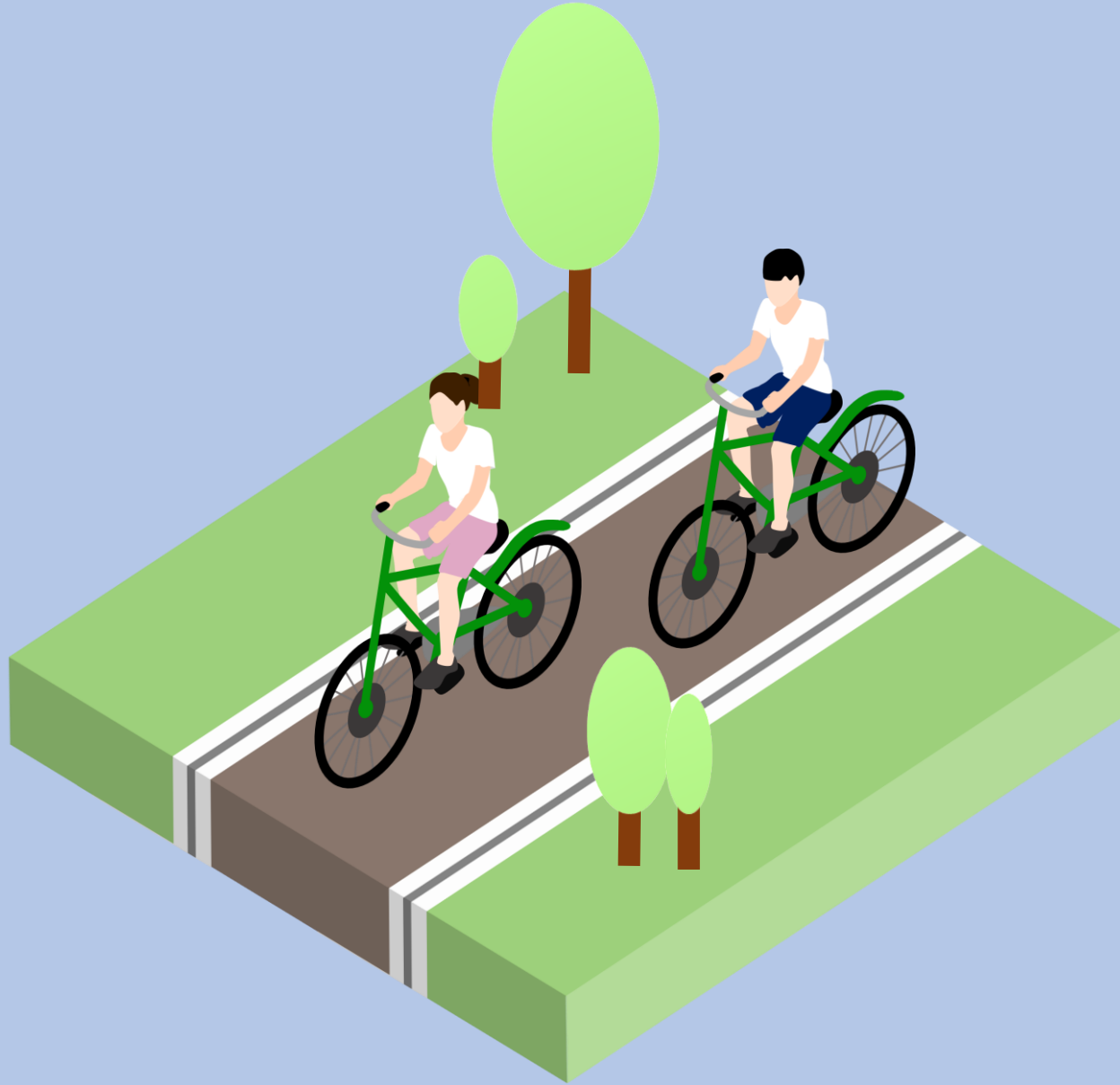
기존에 하고자 했던 경로추천에 대한 부분도 직접 구현이 가능했다면 좋았을 것 같다

날씨같은 조건에 따라 자전거 경로를 추천해주는 기능도 있었다면 좋았을 것 같다

경사도처럼 실제로 자전거를 탈 때의 난이도도 반영할 수 있었다면 좋았을 것 같다

# 08 시연 링크





# Q&A

# 감사합니다.