간편한 여행 문화를 만들다

좋은 날

바닐라라떼
김시아 박승재 신혜수 임준형

좋은 날

Contents


팀장 김시아
front-end
팀원 박승재
back-end
팀원 신혜수
server & 데이터 관리
팀원 임준형
modeling

개요

기존 여행과 관련된 플랫폼은 단순히 인기많은 곳을 추천해주거나, 카테고리 분류 정도로만 여행지를 추천해 준다. 사용자가 여행했던 장소와 유사한 곳의 정보는 쉽게 찾기가 힘들다.

그래서 여행했던 장소의 사진만을 가지고 유사한 곳을 추천받음으로써 사용자의 장소 선택의 고민을 해결할 수 있게 될 것이다.

더불어 지역이나, 테마별 여행지를 추천받을 수도 있다. 우리의 '좋은 날' 서비스는 간단한 이용만으로 여러 가지 여행지를 추천하여 사용자의 고민을 해결할 수 있다.

좋은 날

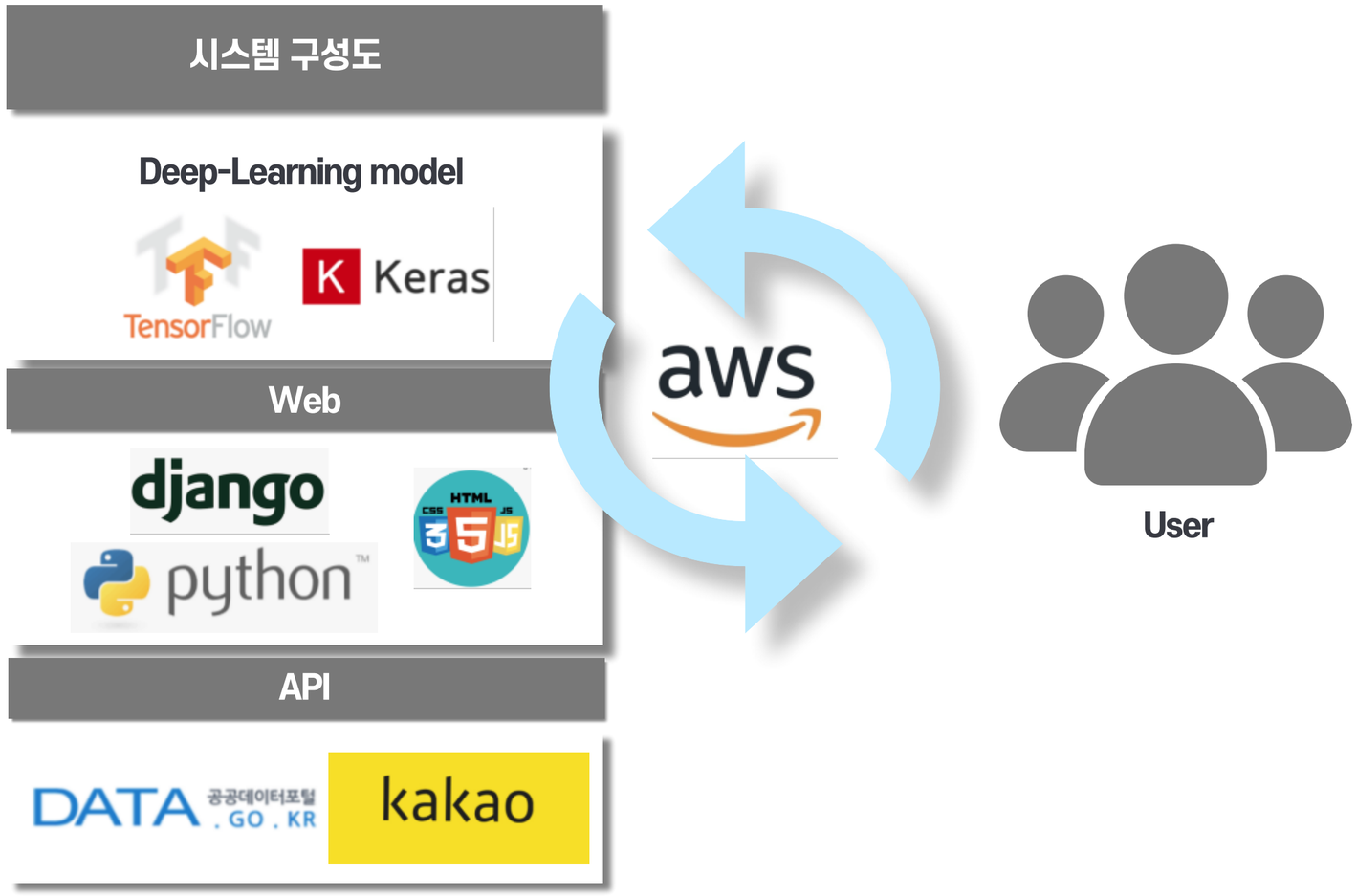
목적

관광지 추천 시스템 개발을 위한 데이터 분석

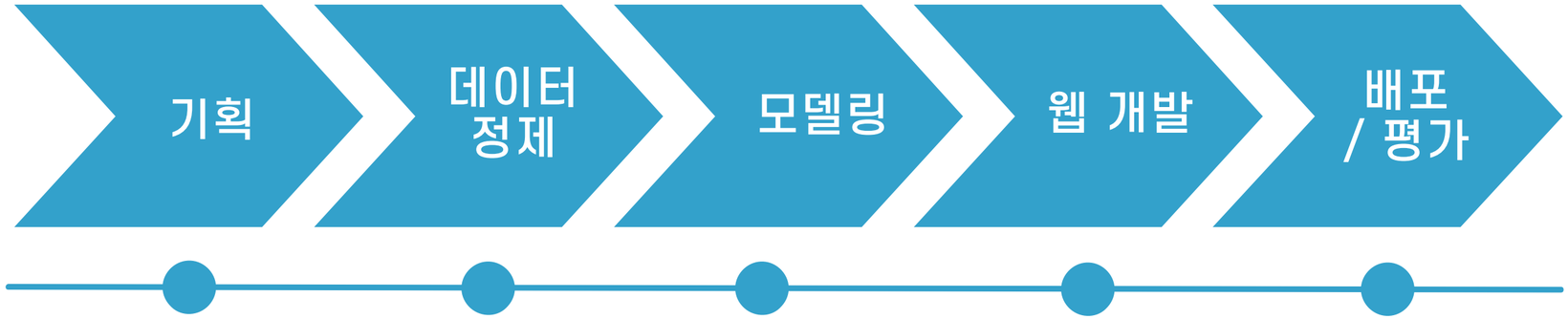
파이썬(장고) 기반의
추천 알고리즘을 사용하여 여행지 추천

장고와 오픈 api를 활용한
웹 서비스 개발

개발 환경


시스템 구성도
Deep-Learning model
TensorFlow
K Keras
Web
django
python
HTML
CSS
JS
API
DATA 공공데이터포털 .GO.KR
kakao
aws
User

프로세스



아이디어
주제 선정

여행지 추천
웹 플랫폼 기획

지역 / 카테고리
여행지 데이터 수집

AI hub
랜드마크 데이터

open API
축제
공공데이터

resnet
알고리즘 선택
후 학습

성능 검증

모바일 환경
맞춤 제작

디버깅

오류수정

자체 피드백

프로젝트 단계별 코드 및 이미지

모델링

```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
  tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
])
```

데이터 증강
매 학습마다 다른 이미지로 학습

model.summary() →

```
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
sequential (Sequential)      (None, 224, 224, 3)       0
_________________________________________________________________
rescaling (Rescaling)        (None, 224, 224, 3)       0
_________________________________________________________________
keras_layer (KerasLayer)     (None, 2048)              23564800
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 255)               522495
=================================================================
Total params: 24,087,295
Trainable params: 24,041,855
Non-trainable params: 45,440
_________________________________________________________________
```

```
Epoch 10/10
536/536 [==============================] - 105s 196ms/step - loss: 0.2348 - accuracy: 0.9333 - val_loss: 0.4431 - val_accuracy: 0.8942
```

프로젝트 단계별 코드 및 이미지

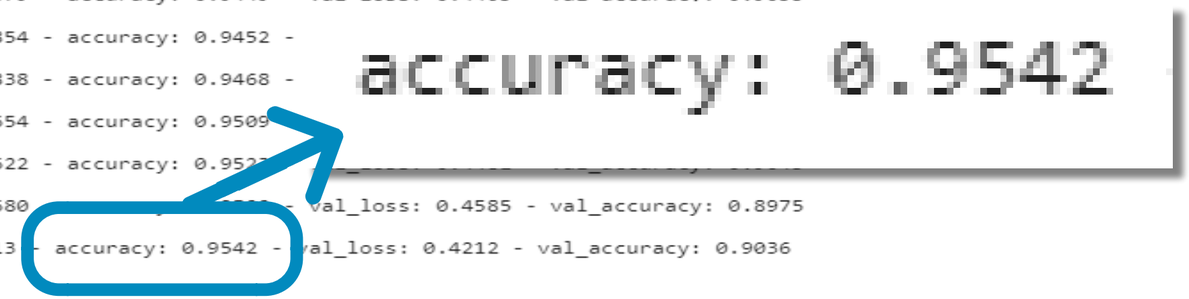
```python
# 파인 튜닝

base_model.trainable = True
fine_tune_epochs = 10
total_epochs =  initial_epochs + fine_tune_epochs

history_fine = model.fit(train_dataset,
                         epochs=total_epochs,
                         initial_epoch=history.epoch[-1],
                         validation_data=validation_dataset)
acc += history_fine.history['accuracy']
val_acc += history_fine.history['val_accuracy']

loss += history_fine.history['loss']
val_loss += history_fine.history['val_loss']
```

```
Epoch 10/20
536/536 [==============================] - 113s 211ms/step - loss: 0.2258 - accuracy: 0.9359 - val_loss: 0.4361 - val_accuracy: 0.8952
Epoch 11/20
536/536 [==============================] - 107s 200ms/step - loss: 0.2163 - accuracy: 0.9402 - val_loss: 0.4405 - val_accuracy: 0.8914
Epoch 12/20
536/536 [==============================] - 110s 205ms/step - loss: 0.2002 - accuracy: 0.9428 - val_loss: 0.4475 - val_accuracy: 0.8935
Epoch 13/20
536/536 [==============================] - 108s 201ms/step - loss: 0.1979 - accuracy: 0.9431 - val_loss: 0.4383 - val_accuracy: 0.8980
Epoch 14/20
536/536 [==============================] - 103s 191ms/step - loss: 0.1879 - accuracy: 0.9449 - val loss: 0.4403 - val accuracy: 0.9033
Epoch 15/20
536/536 [==============================] - 106s 198ms/step - loss: 0.1854 - accuracy: 0.9452 -
Epoch 16/20
536/536 [==============================] - 105s 196ms/step - loss: 0.1838 - accuracy: 0.9468 -
Epoch 17/20
536/536 [==============================] - 102s 190ms/step - loss: 0.1654 - accuracy: 0.9509
Epoch 18/20
536/536 [==============================] - 105s 196ms/step - loss: 0.1622 - accuracy: 0.952
Epoch 19/20
536/536 [==============================] - 112s 209ms/step - loss: 0.1680 -  - val_loss: 0.4585 - val_accuracy: 0.8975
Epoch 20/20
536/536 [==============================] - 97s 182ms/step - loss: 0.1513 - accuracy: 0.9542 - val_loss: 0.4212 - val_accuracy: 0.9036
```



프로젝트 단계별 코드 및 이미지

back-end

```
with open("amend_model_1.json", "r") as f :
    loaded_model_json = f.read()
model = model_from_json(loaded_model_json, custom_objects={'KerasLayer': hub.KerasLayer})
model.load_weights("amend_model_1.h5")

if request.method == "POST":
    landmark = request.FILES['photo']
    default_storage.save('image.jpg', ContentFile(landmark.read()))
    path = 'media/image.jpg'

    img = tf.keras.preprocessing.image.load_img(path, target_size=(224, 224),)
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)
    predictions = model.predict(img_array)
    score = tf.nn.softmax(predictions[0])
    total_score = np.array(score)
    total_sort_score = sorted(total_score)

    context = {
        'first': {
            'name': class_name.iloc[np.where(total_score == total_sort_score[-2])[0][0], 1],
            'address': class_name.iloc[np.where(total_score == total_sort_score[-2])[0][0], 2],
            'long': class_name.iloc[np.where(total_score == total_sort_score[-2])[0][0], 3],
            'lat': class_name.iloc[np.where(total_score == total_sort_score[-2])[0][0], 4],
            'img_url' : 'img/' + img_url_dict[class_name.iloc[np.where(total_score == total_sort
        },
        'second': {
            'name': class_name.iloc[np.where(total_score == total_sort_score[-3])[0][0], 1],
```

이미지 첨부 시 prediction 하여 첨부한 이미지와 가장 유사한 여행지 추천 코드

프로젝트 단계별 코드 및 이미지

```
reco_list =  pd.read_csv('reco_location.csv')
context = {
    'reco_list':{
    }
}
reco_list_temp = {}
if request.method == "POST":
    reco = reco_list[(reco_list['area'] == request.POST['chk_region']) & (reco_list['theme'] == request.POST['chk_theme']
    for item in reco.values:
        reco_list_temp[item[0]] = {
            'name': item[0],
            'address': item[1],
            'long': item[2],
            'lat': item[3],
            'img_url': 'img/' + item[-1]
            }
context['reco_list'] = reco_list_temp
```

여행지 추천을 눌렀을 때, 만들어 둔 csv 파일에서 filter하는 방식으로 여행지를 보여주는 코드

프로젝트 단계별 코드 및 이미지

```
url = f'http://api.visitkorea.or.kr/openapi/service/rest/KorService/searchFestival?serviceKey={key}&numOfRows=100&pageNo=1&MobileOS=ETC&Mobil
event = requests.get(url)
event = event.json()
if event['response']['body']['totalCount'] >= 2:
    for x in event['response']['body']['items']['item']:
        if 'firstimage' not in x:
            x['firstimage'] = 'https://search.pstatic.net/common/?src=http%3A%2F%2Fimgnews.naver.net%2Fimage%2F5002%2F2018%2F10%2F05%2F000107
        festival[x['title']] = x
context['festival'] = festival
```

마음에 드는 여행지를 선택하면, 공공데이터 API를 통해 그 지역의 축제를 받아오는 코드

프로젝트 단계별 코드 및 이미지

```
<button onclick="festival()">축제</button>
<button onclick="restaurant()">음식점</button>
<button onclick="cafe()">카페</button>
<button onclick="store()">편의점</button>
<button onclick="inn()">숙박</button>
<button>
<a href="https://map.kakao.com/link/to/{{map.name}},{{map.lat}},{{map.long}}" style="colo
    <i class="fas fa-route" style="font-size:30px"></i>
</a>
</button>
iv>

ipt>
$('#name').text('{{map.name}}');
$('#address').text('{{map.address}}');
function festival() {
    if ($('.map_wrap').css('display') == 'block'){
        $('.map_wrap').css('display', 'none');
        $('.festival_wrap').css('display', 'flex');
    }
    else{
        removeMarker();
        $('.map_wrap').css('display', 'block');
        $('.festival_wrap').css('display', 'none');
    }
}
function restaurant() {
    if($('.map_wrap').css('display') == 'none'){
        $('.map_wrap').css('display', 'block');
        $('.festival_wrap').css('display', 'none');
    }
    $('#FD6').trigger("click");
```

Detail 페이지에서 카테고리 구현 JS코드

```
var lat = '{{map.lat}}';
var long = '{{map.long}}';
var locPosition = new kakao.maps.LatLng(lat, long);
displayMarker(locPosition);
```

```
var marker = new kakao.maps.Marker({
    map: map,
    position: locPosition

});
map.setCenter(locPosition);
}
```

여행지 선택시, 선택된 여행지 중심 좌표로 보여주는 코드

결과


좋은 날
여행지 검색
여행지 추천

①

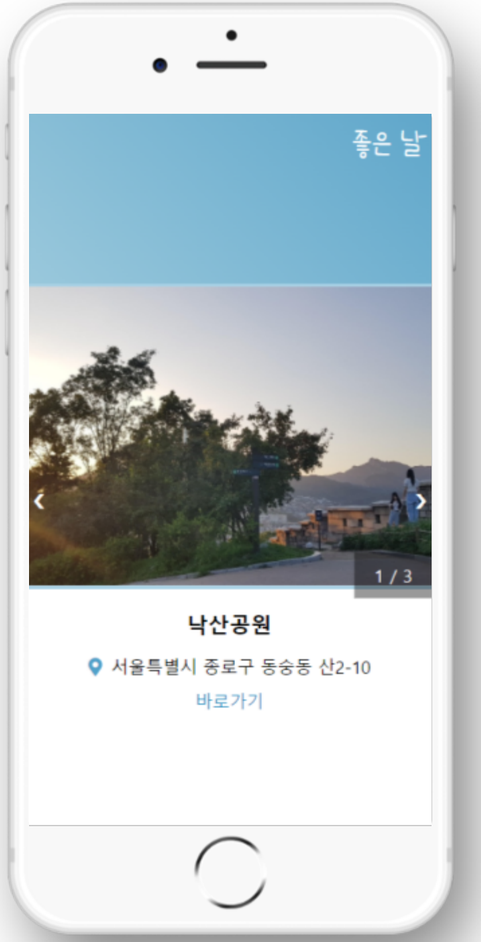

좋은 날
사진 첨부
장소 추천받기
여행지 추천

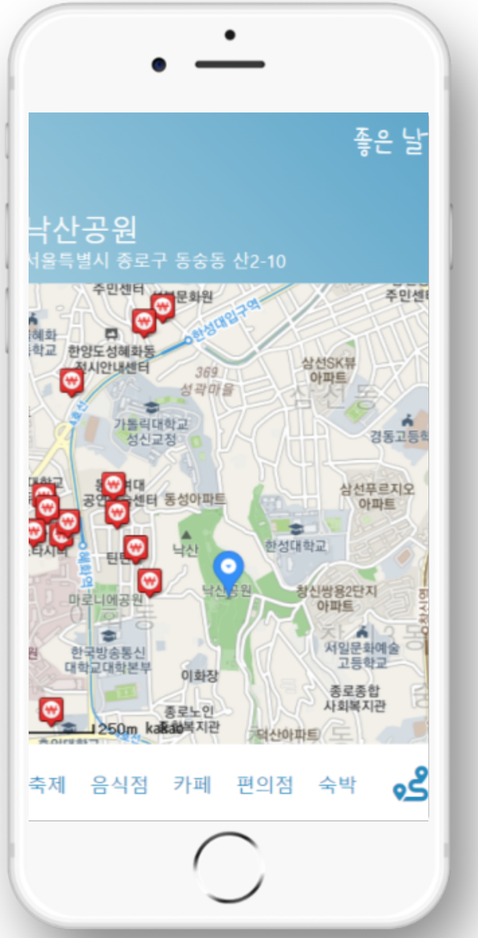
②

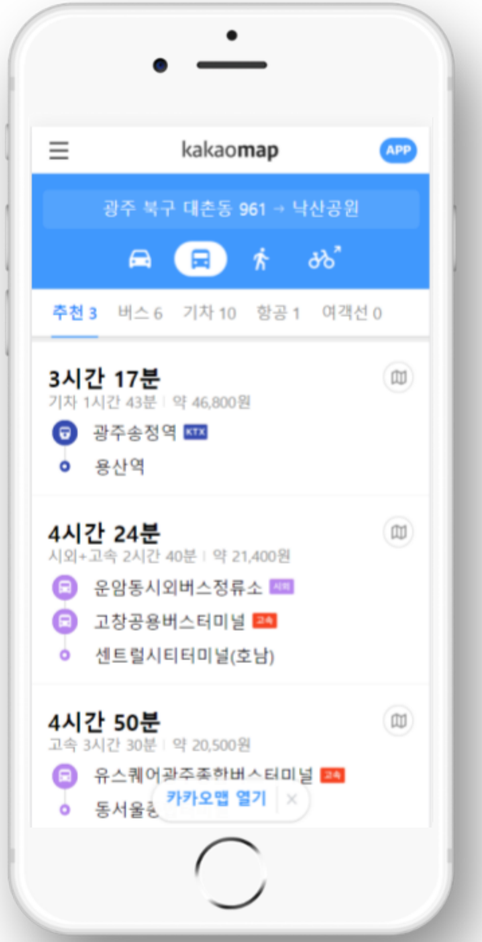

지역
#서울특별시
#부산광역시
#대구광역시
#인천광역시
#광주광역시
#대전광역시
#울산광역시
#제주도
테마
#관광지
#액티
#자연
#체험
장소 추천받기

결과

① 1번 기능 (여행지 검색) 선택시 사용자 화면

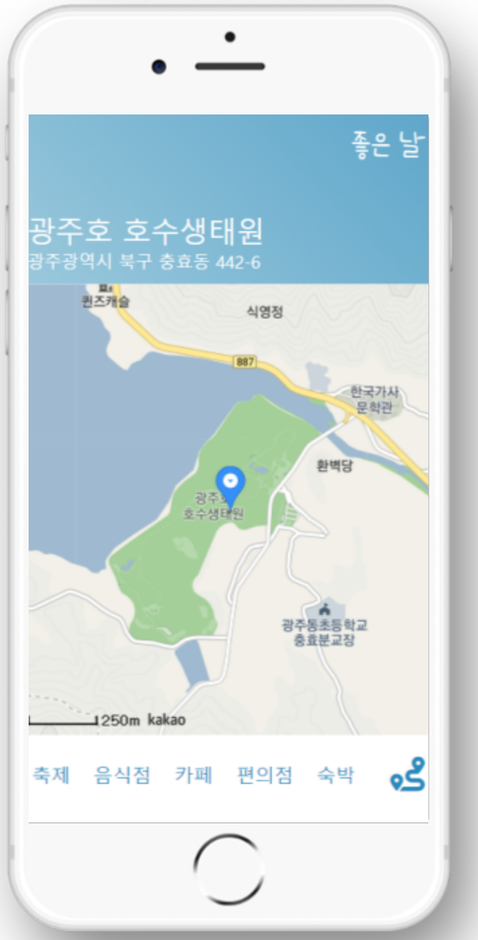


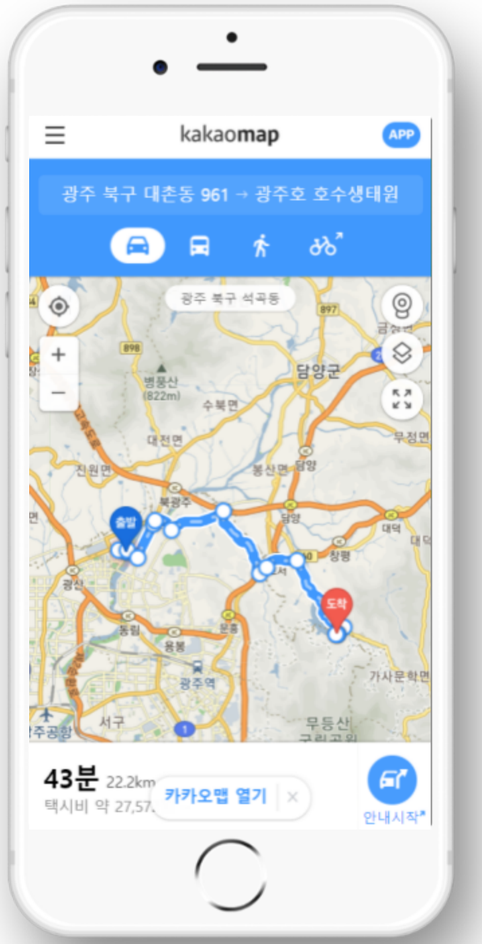




결과

② 2번 기능 (여행지 추천) 선택시 사용자 화면





좋은 날

프로젝트 자체 평가

1) 주요성과

- 모델링 : AI hub 의 지역별 랜드마크 이미지 데이터를 resnet을 이용하여 95% 정확도라는 우수한 성과를 보인것으로 평가
- web : 데이터베이스를 사용하지 않고 lite한 웹 개발
  : 다양한 api를 활용하여 완성도를 높임
  : 자바스크립트를 활용한 다양한 기능 구현
- 기존 플랫폼과의 차별성 : 기존은 텍스트로 여행지 검색 -> 이미지 검색 -> 다르지만 비슷한 곳 추천 -> 공감대 형성 기여

좋은 날

프로젝트 자체 평가

2) 피드백 및 보완점

- 반응형 웹 : 반응형 웹을 구현하고 싶었으나 시간 관계상 모바일 화면에만 맞춘 디자인
- 서버 속도 : 코드 최적화 문제
- 추천 데이터 양 : 구할 수 있는 데이터의 양의 한계와 데이터들의 일관성이 없어 추천할 수 있는 데이터의 양이 한정적임

관련 논문 및 레퍼런스

https://www.tensorflow.org/api_docs/python/tf/keras/Model

https://apis.map.kakao.com/web/documentation/

날씨 좋은 날 우리 여행갈까?

https://imhelloworld.tk/

모바일 환경을 기준으로 제작되었습니다.

감사합니다