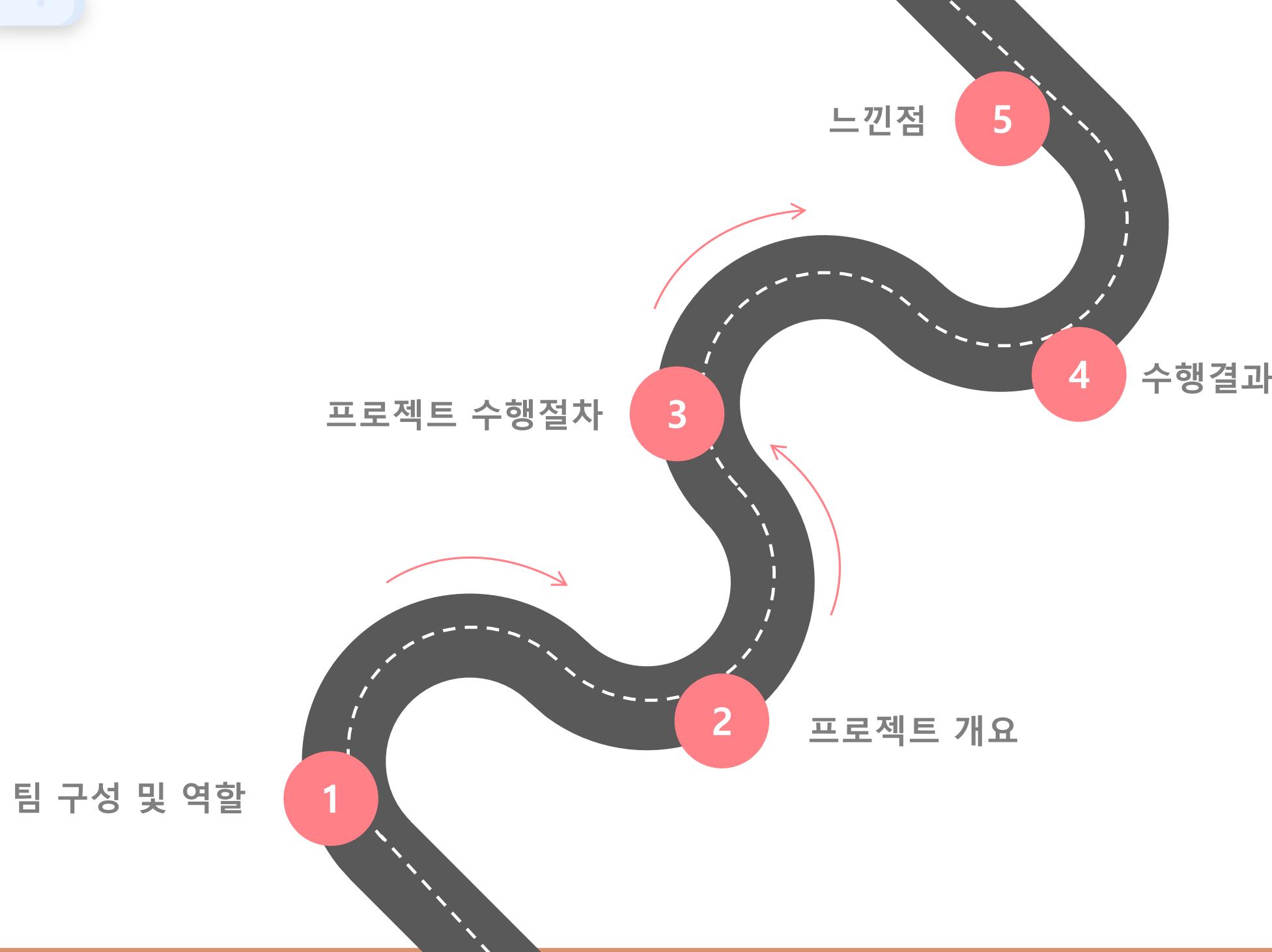




traveler

목차



1

팀 구성 및 역할

Project Teaming and Roles



1. 팀원소개



권력자

김지윤

- 선산 김씨 희귀성으로서의 품격과 신비함 담당
 - DB 개발
 - 추천 알고리즘 구현
 - PPT 제작, 발표
 - 회의 진행 및 정리
 - 프로젝트 진행 및 타임라인 관리



오은영

- 귀여움과 섹시함이 공존하는 이 시대의 마지막 팜므파탈 담당
- WEB 개발
- 아이템 기반 추천 알고리즘 구현
- PPT 제작, 발표



박현수

- 집 가서 브레이브걸스 운전만 해 들어야 하는데..
 - 프로젝트 총괄
 - 유저 기반 추천 알고리즘 구현
 - 머지 마스터
 - 코드 리펙토링 마스터



박윤수

- 담당
- 좋아요 봇 및 팀원 멘탈 관리
- 컨텐츠 기반 추천 알고리즘 구현
- 카카오 지도 API를 통해 추천 정보 웹 구현
- PPT제작

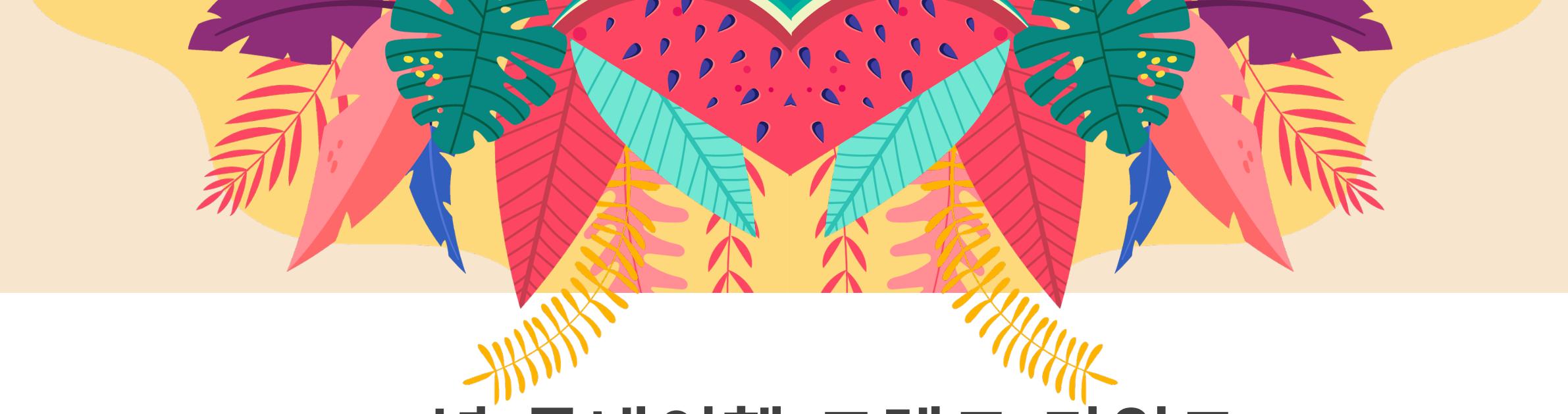


2

프로젝트 개요

Project Outline

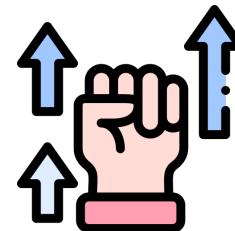




2021년 국내여행 트렌드 키워드



Break



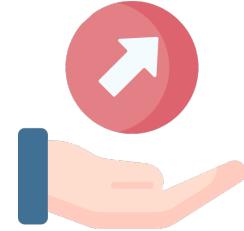
Encourage



Tie



Wherever



Enhance

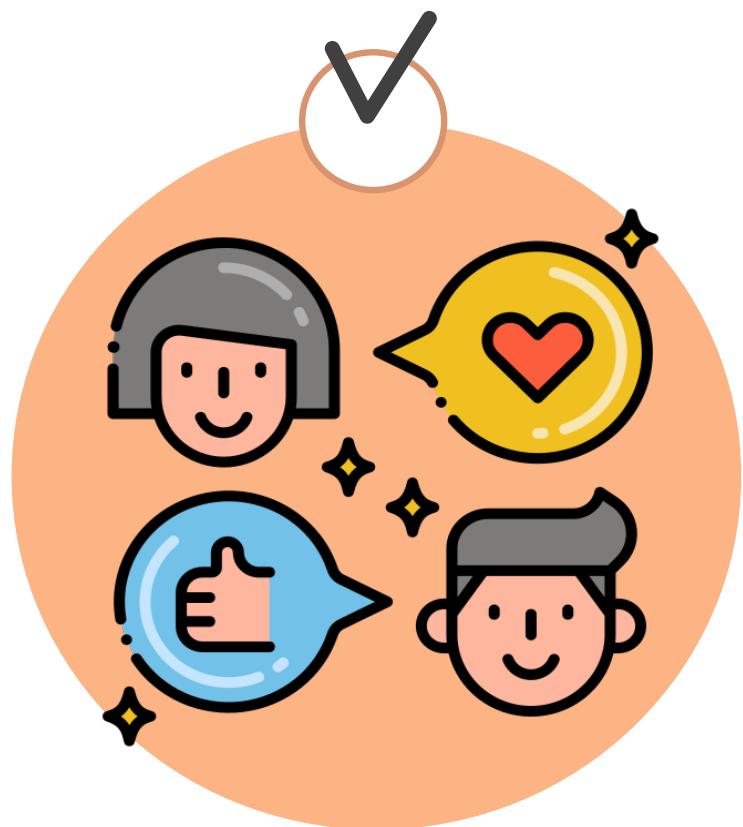


Expect

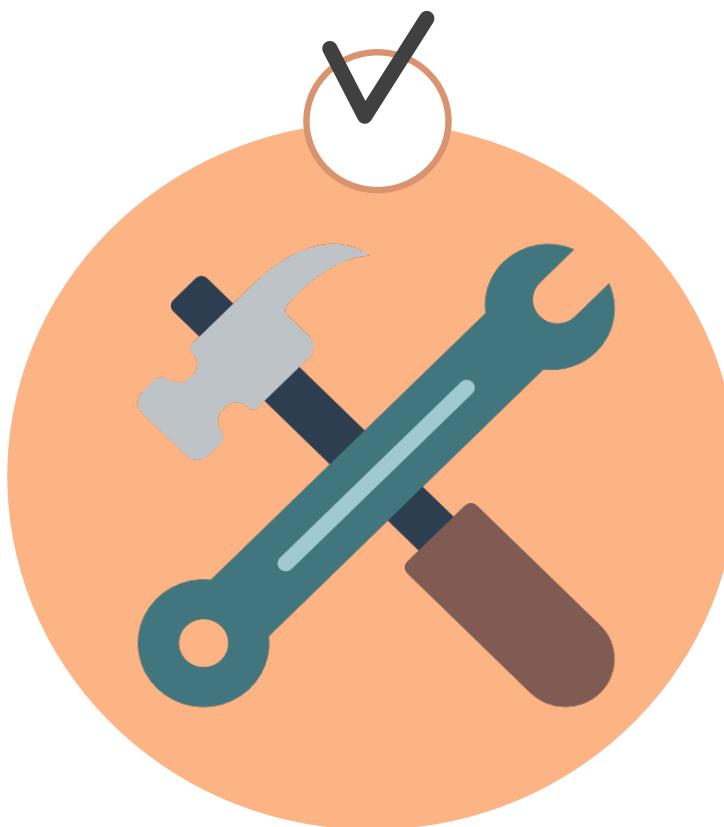


Note

프로젝트 컨셉



고객 지향성

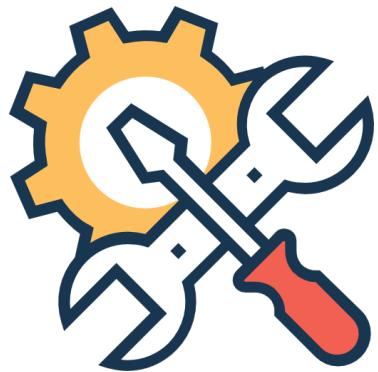


보완성



혁신성

프로젝트 개발 및 협업 환경



OS - Linux, Windows, AWS
DB - MySQL
Web Framework - Django
Language - Python, Javascript,
개발 Tool - Pycharm, MySQL, VS Code, Postman, Colab, Jupyter



git - <https://github.com/Travler-kipilGO/Recomendation-System>
소통 채널 - zoom, slack, kakao talk

사용 API

[1] 한국관광공사_TourAPI 활용 메뉴얼

<https://api.visitkorea.or.kr/openAPI/tourAPI/koreaGuide.do>

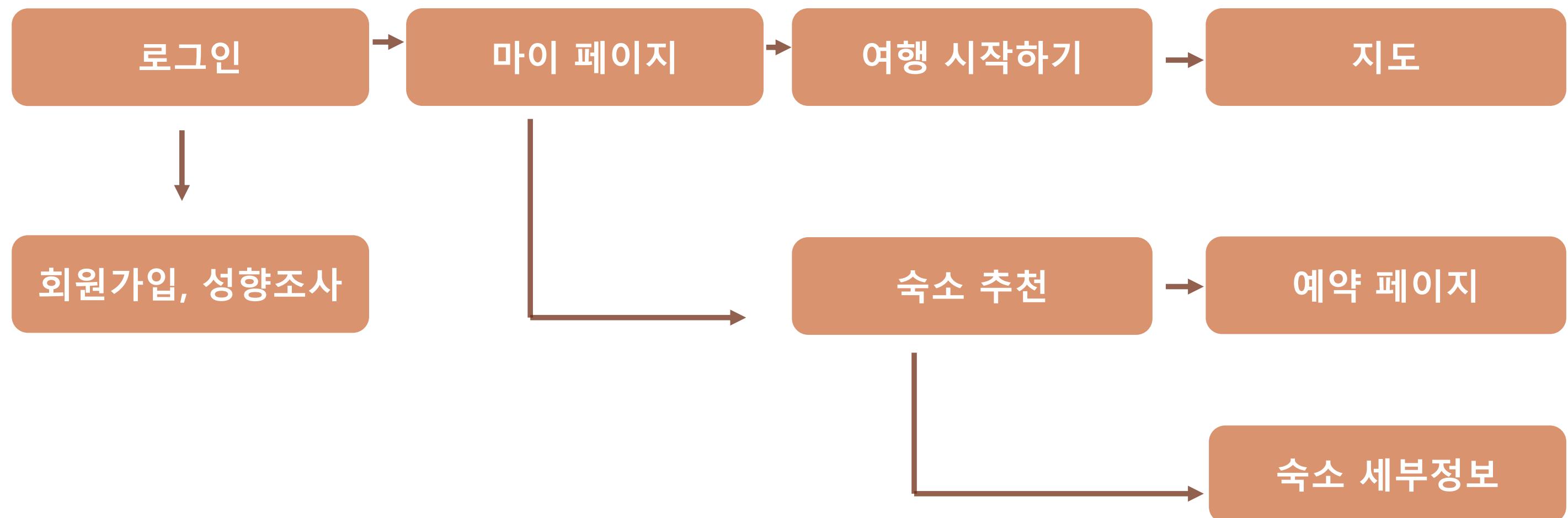
[2] 한국관광공사 여행지데이터

<https://api.visitkorea.or.kr/main.do/>

[3] 카카오 지도 API

<https://apis.map.kakao.com/>

서비스 구성도



3

프로젝트 수행 절차 및 방법

Procedures and methods for project



프로젝트 수행 절차 및 방법



4.05 ~ 4.07



4.07 ~ 4.25



4.25 ~ 4.26

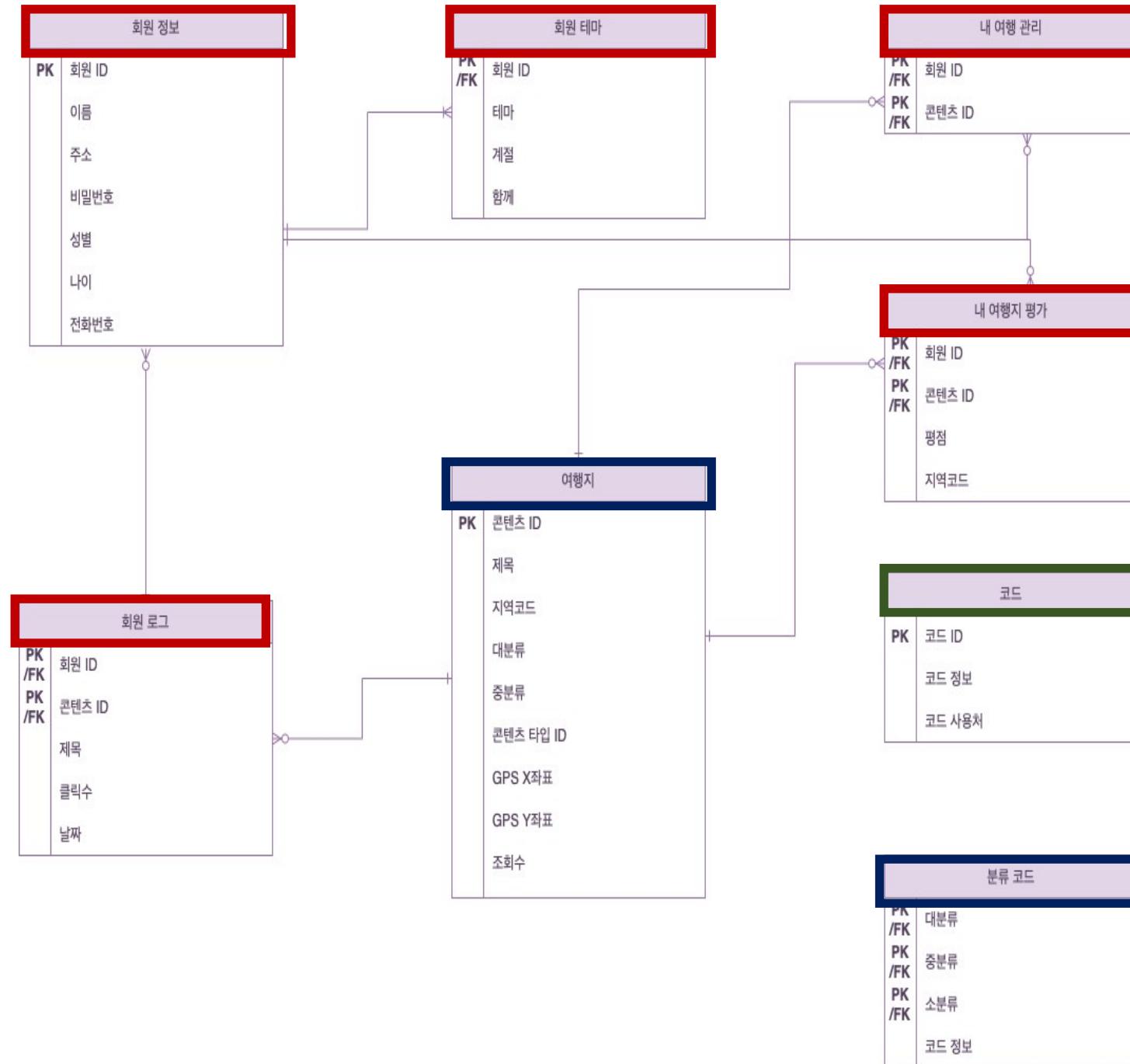
4

프로젝트 수행 결과

Result of Project

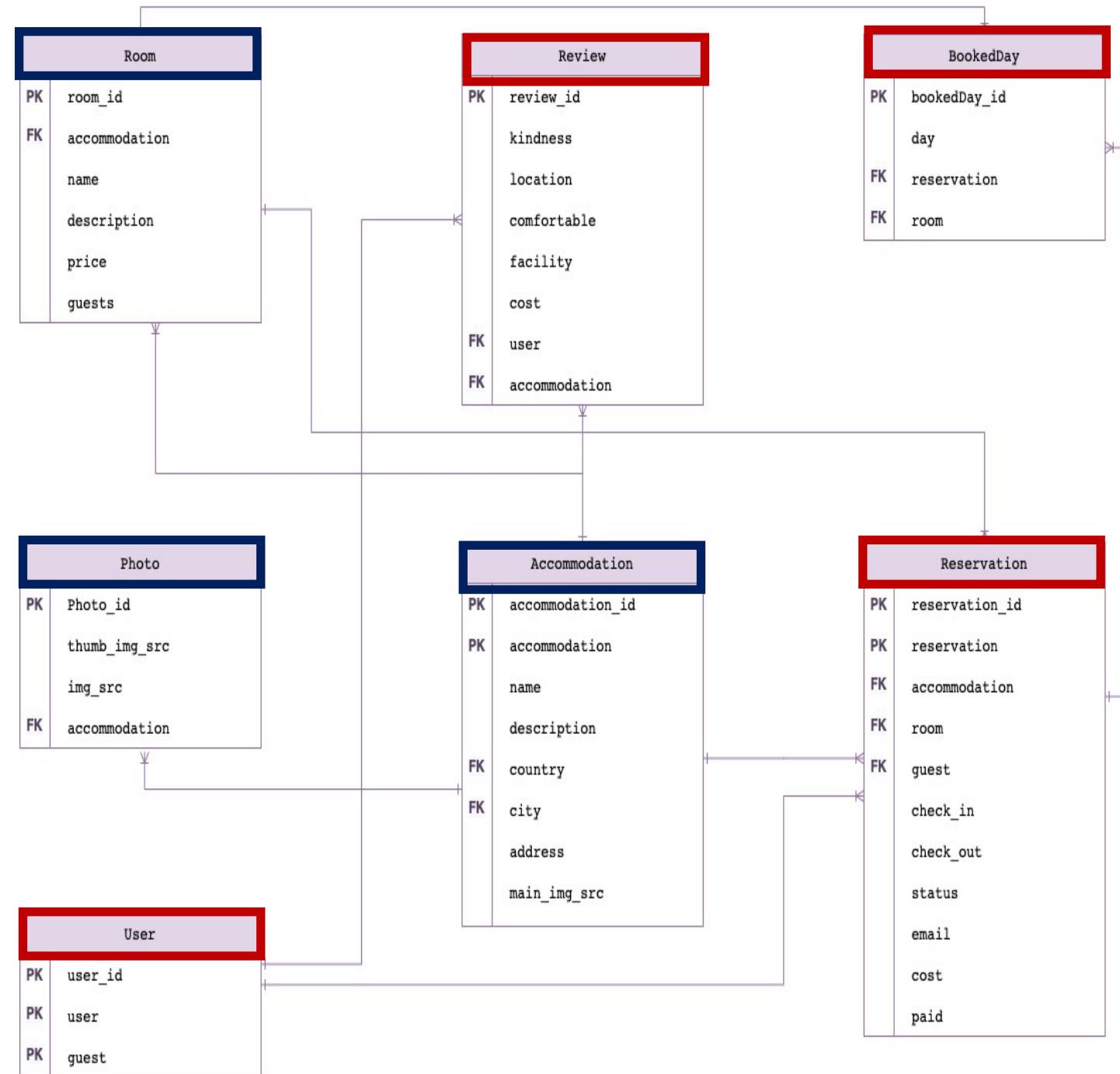


DB 모델링



- 여행지 정보
- 사용자 집계
- 코드

DB 모델링



■ 숙박 정보
■ 사용자 집계

DB 개발 – 사용자 집계 테이블



```
class MyTripData(models.Model):
    user = models.ForeignKey("users.User", on_delete=models.CASCADE)
    travelspot = models.ForeignKey('TravelSpot', on_delete=models.CASCADE)

    def __str__(self):
        return self.user.username + '-' + self.travelspot.title

class TravelRating(models.Model):
    user = models.ForeignKey("users.User", on_delete=models.CASCADE)
    mytripdata = models.ForeignKey('MyTripData', on_delete=models.CASCADE)
    rating = models.IntegerField()

    def __str__(self):
        return self.user.username + ',' + self.mytripdata.travelspot.title +',' + str(self.rating)

class UserLogData(models.Model):
    user = models.ForeignKey("users.User", on_delete=models.CASCADE)
    travelspot = models.ForeignKey('TravelSpot', on_delete=models.CASCADE)
    click = models.IntegerField()
    date = models.DateTimeField()

    def __str__(self):
        return self.user.username + ',' + self.travelspot.title +',' + str(self.click)
```

Travles/models.py

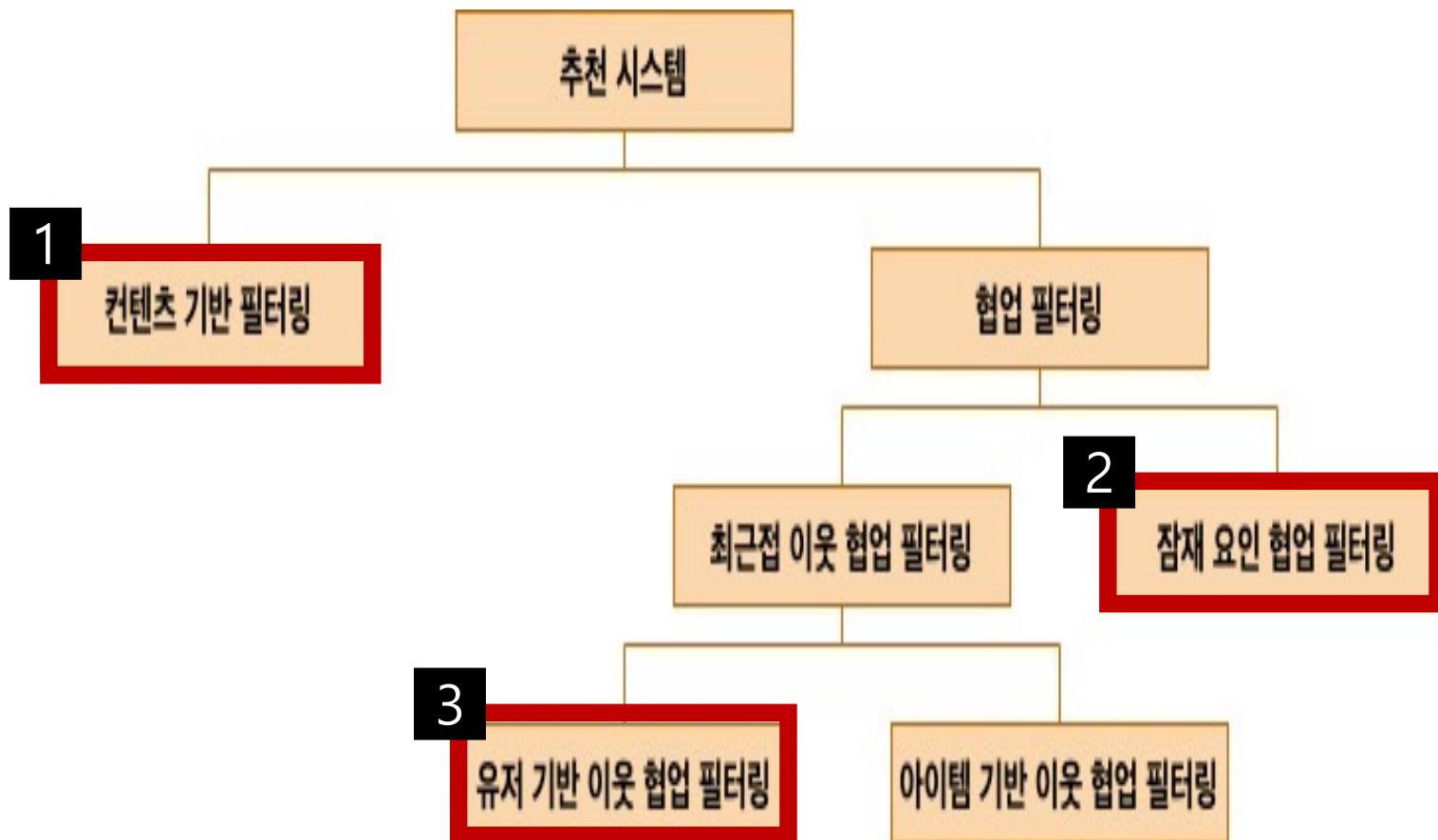
생성

user_log_data

userId	contentcode	contentdes	click	date
pulws5575@google.com	590097	이용노 미술관	1	2020-10-22
nlij3@google.com	130506	남원향토박물관	3	2020-01-12
qrkctmrn@google.com	130719	재미난 박물관	2	2020-11-22
dyzxaspk04@google.com	2611483	최재한 가옥(덕곡댁 최재한 가옥)	3	2020-07-24
fpjzuecm5544@google.com	2652951	그날우리는	2	2020-12-24
imeh939@google.com	1365460	오리마당	1	2020-08-28
blannc459@google.com	1114686	감곡매괴성모순례지성당	2	2020-04-11
lsfbaudi2@google.com	1928421	다려도	3	2020-02-10
dntj1663@google.com	131658	제일컨트리클럽	3	2020-03-12
2296@google.com	346626	용덕사(서울)	3	2020-02-09
sd41@google.com	1621219	경정해수욕장	1	2021-01-21
npqkbjy771@google.com	2678847	블랑로쉐	2	2020-11-11
uxnik4@google.com	127334	일제강점기 건축물들	1	2021-04-03
xifshzo93@google.com	129139	신안 가거도 등대	2	2021-03-27
kuxwzk92@google.com	2022638	하늘물빛정원	3	2020-12-09
yexowhe@google.com	131533	신림체육센터	3	2020-02-19
llyetl11@google.com	132285	영주장 / 신영주번개시장 (5, 10일)	2	2021-04-15
ubkimi8@google.com	126599	상동 빈지소	2	2020-12-28
memz9@google.com	628541	갤러리하우스모텔	1	2020-09-27
stmhxvw9364@google.com	2433583	경북궁 주·야간 전통공연 「고궁음악회」, 2020	2	2021-03-31
quyteyn2826@google.com	1905562	밀라노모델	1	2020-05-13
eviraxj1@google.com	402885	칠형제우리김자탕	3	2020-03-16
hvlisffpz@google.com	1013155	자갈치시장 활어부	1	2020-06-26



추천 알고리즘 개요



- 1 사용자 성향 분석 기반 컨텐츠 추천
- 2 사용자 로그 데이터 기반 컨텐츠 추천
- 3 사용자 평점 데이터 기반 컨텐츠 추천



컨텐츠 기반 필터링 - 친구추천

```
def init():
    surveys = Survey.objects.all()

    survey_df = pd.DataFrame(
        [
            [survey.user.username, survey.theme]
            for survey in surveys
        ],
        columns=['user', 'themacode']
    )
    survey_df['rating'] = 1

    theme_data = survey_df.pivot_table('rating', index='user', columns='themacode').fillna(0).astype(int)
    return theme_data.T.to_dict()
```

```
def dropna(data):
    for i in data:
        for j in data[i]:
            name=i
            theme=j
            value=data[i][j]
            data[i]={travel: value for travel, value in data[i].items() if pd.isnull(value)==False}
    return data
```

컨텐츠 기반 필터링 - 친구추천

```
def sim_pearson(data, n1, n2):
    sumX=0
    sumY=0
    sumSqX=0 # x 제곱합
    sumSqY=0 # y 제곱합
    sumXY=0 # XY 합
    global cnt # 테마 갯수
    cnt=0
    for i in data[n1]:
        if i in data[n2]:
            sumX+=data[n1][i]
            sumY+=data[n2][i]
            sumSqX+=pow(data[n1][i],2)
            sumSqY+=pow(data[n2][i],2)
            sumXY+=(data[n1][i])*(data[n2][i])
            cnt+=1
    global num
    global den
    num=sumXY-((sumX*sumY)/cnt)
    den= (sumSqX-(pow(sumX,2)/cnt))*(sumSqY-(pow(sumY,2)/cnt))
    return num/sqrt(den+0.00001) # 분모=0방지
```

μ_u 는 사용자 u 의 평균 평점

- 상품 i 와 상품 j 간의 Pearson Similarity

$$\text{pearson_sim}(i, j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \mu_j)^2}}$$

μ_i 는 상품 i 의 평균 평점

```
def top_match(data, name, rank=3, simf=sim_pearson):
    simList=[] # 유사한 테마들이 저장될 리스트
    for i in data:
        if name!=i:# 자기 자신 제외
            simList.append((simf(data, name, i), i))
    simList.sort() # 오름차순
    simList.reverse() # 내림차순
    return simList[:rank]

def main(username):
    theme_data_dic = init()
    theme_data_dic_drop = dropna(theme_data_dic)
    return top_match(theme_data_dic_drop, username, 3)
```

컨텐츠 기반 필터링 - 친구추천

```
for survey in surveys:
    themes.append(ClassificationCode.objects.get(cat2=survey.theme))
context['themes'] = themes

friendList = []
friends = friend.main(user.username)
for _, name in friends:
    friendList.append(models.User.objects.get(username=name))

context['friends'] = friendList
context['reviews'] = Review.objects.filter(user=user)

username = user.username
recdata_list = []
try:
    if username != '' and username != 'admin':
        recDatas = userlog.main(username)
        for recdata in recDatas:
            recdata_list.append(recdata[1])

    context['recdatas'] = recdata_list
except:
    context['recdatas'] = ['no recommend']
return context
```

```
class UserProfileView(DetailView):
    model = models.User
    context_object_name = "user_obj"

    def get_context_data(self, **kwargs):
        context = super(UserProfileView, self).get_context_data(**kwargs)
        user = models.User.objects.get(pk=self.kwargs['pk'])

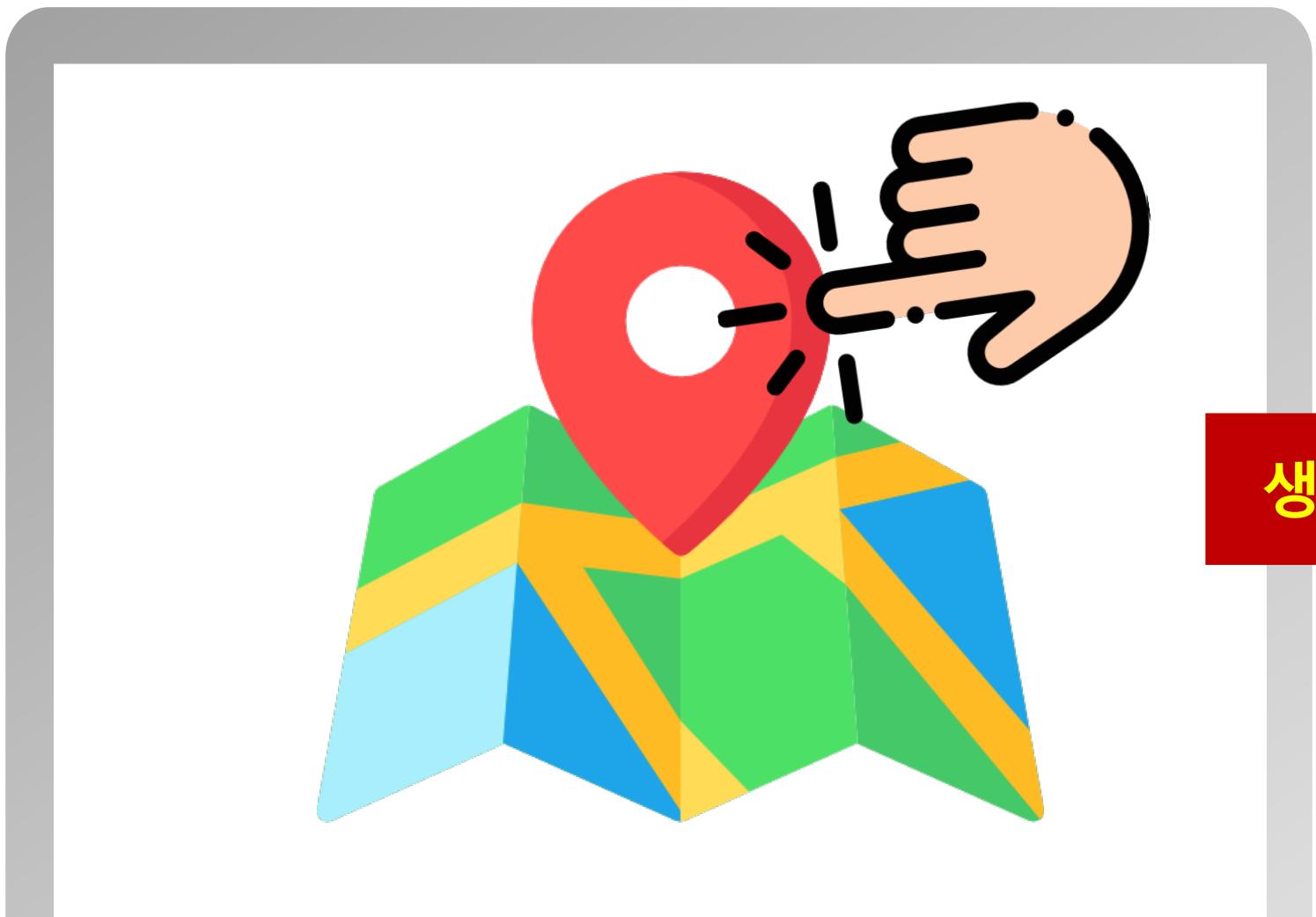
        surveys = models.Survey.objects.filter(user=user)
        themes = []

        for survey in surveys:
            themes.append(ClassificationCode.objects.get(cat2=survey.theme))
        context['themes'] = themes

        return context
```



잠재 요인 협업 필터링 - 여행지 리스트 추천



user_log_data				
userId	contentcode	contentdes	click	date
pulws5575@google.com	590097	이용노 미술관	1	2020-10-22
nlij3@google.com	130506	남원향토박물관	3	2020-01-12
qrkctmrn@google.com	130719	재미난 박물관	2	2020-11-22
dyzxaspk04@google.com	2611483	최재한 가옥(덕곡댁 최재한 가옥)	3	2020-07-24
fpjzuecm5544@google.com	2652951	그날우리는	2	2020-12-24
imeh939@google.com	1365460	오리마당	1	2020-08-28
blannc459@google.com	1114686	감곡매괴성모순례지성당	2	2020-04-11
lsfbaudi2@google.com	1928421	다려도	3	2020-02-10
dntj1663@google.com	131658	제일컨트리클럽	3	2020-03-12
iae2296@google.com	346626	용덕사(서울)	3	2020-02-09
l41@google.com	1621219	경정해수욕장	1	2021-01-21
71@google.com	2678847	블랑로쉐	2	2020-11-11
4@google.com	127334	일제강점기 건축물들	1	2021-04-03
lfshzo93@google.com	129139	신안 가거도 등대	2	2021-03-27
kuxwzk92@google.com	2022638	하늘물빛정원	3	2020-12-09
yexowhe@google.com	131533	신림체육센터	3	2020-02-19
llyetl11@google.com	132285	영주장 / 신영주번개시장 (5, 10일)	2	2021-04-15
ubkimi8@google.com	126599	상동 빈자소	2	2020-12-28
memz9@google.com	628541	갤러리하우스모텔	1	2020-09-27
stmhxvw9364@google.com	2433583	경북궁 주·야간 전통공연 「고궁음악회」2020	2	2021-03-31
quyteyn2826@google.com	1905562	밀라노모텔	1	2020-05-13
eviraxj1@google.com	402885	칠령제우리감자탕	3	2020-03-16
hvlsffpz@google.com	1013155	자갈치시장 활어부	1	2020-06-26



잠재 요인 협업 필터링 - 여행지 리스트 추천

```
def make_train (matrix, percentage = .2):
    test_set = matrix.copy()
    test_set[test_set != 0] = 1 # binary하게 만들기

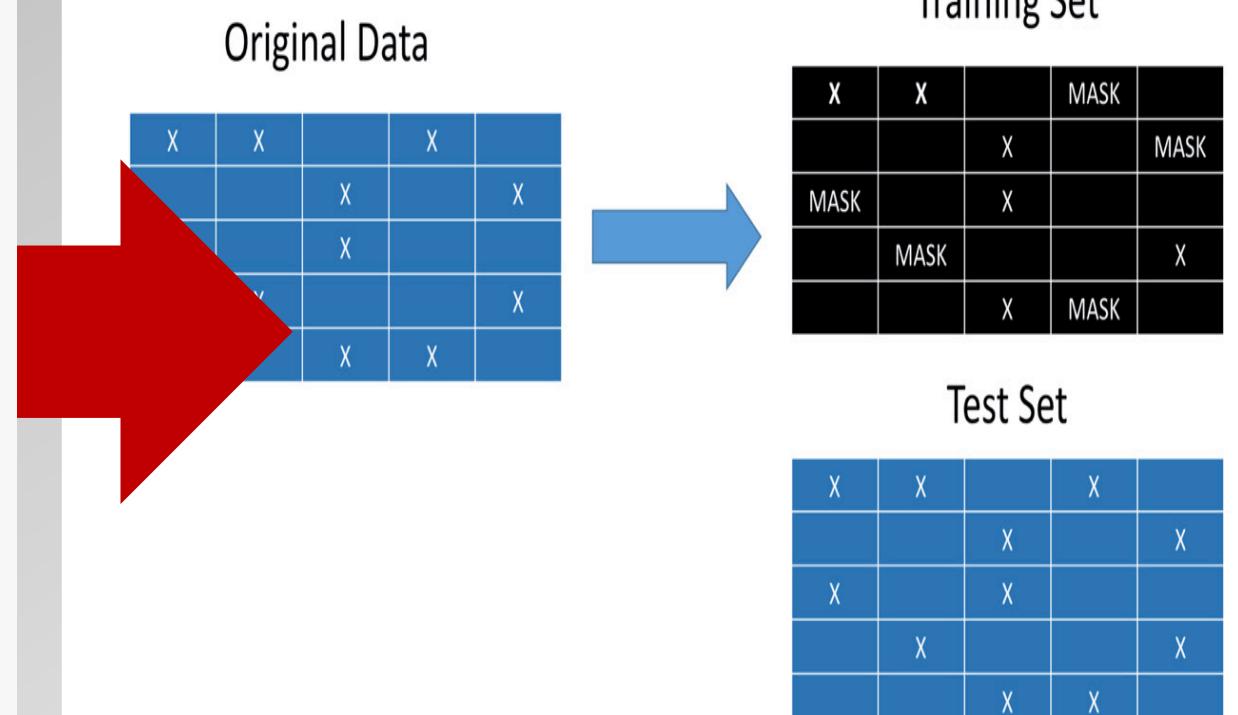
    training_set = matrix.copy()
    nonzero_inds = training_set.nonzero()
    nonzero_pairs = list(zip(nonzero_inds[0], nonzero_inds[1]))

    random.seed(0)
    num_samples = int(np.ceil(percentage * len(nonzero_pairs)))
    samples = random.sample(nonzero_pairs, num_samples)

    user_inds = [index[0] for index in samples]
    item_inds = [index[1] for index in samples]

    # eliminate_zeros(): Remove zero entries from the matrix
    training_set[user_inds, item_inds] = 0
    training_set.eliminate_zeros()

    return training_set, test_set, list(set(user_inds))
```



잠재 요인 협업 필터링 - 여행지 리스트 추천



행렬 분해 (Matrix Factorization)

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	X		X		X	
User 2		X	X			
User 3				X		X
User 4					X	
User 5	X	X		X		X
User 6			X	X		
User 7	X	X	X		X	X
User 8		X		X		
User 9			X			

R

\cong

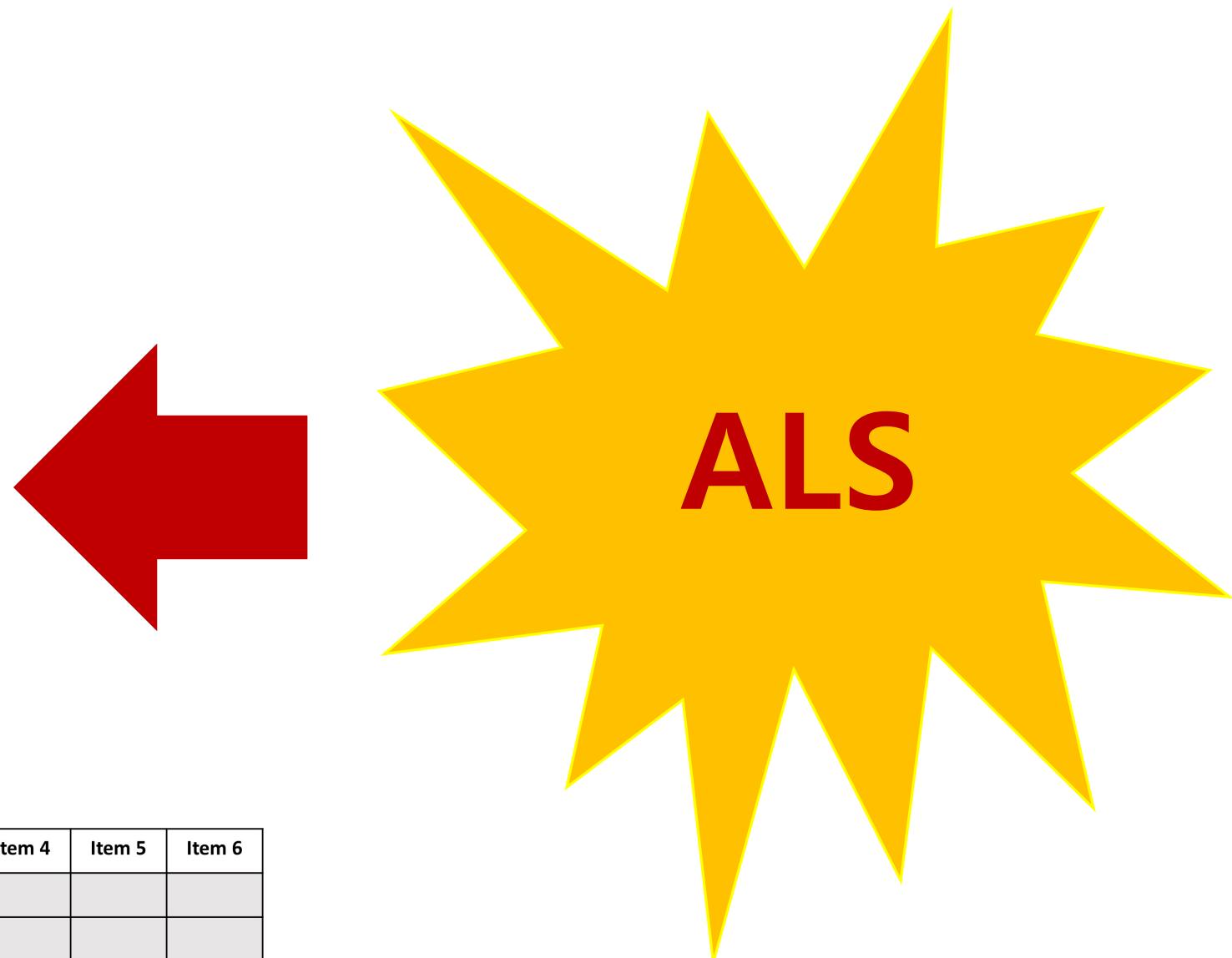
	UF1	UF2
User 1		
User 2		
User 3		
User 4		
User 5		
User 6		
User 7		
User 8		
User 9		

X

U

V

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
IF1						
IF2						





잠재 요인 협업 필터링 - 여행지 리스트 추천

$$C_{ui} = 1 + ar_{ui}$$

$$p_{ui} = x_u^\top y_i$$

$$\min \sum_{u,i} c_{ui} (p_{ui} - x_u^\top y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

$$x_u = (Y^\top C^u Y + \lambda I)^{-1} Y^\top C^u p(u)$$

$$x_u = (Y^\top Y + Y^\top (C^u - I)Y + \lambda I)^{-1} Y^\top C^u p(u)$$

$$y_i = (X^\top X + X^\top (C^i - I)X + \lambda I)^{-1} X^\top C^i p(i)$$

```
yTy = Y.T.dot(Y)
xTx = X.T.dot(X)

# Y를 고정해놓고 X에 대해 반복
# Xu = (yTy + yT(Cu-I)Y + λI)^{-1} yTCuPu
for u in range(num_user):
    conf_samp = conf[u,:].toarray() # Cu
    pref = conf_samp.copy()
    pref[pref!=0] = 1
    # Cu-I: 위에서 conf에 1을 더하지 않았으니까 I를 빼지 않음
    CuI = sparse.diags(conf_samp, [0])
    # yT(Cu-I)Y
    yTCuIY = Y.T.dot(CuI).dot(Y)
    # yTCuPu
    yTCupu = Y.T.dot(CuI+Y_eye).dot(pref.T)

    x[u] = spsolve(yTy + yTCuIY + lambda_eye, yTCupu)

# X를 고정해놓고 Y에 대해 반복
# Yi = (xTx + xT(Cu-I)X + λI)^{-1} xTCiPi
for i in range(num_item):
    conf_samp = conf[:,i].T.toarray()
    pref = conf_samp.copy()
    pref[pref!=0] = 1

    # Ci-I
    CiI = sparse.diags(conf_samp, [0])
    # xT(Ci-I)X
    xTCiIX = X.T.dot(CiI).dot(X)
    # xTCiPi
    xTCiPi = X.T.dot(CiI+ X_eye).dot(pref.T)

    y[i] = spsolve(xTx + xTCiIX + lambda_eye, xTCiPi)
```

잠재 요인 협업 필터링 - 여행지 리스트 추천



```
# 리스트 초기화
store_auc = []
popularity_auc = []

pop_items = np.array(test_set.sum(axis = 0)).reshape(-1) # 모든 유저의 아이템별 구매횟수 합
item_vecs = predictions[1] # 아이템 latent 벡터

for user in altered_users:
    training_row = training_set[user,:].toarray().reshape(-1) # 유저의 훈련데이터
    zero_inds = np.where(training_row == 0) # 가려진 아이템 Index

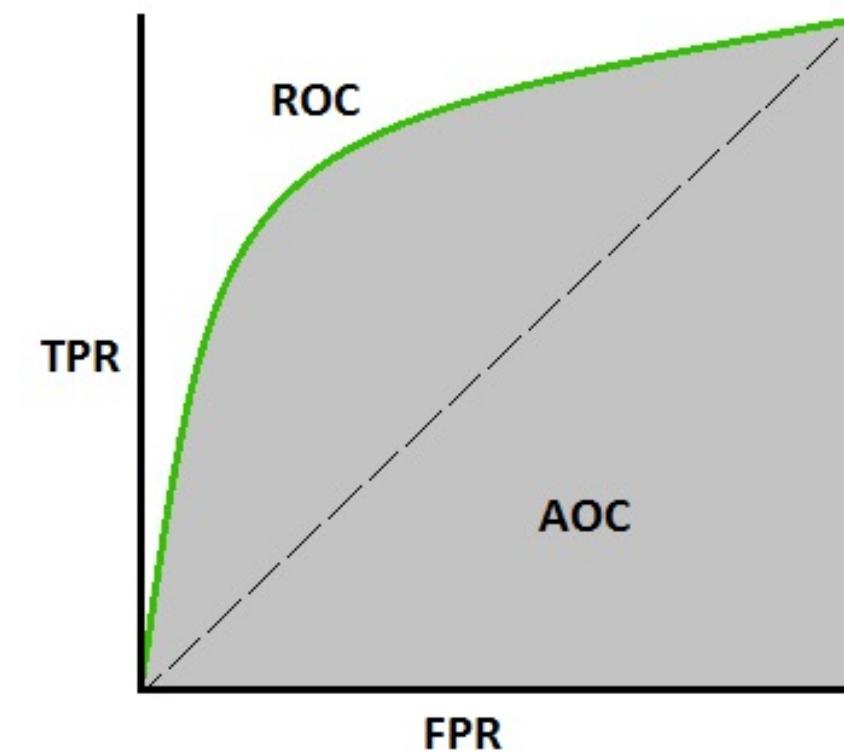
    # 가려진 아이템에 대한 예측
    user_vec = predictions[0][user,:]
    pred = user_vec.dot(item_vecs).toarray()[0,zero_inds].reshape(-1)

    # 가려진 아이템에 대한 실제값
    actual = test_set[user,:].toarray()[0,zero_inds].reshape(-1)

    # 가려진 아이템에 대한 popularity (구매횟수 합)
    pop = pop_items[zero_inds]

    # AUC 계산
    store_auc.append(auc_score(actual, pred))
    popularity_auc.append(auc_score(actual,pop))

return float('%.3f'%np.mean(store_auc)), float('%.3f'%np.mean(popularity_auc))
```





잠재 요인 협업 필터링 - 여행지 리스트 추천

```
pref_vec = product_train[0,:].toarray() # Get the ratings from the training set ratings matrix
pref_vec = pref_vec.reshape(-1) + 1 # Add 1 to everything, so that items not purchased yet become equal to 1
pref_vec[pref_vec > 1] = 0 # Make everything already purchased zero
rec_vector = user_vecs[0,:].dot(item_vecs) # Get dot product of user vector and all item vectors
rec_vector = rec_vector.toarray()

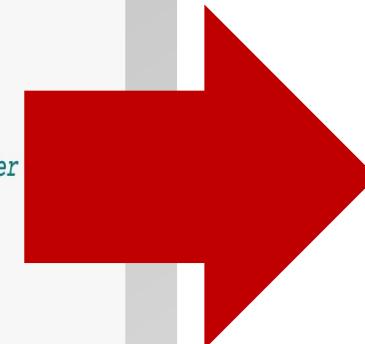
# Scale this recommendation vector between 0 and 1
min_max = MinMaxScaler()
rec_vector_scaled = min_max.fit_transform(rec_vector.reshape(-1,1))[:,0]

recommend_vector = pref_vec*rec_vector_scaled

product_idx = np.argsort(recommend_vector)[::-1][:num_items] # Sort the indices of the items into order
# of best recommendations
rec_list = [] # start empty list to store items

for index in product_idx:
    code = item_list[index]
    rec_list.append([code, item_lookup.contentdes.loc[item_lookup.contentcode == code].iloc[0]])
    # Append our descriptions to the list

codes = [item[0] for item in rec_list]
descriptions = [item[1] for item in rec_list]
final_frame = pd.DataFrame({'contentcode': codes, 'contentdes': descriptions}) # Create a dataframe
return final_frame[['contentcode', 'contentdes']] # Switch order of columns around
```



	contentcode	contentdes
0	628541	갤러리하우스모텔
1	1957963	경광서원
2	988129	거문도 신선바위
3	1979622	강원도 고성의 송지호 캠핑장
4	138920	거제하와이 콘도비치호텔
5	129407	가평 영양잣마을
6	2660518	강정희 간장새우
7	126288	격포해변
8	1619630	거점포해변
9	403194	갯마을풍천장어

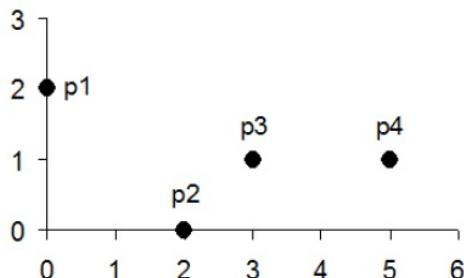
유저 기반 이웃 협업 필터링 - 여행자 추천

유저 정보를 기반으로 한 여행지 추천시스템 구현

- 유클리디안 거리의 정의

- n = number of dimensions (attribute)

- p_k, q_k = value of the k -th dimension



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix

유사도 함수

```
# 유사도 함수
def sim_distance(data, n1, n2):
    sum = 0
    # 두 사용자가 모두 간 여행지를 기준으로 해야해서 i로 변수 통일(j따로 안 써줌)
    for i in data.loc[n1, data.loc[n1, :] >= 0].index:
        if data.loc[n2, i] >= 0:
            sum += pow(data.loc[n1, i]-data.loc[n2, i], 2) # 누적합
    return sqrt(1/(sum+1)) # 유사도 형식으로 출력
```

$$dist = \sqrt{\sum_{k=1}^n (p_k - q_k)^2}$$

매칭 함수

```
# 나와 유사도가 높은 user 매칭 함수
def top_match(data, name, rank=5, simf=sim_distance):
    simList = []
    for i in data.index[-10:]:
        if name != i:
            simList.append((simf(data.loc[name], data.loc[i]), i))
    simList.sort()
    simList.reverse()
    return simList[:rank]
```

추천 함수

```
# 추천 시스템 함수
def recommendation(data, person, simf=sim_distance):
    res = top_match(data, person, len(data))
    score_dic = {}
    sim_dic = {}
    myList = []
    for sim, name in res:
        if sim < 0:
            continue
        for movie in data.loc[person, data.loc[person, :] < 0].index:
            simSum = 0
            if data.loc[name, movie] >= 0:
                simSum += sim * data.loc[name, movie]
            score_dic.setdefault(movie, 0)
            score_dic[movie] += simSum
            sim_dic.setdefault(movie, 0)
            sim_dic[movie] += sim
    for key in score_dic:
        myList.append((score_dic[key] / sim_dic[key], key))
    myList.sort()
    myList.reverse()
    return myList
```

유저 기반 이웃 협업 필터링 - 여행자 추천

유저 정보를 기반으로 한 여행지 추천시스템 구현

유저에 맞는 여행지 추천 함수

```
def reco_data(username, c_type, loc):
    # 여행지 불러오기
    data = pd.DataFrame(TravelSpot.objects.all().values())
    # 여행지 평가 불러오기
    data_r = pd.DataFrame([
        [travelRating.user.username, travelRating.mytripdata.travelspot.content_id, travelRating.mytripdata.travelspot.content_type, travelRating.rating]
        for travelRating in TravelRating.objects.filter(mytripdata__travelspot__content_type=c_type, mytripdata__travelspot__areacode=loc)
    ],
    columns=['user_id', 'content_id', 'content_type', 'rating'])
    data_r[['content_id', 'content_type']] = data_r[['content_id', 'content_type']].astype(int)
    data_r[['rating']] = data_r[['rating']].astype(float)

    # 현재 로그인 된 유저 아이디 받기
    user = username

    # 매트릭스 만들기
    matrix = pd.pivot_table(data_r, index="user_id", columns="content_id", values='rating')
    matrix.fillna(-1, inplace=True)

    # 추천 받기
    li = []
    for rate, c_id in recommendation(matrix, user):
        li.append((rate, data.loc[ data['content_id'] == c_id, 'title'].values[0]))
    reco_li = li[:5]

    # title 기반으로 좌표값 불러오기
    t_li = []
    x_li = []
    y_li = []
    for i in range(len(reco_li)):
        t_li.append(data[ data['title']== reco_li[i][1]].title.values[0])
        x_li.append(data[ data['title']== reco_li[i][1]].mapx.values[0])
        y_li.append(data[ data['title']== reco_li[i][1]].mapy.values[0])

    # 추천 결과
    context = {}
    context['title'] = t_li
    context['mapx'] = x_li
    context['mapy'] = y_li
    print('recommend')
    return context
```

#api데이터 호출 함수

```
def search_data(username, c_type, loc):
    # dataset = get_api_data(1000)

    context = {}
    try:
        if username != None:
            return travel.reco_data(username, c_type, loc)
    except:
        print('search')
        # 여행지 불러오기
        data = models.TravelSpot.objects.all()
        # 여행지 평가 불러오기
        title_li = []
        x_li = []
        y_li = []
        for i in range(0, len(data)):
            if (data[i].content_type == int(c_type)) & (data[i].areacode == int(loc)):
                title_li.append(data[i].title)
                x_li.append(data[i].mapx)
                y_li.append(data[i].mapy)

        context['title'] = title_li[:5]
        context['mapx'] = x_li[:5]
        context['mapy'] = y_li[:5]

    return context
```

유저 기반 이웃 협업 필터링 - 여행자 추천

유저 정보를 기반으로 한 여행지 추천시스템 구현

유저에 맞는 여행지 추천 함수

```
def reco_data(username, c_type, loc):
    # 여행자 플리오기
    data = pd.DataFrame(travelRating.mytripdata.travelspot.initial())
    # 여행지 평가 불러오기
    data_r = pd.DataFrame([
        [travelRating.user.username, travelRating.mytripdata.travelspot.content_id, travelRating.mytripdata.travelspot.content_type, travelRating.rating]
    ],
    columns=['user', 'content_id', 'content_type', 'rating'])
    # 현재 로그인 유저
    user = username

    # 매트릭스 만들기
    matrix = pd.pivot_table(data_r, index='user', columns='content_id', values='rating')
    matrix.fillna(-1, inplace=True)

    # 추천 받기
    li = []
    for rate, c_id in matrix[user].iteritems():
        li.append((rate, c_id))
    reco_li = li[:5]

    # title 저장하기
    t_li = []
    x_li = []
    y_li = []
    for i in range(len(reco_li)):
        t_li.append(reco_li[i][1])
        x_li.append(100 + i * 10)
        y_li.append(100 + i * 10)

    # 추천 결과
    context = {}
    context['title'] = t_li
    context['mapx'] = x_li
    context['mapy'] = y_li
    print('recommend')
    return context
```

실행 결과

```
[(4.98, '검무산'),
 (4.98, '강화 용두레마을'),
 (4.98, '가지산도립공원(울주)'),
 (4.97, '신흥진식당'),
 (4.96, '충주 탄금호 무지개길'),
 (4.94, '게스트하우스강남'),
 (4.94, '강릉커피거리'),
 (4.94, '갯벌부흥펜션'),
 (4.92, '개성순대주식회사'),
 (4.92, '갤러리 라메르')]
```

#api데이터 호출 함수

```
def search_data(username, c_type, loc):
    # dataset = get_api_data(1000)

    context = {}
    try:
        if username != None:
            context['c_type', loc]
            context['recommend']
            context['search']
            context['recommend']
            context['recommend']
            context['recommend']
            context['recommend']
            context['recommend']
```



유저 기반 이웃 협업 필터링 – 여행자 추천

카카오 지도 API를 통해 추천데이터 시각화

지도 생성

```
var mapContainer = document.getElementById('map'), // 지도를 표시할 div
    mapOption = {
        center: new kakao.maps.LatLng(37.498004414546934, 127.02770621963765), // 지도의 중심좌표
        center: new kakao.maps.LatLng(context.areacode[0], context.areacode[1]), // 지도의 중심좌표
        level: 8 // 지도의 확대 레벨
    };

var map = new kakao.maps.Map(mapContainer, mapOption); // 지도를 생성합니다
```

여행지 태입별 GPS좌표가 담길 배열 생성

```
// 관광지 마커가 표시될 좌표 배열입니다
var tourPositions = [];
for (i of [...Array(context.tour_context.title.length).keys()]) {
    // console.log(context.mapy[i], context.mapx[i])
    var y = context.tour_context.mapy[i];
    var x = context.tour_context.mapx[i];
    tourPositions.push(new kakao.maps.LatLng(y, x))
}
```

마커, 오버레이 배열

```
var tourMarkers = []; // 관광지 마커 객체를 가지고 있을 배열입니다
tourOverlay = [], // 관광지 오버레이
```

생성된 마커 배열 추가

```
createTourMarkers(); // 관광지 마커를 생성하고 관광지 마커 배열에 추가합니다
```

마커 표시

```
changeMarker(context.content_type); // 지도에 관광지 마커가 보이도록 설정합니다
```

마커 생성 함수

```
// 마커 이미지의 주소와, 크기, 옵션으로 마커 이미지를 생성하여 리턴하는 함수입니다
function createMarkerImage(src, size, options) {
    var markerImage = new kakao.maps.MarkerImage(src, size, options);
    return markerImage;
}
```

마커 정보 생성 함수

```
// 좌표와 마커 이미지를 받아 마커를 생성하여 리턴하는 함수입니다
function createMarker(position, image) {
    var marker = new kakao.maps.Marker({
        position: position,
        image: image
    });

    return marker;
}
```

유저 기반 이웃 협업 필터링 - 여행자 추천

카카오 지도 API를 통해 추천데이터 시각화

여행지 타입별 마커, 오버레이 배열 추가 함수

```
// 관광지 마커를 생성하고 관광지 마커 배열에 추가하는 함수입니다
function createTourMarkers() {

    for (var i = 0; i < tourPositions.length; i++) {

        // var imageSize = new kakao.maps.Size(22, 26),
        var imageSize = new kakao.maps.Size(50, 60),
            imageOptions = {
                spriteOrigin: new kakao.maps.Point(10, 0),
                spriteSize: new kakao.maps.Size(36, 98)
            };

        // 마커이미지와 마커를 생성합니다
        // var markerImage = createMarkerImage(coffeemarkerImageSrc, imageSize, imageOptions),
        var markerImage = createMarkerImage(tourmarkerImageSrc, imageSize),
            marker = createMarker(tourPositions[i], markerImage);

        // 커스텀 오버레이에 표출될 내용으로 HTML 문자열이나 document element가 가능합니다
        var content = `<div class="customoverlay">
            <a href="https://api.visitkorea.or.kr/search/commonList.do" target="_blank">
                <span class="title">${context.tour_context.title[i]}</span>
            </a>
        </div>`;

        // 커스텀 오버레이를 생성합니다
        var customOverlay = new kakao.maps.CustomOverlay({
            map: map,
            position: tourPositions[i],
            content: content,
            yAnchor: 0.5
        });

        // 생성된 마커를 관광지 마커 배열에 추가합니다
        tourMarkers.push(marker);
        tourOverlay.push(customOverlay);
    }
}

// 관광지 마커들의 지도 표시 여부를 설정하는 함수입니다
function setTourMarkers(map) {
    for (var i = 0; i < tourMarkers.length; i++) {
        tourMarkers[i].setMap(map);
        tourOverlay[i].setMap(map);
    }
}
```

클릭 이벤트

```
// 카테고리를 클릭했을 때 type에 따라 카테고리의 스타일과 지도에 표시되는 마커를 변경합니다
function changeMarker(type) {

    var tourMenu = document.getElementById('tourMenu');
    var cultureMenu = document.getElementById('cultureMenu');
    var festivalMenu = document.getElementById('festivalMenu');
    var leportsMenu = document.getElementById('leportsMenu');
    var stayMenu = document.getElementById('stayMenu');
    var shoppingMenu = document.getElementById('shoppingMenu');
    var restaurantMenu = document.getElementById('restaurantMenu');
```

클릭시 변경 이벤트

```
// 관광지 카테고리가 클릭됐을 때
if (type === 'tour') {

    // 관광지 카테고리를 선택된 스타일로 변경하고
    tourMenu.className = 'menu_selected';

    // 문화시설과 축제공연 카테고리는 선택되지 않은 스타일로 바꿉니다
    cultureMenu.className = '';
    festivalMenu.className = '';
    leportsMenu.className = '';
    stayMenu.className = '';
    shoppingMenu.className = '';
    restaurantMenu.className = '';

    // 관광지 마커들만 지도에 표시하도록 설정합니다
    setTourMarkers(map);
    setCultureMarkers(null);
    setFestivalMarkers(null);
    setLeportsMarkers(null);
    setStayMarkers(null);
    setShoppingMarkers(null);
    setRestaurantMarkers(null);
```

지도컨트롤러

```
// 지도에 확대 축소 컨트롤을 생성한다
var zoomControl = new kakao.maps.ZoomControl();

// 지도의 우측 아래에 확대 축소 컨트롤을 추가한다
map.addControl(zoomControl, kakao.maps.ControlPosition.BOTTOMRIGHT);
```

유저 기반 이웃 협업 필터링 - 여행자 추천

실행 결과

여행지 타입별 마커, 오버레이 배열 추가 함수

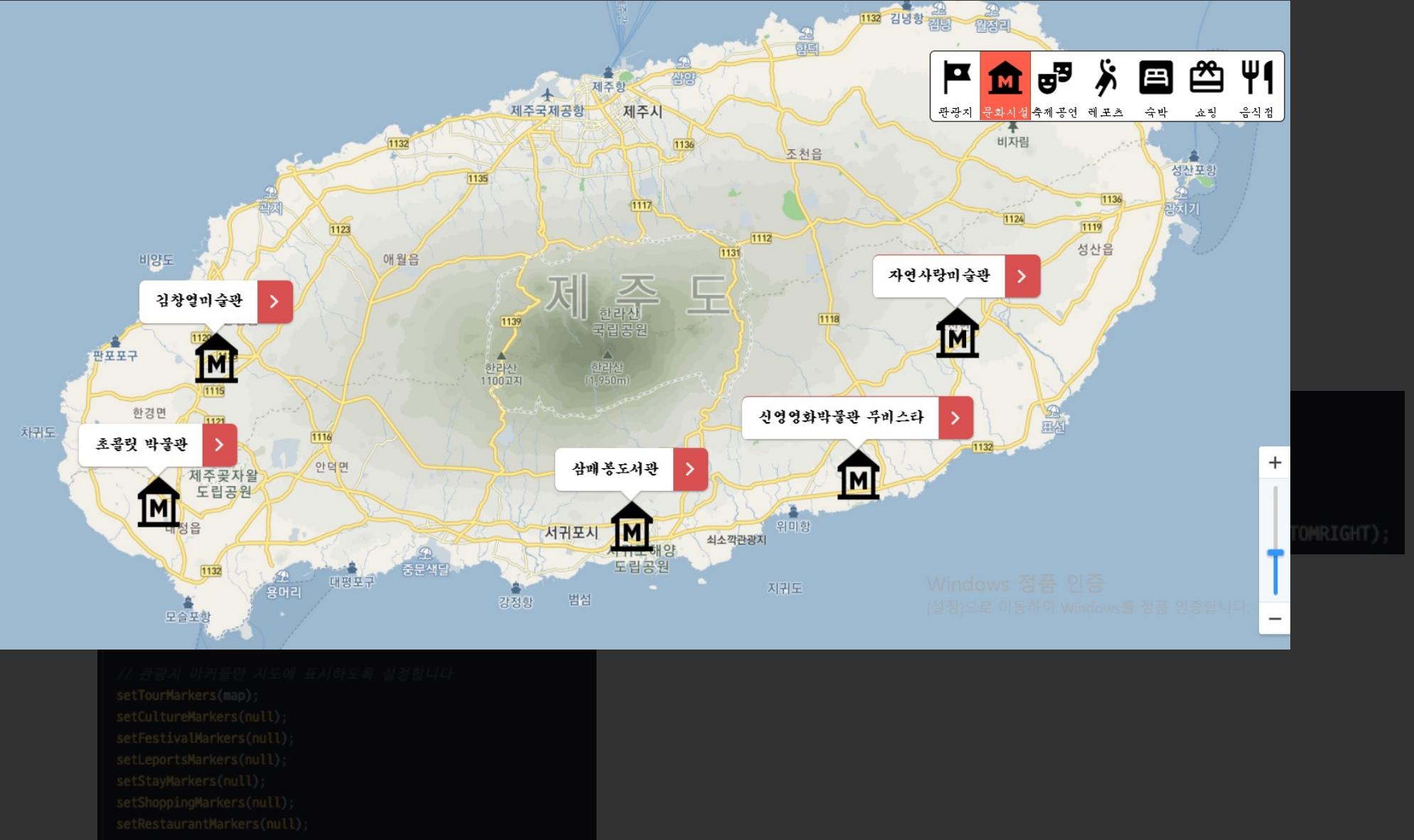
클릭 이벤트

```
// 관광지 마커를 생성하고 관광지 마커 배열에 추가하는 함수입니다
function createTourMarkers() {
    for (var i = 0; i < tourPositions.length; i++) {
        // var imageSize = new kakao.maps.Size(22, 26),
        var imageSize = new kakao.maps.Size(50, 60),
            imageOptions = {
                spriteOrigin: new kakao.maps.Point(10, 0),
                spriteSize: new kakao.maps.Size(36, 98)
            };
        // 마커이미지와 마커를 생성합니다
        // var markerImage = createMarkerImage(coffeemarkerImageSrc, imageSize);
        var markerImage = createMarkerImage(tourmarkerImageSrc, imageSize);
        marker = createMarker(tourPositions[i], markerImage);

        // 커스텀 오버레이에 표출될 내용으로 HTML 문자열이나 document Element 등이 가능합니다
        var content = `<div class="customoverlay">
            <a href="https://api.visitkorea.or.kr/search/commonList.do" title=" ${context.tour_context.title[i]} " target="_blank">${context.tour_context.title[i]}</a>
        </div>`;
        // 커스텀 오버레이를 생성합니다
        var customOverlay = new kakao.maps.CustomOverlay({
            map: map,
            position: tourPositions[i],
            content: content,
            yAnchor: 0.5
        });

        // 생성된 마커를 관광지 마커 배열에 추가합니다
        tourMarkers.push(marker);
        tourOverlay.push(customOverlay);
    }
}

// 관광지 마커들의 지도 표시 여부를 설정하는 함수입니다
function setTourMarkers(map) {
    for (var i = 0; i < tourMarkers.length; i++) {
        tourMarkers[i].setMap(map);
        tourOverlay[i].setMap(map);
    }
}
```





로그인, 로그아웃

```
class LoginView(FormView):
    template_name = "users/login.html"
    form_class = forms.LoginForm

    def form_valid(self, form):
        email = form.cleaned_data.get("email")
        password = form.cleaned_data.get("password")
        user = authenticate(self.request, username=email, password=password)
        if user is not None:
            login(self.request, user)

        return super().form_valid(form)

    def get_success_url(self):
        next_arg = self.request.GET.get("next")
        if next_arg is not None:
            return next_arg
        else:
            print(self.request.user.id)
            return reverse_lazy("users:profile", kwargs={'pk': self.request.user.id})

    def log_out(request):
        logout(request)
        return redirect(reverse("core:login"))
```



회원가입, 회원정보수정

```
class SignupView(FormView):
    template_name = "users/signup.html"
    form_class = forms.SignUpForm
    success_url = reverse_lazy("users:survey")

    def form_valid(self, form):
        form.save()
        email = form.cleaned_data.get("email")
        password = form.cleaned_data.get("password")
        user = authenticate(self.request, username=email, password=password)
        if user is not None:
            login(self.request, user)
        return super().form_valid(form)
```

```
class UpdateProfileView(UpdateView):
    model = models.User
    template_name = "users/update_profile.html"
    fields = (
        "first_name",
        "last_name",
    )

    def get_object(self, queryset=None):
        return self.request.user

    def get_form(self, form_class=None):
        form = super().get_form(form_class=form_class)
        form.fields["first_name"].widget.attrs = {"placeholder": "First name"}
        form.fields["last_name"].widget.attrs = {"placeholder": "Last name"}
        return form
```

```
class UpdatePasswordView(PasswordChangeView):
    template_name = "users/update-password.html"

    def get_form(self, form_class=None):
        form = super().get_form(form_class=form_class)
        form.fields["old_password"].widget.attrs = {"placeholder": "Current password"}
        form.fields["new_password1"].widget.attrs = {"placeholder": "New password"}
        form.fields["new_password2"].widget.attrs = {
            "placeholder": "Confirm new password"
        }
        return form

    def get_success_url(self):
        return self.request.user.get_absolute_url()
```

5

느낀점

Impression



5. 느낀점



김지윤

이번 프로젝트 하면서 어려움이 있었나요??
어떻게 극복 하셨나요?



박현수



오은영

세미 프로젝트가 끝나고 한달만에 장고를 다시 하니까 그동안 까
먹었던 부분도 있었고 새롭게 배워나가야 할 부분도 있었습니다.



김지윤

생각의 전환을 하고 싶다. 무언가를 이루어야 그것이 진정한 극복일까?
버릴 때는 버리는 것이 또한 극복을 했다고 볼 수도 있다고 생각한다.
그래서 시간 안에 못하겠다 싶은 부분은 과감히 버렸다.



박윤수

기능 구현에 있어 어려움이 있었다. 그동안 많이 배워서 자신감에 차
있었는데, 막상 기능을 구현하려고 보니 에러가 너무 많았다. 단계를
쪼개고 하나씩 테스트 하는 것이 중요하는 것을 깨달았다.

5. 느낀점



박현수

프로젝트에서 잘한 부분이 있나요?



오은영

웹페이지를 만들면서 자바스크립트와 css.에 대해 알게되었고 맙은 부분에 대해서는 잘 해낸거 같습니다.



김지윤

프로젝트에서 소통을 하는데 있어, 정리를 잘 한 것 같다.



박윤수

기능을 구현하려고 열심히 배웠다. 장고도 좀더 활용해보고 투박하지만 추천데이터를 지도에 표시하는 것을 자바스크립트로 구현하였다.



박현수

5. 느낀점



오은영

프로젝트에서 아쉬운 부분이 있나요?

컨디션 조절을 잘하지 못해서, 그 부분이 아쉽다. 시간이 지체되는 경우가 있었는데, 그럴 때는 과감히 버렸어야 했는데 의사결정이 늦은 부분이 있었다. 그 부분이 아쉬웠다.



김지윤



박윤수

추천 시스템을 좀더 고도화를 시키지 못한 것
이 너무 아쉬웠다. 욕심이 많았다. 너무 여러
가지를 하려고 했던 것 같다.



박현수



오은영

처음 계획했던 기능들을 모두 넣지 못해서 아쉬워요!

5. 느낀점



김지윤

프로젝트가 진로설계, 취업분야 탐색 및 결정에 도움이 되었나요?

프로젝트에 앞서 어떤 것을 잘 할 수 있을까 고민을 많이 했는데 프로젝트 속에서 잘 해야 될 것과 손에 잘 맞는 것을 알게 되었던 점이 도움이 되었다.



박윤수



박현수

아직은 더 경험이 필요한것 같습니다.



오은영



김지윤

프로젝트를 진행할 때, 내 포지션에 대해 생각을 많이 한 것 같다. 특히 내가 잘하는 것, 못하는 것에 대해 여러모로 생각해보는 시간이 있었는데, 이를 통해 나에 대해 더 많이 알게 되었고, 이를 기반으로 진로를 구체화 시켰다.

5. 느낀점



박윤수

더 하고 싶은 말 있나요?



박현수



오은영

다들 고생 많았고 짧은 기간이였지만 급속도로 친해져서 좋았습니당~!



김지윤

팀원들이 정말 좋은 사람들이라 좋았다. 원래 낯을 많이 가리는 부끄럼장이 막내 타입이라 남이랑 친해지기 어려워하는데, 이번에 팀원들과 많이 친해진 것 같다. 친구가 늘어서 기쁘다.



박윤수

좋은 팀원들 만나서 정말 좋았고, 좋은 팀원들 덕에 조금 더 성장할 수 있어 좋았습니다.