Busan for U

4 조

CONTENTS

Busan For U Web-Site 주제선정 배경

혼자 또는 가족, 연인과 같은 동반자와 여행 계획을 짜거나 어떤 볼거리, 놀거리 들이 있는지 알아볼 때 자주 여행을 다니던 사람이 아니라면 어떤 곳으로 여행가는 것이 알차고 재밌게 보낼 수 있는지에 큰 어려움을 느끼게 됩니다.

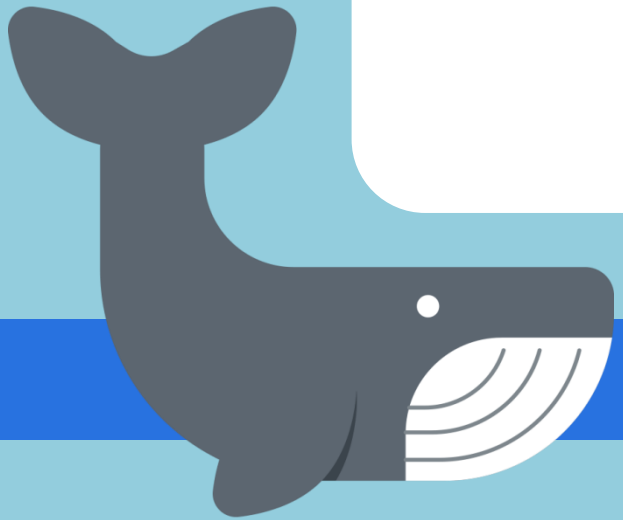
그럴 때 저희 Busan For U와 같은 웹 페이지가 있다면 자신의 나이대에 어떤 테마를 좋아하며 누구와 함께 가는지 선택하고 추천 받아 사용자가 느낄 선택의 어려움은 줄여주고, 여행 만족도는 높여주는 행복한 여행을 즐길 수 있도록 하기위해 구상하게 되었습니다.

Busan For U는 부산여행지 추천 사이트

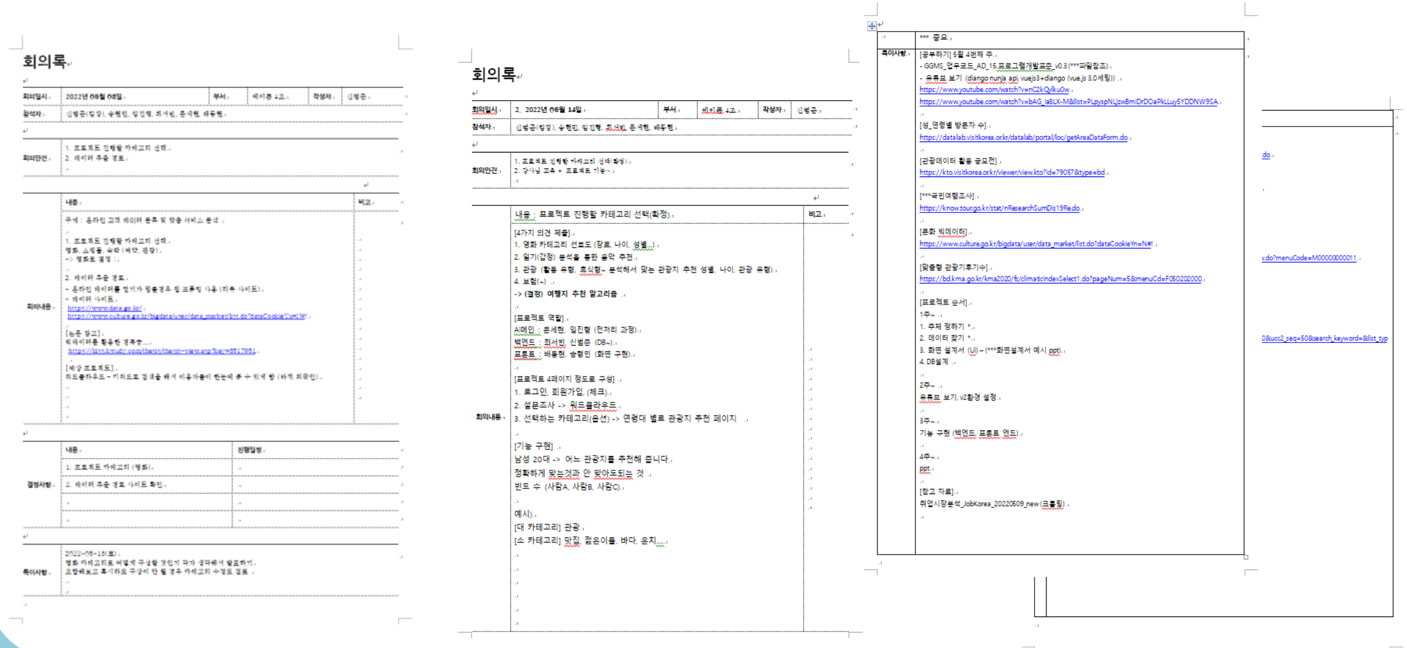
설문조사 페이지를 통해 연령, 동반자, 테마를 선택하여
사용자에게 알맞은 부산 관광 명소를 추천해 주는 사이트

사용자 로그인 > 사용자 선호도 선택

선호도 기반 관광지 추천 > 관광지 코스안내, 커뮤니티 기능



프로젝트 일지 및 계획표



회의록

회의일시	2022년 06월 08일	부서	에이콘 4조	작성자	신범준
참석자	신범준(팀장), 송현민, 임진형, 최서빈, 문세현, 배동현				

회의안건	1. 프로젝트 진행할 카테고리 선택 2. 데이터 추출 경로

	내용	비고
회의내용	주제 : 온라인 고객 데이터 분류 및 맞춤 서비스 분석 1. 프로젝트 진행할 카테고리 선택 영화, 쇼핑몰, 숙박 (예약, 전망) -> 영화로 결정 : 2. 데이터 추출 경로 - 온라인 데이터를 얻기가 힘들경우 웹 크롤링 사용 (리뷰 사이트) - 데이터 사이트 https://www.data.go.kr/ https://www.culture.go.kr/bigdata/user/data_market/list.do?dataCookieYn=N#! [논문 참고] 빅데이터를 활용한 경북권... https://kiss.kstudy.com/thesis/thesis-view.asp?key=3517951 [예상 프로젝트] 워드클라우드 – 키워드로 검색을 해서 이용자들이 한눈에 볼 수 있게 함 (하게 되면)	

	내용	진행일정
결정사항	1. 프로젝트 카테고리 (영화)	
	2. 데이터 추출 경로 사이트 확인	

특이사항	2022-06-10(토) 영화 카테고리로 어떻게 구성할 것인지 각자 생각해서 발표하기. 조합해보고 혹시라도 구상이 안 될 경우 카테고리 수정도 검토

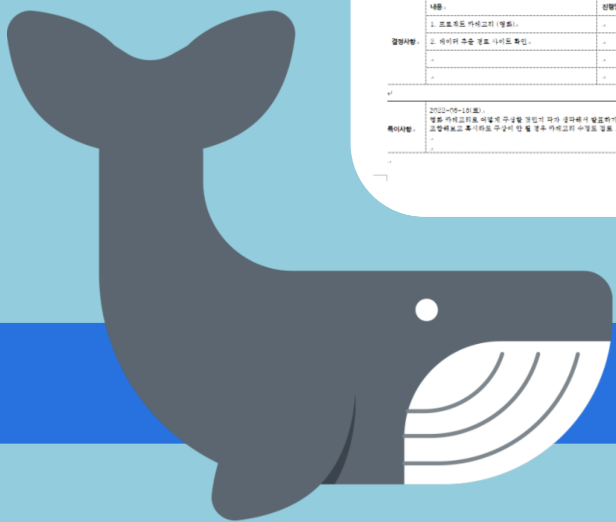
회의록

회의일시	2. 2022년 06월 14일	부서	에이콘 4조	작성자	신범준
참석자	신범준(팀장), 송현민, 임진형, 최서빈, 문세현, 배동현				

회의안건	1. 프로젝트 진행할 카테고리 선택(확정) 2. 강사님 교육 + 프로젝트 기능~

	내용 : 프로젝트 진행할 카테고리 선택(확정)	비고
회의내용	[4가지 의견 제출] 1. 영화 카테고리 선호도 (장르, 나이, 성별...) 2. 일기(감정) 분석을 통한 음악 추천 3. 관광 (활동 유형, 휴식형~ 분석해서 맞는 관광지 추천 성별, 나이, 관광 유형) 4. 보험(~) -> (결정) 여행지 추천 알고리즘 [프로젝트 역할] AI메인 : 문세현, 임진형 (전처리 과정) 백엔드 : 최서빈, 신범준 (DB~) 프론트 : 배동현, 송현민 (화면 구현) [프로젝트 4페이지 정도로 구성] 1. 로그인, 회원가입, (체크) 2. 설문조사 -> 워드클라우드 3. 선택하는 카테고리(옵션) -> 연령대 별로 관광지 추천 페이지 [기능 구현] 남성 20대 -> 어느 관광지를 추천해 줍니다. 정확하게 맞는것과 안 맞아도되는 것 빈도 수 (사람A, 사람B, 사람C) 예시) [대 카테고리] 관광 [소 카테고리] 맛집, 젊은이들, 바다, 운치.....	

	*** 중요
특이사항	[공부하기] 5월 4번째 주 - GGMS_업무코드_AD_15.프로그램개발표준_v0.3 (***파일참조) - 유튜브 보기 (django nunja api, vuejs3+django (vue.js 3.0세팅)) https://www.youtube.com/watch?v=nC2kQylku0w https://www.youtube.com/watch?v=bAG_la8LX-M&list=PLpyspNLjzwBmIDrDOaPkLLuy5YDDNW9SA [성_연령별 방문자 수] https://datalab.visitkorea.or.kr/datalab/portal/loc/getAreaDataForm.do [관광데이터 활용 공모전] https://kto.visitkorea.or.kr/viewer/view.kto?id=79057&type=bd [***국민여행조사] https://know.tour.go.kr/stat/nResearchSumDis19Re.do [문화 빅데이터] https://www.culture.go.kr/bigdata/user/data_market/list.do?dataCookieYn=N#! [맞춤형 관광기후지수] https://bd.kma.go.kr/kma2020/fs/climaticIndexSelect1.do?pageNum=5&menuCd=F050202000 [프로젝트 순서] 1주~ 1. 주제 정하기 * 2. 데이터 찾기 * 3. 화면 설계서 (UI) – (***화면설계서 예시 ppt) 4. DB설계 2주~ 유튜브 보기, v2환경 설정 3주~ 기능 구현 (백엔드, 프론트 엔드) 4주~ ppt [참고 자료] 취업시장분석_JobKorea_20220509_new (크롤링)



1.프로젝트 소개

프로젝트 일지 및 계획표

Team 4											
	1 주차								2 주차		
날짜	5/15	16	17	18	19	20	21	22	23	24	25
일정	프로젝트 주제 선정 및 적정 Data 검색						Data 선정 및 페이지 구성 (Meet)		Part 별 학습 및 setting		
세부 내용	주제별 장단점 비교 & 주제 선정			학습 DATA 검색			최종 Data 선정	Page UI & 도안 작성	part 학습 & tool, git setting		
	2 주차				3 주차						
날짜	26	27	28	29	30	31	6/1	2	3	4	5
일정	Part 별 Programming & 연동										
세부 내용	Front - end : 각 Page별 UI 구성(html, css, bootstrap) , 기능 구현(javascript, Vue.js)										Front & Back & Big Data 연동
	Back - end : Database 구축(Oracle, Maria DB), 기능 구현(Python, Django-Ninja)										
	Big Data : Data 전처리, Model 구성, Exel 작업, Machine-learning(Python)										
	4 주차							발표주			
날짜	6	7	8	9	10	11	12	13	14	15	16
일정	미흡점 보완 & 후보기능 추가				최종 점검 및 오류 수정			ppt 제작		발표 준비	D-Day
세부 내용	미완성 기능 구현 및 미흡점 보완 모든 기능 구현 시 후보기능 추가										발표

1.프로젝트 소개

프론트엔드

배동*

전체적인 UI설계 및 제작 Vue.js 구현

송형*

웹 틀 제작 및 Vue.js 구현

백엔드

최서*

전체적인 기능 구현 및 세부 기능 구현

신범*

자료 수집 및 PPT 제작

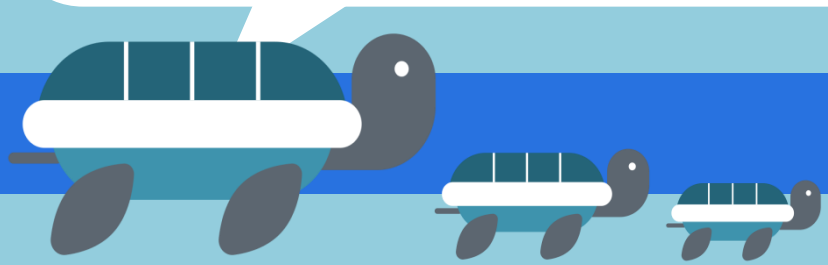
AI

임진*

데이터 수집, AI 알고리즘 분석

문세*

데이터 수집, AI 데이터 전처리 및 알고리즘 분석



2. 개인역할

Infra

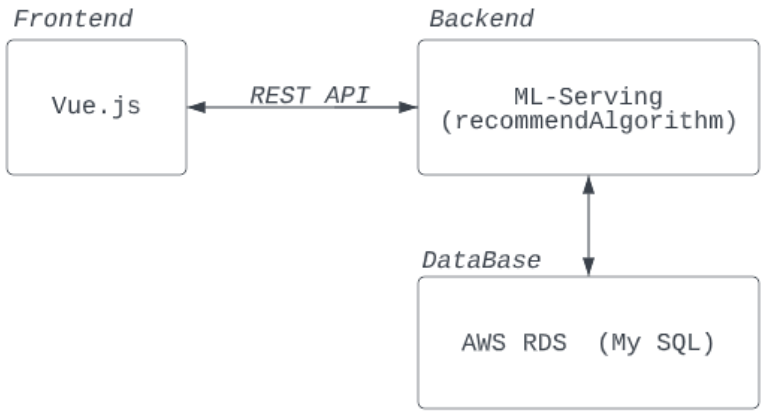
REST API

Frontend

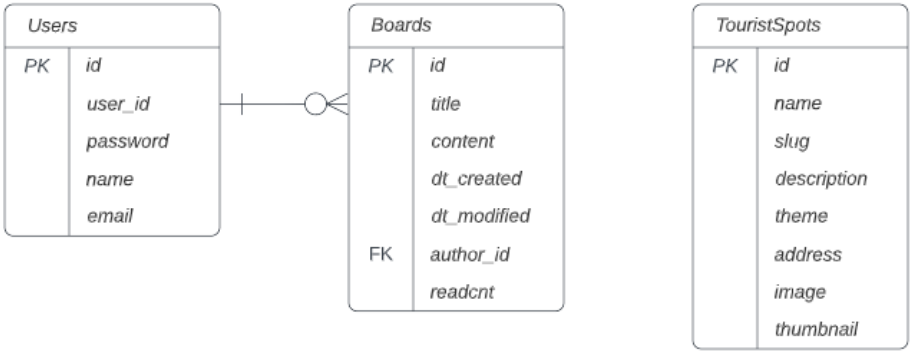
Database

3. Tech Stack

흐름도



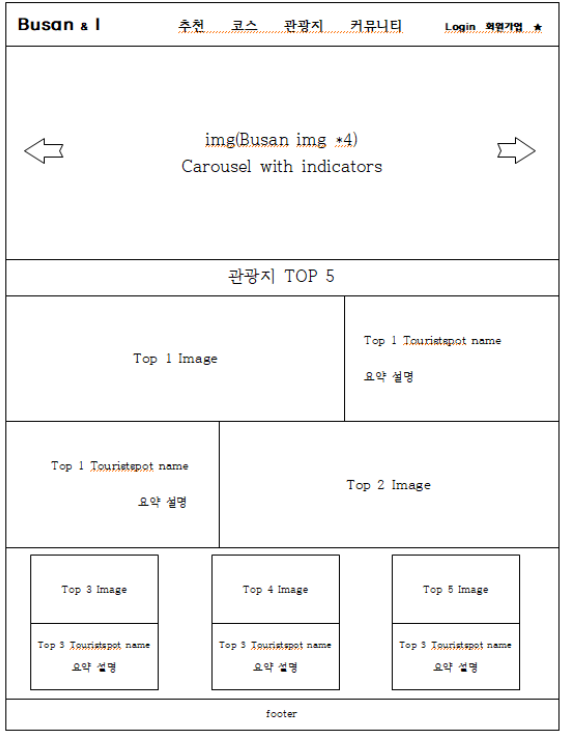
DBMS ER Diagram



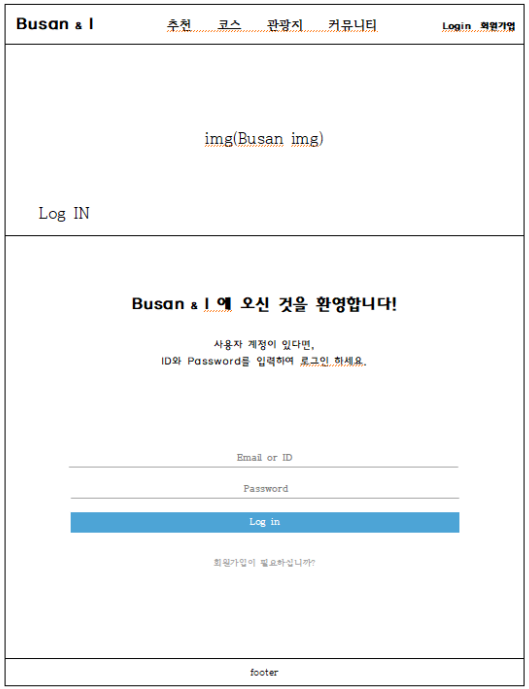
4. 설계 및 화면 구성

와이어프레임

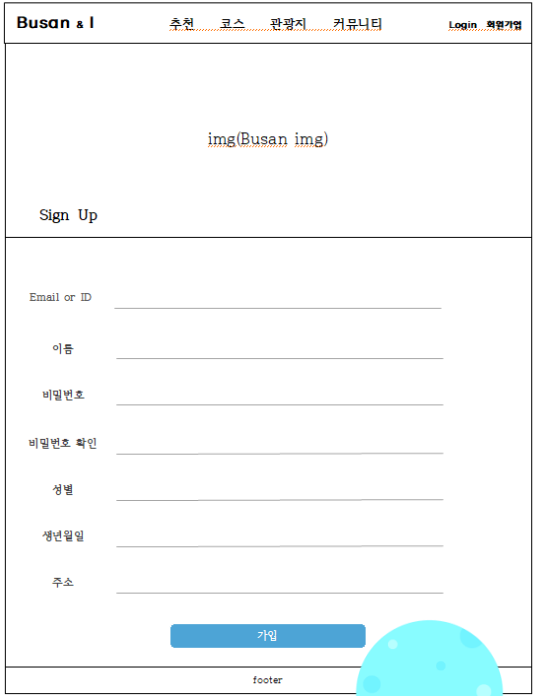
홈(Home) Page



로그인(Log_in) Page



회원가입(Sign_up) Page

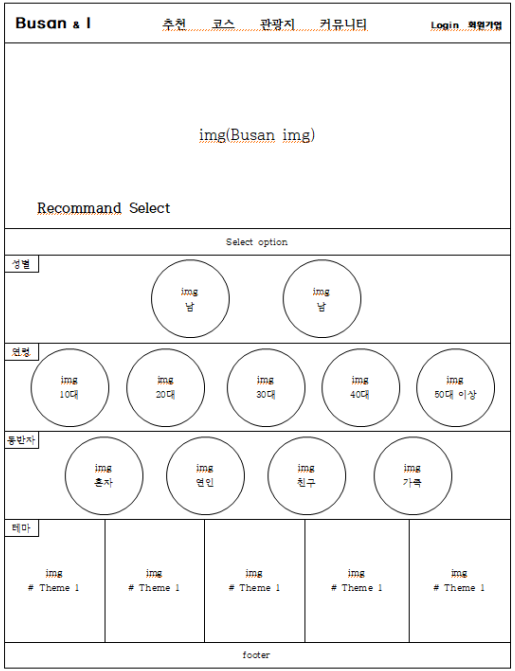


4. 설계 및 화면 구성

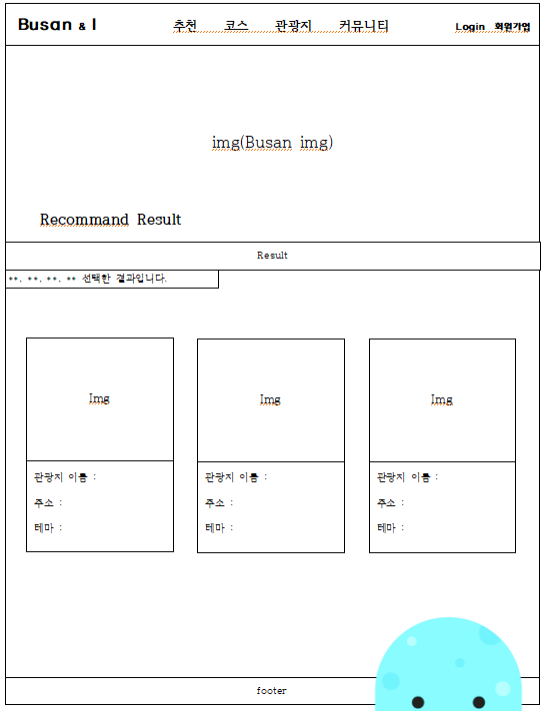
와이어프레임

마이페이지(Mypage) - 회원정보 수정

추천 선택(Recommand_select) Page



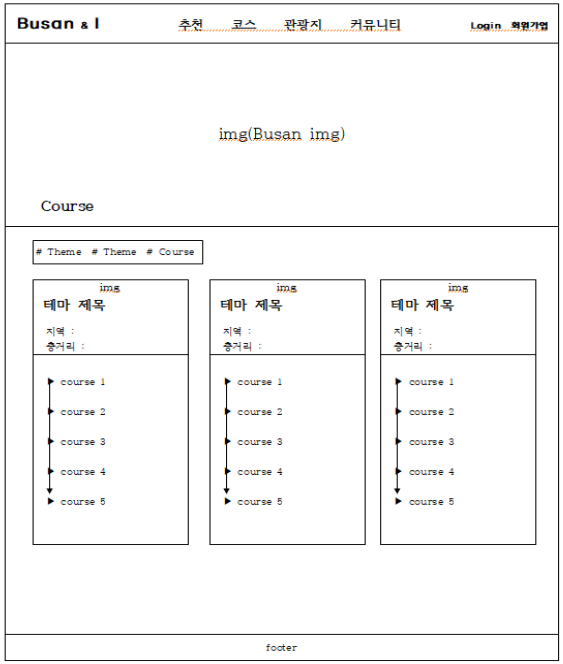
추천 결과(Recommand_result) Page



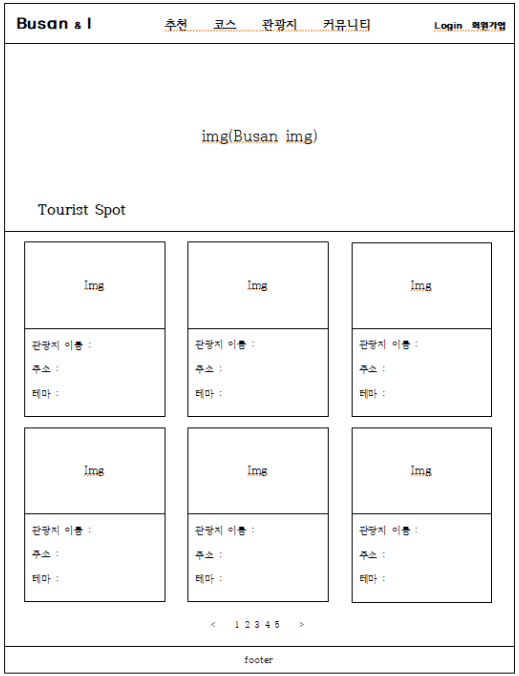
4. 설계 및 화면 구성

와이어프레임

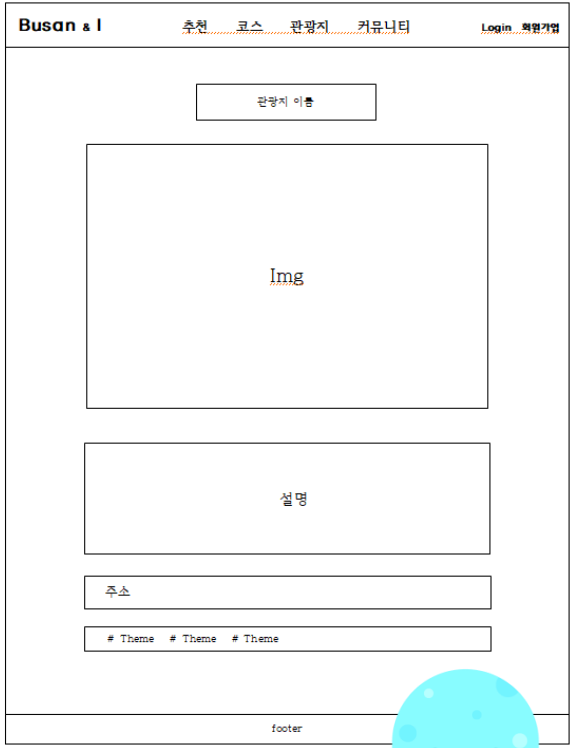
코스 추천(Course) Page



관광지(Touristspot) Page - 즐겨찾기 페이지와 동일



관광지 상세보기(Touristspot_details) Page



4. 설계 및 화면 구성

Recommend Model

Data Collecting

부산광역시 빅데이터플랫폼

한국관광 데이터랩 Data Lab

부산 관광객 데이터 (1022개)

Column	내용
Location	관광지명
Age	나이
Gender	성별
Companions	동반자
Education level	학력
Transportation	교통수단
Visit month	방문월

부산 내 관광지 데이터 (201개)

Column	내용
PLACE_NM	관광지명
TRRSRT_ROAD_NM_ADDR	관광지 주소
GRP_NM	동반자
SEASON_NM	계절
IEM_NM	관광지 테마
CL_NM	위치 특징

location	age	gender	companions	education level	transporation	visit month
용두산공원	10	여	친구	중졸	대중교통	10
보수동책방골목	50	남	커플	대학원졸	승용차	9
해운대해수욕장	30	여	커플	고졸	승용차	8
백양산	50	여	커플	고졸	승용차	7
부산시민공원	40	여	아이	고졸	대중교통	7
서면시장	20	여	가족	대졸	승용차	4
해운대해수욕장	30	여	커플	고졸	승용차	11
부산아쿠아리움	10	여	커플	중졸	대중교통	5
해운대해수욕장	10	남	친구	중졸	도보	8
대항어촌마을	20	여	가족	중졸	자전거	5
사직야구장	20	남	친구	대졸	대중교통	5
부산현대미술관	30					
을숙도생태공원	30					
해동용궁사	50					
해동용궁사	20					
서면시장	50					
유엔기념공원	20					
유엔기념공원	20					
송도해수욕장	10					

PLACE_NM	GUGUN_N	TRRSRT_LA	TRRSRT_LO	TRRSRT_R	TEL_NO	GRP_NM	SEASON_N	IEM_NM	CL_NM	ETC_CN
용소웰빙공원	기장군	35.24629	129.2041	부산 기장	5.17E+08	가족, 아이	봄, 가을	자연,공원	바다,도시	사계절 나
몰운대 트레킹	사하구	35.04651	128.9627	부산광역시	5.12E+08	가족, 아이	사계절	자연,걷기	바다	이곳은 부
일광해수욕장 더	기장군	35.26255	129.2337	부산 기장	5.17E+08	가족,아이,	봄, 가을	자연,공원	바다,도시	약 10분
IKEA(이케아)등	기장군	35.19029	129.2103	부산광역시	16704532	가족,부모,	사계절	쇼핑	도시	파란색 오
사상근린공원	사상구	35.15885	128.9931	부산광역시	7.04E+09	가족,아이	봄,여름,가	자연,걷기	산	아이들이
을숙도생태공원	사하구	35.11677	128.9492	부산광역시	5.12E+08	가족,부모,	사계절	자연,공원,	기타	을숙도는
삼락생태공원	사상구	35.16873	128.9736	부산광역시	5.13E+08	가족,부모,	사계절	자연,공원,	기타	을숙도 생
대저생태공원	강서구	35.20498	128.9827	부산광역시	5.2E+08	가족,부모,	사계절	자연,공원,	기타	계절마다
신평소공원	기장군	35.29378	129.2606	부산 기장	5.17E+08	가족,아이,	봄, 가을	자연,공원,	바다,도시	비대면 여
흰여울마을	영도구	35.07828	129.0453	부산광역시	5.14E+08	가족,부모,	봄,여름,가	자연,걷기	기타	감천문화
감천문화마을	사하구	35.09731	129.0103	부산광역시	5.12E+08	가족,부모,	봄,여름,가	자연,걷기	기타	알록달록

5.제작방법

Recommend Model

Data Analysis

```
loc_num              1.000000
companions           0.052395
gender               0.034142
visit month          0.006530
education level     -0.007324
transporation       -0.020811
age                 -0.032496
Name: loc_num, dtype: float64
```

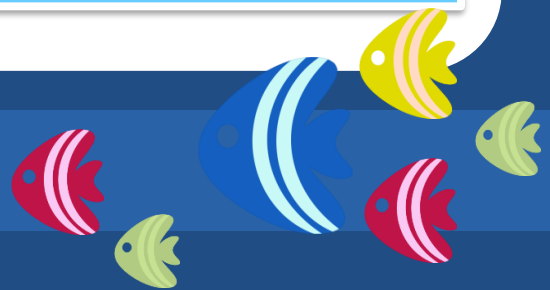
```
예측값 :  [27 19 48 27 27 48 48 48 48 27]
실제값 :  [37  2 38 49 48  3  6  2 48 43]
accuracy :  0.05859375
```

✓ 관광지(loc_num)와 관광객 데이터의 변수 간의 상관계수 분석 결과 상관관계가 매우 낮은 것으로 확인

✓ RandomForestClassifier를 통한 예측모델 실행 시
매우 낮은 정확도를 보임

✓ 기존 구상한 사용자 기반 협업 필터링 모델 사용은 어려울 것으로 판단

관광지 데이터 중심의 컨텐츠 기반 필터링 모델 사용



5.제작방법

Recommend Model

Model Selection

콘텐츠 기반 필터링(Content based filtering) 추천 모델

location	theme	group	age
APEC나루공원	자연,공원	가족, 아이, 커플	20,30,40
IKEA(이케아			
감천문화마			
광안리 해양			
광안리해수			
국제시장			
낙조분수			
대항어촌마			
드림씨어터			
렛츠런파크			
물운대 몽돌			
민락수변공			
민락회타운			
백양산			
범어사			
벡스코			

```
# 모델
count_vector = CountVectorizer(ngram_range=(1, 3))
c_vector_theme = count_vector.fit_transform(data['theme'].values.astype('U'))

theme_c_sim = cosine_similarity(c_vector_theme, c_vector_theme).argsort()[:,::-1]

def get_recommend_traval_list(df, traval_list, top=60):
    target_traval_index = df[df['location'] == traval_list].index.values
    print('테마별 target : ',target_traval_index)
    sin_index = theme_c_sim[target_traval_index,:top].reshape(-1)
    print('테마별 sin : ',sin_index.shape)
    sin_index = sin_index[sin_index != target_traval_index]
    print('테마별 sin 2 : ',sin_index.shape)
    result = data.iloc[sin_index].sort_values('score', ascending=False)#[:4]
    # score는 정렬을 위한 것
    return result

add1 = '걷기'
add2 = '쇼핑'
add3 = '자연'
add4 = '공원'
add5 = '이색여행'
add6 = '문화'
add7 = '역사'
add8 = '체험'
add9 = '축제'

list = add1 + '|' + add3
print('추천리스트 :', list)
```

- ✓ 관광지들 간의 유사도(similarity) 를 기반으로 추천

- 유사도 수치 계산을 위해 아이템을 벡터 형태로 표현하고, 이들 벡터 간의 코사인 유사도(Cosine similarity)를 계산하는 방법을 사용.

- ✓ 협업 필터링(Collaborative Filtering)의 일환으로 기존 관광객 데이터 중 관광지별 동반자 형태, 나이에서 상관관계가 큰 변수 값을 포함하여 유사도 측정.

Recommend Model

Process

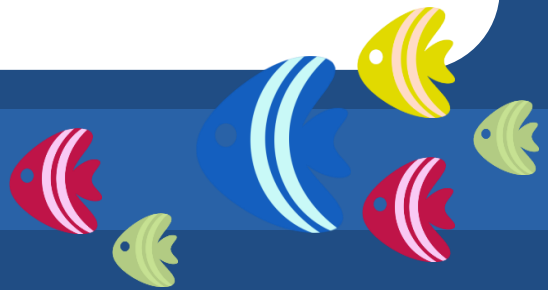
나이 선택
반영 리스트

동반자 선택
반영 리스트

테마 선택
반영 리스트

추천 여행지
3개 출력

- ✓ 각 단계별로 선택된 변수를 포함한 여행지 리스트 간의 코사인 유사도 계산
- ✓ 추천 리스트를 분류하여 최종 추천 여행지 출력



5.제작방법

Frontend

App.vue 1 ×

busanproject > Busan > busan-vue > src > App.vue > {} "App.vue" > script

```vue
<template>
  <div id="app">
    <nav_bar v-bind:propsdata="userList"></nav_bar>
    <transition name="fade">
    <router-view v-bind:propsdata="userList" v-bind:userselect="select" v-bind:selecteddata="userSelectResult" v-on:saved="getUserList" v-
    </transition>
    <!-- v-bind:하위컴포넌트 속성명="상위 컴포넌트 전달할 데이터명" -->
    <Main_footer></Main_footer>
  </div>
</template>

<script>
import axios from 'axios';
import footer from './components/include/footer.vue';
import nav_bar from './components/include/nav_bar.vue';

let url = "http://127.0.0.1:8000/";  // 장고 drf 서버 주소

export default {
  name: 'App',
  components: {
    'Main_footer': footer,
    'nav_bar': nav_bar,
  },
  data:() =>{
    return{
      userList:[],
      userSelectResult:[],
      user:{
```

SPA(Single Page Application)

상위 Vue인 App.vue file에 Header.vue, Footer.vue와 같이
모든 Page에 출력되는 Include와Vue-Router의 구조를 선언하고

Page 이동시 Main화면만 교체, 출력하여 싱글 페이지 애플리케이션 구현

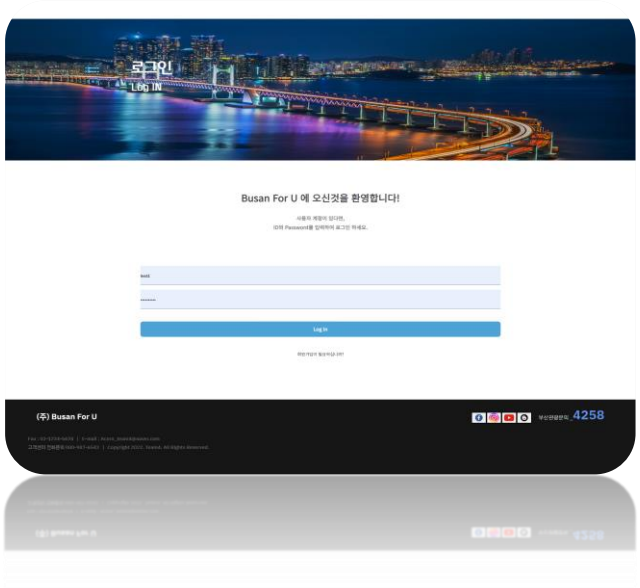
VUE ROUTER

5.제작방법

Frontend

Vue 와 Django REST framework 간 Rest 통신

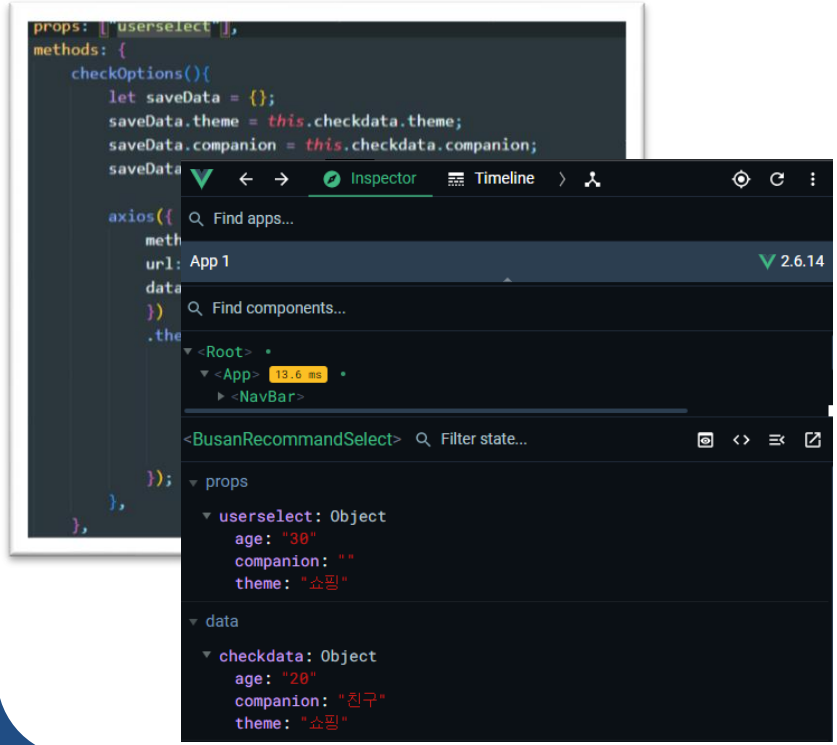
Front와 Back을 분리하여 Rest API 적용
Promise 기반의 HTTP 통신 라이브러리 Axios를 통해서 Data를 Jason file로 변환하여 CRUD 진행



Django REST API 와 REST 연결로 Token을 통한 사용자 Login 인증정보 확인
페이지 전환 시 Login 유지

5.제작방법

Frontend



Props Data 활용 Data bind

Props 기능을 활용 상위 Vue file, 하위 Vue file 간의 Data transfer 구현

Emit을 사용 하위 Vue file에서 상위 Vue file로 Data load
Props를 사용 상위 Vue file에서 하위 Vue file로 Data transfer

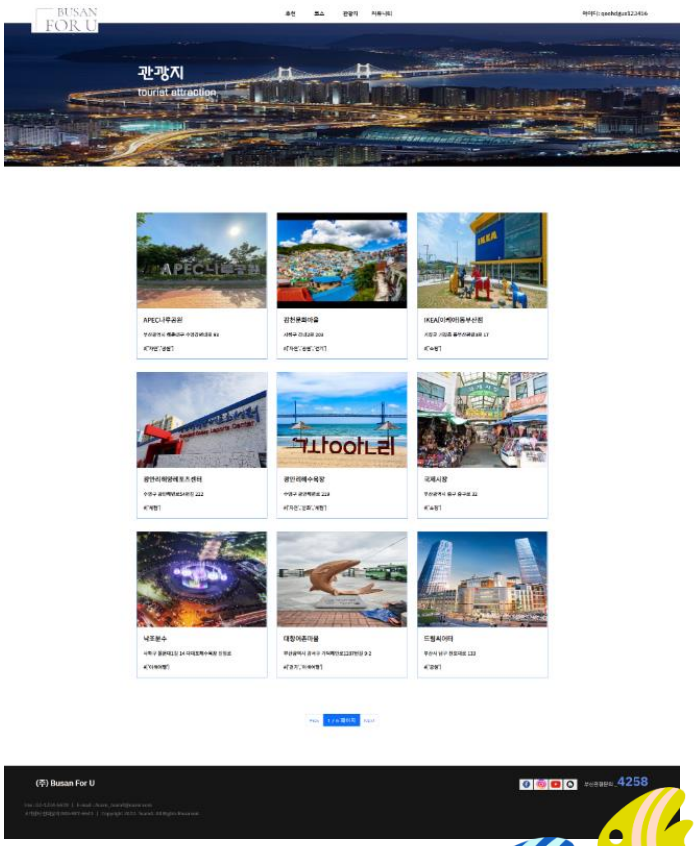
Frontend

BusanCommunitydetail.vue 1

busanproject > Busan > busan-vue > src > components > BusanCommunitydetail.vue > {} "BusanCommunitydetail.vue" > template > m

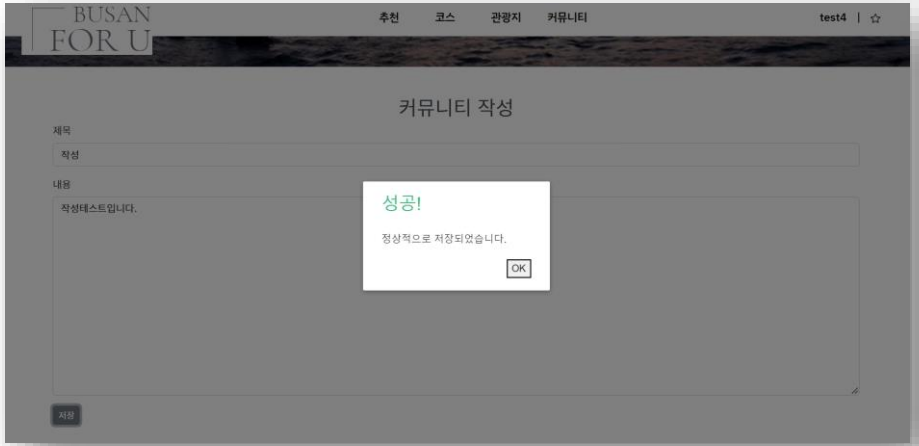
```
    },
    getPosts() {
     axios.get(`/api/v1/board/${this.$route.params.id}/`)
     .then((response) => this.items = response.data)
     .catch((err) => console.log(err))
    },
    readMode() {
      this.disabled = true
    },
    setMode() {
      this.disabled = false
    },
    setPosts() {
      let params = {
        "title" : this.items.title,
        "content" : this.items.content,
      }
      axios.put(`/api/v1/board/${this.$route.params.id}/`,
      JSON.stringify(params), {headers: { 'content-type':
      'application/json',
      // 나중에 받아올 토큰값을 적는다. 현재 임시로 test3의 토큰값을 적어놓음.
      // 'Authorization': 'token d548c2da5d0b412017c3bad825397d0427f8e956ff7d45c4b9b940d05976458d'
      'Authorization': `token ${this.token}`
      }}
      ).then(res => {
        console.log(res.data)
        this.showModal = !this.showModal;

        this.handleService()
      }).catch(e => {
```

CRUD의 GET, POST시 발생하는
Response의 데이터를 받아
MySQL DB에 있는 자료들을 출력.



5.제작방법

Frontend



Vue.js 의 기능인 애니메이션 트랜지션, 모달 적용.
Bootstrap-Vue, CSS3 를 활용 UI 구현

```
            </div>
    </div>
    <div class="box" style="height: 2300px;">
    <br>
    <h2 class="text-center">관광지</h2>
    <br>
    <BusanTouristspotList :list-array="items"></BusanTouristspotList>
    </div>
  </main>
</template>

<script>
import axios from 'axios';
import BusanTouristspotList from './BusanTouristspotlist.vue';

export default {
 components: { BusanTouristspotList },
 data() {
     return {
         items: []
     }
 },
```

Vue.js 문법 적용으로 Script Part 간소화, 모듈화

5.제작방법

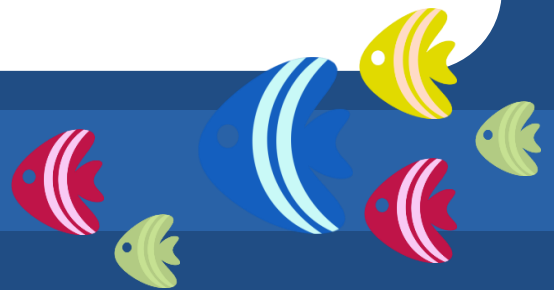
Backend

Serializer : Board, Touristsopt

fields의 데이터를 json 형식으로 전달하는 역할

```python
class BoardSerializer(serializers.ModelSerializer):
    class Meta:
        model = Board
        fields = [
            "id",
            "author_name",
            "title",
            "content",
            "dt_created",
            "dt_modified",
            "readcnt",
        ]

    author_name = serializers.SerializerMethodField("get_authors_name")

    def get_authors_name(self, obj):
        return obj.author.name
```

```python
class TouristSpotSerializer(serializers.ModelSerializer):
    class Meta:
        model = TouristSpot
        fields = (
            "id",
            "name",
            "description",
            "theme",
            "address",
            "get_image",
            "get_thumbnail"
        )
```



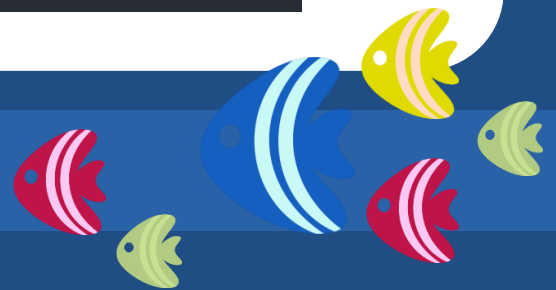
5.제작방법

Backend

Serializer : User

```python
class CreateUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ["user_id","email", "name", "password"]
        extra_kwagrs = {
            "password": {"write_only": True},
            "username": {"read_only": True},
        }

    def validate_password(self, value):
        try:
            validators.validate_password(value)
        except ValidationError as exc:
            raise serializers.ValidationError(str(exc))
        return value

    def create(self, validated_data):
        user = super().create(validated_data)
        user.set_password(validated_data["password"])

        user.is_active = True
        user.save()
        return user
```

```python
# 로그인
class LoginUserSerializer(serializers.Serializer):
    user_id = serializers.CharField()
    password = serializers.CharField()

    def validate(self, data):
        user = authenticate(**data)
        if user and user.is_active:
            user.last_login = timezone.now()
            user.save(update_fields=["last_login"])
            return user
        raise serializers.ValidationError("Unable to log


# 접속 유지 확인
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ["user_id", "name", "email"]
```



5.제작방법

Backend

View : Board

BoardAPI

```python
class BoardAPI(generics.ListCreateAPIView):
    queryset = Board.objects.all()
    serializer_class = BoardSerializer
    authentication_classes = [TokenAuthentication]
    permission_classes = [IsAuthenticatedOrReadOnly]

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

GET, POST Method 두 가지의 요청이 가능

BoardAPI의 경우 로그인 하지 않아도 게시물을 읽을 수 있지만 새로운 게시물을 작성할 수는 없음

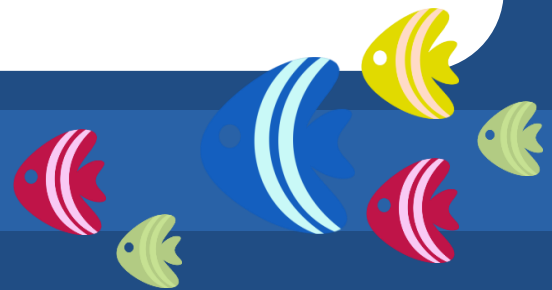
BoardDetailAPI

```python
class BoardDetailAPI(generics.RetrieveUpdateDestroyAPIView):
    queryset = Board.objects.all()

    serializer_class = BoardDetailSerializer
    authentication_classes = [TokenAuthentication]
    permission_classes = [IsOwnerOrReadOnly]
```

GET, PUT, DELETE Method 세 가지의 요청이 가능

세부 게시물도 역시 로그인 하지 않은 유저도 보는 것은 가능하지만 수정과 삭제는 로그인한 유저만 가능함.

이를 구현한 것이 ' permission_classes ' 의 IsOwnerOrReadOnly이며 이는 permission을 커스텀 하여 만듦



Backend

View : Touristspot

TouristSpotsList

GET, POST Method 두 가지의 요청 가능

TouristSpotDetail

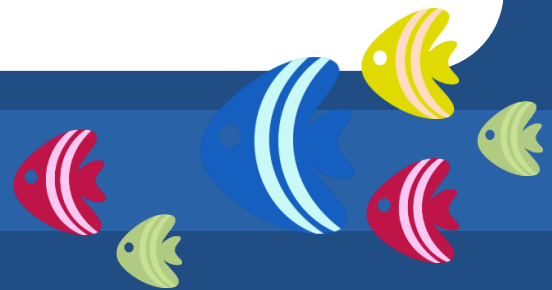
GET, PUT, DELETE Method 세 가지의 요청

```python
class TouristSpotsList(generics.ListCreateAPIView):
    queryset = TouristSpot.objects.all()
    serializer_class = TouristSpotSerializer

class TouristSpotDetail(generics.RetrieveUpdateDestroyAPIView):
    queryset = TouristSpot.objects.all()
    serializer_class = TouristSpotSerializer
```

```python
@api_view(['POST'])
def recommendList(request):
    theme = request.data.get('theme', '')
    companion = request.data.get('companion', '')
    age = request.data.get('age', '')

    a = recommed(theme, companion, age)
    arr = a.tolist()
    spots = TouristSpot.objects.filter(Q(name=arr[0]) | Q(name=arr[1]) | Q(name=ar

    serializer = TouristSpotSerializer(spots, many=True)
    return Response(serializer.data)
```

recommendList

클라이언트로 부터 테마, 동반자, 나이를 받아와서 제작한 AI 추천 모델에 넣고 나온 결과값 관광지 데이터를 3개를 내보냄



5.제작방법

Backend

View : User

```python
class RegistrationAPI(generics.GenericAPIView):
    serializer_class = CreateUserSerializer

    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.save()
        return Response(
            {
                "user": UserSerializer(
                    user, context=self.get_serializer_context(
                ).data,
                "token": AuthToken.objects.create(user)[1],
            }
        )
```

```python
class LoginAPI(generics.GenericAPIView):
    serializer_class = LoginUserSerializer

    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data
        return Response(
            {
                "user": UserSerializer(
                    user, context=self.get_serializer_context()
                ).data,
                "token": AuthToken.objects.create(user)[1],
            }
        )


class UserAPI(generics.RetrieveUpdateAPIView):
    permission_classes = [
        permissions.IsAuthenticated,
    ]
    serializer_class = UserSerializer

    def get_object(self):
        return self.request.user
```

RegistrationAPI

knox를 이용해 가입시 유저 token을 부여

LoginAPI

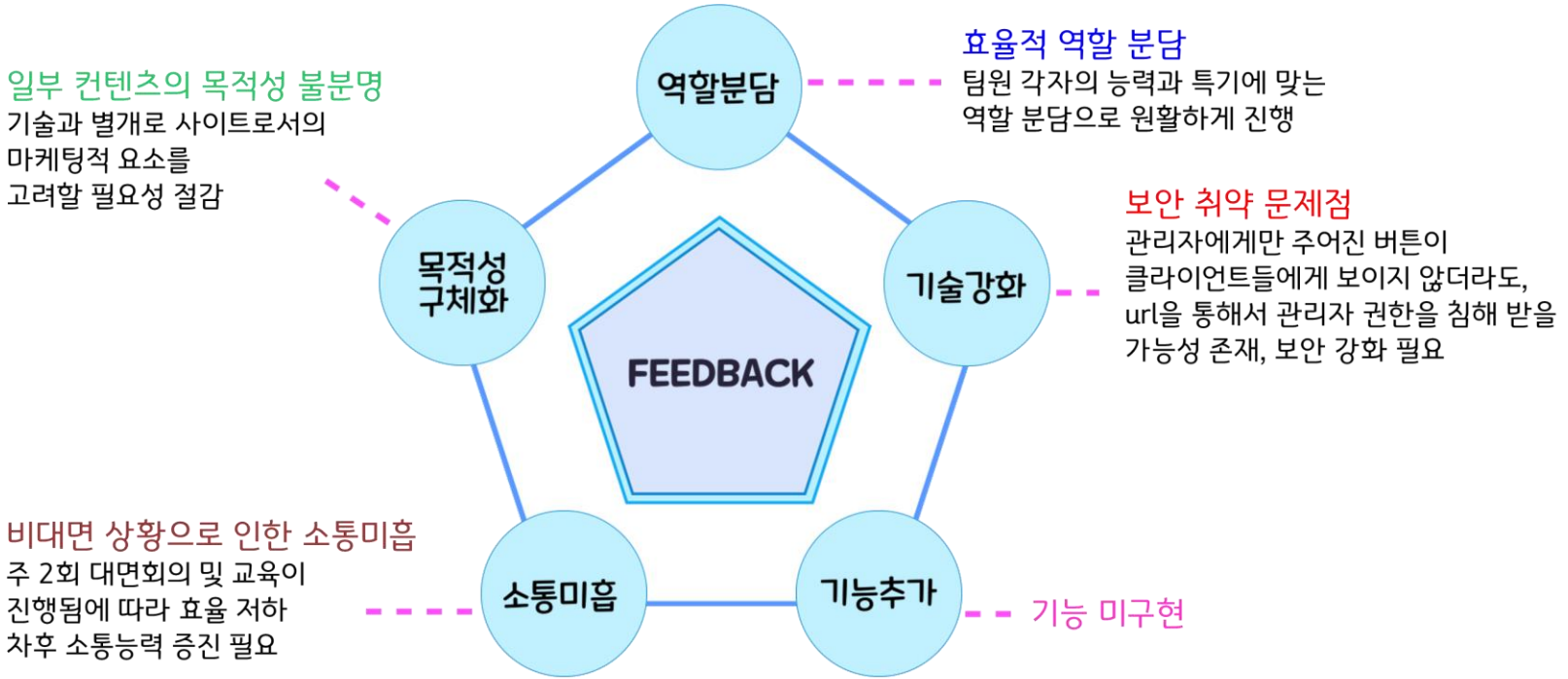
이메일 아이디와 비밀번호로 로그인하며, 로그인시 token이 발급

UserAPI

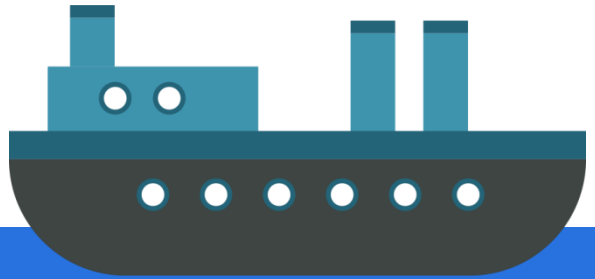
user token으로 인증된 유저의 아이디와 이름을 확인

5.제작방법

사이트 시연



6. 개선사항 및 발전방향논의



THANK YOU !