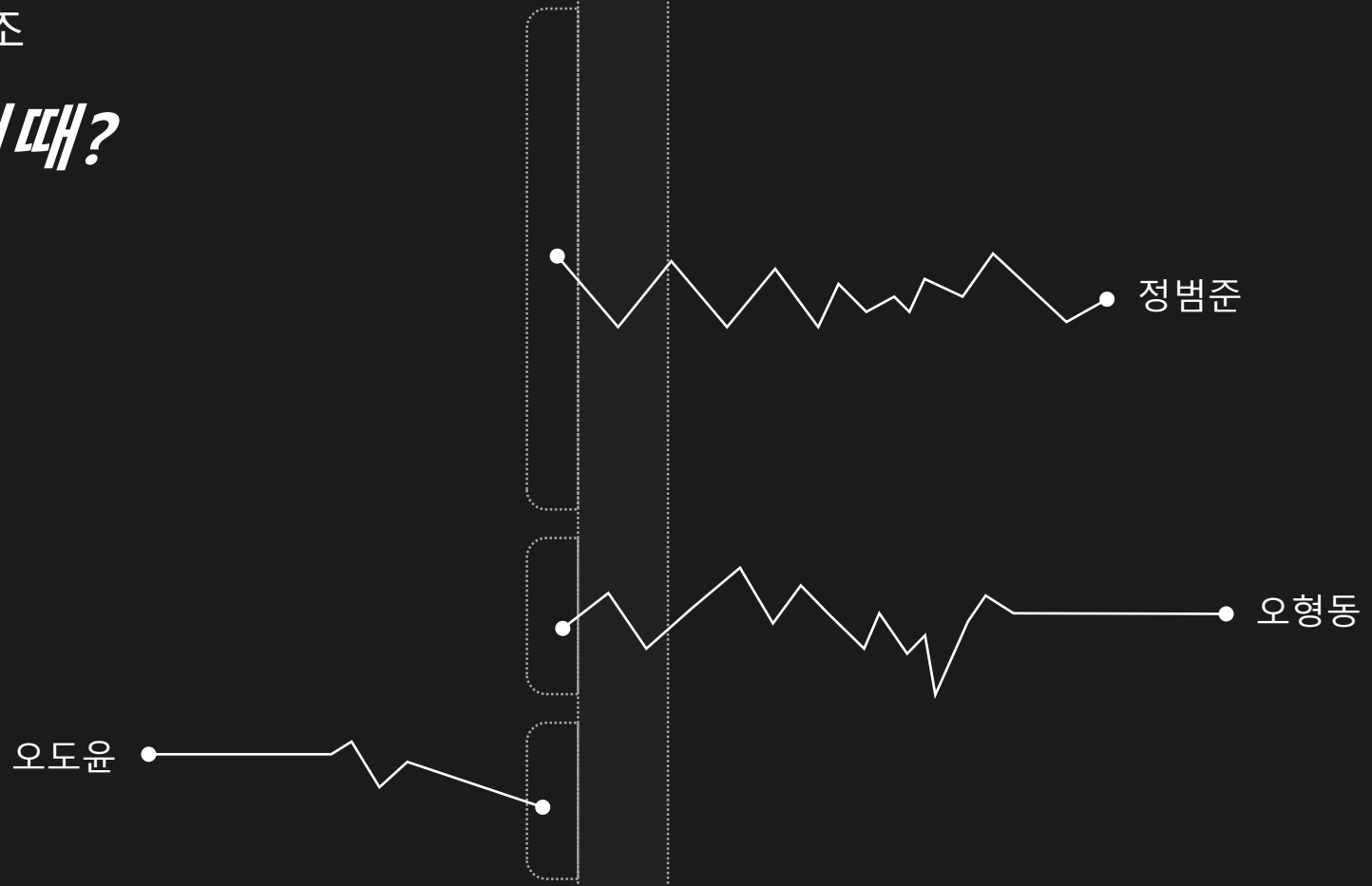
오마카세 쿠다사이조

오늘 여기 어때?



목 차

프로젝트 소개

- 개발 동기
- 프로젝트 목표
- 기능 구현

웹 크롤링

- 구석구석 홈페이지
- 코드

MYSQL

- 테이블 구성
- 데이터 입력
- Django와 연계

Django

- 모델 구현
- 목적에 맞는 html 페이지 구성

프로젝트 소개

개발 동기

✓ 정보과다의 시대, 갈 곳은 많고 선택하기는 어렵다.

✓ 데이트 하는 연인에게도 마찬가지

✓ 데이트 코스를 위해 고민하는 연인들을 위해 음식점, 카페, 가볼만 한 관광지를 추천하는 서비스 제공

프로젝트 목표

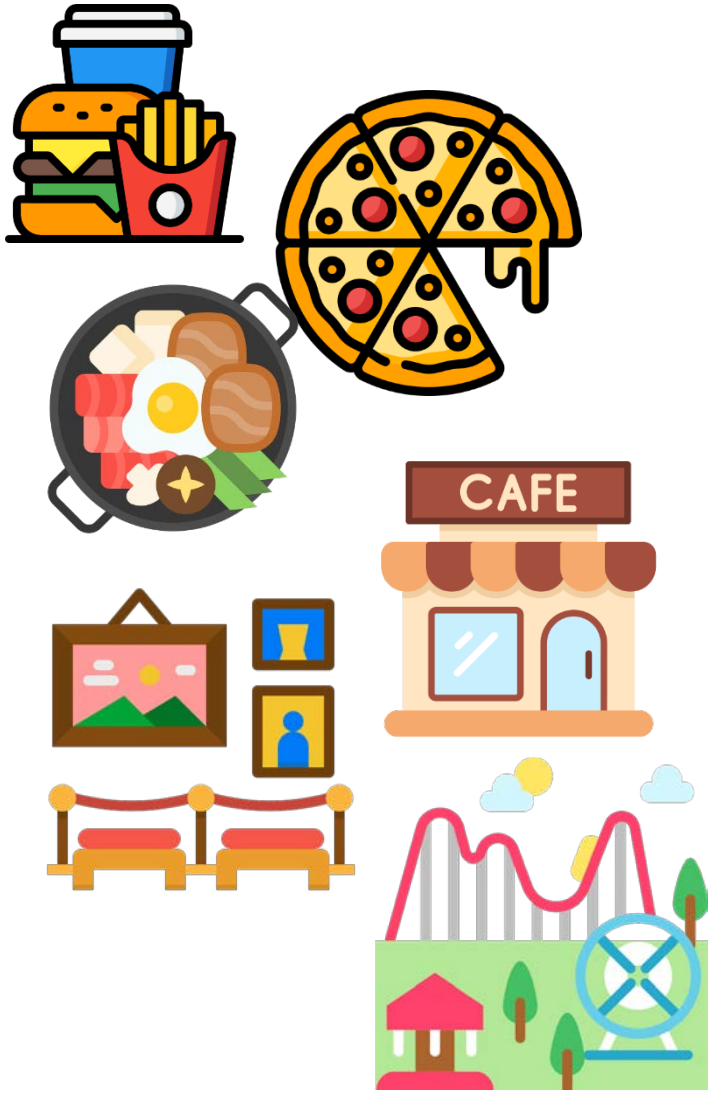
✓ 서울 중심을 기준으로 데이트 코스를 연인에게 추천

✓ 추천 코스를 방문한 이들의 리뷰를 모아 커뮤니티 구축 및 사용자간 소통 활성화

기능 구현

✓ 지역을 기준으로 장소 추천
- 맛집
- 카페
- 관광지

✓ 회원이 참조한 데이트 코스에 대한 리뷰 작성 기능



웹 크롤링 대상사이트

✓ 음식점, 카페, 가볼만한 곳 모두에 대한 정보를 가지고 있는 사이트 탐색

✓ 대상 사이트 : 구석구석 홈페이지

- 전국에 있는 맛집, 관광지, 카페들에 대한 정보와 축제정보를 공유하는 사이트

웹 크롤링 방법

✓ 프로젝트 기간의 제약으로 크롤링 대상을 한정

- 서울 중구를 기준으로 8km 이내의 음식점, 카페, 가볼만한 곳 검색

✓ 상세정보 검색을 위해 검색한 장소들의 상호명을 리스트에 추가

- 식당 : rest_list
- 카페 : cafe_list
- 가볼만한 곳 : place_list

✓ 상세정보를 담을 리스트 선언

- 식당 : rest_detail_list
- 카페 : cafe_detail_list
- 가볼만한 곳 : place_detail_list

코드 작성

```
# 동적 크롤링을 사용하기 위한 기본적인 모듈 import
from selenium.webdriver import Chrome, ChromeService
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup
from urllib.request import urlopen
from selenium.webdriver.common.keys import Keys
import time

# 동적 크롤링을 위한 driver 준비 및 url 입력, 받아오기
driver = Chrome(service = ChromeService("chromedriver.exe"))
url = "https://korean.visitkorea.or.kr/mylocation/mylocation.do"
driver.get(url)
time.sleep(5)

# 상세페이지 크롤링을 위해 지도 검색에서 식당, 카페, 가볼만한 곳의 상호명을 리스트 형태로 저장
rest_list = list()
for i in driver.find_elements(by = By.CLASS_NAME, value = "tit > a"):
    rest_list.append(i.text)

# 상세페이지 정보를 리스트 형태로 저장
rest_detail_list = list()
rest_num = 0
```

코드 작성

```
# 상세페이지에 접근하기 위해 반복문 사용
 for i in rest_list:
    title_check = 0
    url = "https://korean.visitkorea.or.kr/search/search_list.do?keyword=" + i
    driver.get(url)
    time.sleep(3)
    driver.find_element(By.CSS_SELECTOR, "#tabView1 > a").click()
    time.sleep(3)

    soup = BeautifulSoup(driver.page_source, "html.parser")
    time.sleep(3)

    # 키워드로 검색 후 카드가 존재할 경우, 카드를 클릭하고
    # 그 외의 경우엔 리스트에 담겨진 상호명과 일치할 경우 클릭
    # title_check 라는 변수를 선언하여 검색결과가 있을 경우 title_check를 1로 변환하고
    # 검색결과가 존재하지 않을 경우 다음 페이지로 넘어 가게끔 if 문을 활용하여 진행
    if(soup.find("a", id = "properTitle",string = i)):
        driver.find_element(By.CSS_SELECTOR, "#properTitle").click()
        title_check = 1
    else:
        name = driver.find_elements(By.CSS_SELECTOR, "#listBody > ul > li > div.area_txt > div.tit > a")
        for j in name:
            print(j.text, i)
            if j.text == i:
                j.click()
                title_check = 1
                break
```

코드 작성

```
if (title_check == 0):
    continue

time.sleep(4)

# 상세페이지 크롤링을 위해 soup에 데이터 저장
soup = BeautifulSoup(driver.page_source, "html.parser")
time.sleep(3)

# 상세페이지 크롤링
for tag in soup.select("br"):
    tag.extract()

rest_name = i
rest_img = rest_num
rest_addr = soup.find("strong", string= "주소").next_sibling.string
if soup.find("strong", string= "대표메뉴") is None:
    rest_menu = "식사"
else:
    rest_menu = soup.find("strong", string= "대표메뉴").next_sibling.string
if soup.find("strong", string= "영업시간") is None:
    rest_time = "문의바람"
else:
    rest_time = soup.find("strong", string= "영업시간").next_sibling.text
```

웹 크롤링

코드 작성

```
    if (driver.find_element(By.CSS_SELECTOR, "#topCp > div > h3 > em").text != ""):
        rest_detail = driver.find_element(By.CSS_SELECTOR, "#topCp > div > h3 > em").text
    else:
        rest_detail = driver.find_element(By.CSS_SELECTOR, "#detailGo > div:nth-child(2) > div > div.inr_wrap > div
> p").text

    rest_store = [rest_name,rest_img,rest_addr,rest_menu,rest_time,rest_detail]
    rest_detail_list.append(rest_store)

    # 이미지 저장
    with open(f"./rest_img/{rest_num}.jpg", "wb") as f:
        f.write(urlopen(soup.select_one("div.swiper-slide.swiper-slide-active > img").get("src")).read())



    rest_num = rest_num + 1
```

```
# 상세페이지 크롤링에서 받아오는 정보를 정리하기 위해 .을 기준으로
# 앞부분의 정보만 크롤링
for i in rest_detail_list:
    i[5] = i[5].split(".")[0]

# 받아온 데이터를 커서를 사용하여 DB에 저장
import pymysql

conn = pymysql.connect(host = "127.0.0.1", user = "root", password = "0000", db = 'team_pj', charset = 'utf8', port
= 3306)
cur = conn.cursor()

for data in rest_detail_list:
    sql = f"""insert into rest (rest_name,rest_img,rest_addr,rest_menu,rest_time,rest_detail)
        values ('{data[0]}','{data[1]}','{data[2]}','{data[3]}','{data[4]}','{data[5]}')"""
    cur.execute(sql)

conn.commit()
cur.close()
conn.close()
```

데이터베이스 구축

✓ 데이터베이스 : MYSQL 에 커서를 이용하여 데이터 저장

- 해당 데이터는 데이터 내보내기를 통해 CSV로 추출
- 데이터테이블을 새로 구축

✓ Dbeaver를 통해 구축된 데이터베이스 확인

- 식당 : rest table
- 카페 : cafe table
- 가볼만한 곳 : place table

MYSQL

코드 작성

```
# DBeaver
# 크롤링한 정보를 DB에 입력하기 위해 데이터베이스 생성
create database team_pj;

USE team_pj;

# 식당, 카페, 가볼만한 곳에 맞는 테이블 생성 후 크롤링한 정보를 입력받음
# 해당 정보를 csv로 저장하여 데이터 유지
create table cafe (
      cafe_num int auto_increment primary key,
      cafe_name varchar(30) not null,
      cafe_img int not null,
      cafe_addr varchar(40) not null,
      cafe_menu varchar(50) not null,
      cafe_time varchar(75) not null,
      cafe_detail varchar(120) not null
);

create table place(
      place_num int auto_increment primary key,
      place_name varchar(50) not null,
      place_img int not null,
      place_addr varchar(40) not null,
      place_time varchar(400) not null,
      place_detail varchar(300) not null
);

create table rest(
      rest_num int auto_increment primary key,
      rest_name varchar(50) not null,
      rest_img int not null,
      rest_addr varchar(40) not null,
      rest_menu varchar(150) not null,
      rest_time varchar(400) not null,
      rest_detail varchar(700) not null
```

코드 작성

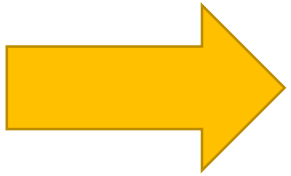
```
# 데이터 전처리
SELECT *
FROM place p
WHERE p.place_addr NOT LIKE '서울%'

DELETE FROM place
WHERE place_addr NOT LIKE '서울%';

SELECT *
FROM rest r
WHERE r.rest_addr NOT LIKE '서울%'

DELETE FROM rest
WHERE rest_addr  NOT LIKE '서울%';

SELECT *
FROM cafe c
WHERE c.cafe_addr NOT LIKE '서울%'

DELETE FROM cafe
WHERE cafe_addr NOT LIKE '서울%';

-- rest 테이블의 중복 데이터 삭제
select rest_name ,count(rest_name)
from rest group by rest_name having count(rest_name) > 1;
select * from rest where rest_name = "진동횟집";
delete from rest where rest_name = "진동횟집" and rest_num = 37;
select * from rest where rest_name = "동경";
delete from rest where rest_name = "동경" and rest_num = 53;
select * from rest where rest_name = "문화식당";
delete from rest where rest_name = "문화식당" and rest_num = 335;
```

데이터베이스 구축

✓ MYSQL과 Django 연결

- 파이참에서 가상환경 설정 및 프로젝트 생성
- 연동될 앱 작성
- 터미널에서 명령어 입력 : python manage.py inspectdb > models.py
  - 해당 과정을 진행하면 models.py에 직접 작성할 필요없이 DB의 저장된 테이블을 불러옴
  - Migrations 진행

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}
```



```
# 데이터베이스와 연결되어 아래와 같이 DATABASES 정보 갱신
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'team_pj',
        'USER': 'root',
        'PASSWORD': '0000',
        'HOST': '127.0.0.1',
        'PORT': '3307',
    }
}
```

웹 페이지 설계 및 기능

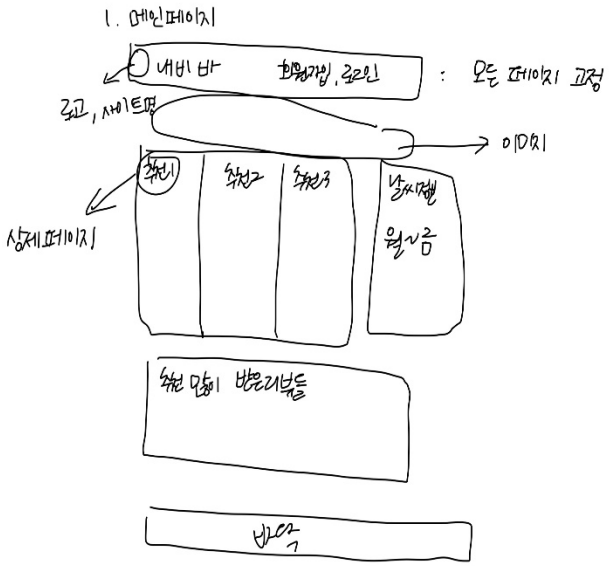
NOW_DATE PROJECT

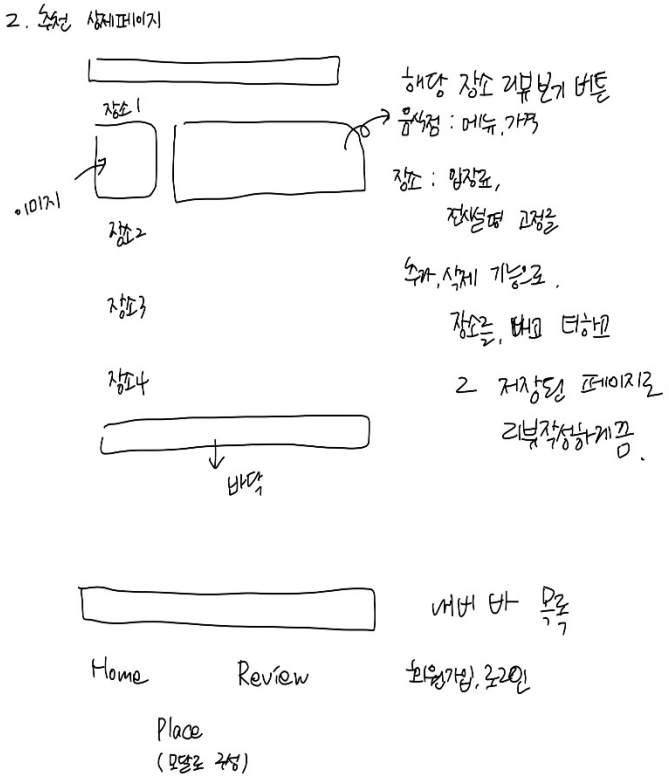
DATE APP

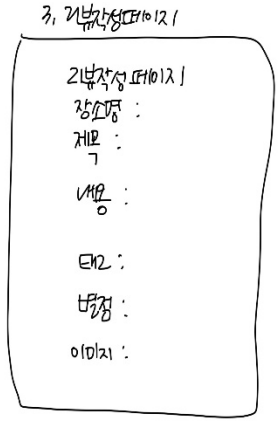
- 추천 기능 구현
- 기본페이지 구성
- 장소페이지 구성
- 리뷰페이지 구성

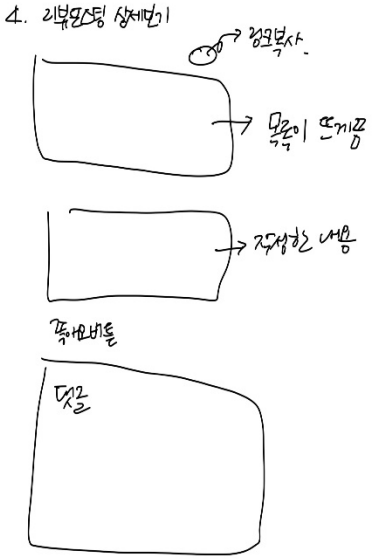
USERS APP

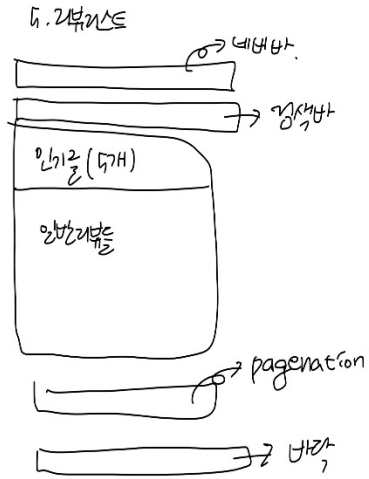
- 로그인 / 로그아웃 구현
  - main_area.html
- 회원가입 기능 구현
  - login.html
  - signup.html
  - form_errors.html











6. 회원가입페이지

ID

password

password 확인

이메일

음식 태그 > 체크박스

취미 태그

음식 : 한식, 중식, 일식, 양식, 분식, 기타.

취미 :

[redacted]

코드 작성

base.html
navbar.html
footer.html

```
{% load static %}
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>{% block head_title %}오늘, 여기 어때?{% endblock %}</title>
    <!-- bootstrap -->
    <link href="{% static 'date/bootstrap/bootstrap.min.css' %}" rel="stylesheet" type="text/css">
    <!-- fontawesome -->
    <script src = "https://kitfontawesome.com/539842d0c4.js" crossorigin="anonymous"></script>
    <!-- 홈페이지 전체 폰트 적용 -->
    <link href="{% static 'style.css' %}" rel="stylesheet" type="text/css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap" rel="stylesheet">
</head>
<body class="d-flex flex-column min-vh-100">

{% include "date/navbar.html" %}
  <div class="container my-3">
        <div class="container" id = "main-area">
            {% block main_area %}
            {% endblock %}
        </div>
  </div>
{% block content %}
{% endblock %}

{% block signup %}
{% endblock %}

{% include "date/footer.html" %}

<!-- 부트스트랩 사용을 위한 스크립트 -->
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.min.js" integrity="sha384-
Rx+T1VzGupg4BHQYs2gCW9It+akI2MM/mndMCy36UVfodzcJcF0GGLxZIzObiEfa" crossorigin="anonymous"></script>

</body>
</html>
```

코드 작성

base.html
navbar.html
footer.html

```
<nav class="navbar navbar-expand-lg" style= "background-color:#FC4C64;">
    <div class="container">
      <div class="container-fluid">
        <a class="navbar-brand text-white" href="/">오늘, 여기 어때?</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle
navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNavDropdown">
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link active text-white" aria-current="page" href="/">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link text-white" href="/place/">Place</a>
            </li>
            <li class="nav-item">
              <a class="nav-link text-white" href="/review_list/">Review</a>
            </li>
          </ul>
```

```
        <!--로그인 버튼 생성 me-auto = ms-auto 같은 개념-->
        <ul class="navbar-nav ms-auto">
         {% if user.is_authenticated %}
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle text-white" href="#" role="button" data-bs-
oggle="dropdown" aria-expanded="false">
                    {% if user.socialaccount_set.all.first.get_avatar_url %}
                      <img class="rounded-circle" width="25px" src="{{
ser.socialaccount_set.all.first.get_avatar_url }}">
                    {% else %}
                      <img class="rounded-circle" width="25px" src="https://api.dicebear.com/6.x/pixel-
t/svg?seed={{ user.username }}">
                    {% endif %}
                     
                    {{ user.username }}
                </a>
                <ul class="dropdown-menu">
                  <li><a class="dropdown-item" href="/myreview/"> 내 리뷰 보기</a></li>
                  <li><a class="dropdown-item" href="{% url 'users:logout' %}"> log out</a></li>
                </ul>
            </li>
        {% else %}
            <li class="nav-item">
                <a class="nav-link text-white" href="{% url 'users:login' %}">로그인</a>
            </li>
        {% endif %}
```

코드 작성

base.html
navbar.html
footer.html

```
<footer class="py-3 mt-auto" style= "background-color:#FC4C60;" >
    <div class="container"><p class="m-0 text-center text-white"> Now, Date ? &copy; Omakase Kudasai 2023</p>
</div>
</footer>
```

웹 페이지 설계 및 기능 – DATE APP

✓ 메인페이지(main_area.html)

- 크롤링한 데이터를 무작위 추출하여, 식당 + 카페 + 관광지의 조합으로 선택할 수 있는 3개의 추천탭 제공
- 우측 상단에 위치정보에 따른 날씨를 제공하는 위젯을 배치하여 선택에 고려할 수 있게끔 유도
- 화면 하단에 별점이 높은 리뷰들을 보여주어 편의성 제공

코드 작성

views.py

```python
#메인페이지 구성을 위한 뷰
class MainPage(ListView):
    model = Cafe    #cafe 모델사용
    template_name = "date/main_area.html"    #템플릿사용

    #가져올 데이터 설정
    def get_context_data(self, **kwargs):
        context =super().get_context_data()
        #주소 설정을 위한 데이터
        context["addr_names"] = Addr.objects.all()
        #추천코스중 카페를 구성하기 위한 데이터
        #카페데이터중 랜덤으로 하나를 가져옴
        cafe_max = Cafe.objects.all().count()
        cafe_ran = random.randint(0,cafe_max-1)
        context["cafe_recommed_1"] = Cafe.objects.all()[cafe_ran]
        cafe_ran = random.randint(0, cafe_max-1)
        context["cafe_recommed_2"] = Cafe.objects.all()[cafe_ran]
        cafe_ran = random.randint(0,cafe_max-1)
        context["cafe_recommed_3"] = Cafe.objects.all()[cafe_ran]
```

```python
#추천코스중 식당을 구성하기 위한 데이터
#식당데이터중 랜덤으로 하나를 가져옴
rest_max = Rest.objects.all().count()
rest_ran = random.randint(0, rest_max-1)
context["rest_recommed_1"] = Rest.objects.all()[rest_ran]
rest_ran = random.randint(0, rest_max-1)
context["rest_recommed_2"] = Rest.objects.all()[rest_ran]
rest_ran = random.randint(0, rest_max-1)
context["rest_recommed_3"] = Rest.objects.all()[rest_ran]

#추천코스중 관광지를 구성하기 위한 데이터
#관광지데이터중 랜덤으로 하나를 가져옴
place_max = Place.objects.all().count()
place_ran = random.randint(0, place_max-1)
context["place_recommed_1"] = Place.objects.all()[place_ran]
place_ran = random.randint(0, place_max-1)
context["place_recommed_2"] = Place.objects.all()[place_ran]
place_ran = random.randint(0, place_max-1)
context["place_recommed_3"] = Place.objects.all()[place_ran]

#별점높은 순으로 정렬후 3개를 가져옴
context["review_top"] = Review.objects.all().order_by('-score')[:3]

return context
```

코드 작성

views.py

```python
#메인페이지에서 주소를 클릭했을 때 보여질 뷰
class SelectPage(ListView):
    models = Cafe   #모델은 cafe를 사용
    template_name = "date/main_area.html"   #템플릿설정

    def get_queryset(self):
        cafe_list = Cafe.objects.all()

        return cafe_list

    #가져올 데이터를 설정
    def get_context_data(self, **kwargs):
        context =super().get_context_data()
        # 주소창으로 받아온 데이터
        q = self.kwargs["q"]
        #주소를 설정하기 위한 데이터
        context["addr_names"] = Addr.objects.all()

        # 추천코스중 카페를 구성하기 위한 데이터
        # 카페 데이터중 설정된 주소에 맞는 데이터를 랜덤으로 가져옴
        cafe_max = Cafe.objects.filter(cafe_addr__contains=q).count()
        cafe_ran = random.randint(0,cafe_max-1)
        context["cafe_recommed_1"] = Cafe.objects.filter(cafe_addr__contains=q)[cafe_ran]
        cafe_ran = random.randint(0, cafe_max-1)
        context["cafe_recommed_2"] = Cafe.objects.filter(cafe_addr__contains=q)[cafe_ran]
        cafe_ran = random.randint(0,cafe_max-1)
        context["cafe_recommed_3"] = Cafe.objects.filter(cafe_addr__contains=q)[cafe_ran]

        # 추천코스중 식당을 구성하기 위한 데이터
        # 식당 데이터중 설정된 주소에 맞는 데이터를 랜덤으로 가져옴
        rest_max = Rest.objects.filter(rest_addr__contains=q).count()
        rest_ran = random.randint(0, rest_max-1)
        context["rest_recommed_1"] = Rest.objects.filter(rest_addr__contains = q)[rest_ran]
        rest_ran = random.randint(0, rest_max-1)
        context["rest_recommed_2"] = Rest.objects.filter(rest_addr__contains = q)[rest_ran]
        rest_ran = random.randint(0, rest_max-1)
        context["rest_recommed_3"] = Rest.objects.filter(rest_addr__contains = q)[rest_ran]

        # 추천코스중 관광지를 구성하기 위한 데이터
        # 관광지 데이터중 설정된 주소에 맞는 데이터를 랜덤으로 가져옴
        place_max = Place.objects.filter(place_addr__contains=q).count()
        place_ran = random.randint(0, place_max-1)
        context["place_recommed_1"] = Place.objects.filter(place_addr__contains=q)[place_ran]
        place_ran = random.randint(0, place_max-1)
```

코드 작성

main_area.html

```
<!--  주소를 설정하기 위한 부분  -->
<div class="row">
    {% for i in addr_names %}
    <div class="col">
        <span class="badge badge-pill badge-primary"><a style="text-decoration: none; color: black"
              href="/main_area/{{i.addr_name}}/">{{i.addr_name}}</a></span>
    </div>
    {%endfor%}
    </div>
    <br/>
<!--    추천코스를 보여주기 위한 부분-->
<div class="row d-flex justify-content-end">
    <div class="col-lg-9" id = "recommandation">
        <div class="row  text-center">
            <!--   추천코스 1번-->
            <div class="col-lg-4 " id="rocom_1">
                <div class="card h-100 w-100">
                    <div class="card-header text-white" style="background-color: #FC4C60;">
                        <h3 style="font-family: 'Gowun Dodum', sans-serif; ">추천1</h3>
                    </div>
                    <div class="card-body">
                        <div class="row h-100 w-100 align-items-center">
                            <div class="col">
                            <a href="/cafedetail/{{cafe_recommed_1.cafe_name}}/" style="text-decoration:
none;color: black"><p>{{cafe_recommed_1.cafe_name}}</p></a>
```

코드 작성

main_area.html

```
        <!-- 날씨 위젯을 보여주는 부분-->
        <div class="col-lg-3" id ="weather">
            <div id="ww_88067e04ea43f" v='1.3' loc='auto' a='{"t":"horizontal","lang":"ko","sl_lpl":1,"ids":
[],"font":"Arial","sl_ics":"one","sl_sot":"celsius","cl_bkg":"#FC4C64;","cl_font":"#FFFFFF","cl_cloud":"#FFFFFF
","cl_persp":"#FFFFFF","cl_sun":"#FFC107","cl_moon":"#FFC107","cl_thund":"#FFFFFF"}'>
                Weather Data Source:
                <a href="https://wetterlang.de/wetter_14_tage/" id="ww_88067e04ea43f_u" target="_blank">
                    Wetter fur 14 tage
                </a>
            </div>
        </div>
    </div>
<!-- 날씨 위젯 스크립트 -->
<script async src="https://app2.weatherwidget.org/js/?id=ww_88067e04ea43f"></script>
{% endblock %}
```

코드 작성

main_area.html

```
<!-- 별점 높은 순으로 리뷰 3개를 보여주는 부분 -->
<div class="row text-center" id = "review">
    <div class="col-lg-12" id = "Most_like">
        <div class="row d-flex justify-content-end">
            {% for review in review_top %}
            <div class="col-lg-4">
                <div class="card h-100">
                    <div class="card-header text-white h-100" style="background-color: #FC4C60;">
                        <h3>{{ review.title }}</h3>
                    </div>
                    <div class="card-body">
                        <a>{{ review.author }}</a>
                        </br>
                        <a>{{ review.content |  truncatewords_html:40 }}</a>
                    </div>
                    <div class="card-footer text-white" style="background-color: #FC4C60;">
                        <a class="btn text-white" style="background-color: #FC4C64;"
href="/review_list/{{review.pk}}/">상세 보기</a>
                    </div>
                </div>
            </div>
            {%endfor %}
            </div>
        </div>
    </div>
</div>
```

코드 작성

main_area.html

```
                            <img src="/static/date/imgs/cafe_img/{{cafe_recommed_1.cafe_img}}.jpg" alt="
{{cafe_recommed_1.cafe_img}}"width="150" height="150">
                            <br/><br/>
                            <a href="/restdetail/{{rest_recommed_1.rest_name}}/" style="text-decoration:
none;color: black"><p>{{rest_recommed_1.rest_name}}</p></a>
                            <img src="/static/date/imgs/rest_img/{{rest_recommed_1.rest_img}}.jpg" alt="
{{rest_recommed_1.rest_img}}"width="150" height="150">
                            <br/><br/>
                            <a href="/placedetail/{{place_recommed_1.place_name}}/" style="text-decoration:
none;color: black"><p>{{place_recommed_1.place_name}}</p></a>
                            <img src="/static/date/imgs/place_img/{{place_recommed_1.place_img}}.jpg" alt="
{{place_recommed_1.place_img}}"width="150" height="150">
                            <br/><br/>
                            </div>
                        </div>
                    </div>
                    <div class="card-footer" style="background-color: #FC4C60;">
                        <a type="button" class="btn text-white" style="background-color: #FC4C64;"
href="/cos/{{cafe_recommed_1.cafe_num}}/{{rest_recommed_1.rest_num}}/{{place_recommed_1.place_num}}/">상세보기
</a>
                    </div>
                </div>
            </div>
```

웹 페이지 설계 및 기능 – DATE APP

✓ 장소 소개 페이지(place.html)

- 추천 장소 외에도 크롤링한 데이터의 다른 장소를 구별로 나타나게 함
- 목적지에 링크를 걸어두어 장소 상세정보 페이지로 링크
  - place_detail.html

코드 작성

views.py

```python
#장소페이지에서 초기화면을 보여주기위한 뷰
class PlaceList(ListView):
    models = Cafe   #모델은 cafe를 사용
    template_name = "date/place.html"   #템플릿 설정

    #가져올 데이터를 설정
    def get_context_data(self, **kwargs):
        context =super().get_context_data()
        #주소 설정을 위한 데이터
        context["addr_names"] = Addr.objects.all()

        return context
    def get_queryset(self):
        cafe_list = Cafe.objects.all()

        return cafe_list
```

```python
#장소페이지에서 카페를 클릭했을 때 보여줄 페이지
class PlaceCafe(ListView):
    models = Cafe   #모델은 cafe를 사용
    template_name = "date/place.html"   #템플릿 설정

    def get_queryset(self):
        cafe_list = Cafe.objects.all()

        return cafe_list

    #가져올 데이터를 설정
    def get_context_data(self, **kwargs):
        context = super().get_context_data()
        #주소를 설정하기 위한 데이터
        context["addr_names"] = Addr.objects.all()
        #카페,식당,관광지를 구분하기 위한 데이터
        context["crp_check"] = 1
        #카페,식당,관광지를 클릭했는지 구분하기 위한 데이터
        context["check"] = 1

        return context
```

```python
#장소페이지에서 카페와 장소를 클릭했을 때 보여줄 뷰
class PlaceCafeLoc(ListView):
    models = Cafe   #모델은 cafe를 사용
    template_name = "date/place.html"   #템플릿을 설정

    #가져올 데이터를 설정
    def get_queryset(self):
        #주소창으로 받아온 데이터
        q = self.kwargs["q"]
        #주소창으로 받아온 데이터로 리스트 조회
        cafe_list_loc = Cafe.objects.filter(cafe_addr__contains=q)

        return cafe_list_loc

    #가져올 데이터를 설정
    def get_context_data(self, **kwargs):
        context = super().get_context_data()
        # 주소를 설정하기 위한 데이터
        context["addr_names"] = Addr.objects.all()
        # 카페,식당,관광지를 구분하기 위한 데이터
        context["crp_check"] = 1
        # 카페,식당,관광지를 클릭했는지 구분하기 위한 데이터
        context["check"] = 1

        return context
```

코드 작성

place.html

```
<div class="container">
    <!--  카페,식당,관광지 중 하나를 선택하는 부분-->
    <div class="row d-flex justify-content-center">
        <div class="col-2">
            <span class="badge badge-pill badge-primary"><a style="text-decoration: none; color: black"
href="/place/cafe/">카페</a></span>
        </div>
        <div class="col-2">
            <span class="badge badge-pill badge-primary"><a style="text-decoration: none; color: black"
href="/place/rest/">식당</a></span>
        </div>
        <div class="col-2">
            <span class="badge badge-pill badge-primary"><a style="text-decoration: none; color: black"
href="/place/place/">관광지</a></span>
        </div>
</div>
<br/>
```

코드 작성

place.html

```
<!--카페,식당,관광지를 선택했을 때 주소를 설정하는 부분-->
<!-- 카페,식당,관광지중 하나를 선택해야 이부분이 보임 -->
{% if crp_check == 1 %}
    <div class="row">
        {% for i in addr_names %}
        <div class="col">
            {%if check == 1%}
                <span class="badge badge-pill badge-primary"><a style="text-decoration: none;  color: black"
href="/place/cafe/{{i.addr_name}}/">{{i.addr_name}}</a></span>
            {%endif%}
            {%if check == 2%}
                <span class="badge badge-pill badge-primary"><a style="text-decoration: none;  color: black"
href="/place/rest/{{i.addr_name}}/">{{i.addr_name}}</a></span>
            {%endif%}
            {%if check == 3%}
                <span class="badge badge-pill badge-primary"><a style="text-decoration: none;  color: black"
href="/place/place/{{i.addr_name}}/">{{i.addr_name}}</a></span>
            {%endif%}
        </div>
        {%endfor%}
    </div>
    <br/>
{%endif%}
```

코드 작성

place.html

```
<!-- 리스트를 보여주는 부분 -->
<div class="container d-flex justify-content-center">
    <div class="container">
        <!-- 카페를 골랐을 때 카페리스트가 보이는 부분 -->
        {%for cafe in cafe_list%}
            <div class="card">
                <div class="card-header" style="background-color: #FC4C60;">
                    <a href="/cafedetail/{{cafe.cafe_name}}/" style="text-decoration: none; color: white"> <h3>
{{cafe.cafe_name}}</h3></a>
                </div>
                <div class="card-body">
                    <div class="d-flex justify-content-center">
                        <img class="" src="/static/date/imgs/cafe_img/{{cafe.cafe_img}}.jpg" alt="
{{cafe.cafe_img}}"width="300">
                    </div>
                        <h5>주소</h5> <p>{{cafe.cafe_addr}}</p>
                        <h5>대표메뉴</h5> <p>{{cafe.cafe_menu}}</p>
                        <h5>영업시간</h5> <p>{{cafe.cafe_time}}</p>
                        <h5>상세설명</h5> <p>{{cafe.cafe_detail}}</p>
                </div>
            </div>
            </br>
        {% endfor %}
```

코드 작성

place_detail.html

```
<div class="container">
    {% if check == 1 %}
    <div class="card">
        <div class="card-header">
            <h3>{{cafe_detail_list.cafe_name}}</h3>
        </div>
        <div class="card-body">
            <div class="d-flex justify-content-center">
                <img class="" src="/static/date/imgs/cafe_img/{{cafe_detail_list.cafe_img}}.jpg" alt="
{{cafe_detail_list.cafe_img}}"width="300">
            </div>
            <h5>주소</h5> <p>{{cafe_detail_list.cafe_addr}}</p>
            <h5>대표메뉴</h5> <p>{{cafe_detail_list.cafe_time}}</p>
            <h5>영업시간</h5> <p>{{cafe_detail_list.cafe_time}}</p>
            <h5>상세설명</h5> <p>{{cafe_detail_list.cafe_detail}}</p>


        </div>
    </div>
    {% endif %}
```

- views.py에서 설정한 check(클릭)의 값에 따라 식당, 카페, 관광지 각각의 정보를 받아옴

웹 페이지 설계 및 기능 – DATE APP

✓ 리뷰 리스트 페이지(review_list.html)

- 추천된 장소의 목록을 받아와 리뷰를 작성
- 리스트는 작성시간이 최신순으로 상위에 배치되게끔 작업
- 해당 리뷰의 제목을 클릭하면 리뷰의 상세내용을 볼 수 있게 링크

✓ 리뷰 작성 페이지(review_form.html) / 리뷰 수정 페이지(review_update.html)

- 부트스트랩의 crispy form을 활용하여 폼 형식으로 작성하게끔 구성
- 별점 평가 기능을 활용(css와 widget 활용)
  - star_score.html

코드 작성

views.py

```python
# 리뷰 생성을 위한 view 작성(CBV)
class ReviewCreate(CreateView):
    # model은 Review 모델을 사용
    model = Review
    # Forms.py에서 작성한 ReviewWrite를 통째로 사용
    form_class = ReviewWrite
    # 템플릿 지정
    template_name = "date/review_form.html"

    # 요청이 유효할 경우의 함수 작성
    def form_valid(self, form):
        current_user = self.request.user
        # 사용자가 로그인한 상태라면
        if current_user.is_authenticated:
            # current_user를 author에 저장
            form.instance.author = current_user
            # cafe_num을 q1에 저장, 리뷰 상세페이지의 주소로 활용하기 위함
            form.instance.cafe_num = (self.kwargs["q1"])
            # rest_num을 q2에 저장, 리뷰 상세페이지의 주소로 활용하기 위함
            form.instance.rest_num = (self.kwargs["q2"])
            # place_num을 q3에 저장, 리뷰 상세페이지의 주소로 활용하기 위함
            form.instance.place_num = (self.kwargs["q3"])
            # 태그와 관련된 작업을 하기 전에 form_valid() 결과값을 response에 저장
            response = super().form_valid(form)
            # 저장된 response 값을 반환
            return response
        # 그 외의 경우 대문페이지로 이동
        else:
            return redirect("/")

    # 성공할 경우 my_review 라는 주소를 가진 페이지로 이동
    def get_success_url(self):
        return reverse('my_review')
```

[redacted]

코드 작성

views.py

```python
# 리뷰 상세페이지 구현(CBV)
class ReviewDetail(DetailView):
    # model은 Review 모델을 사용
    model = Review
    # 템플릿 지정
    template_name = "date/review_detail.html"

    # 작성된 리뷰를 불러오기 위해 함수 작성
    def get_context_data(self, **kwargs):
        # get_context_data 메서드를 호출하여 초기 컨텍스트를 가져옴
        context = super().get_context_data()

        pk = (self.kwargs["pk"])
        # review에 Review모델에 작성된 내용을 pk를 기준으로 읽어들임
        review = Review.objects.get(pk=pk)

        # 추천이 다 다르게 진행되기 때문에 해당 정보를 불러오기 위해 번호 정보를 지정
        # Review모델에서 선언한 cafe_num, rest_num, place_num 활용
        cafe = review.cafe_num
        rest = review.rest_num
        place = review.place_num

        # context를 통해 카페, 식당, 관광지의 정보와 별점 평가된 정보를 불러옴
        context["cafe_detail_list"] = Cafe.objects.get(cafe_num=cafe)
        context["rest_detail_list"] = Rest.objects.get(rest_num=rest)
        context["place_detail_list"] = Place.objects.get(place_num=place)
        context["form"] = Star(instance=review)
        # context를 반환
        return context
```

[redacted]

코드 작성

views.py

```
# 리뷰 수정 view 작성(FBV)
def ReviewUpdate(request, pk):
    # 이전 글의 데이터를 받아 옴
    review = get_object_or_404(Review, pk=pk)

    # 글을 수정하기 위해 페이지에 접속 후 제출을 눌렀을 때, POST 방식을 사용한다는 전제를 두고 있기 때문에
    # form = ReviewWrite(request.POST, instance = post)로 활용
    if request.method == "POST":
        form = ReviewWrite(request.POST, instance = review)

        # form이 유효할 경우
        if form.is_valid():
            review = form.save(commit=False)
            # 수정된 내용을 저장
            review.save()
            # 수정하기를 눌렀던 상세페이지로 돌아가게 함
            return redirect('review_detail', pk = review.pk)

    # 글을 수정하기 위해 페이지에 처음 접속했을 때(url로 get방식을 활용하기 때문에,
    # form = ReviewWrite(instance = review)에 request.POST를 집어넣을 필요가 없음
    else:
        form = ReviewWrite(instance = review)
        return render(request, 'date/review_update.html', {"form" : form})
```

코드 작성

```python
# 내가 작성한 리뷰만 볼 수 있는 View(CBV)
class MyReview(ListView):
    # model은 Review 모델을 사용
    model = Review
    # 페이지네이션 기능 사용(Django에 탑재된 기능, import 필요)
    paginate_by = 5
    # 게시글 최신순 정렬
    ordering = ['-created_at']
    # 템플릿 설정
    template_name = "date/my_review.html"

    # 내가 작성한 리뷰를 불러오기 위해 함수 작성
    def get_context_data(self, **kwargs):
        context = super().get_context_data()
        # current_user : 현재 로그인된 사용자를 나타내는 속성
        current_user = self.request.user
        # 작성자가 현재 로그인된 사용자인 Review의 정보만 담아올 수 있게 함
        context["review_list"] = Review.objects.filter(author=current_user)
        # context 값 출력
        return context
```

```python
# 리뷰 삭제를 위한 view 작성(FBV)
def ReviewDelete(request, pk):
    # review의 내용을 담아옴
    review = get_object_or_404(Review, pk=pk)

    # 해당 삭제요청을 한 사용자와 작성자가 일치하는 경우
    if request.user.is_authenticated and request.user == review.author:
        # 리뷰를 삭제하고 리뷰의 상세페이지로 리다이렉트
        review.delete()
        return redirect("/myreview/")
    # 그 외의 경우에는 권한 거부 예외를 발생시킴
    else:
        PermissionDenied
```

코드 작성

widgets.py

```python
# 별점평가에 사용하기 위해 starWidget 클래스 작성
# forms의 형태를 사용하고 TextInput을 통해 값을 입력받음
class starWidget(forms.TextInput):
    # input_type 설정
    input_type = 'rating'
    # 템플릿 지정
    template_name = "date/star_score.html"

    class Media:
        # 해당 위젯에서 사용할 css, js 불러오기
        css = {
            'all': [
                'widgets/rateit.css',
            ],
        }
        js = [
            "//code.jquery.com/jquery-3.4.1.min.js",
            'widgets/jquery.rateit.min.js',
        ]

    # star_score.html에 사용되는 attr 생성(별의 갯수를 지정하기 위함)
    def build_attrs(self, *args, **kwargs):
        attrs = super().build_attrs(*args, **kwargs)
        # 최소, 0개, 최대 5개, 1칸씩 이동 가능하도록 범위 설정
        attrs.update({
            'min': 0,
            'max': 5,
            'step': 1,
        })
        # attr 값 출력
        return attrs
```

코드 작성

star_score.html

```
<!-- static 폴더 경로 로드 -->
{% load static %}
<!-- 별점을 입력받기 위해 Django 내부에 딸린 input.html을 포함시킴 -->
{% include "django/forms/widgets/input.html" %}


{{ form.media }}
<!-- 별모양 표시하도록 div 작성-->
<!-- input 타입은 표현되지 않도록 display:none으로 설정-->
<div id="star_{{ widget.attrs.id }}" class="rateit" data-rateit-backingfld="#{{ widget.attrs.id }}"></div>
<input type="rating" name="score" id="id_score" min="0" max="5" step="1" style="display: none;">
```

웹 페이지 설계 및 기능 – USERS APP

✓ 로그인, 로그아웃, 회원가입 등의 기능을 구현

- login.html : 로그인
- signup.html : 회원가입
- forms_errors.html : 에러 발생시 출력될 페이지 작성

코드 작성

views.py

```python
def signup(request): # signup 함수 선언
    if request.method == "POST": # 만약 request.method가 POST라면
        form = UserForm(request.POST) # UserForm은 사용자 정보를 입력받는 Form 클래스
        if form.is_valid(): # form의 유효성 검사(필수 데이터 입력과 형식이 올바른지)
            form.save() # 유효하다면 저장
            username = form.cleaned_data.get('username') #'username'을 form에서 가져와 username에 선언
            raw_password = form.cleaned_data.get('password1') #'password1'을 form에서 가져와 raw_password에 선언
            user = authenticate(username=username, password=raw_password)  # 사용자 인증
            login(request, user)  # 로그인 상태로 변경
            return redirect('/') # 127.0.0.1:8000/ 페이지로 이동
    else: # request.method가 POST가 아니라면
        form = UserForm() # 유효하지 않은 요청이라면 다시 사용자 정보를 입력하는 폼 생성
    return render(request, 'users/signup.html', {'form': form})
    # 폼을 포함한 'users/signup.html'을 렌더링해서 보여줌.
```

코드 작성

login.html

```
<form method="post" action="{% url 'users:login' %}"> <!-- POST 메소드로 데이터를 서버에 전송해주는 폼 생성. -->
  {% csrf_token %} <!-- 보안을 위해 CSRF 토큰 추가. -->
  {% include 'users/form_errors.html' %}<!-- 오류 메시지를 출력하는 템플릿 추가. -->
    <div>
      <label> User ID </label>
      <input type="text" name="username"  id="username"> <!-- 아이디를 입력받는 인풋 생성. -->
    </div>
  </br>
    <div>
      <label> User PW </label>
      <input type="password"name="password" id="password"> <!-- 비밀번호를 입력받는 인풋 생성. -->
    </div>
  </br>
  <div class="d-grid gap-3 col-6 mx-auto">
    <button class="btn btn-long text-white" style="background-color:#FC4C60;" type="submit">로그인</button>
      <!-- 로그인 버튼 -->
    <a class ="btn text-white "style="background-color:#FC4C60;" href="{% url 'users:signup' %}">회원가입</a>
      <!-- 회원가입 버튼 -->
    </br>
  </div>
</form>
```

코드 작성

form_errors.html

```
{% if form.errors %}<!-- 만약에 폼 검증 중 오류가 발생하면 아래의 코드 실행. -->
    {% for field in form %} <!-- 각 필드에 대한 오류를 순회 하면서 출력 -->
        {% for error in field.errors %}  <!-- 필드 오류를 출력. -->
            <div class="alert alert-danger">
                <strong>{{ field.label }}</strong>
                {{ field.errors }}
            </div>
        {% endfor %}
    {% endfor %}
    {% for error in form.non_field_errors %}   <!-- 넌필드 오류를 출력. -->
        <div class="alert alert-danger">
            <strong>{{ error }}</strong>
        </div>
    {% endfor %}
{% endif %}
```

코드 작성

urls.py

```
# date app
urlpatterns = [
    path("", views.MainPage.as_view(), name="main_page"),
    path("review_list/", views.ReviewList.as_view(), name="review_list"),
    path("main_area/<str:q>/",views.SelectPage.as_view(), name="select_page"),
    path("place/",views.PlaceList.as_view(),name="place_list"),
    path("place/cafe/",views.PlaceCafe.as_view(),name = "place_cafe"),
    path("place/rest/",views.PlaceRest.as_view(),name = "place_rest"),
    path("place/place/",views.PlacePlace.as_view(),name = "place_place"),
    path('login/', auth_views.LoginView.as_view(), name='login'),
    path("place/cafe/<str:q>/",views.PlaceCafeLoc.as_view(),name = "place_cafe_loc"),
    path("place/rest/<str:q>/",views.PlaceRestLoc.as_view(),name = "place_rest_loc"),
    path("place/place/<str:q>/",views.PlacePlaceLoc.as_view(),name = "place_rest_loc"),
    path("cafedetail/<str:q>/",views.CafeDetail.as_view(), name = "cafe_detail"),
    path("restdetail/<str:q>/",views.RestDetail.as_view(), name = "rest_detail"),
    path("placedetail/<str:q>/",views.PlaceDetail.as_view(), name = "place_detail"),
    path("cos/<int:q1>/<int:q2>/<int:q3>/", views.CosPage.as_view(), name="cos_page"),
    path("write/<int:q1>/<int:q2>/<int:q3>/", views.ReviewCreate.as_view(), name="review_write"),
    path("myreview/", views.MyReview.as_view(), name="my_review"),
    path("review_list/<int:pk>/", views.ReviewDetail.as_view(), name="review_detail"),
    path("myreview/<int:pk>/",views.ReviewDetail.as_view(), name="review_my_review_detail"),
    path("review_update/<int:pk>/", views.ReviewUpdate, name="review_update"),
    path("review_delete/<int:pk>/", views.ReviewDelete, name="review_delete"),
]
```

```
# users
app_name = "users"
urlpatterns = [
  path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'), #
django.contrib.auth의 views를 import해서 따로 views에 입력하지 않고 기능 구현
  path('logout/', auth_views.LogoutView.as_view(), name='logout'), # django.contrib.auth의 views를 import해서 따로
views에 입력하지 않고 기능 구현
  path('signup/', views.signup, name='signup'),
]
```

감사합니다.

Q & A