

캡스톤 디자인 I

종합설계 프로젝트

| | |
|--------|-------------|
| 프로젝트 명 | 옷때? (OTTE?) |
| 팀 명 | 옷마이갓 |
| 문서 제목 | 중간보고서 |

| | |
|---------|------------|
| Version | 1.6 |
| Date | 2020-04-20 |

| | |
|------|-----------|
| 팀원 | 황 효빈 (조장) |
| | 송 현화 |
| | 정 예빈 |
| | 권 민수 |
| | 이 재호 |
| | 주 가구 |
| 지도교수 | 이 경용 교수님 |

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 "옷때? (OTTE?)"를 수행하는 팀 "옷마이갓"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "옷마이갓"의 팀원들의 서면 허락 없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


| | |
|-----------------|------------------------------|
| Filename | 21조중간보고서-옷때(OTTE).doc |
| 원안작성자 | 황효빈, 송현화, 정예빈, 권민수, 이재호, 주가구 |
| 수정작업자 | 황효빈, 송현화, 정예빈, 권민수, 이재호, 주가구 |

| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
|------------|-------|----------|----------|--------------------------------|
| 2020-04-09 | 정예빈 | 1.0 | 최초 작성 | 프로젝트 목표 작성 |
| 2020-04-09 | 권민수 | 1.1 | 최초 작성 | 수행 내용 및 중간 결과, 향후 추진계획 작성 |
| 2020-04-09 | 황효빈 | 1.2 | 최초 작성 | 수정된 연구내용 및 추진 방향, 고충 및 건의사항 작성 |
| 2020-04-10 | 황효빈 | 1.3 | 내용 수정 | 전체 문서 내용 정리 |
| 2020-04-18 | 황효빈 | 1.4 | 내용 수정 | 수정된 연구내용 및 추진 방향 수정 |
| 2020-04-18 | 권민수 | 1.5 | 내용 수정 | 향후 추진 계획 수정 |
| 2020-04-20 | 황효빈 | 1.6 | 최종 수정 | 문서 전체 수정 |

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

목 차

| | | |
|-------|-------------------------------------|----|
| 1 | 프로젝트 목표..... | 4 |
| 2 | 수행 내용 및 중간결과 | 5 |
| 2.1 | 계획서 상의 연구내용 | 5 |
| 2.1.1 | 옷 카테고리 분석..... | 5 |
| 2.1.2 | 날씨 정보 요청 및 저장 | 7 |
| 2.1.3 | 날씨 기반 옷 추천 | 9 |
| 2.1.4 | 클라이언트(Single Page Application)..... | 10 |
| 2.1.5 | API 서버..... | 10 |
| 2.2 | 수행내용 | 11 |
| 2.2.1 | 옷 카테고리 분석..... | 11 |
| 2.2.2 | 날씨 정보 요청 및 저장 | 17 |
| 2.2.3 | 날씨 기반 옷 추천 | 20 |
| 2.2.4 | 클라이언트(Single Page Application)..... | 21 |
| 3 | 수정된 연구내용 및 추진 방향..... | 29 |
| 3.1 | 수정사항 | 29 |
| 4 | 향후 추진계획..... | 31 |
| 4.1 | 향후 계획의 세부 내용..... | 31 |
| 4.1.1 | 클라이언트(Single Page Application)..... | 31 |
| 4.1.2 | API 서버..... | 32 |
| 4.1.3 | 개선이 필요한 기능 | 33 |
| 4.1.4 | 추가가 필요한 기능 | 33 |
| 5 | 고충 및 건의사항 | 34 |

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

1 프로젝트 목표

사람마다 더위, 추위를 타는 정도와 각자 소유하고 있는 옷 등이 모두 다르다. 그러므로 본 프로젝트에서는 사용자 개인 맞춤에 중점을 두어, "오늘 뭐 입지?"라는 고민을 덜어주는 웹 서비스를 만드는 것을 목표로 한다. 대략적인 기능의 흐름은 다음과 같다.

1. 사용자가 소유한 옷들은 미리 사진을 찍어 이미지 형태로 저장해 둔다. 이 때 옷의 대분류(아우터, 상의, 바지, 치마, 원피스)와 30여가지의 소분류(롱패딩, 니트 등) 카테고리가 자동으로 분석되며 옷의 별칭을 같이 등록할 수 있다.
2. 사용자는 실제로 자신이 입었던 코드를 기록할 수 있다. 심플, 스트릿 등 5가지의 스타일을 선택하여 저장할 수 있으며 코드의 별칭도 같이 등록할 수 있다.
3. 이후 사용자는 자신이 입었던 코드에 대한 리뷰를 남길 수 있다. 리뷰에는 해당 코드를 입었던 날의 체감온도, 기온, 풍속 등의 날씨 정보와 함께 코멘트가 저장된다. 또한 해당 날씨에 대한 코드의 적절성을 5단계로 선택하여 기록할 수 있다.
4. 어떤 옷을 입을지 고민이 될 때 현재 체감온도와 유사한 날에 입었던 자신의 코디와 리뷰를 통해 스스로 날씨에 적절한 옷을 매치하여 입을 수 있다.
5. 여러 사용자들이 현재 체감온도와 유사한 날에 입었던 코디의 리뷰 중 날씨에 대한 코디 적절성이 높은 리뷰를 수집한다. 이후 대분류 기준 가장 높은 비율을 차지하는 소분류 카테고리를 텍스트 형태로 추천하며 일치하는 사용자의 옷을 이미지 형태로 보여준다.

위와 같이, 옷때?(OTTE?)를 통해 다음과 같은 기대효과를 얻을 수 있다. 사용자는 날씨에 알맞은 코디를 할 수 있고, 어떤 옷을 입어야 할지 고민하는 시간을 단축시킬 수 있다. 또한 사용자가 소유한 옷들을 쉽게 파악할 수 있어, 자신의 옷을 효율적으로 관리할 수 있다.

2 수행 내용 및 중간결과

2.1 계획서 상의 연구내용

2.1.1 옷 카테고리 분석


사용자가 옷을 등록할 때, 옷 이미지를 기반으로 카테고리를 추론하는 기능을 제공해야 한다. 따라서 모델을 학습시키고, AWS SageMaker에 배포하여 사용할 수 있도록 한다.

1. 모델 학습에 사용할 데이터를 각 카테고리별로 모은다.

카테고리는 다양한 인터넷 의류 쇼핑몰의 옷 카테고리를 참고한다. 대분류와 소분류 카테고리는 다음과 같다.

| 대분류 | 소분류 | 대분류 | 소분류 |
|-----|---------|-----|-------|
| 상의 | 반팔 티셔츠 | 아우터 | 블레이저 |
| | 긴팔 티셔츠 | | 패딩 |
| | 반팔 셔츠 | | 조끼 패딩 |
| | 긴팔 셔츠 | | 롱 패딩 |
| | 맨투맨 | | 야구 점퍼 |
| | 터틀넥 | | 항공 점퍼 |
| | 후드티 | | 바람막이 |
| | 니트 | | 야상 |
| | 블라우스 | | 무스탕 |
| | 끈나시 | | 코트 |
| | 민소매 | | 트랙탑 |
| | | | 가죽 자켓 |
| | | | 청자켓 |
| 치마 | 미니스커트 | | 가디건 |
| 하의 | 롱스커트 | 한벌옷 | 원피스 |
| | 반바지 | | |
| | 하트팬츠 | | |
| | 슬랙스 | | |
| | 청바지 | | |
| | 골덴 바지 | | |
| | 트레이닝 바지 | | |

[그림 2-1] 대분류와 소분류 카테고리

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

2. TeachableMachine 서비스를 이용해 모델을 만든다.

TeachableMachine은 간단한 데이터들을 통해 나만의 모델을 학습시키고 저장할 수 있도록 해주는 서비스로, 현재 무료로 서비스되고 있다. 이를 통해서 이미지, 영상, 음성 데이터를 통해 학습시킬 수 있다. 또한 학습시킨 모델은 Tensorflow의 savedmodel 형태로 저장할 수 있다. (<https://teachablemachine.withgoogle.com/>)

3. 저장한 savedmodel 파일을 AWS SageMaker에서 제공하는 Tensorflow Serving Container를 이용, 배포 가능한 모델의 형태로 AWS SageMaker에 저장한다.

4. AWS SageMaker에 저장시킨 모델을 public DNS로 접근할 수 있는 엔드포인트로 배포한다.

5. 이후 이 엔드포인트에 AWS SageMaker Python SDK와 계정정보를 이용해 Python 코드 내에서 접근할 수 있다.

2.1.2 날씨 정보 요청 및 저장

사용자가 코디 리뷰 정보를 등록할 때, 사용자가 입력한 지역과 시간 정보를 기반으로 날씨 정보를 받아와 데이터베이스에 함께 저장해야한다. 이를 위해 공공데이터포털의 기상청 Open API를 이용한다.

1. 공공데이터포털의 '기상청 날씨예보 정보 API – 동네예보 조회서비스'를 이용한다

동네예보 조회서비스는 자체의 격자 정보와 시간 정보를 입력으로 받아, 해당 입력에 맞는 다양한 데이터를 반환해준다. 회원가입을 하고, API 키를 발급 받아 해당 서비스를 이용할 수 있으며 반환되는 데이터들은 다음과 같다.

| 예보구분 | 항목값 | 항목명 | 단위 | 압축bit수 |
|------|-----|----------|-----------|--------|
| 동네예보 | POP | 강수확률 | % | 8 |
| | PTY | 강수형태 | 코드값 | 4 |
| | R06 | 6시간 강수량 | 범주 (1 mm) | 8 |
| | REH | 습도 | % | 8 |
| | S06 | 6시간 신적설 | 범주(1 cm) | 8 |
| | SKY | 하늘상태 | 코드값 | 4 |
| | T3H | 3시간 기온 | °C | 10 |
| | TMN | 아침 최저기온 | °C | 10 |
| | TMX | 낮 최고기온 | °C | 10 |
| | UUU | 풍속(동서성분) | m/s | 12 |
| | VVV | 풍속(남북성분) | m/s | 12 |
| | WAV | 파고 | M | 8 |
| | VEC | 풍향 | m/s | 10 |
| | WSD | 풍속 | 1 | 10 |

[그림 2-2] 기상청 API 동네예보 조회서비스의 데이터 반환 형식


2. 클라이언트로부터 지역 정보와 시작 시간, 종료시간을 받고, 이를 이용해 해당하는 날씨 정보를 기상청 API에게 요청한다.

지역 정보는 자체의 (x, y) 격자 정보를 가지기 때문에, 지역 이름을 받았을 때 이를 알맞은 (x, y) 격자 정보로 바꿔주는 방법이 필요하다. 기상청에서는 다음과 같은 csv 파일을 제공하며 우리는 이를 활용한다. 먼저 해당 파일을 json 파일로 변환시키고, 각각의 위치정보는 행 번호를 key로 하는 딕셔너리(Dictionary) 형태로 저장한다. 이후 클라이언트에게 행 번호를 받으면 바로 그에 맞는 (x, y) 좌표로 변환할 수 있도록 한다.

| | A | B | C | D | E | F | G |
|----|-------|-----|-------------|------|------|---|---|
| 1 | 1단계 | 2단계 | 3단계 | 격자 X | 격자 Y | | |
| 2 | 서울특별시 | | | 60 | 127 | | |
| 3 | 서울특별시 | 종로구 | | 60 | 127 | | |
| 4 | 서울특별시 | 종로구 | 청운효자동 | 60 | 127 | | |
| 5 | 서울특별시 | 종로구 | 사직동 | 60 | 127 | | |
| 6 | 서울특별시 | 종로구 | 삼청동 | 60 | 127 | | |
| 7 | 서울특별시 | 종로구 | 부암동 | 60 | 127 | | |
| 8 | 서울특별시 | 종로구 | 평창동 | 60 | 127 | | |
| 9 | 서울특별시 | 종로구 | 무악동 | 60 | 127 | | |
| 10 | 서울특별시 | 종로구 | 교남동 | 60 | 127 | | |
| 11 | 서울특별시 | 종로구 | 가회동 | 60 | 127 | | |
| 12 | 서울특별시 | 종로구 | 종로1.2.3.4가동 | 60 | 127 | | |
| 13 | 서울특별시 | 종로구 | 종로5.6가동 | 60 | 127 | | |
| 14 | 서울특별시 | 종로구 | 이화동 | 60 | 127 | | |
| 15 | 서울특별시 | 종로구 | 혜화동 | 60 | 127 | | |
| 16 | 서울특별시 | 종로구 | 창신제1동 | 61 | 127 | | |
| 17 | 서울특별시 | 종로구 | 창신제2동 | 60 | 127 | | |
| 18 | 서울특별시 | 종로구 | 창신제3동 | 60 | 127 | | |
| 19 | 서울특별시 | 종로구 | 송인제1동 | 60 | 127 | | |
| 20 | 서울특별시 | 종로구 | 송인제2동 | 61 | 127 | | |
| 21 | 서울특별시 | 중구 | | 60 | 127 | | |
| 22 | 서울특별시 | 중구 | 소공동 | 60 | 127 | | |
| 23 | 서울특별시 | 중구 | 회현동 | 60 | 126 | | |

[그림 2-3] 기상청에서 제공하는 지역 정보 csv 파일

3. 데이터를 성공적으로 받아오면, 필요한 데이터를 가공하여 사용한다.

| | | | |
|---|-------------------------|-------------|-------------|
|  <div> <p>국민대학교</p> <p>소프트웨어학부</p> <p>캡스톤 디자인 I</p> </div> | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

2.1.3 날씨 기반 옷 추천

현재와 유사한 날씨에 여러 사용자들이 남긴 코디 리뷰를 바탕으로 사용자에게 오늘 날씨에 적절한 옷이 무엇인지 카테고리 형태로 보여줄 수 있어야 한다.

1. 클라이언트로부터 오늘의 날씨에 대한 정보 중 최저/최고 체감온도를 받는다.
2. 오늘과 유사한 날씨에 작성된 리뷰들 중 '날씨의 코디 적절성'을 기준으로 여러 사용자들의 코디 리뷰들을 모두 추출한다.


날씨의 유사도는 최저/최고 체감온도를 기준으로 구분한다. 체감온도는 인체가 덥거나 춥다고 느끼는 정도를 수치로 계산한 온도이다. 사람은 기온 뿐 만 아니라 풍속, 습도 등 여러가지 요인에 따라 느끼는 것이 다르기 때문에 우리는 체감온도를 기준으로 두기로 하였다. 최저/최고 체감온도가 모두 $\pm 2^{\circ}\text{C}$ 이내로 차이가 나는 경우만 유사한 날씨로 구분하여 추출한다. 우리나라 기상청에서는 2001년 캐나다에서 열린 'Joint Action Group for Temperature Indices(JAG/TI)' 회의에서 발표한 체감온도 계산법을 사용하며 공식은 다음과 같다.

$$\text{체감온도}(^{\circ}\text{C}) = 13.12 + 0.6215 \times T - 11.37 \times V^{0.16} + 0.3965 \times V^{0.16} \times T$$

$T = \text{기온}(^{\circ}\text{C}), V = \text{풍속}(\text{km/h})$

[그림 2-4] 체감온도 계산법

3. 추출한 리뷰들에 포함된 모든 옷들을 추출하고, 해당 옷들은 대분류 카테고리별로 정리한다.
4. 대분류 카테고리별로 가장 빈도수가 높게 나온 소분류 카테고리를 추출한다
5. 결과를 반환한다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

2.1.4 클라이언트(Single Page Application)

클라이언트 애플리케이션은 Single Page Application 형태로 개발하며, Vue.js 프레임워크를 이용한다. 개발은 크게 3단계로 나눈다.

1. 각각의 페이지 설계 및 라우팅 계획 수립
2. API 서버와 통신하지 않고, 정적인 데이터로 이루어진 페이지 구현
3. API 서버와 통신하는 동적인 페이지 구현


개발이 끝나면 AWS Amplify를 이용하여 간단한 방법으로 애플리케이션을 배포하고, Public DNS를 통해 접근할 수 있도록 한다.

2.1.5 API 서버

API 서버는 클라이언트의 Single Page Application에게 데이터베이스를 이어주는 가교 역할을 해주며, 고정된 JSON 형태로 데이터만 주고받는 형식을 사용한다. 데스크탑이든 모바일이든 상관없이 데이터를 주고받고, 클라이언트 쪽에서 상황에 맞게 데이터를 가공하여 보여주면 되므로 이러한 방식을 사용했다. 개발은 크게 4단계로 나눈다.

1. 제공해야 하는 API 설계 및 구현 방법 계획
2. 기본적인 CRUD(Create, Retrieve, Update, Delete) 엔드포인트 구현
3. 추가적인 엔드포인트 구현
4. 테스트 코드 작성 및 유닛 테스트 진행

개발이 끝나면 Apache와 연동하여 클라이언트에서 접근할 수 있도록 EC2 인스턴스에 배포한다.

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

2.2 수행내용

2.2.1 옷 카테고리 분석

1. 모델 학습에 사용할 데이터를 카테고리별로 수집한다.

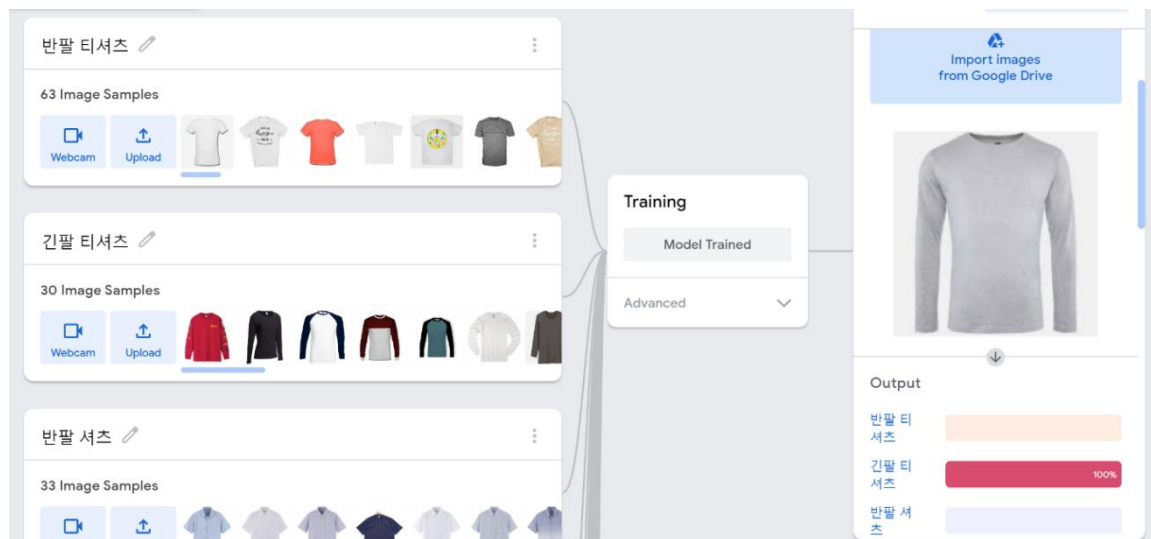
대분류와 소분류 카테고리별로 이미지를 모았으며, 이미지를 모으는 작업은 정확도를 위해 수작업으로 배경이 단색인 각각의 사진을 모았다.



[그림 2-5] 카테고리 별 배경이 단색인 사진 수집

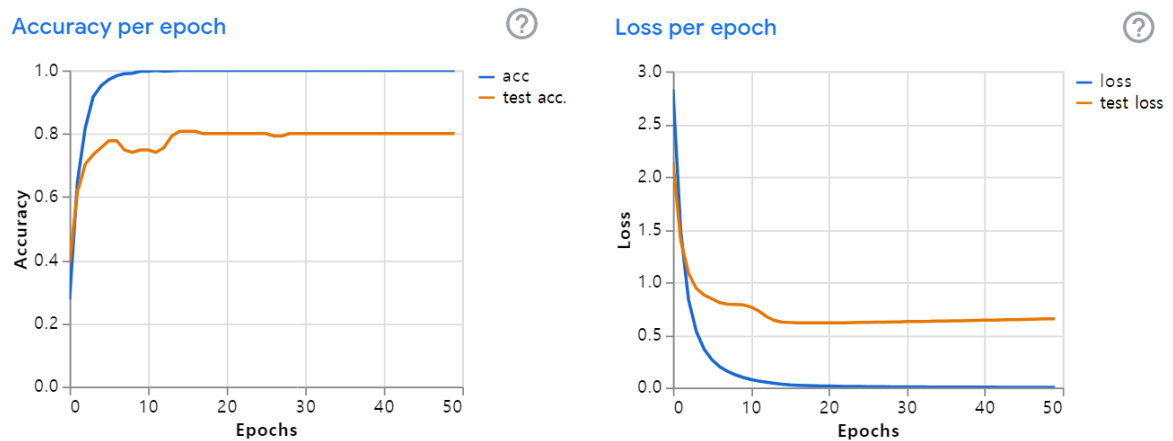
2. TeachableMachine 서비스를 이용해 모델을 만든다.

- 앞에서 수집한 이미지를 각 분류에 맞게 넣어주고, 학습을 진행하였다.




[그림 2-6] TeachableMachine 서비스를 이용한 모델 생성

- 0.001의 학습률과 16의 배치 사이즈로 50세대의 학습을 돌렸으며, 다음과 같은 결과를 보였다.



[그림 2-7] 학습 결과

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 학습시킨 모델은 Tensorflow savedmodel 형식으로 저장했다.



[그림 2-8] 모델 저장 형식

3. 저장한 savedmodel 파일을 AWS SageMaker에서 제공하는 Tensorflow Serving Container를 이용, 배포 가능한 모델의 형태로 AWS SageMaker에 저장한다.

| |
|---|
| tensorflow_keras_CIFAR10.ipynb |
| tensorflow_moving_from_framework_mode_to_script_mode.ipynb |
| tensorflow_script_mode_horovod.ipynb |
| tensorflow_script_mode_pipe_mode.ipynb |
| tensorflow_script_mode_quickstart.ipynb |
| tensorflow_script_mode_training_and_serving.ipynb |
| tensorflow_script_mode_using_shell_commands.ipynb |
| tensorflow_serving_container.ipynb |
| tensorflow_serving_pretrained_model_elastic_inference.ipynb |
| tf-eager-sm-scriptmode.ipynb |


[그림 2-9] AWS 제공 Container

- 모델을 S3에 저장한다.

```
from sagemaker.session import Session

model_data = Session().upload_data(path='clothes_model.tar.gz', key_prefix='model')
print('model uploaded to: {}'.format(model_data))
```

[그림 2-10] S3에 모델 저장

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 해당 모델을 엔드포인트로 배포 가능한 형식으로 저장하고, 엔드포인트를 생성한다.

```
from sagemaker.tensorflow.serving import Model

# Use an env argument to set the name of the default model.
# This is optional, but recommended when you deploy multiple models
# so that requests that don't include a model name are sent to a
# predictable model.
env = {'SAGEMAKER_TFS_DEFAULT_MODEL_NAME': 'clothes_model'}

model = Model(model_data=model_data, role=sagemaker_role, framework_version='1.14', env=env)
```

[그림 2-11] 엔드포인트 생성

4. AWS SageMaker에 저장한 모델을 Public DNS로 접근할 수 있는 엔드포인트로 배포한다.

| 엔드포인트 | | 🔄 | 엔드포인트 업데이트 | 작업 ▼ | 엔드포인트 생성 |
|--------------------|---|---|-------------|------------------------|----------|
| 🔍 엔드포인트 검색 | | <div> <div><</div> <div>1</div> <div>></div> <div>⚙️</div> </div> | | | |
| 이름 ▼ | ARN | 생성 시간 ▼ | 상태 ▼ | 최종 업데이트 날짜 | |
| ○ clothes-30-model | arn:aws:sagemaker:ap-northeast-2:684188911663:endpoint/clothes-30-model | Apr 05, 2020 09:03 UTC | 🟢 InService | Apr 05, 2020 09:08 UTC | |

[그림 2-12] 엔드포인트로 모델 배포

5. 이후 이 엔드포인트에 AWS SageMaker, Python SDK와 계정정보를 이용해 Python 코드 내에서 접근할 수 있다.

[그림 2-13]의 코드로 배포한 엔드포인트를 이용해 결과를 받아온다.



```
def execute_inference(image):
    """
    Receives image and executes
    inference against sagemaker endpoint.
    """

    boto_session = boto3.Session(aws_access_key_id=settings.AWS_ACCESS_KEY_ID,
                                  aws_secret_access_key=settings.AWS_SECRET_ACCESS_KEY)
    sess = sagemaker.Session(boto_session=boto_session)

    ENDPOINT_MODEL = 'clothes-30-model'

    predictor = RealTimePredictor(endpoint=ENDPOINT_MODEL,
                                   sagemaker_session=sess,
                                   content_type='application/json',
                                   accept='application/json')

    # Convert tensor to JSON format.
    image = image.tolist()
    image = json.dumps(image)

    # Get a prediction from the endpoint.
    result = predictor.predict(image)
    # Convert result into python dict.
    result = json.loads(result)


    return result
```

[그림 2-13] 엔드포인트 이용

- 입력으로 이미지의 정보가 담긴 텐서를 받는다.

```
✓ image: [[[[...], [...], [...], [...], [...], [...], [...], [...], [...], ...]]
```

[그림 2-14] 이미지 정보가 담긴 텐서

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 분석이 끝나면 다음과 같이 추론 결과를 각각 클래스의 확률 형태로 받는다.

```

✓ result: {'predictions': [[...]]}
✓ 'predictions': [[5.70503511e-09,
  ✓ 0: [5.70503511e-09, 0.00199416,
    00: 5.70503511e-09
    01: 0.00199416
    02: 3.00376769e-06
    03: 0.987620115
    04: 8.43962937e-07
    05: 0.000423914782
    06: 4.26048175e-08
    07: 0.00700465078
    08: 1.45532322e-05
  
```

[그림 2-15] 분석 후 추론 결과

- 이후 인덱스를 이용해 결과 대분류, 소분류 카테고리를 각각 구할 수 있게 된다.

```

upper: '상의'   lower: '긴팔셔츠'

```

[그림 2-16] 도출된 대분류, 소분류 카테고리

2.2.2 날씨 정보 요청 및 저장

1. 공공데이터포털의 '기상청 날씨예보 정보 API - 동네예보 조회서비스'를 이용한다.

[그림 2-17]과 같이 신청을 통해 API 키를 발급 받았다.

>

기본정보

| | | | | | | |
|--------|-------------------------|-------|-----------------|-----------|------|----|
| 서비스명 | 동네예보 조회서비스 | | <div>상세설명</div> | | | |
| 서비스 유형 | REST | 일일트래픽 | 0 | 평균응답속도(초) | 0 | |
| 심의여부 | 자동승인 | 신청유형 | 개발계정 연장신청 | | 처리상태 | 승인 |
| 활용기간 | 2020-02-28 ~ 2022-03-12 | | | | | |

>

서비스정보

| | | | | | |
|----------------|--|--|--|--|--|
| 일반 인증키 (UTF-8) | <div></div> <div>복사</div> | | | | |
| End Point | http://apis.data.go.kr/1360000/VilageFcstInfoService | | | | |
| 데이터포맷 | JSON+XML | | | | |
| 참고문서 | 기상청18_동네예보_조회서비스_오픈API활용가이드.zip | | | | |

[그림 2-17] API 키 발급

2. 클라이언트로부터 지역 정보와 시작 시간, 종료 시간을 받아서 해당하는 날씨 정보를 기상청 API에 요청한다.


지역 정보는 기상청에서 제공받은 CSV 파일을 이용하여 [그림 2-18]과 같은 JSON 형태로 저장해 X좌표와 Y좌표로 변환할 수 있도록 하였다. 그리고 받아온 정보를 통해 URL을 [그림 2-19]와 같이 생성한다.

```

'0': {'full_address': '서울특별시', 'x': '60', 'y': '127'}
'1': {'full_address': '서울특별시 종로구', 'x': '60', 'y': '127'}
'2': {'full_address': '서울특별시 종로구 청운효자동', 'x': '60', 'y': '127'}
'3': {'full_address': '서울특별시 종로구 사직동', 'x': '60', 'y': '127'}
'4': {'full_address': '서울특별시 종로구 삼청동', 'x': '60', 'y': '127'}

```

[그림 2-18] x, y좌표 변환 JSON 파일

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

```

url = "http://apis.data.go.kr/1360000/VilageFcstInfoService/getVilageFcst?"
key = "serviceKey=" + ServiceKey
numOfRows = "&numOfRows=100"
typeOfData = "&dataType=JSON"
date = "&base_date=" + convert_api_date
time = "&base_time=" + convert_api_time

x = (json_data[str(location)][ 'x' ])
y = (json_data[str(location)][ 'y' ])

nx = "&nx=" + x
ny = "&ny=" + y

api_url = url + key + numOfRows + typeOfData + date + time + nx + ny
data = urllib.request.urlopen(api_url).read().decode('utf8')

```

[그림 2-19] URL 생성

3. 데이터를 성공적으로 받아오면, 필요한 데이터를 가공하여 사용한다.


- 받아온 데이터는 [그림 2-20]과 같은 형식으로 변환된다.

```

{
  'item': [
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'POP', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    },
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'PTY', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    },
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'REH', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    },
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'SKY', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    },
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'T3H', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    },
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'UUU', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    },
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'VEC', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    },
    {
      'baseDate': '20200408', 'baseTime': '0500', 'category': 'VWV', 'fcstDate': '20200408', 'fcstTime': '0500', 'fcstValue': 100
    }
  ]
}

```

[그림 2-20] 데이터 가공

| | | | |
|---|-------------------------|-------------|-------------|
|  <div> <p>국민대학교</p> <p>소프트웨어학부</p> <p>캡스톤 디자인 I</p> </div> | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 체감온도는 공식에 맞게 계산한다.

체감 온도 구하기

```
def getWCI(temperature, wind_velocity):
    """
    기온과 풍속을 입력 받아 체감온도를 반환한다.
    예시 : 12, 9.5
    """
    WCI = 13.12 + 0.6215 * temperature - 11.37 * math.pow(wind_velocity, 0.16) \
        + 0.3965 * temperature * math.pow(wind_velocity, 0.16)

    return WCI
```

[그림 2-21] 체감온도 계산

- 최종으로 다음과 같은 데이터들을 데이터베이스에 저장한다.

```

    < passing_data: {'P
      'POP': '0'
      'PTY': '0'
      'REH': '25'
      'SKY': '1'
      'T3H': '13'
      'UUU': '2.8'
      'VEC': '298'
      'VVV': '-0.9'
      'WSD': '1.7'
      'R06': '0'
      'S06': '0'
      'TMX': '15.0'
      'TMN': '4.0'
      'WCI': 14.43
      'WCIMAX': 14.43
      'WCIMIN': 14.43
  
```

[그림 2-22] 데이터베이스에 저장될 데이터

2.2.3 날씨 기반 옷 추천

1. 클라이언트로부터 오늘의 날씨에 대한 정보를 받는다.

URL의 쿼리 스트링 형태로 받고 이를 float 형태로 파싱한다.

```
/clothes/today_category/?min_sensible_temp=2&max_sensible_temp=15
```

[그림 2-23] URL의 쿼리 스트링

```
max_sensible: 15.0
min_sensible: 2.0
```

[그림 2-24] float 형태로 파싱

2. 유사한 날씨에 남긴 리뷰 중 날씨의 코디 적합성에 긍정적인 평가를 남긴 리뷰들을 모두 추출한다.

코디들의 id가 다음과 같이 추출된다.

```

filtered_clothes_set_id
0: 10
1: 21
2: 7
3: 48

```

[그림 2-25] 코디 id 추출


3. 추출한 리뷰들을 바탕으로 해당 리뷰에 포함된 모든 옷들을 추출하고, 해당 옷을 대분류 카테고리별로 정리한다.

```

analysis_upper_category_dict: {'상의': ['곤나시', '반팔티셔츠',
> '하의': []
> '원피스': ['원피스', '원피스', '원피스', '원피스', '원피스']
> '아우터': ['무스탕', '코트', '숏패딩', '코트']
> '치마': ['미니스커트', '미니스커트', '롱스커트', '미니스커트']
> '상의': ['곤나시', '반팔티셔츠', '반팔티셔츠']

```

[그림 2-26] 대분류 카테고리별로 정리된 추출된 옷

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 대분류 카테고리별로 가장 빈도수가 높게 나온 소분류 카테고리를 추출한다

```

analysis_upper_category_dict
{
  '하의': ''
  '원피스': '원피스'
  '아우터': '코트'
  '치마': '미니스커트'
  '상의': '반팔티셔츠'
}

```


[그림 2-27] 대분류 카테고리별 가장 빈도수 높은 소분류 카테고리

- 결과를 반환한다.

2.2.4 클라이언트(Single Page Application)

- 각각의 페이지 설계 및 라우팅 계획 수립

로그인, 회원가입, 옷장, 옷 상세정보, 옷 등록, 메인, 코디 옷장, 코디 상세 정보 등의 페이지를 설계하였고, 라우팅 계획을 수립하였다. 또한 [그림 2-28]과 같이 각각의 페이지에 대한 상세 설계 내용 기술을 진행하였다. 자세한 정보는 해당 링크에서 확인 가능하다. (<https://bit.ly/2VamGA9>)

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

로그인 화면

웹 페이지 접속 시 첫 화면.


라우팅

/login HTML ▾

디자인

기본 화면

sign up




id :

password :

login

첫 진입 시 화면

sign up



id : id가 존재하지 않습니다

password : password가 일치하지 않습니다


login

로그인 실패 시

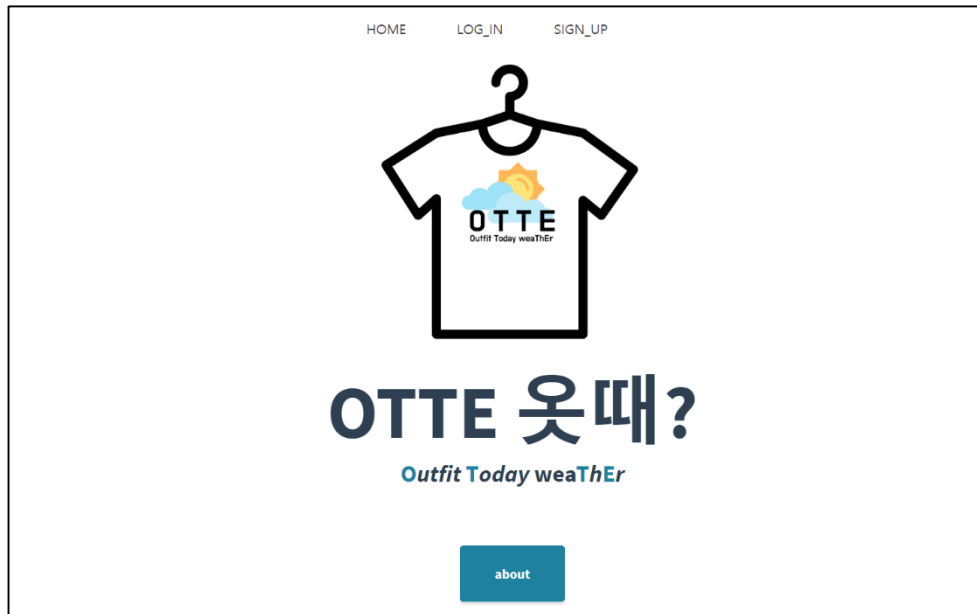
[그림 2-28] 로그인 페이지 상세 설계 내용

2. API 서버와 통신하지 않고, 정적인 데이터로 이루어진 페이지 구현

구현된 핵심 페이지들은 아래와 같으며 상세 디자인은 수정될 예정이다.

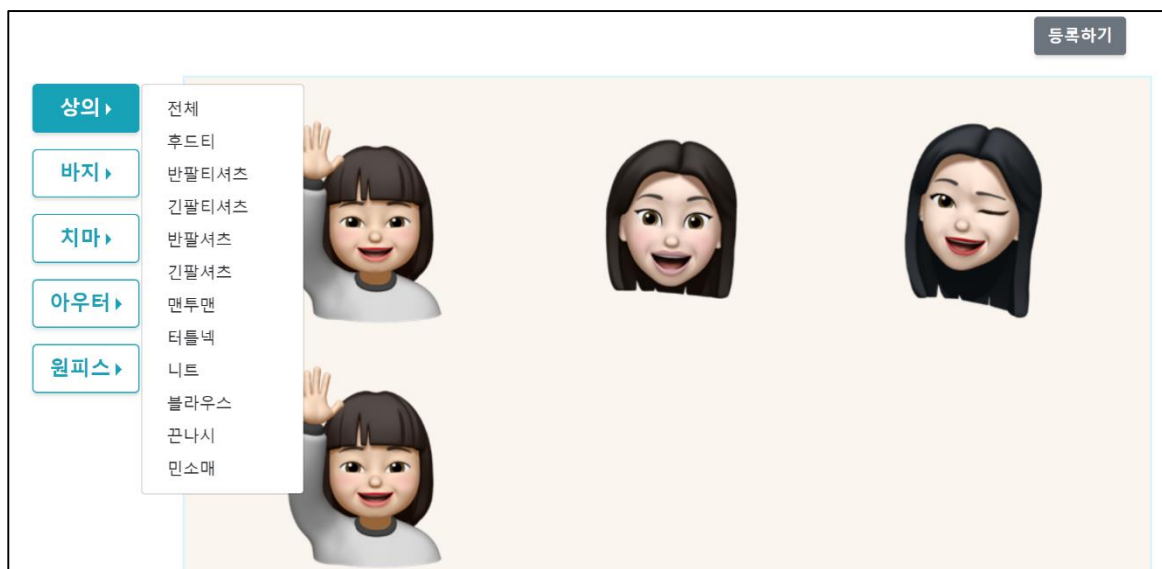
| | | | |
|--|-------------------------|-------------|-------------|
|  <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div> | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 메인페이지



[그림 2-29] 메인페이지

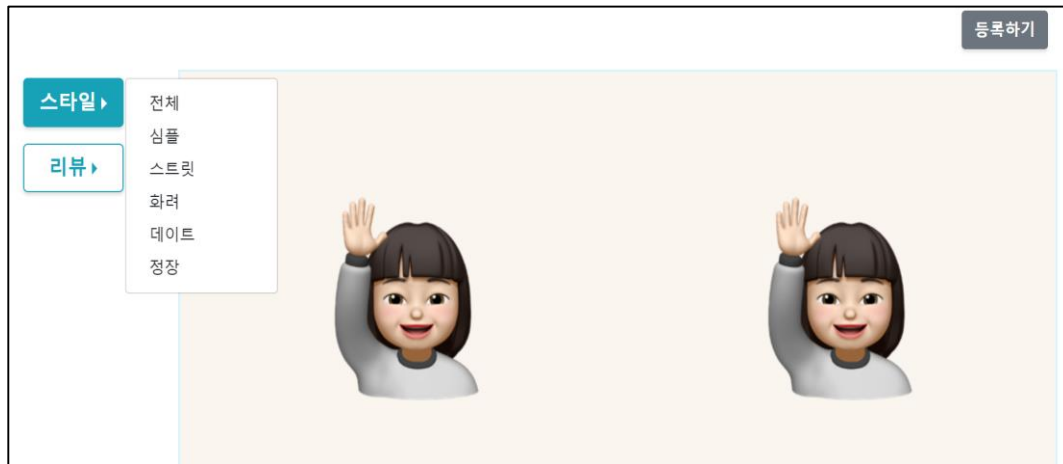
- 옷장 페이지



[그림 2-30] 옷장페이지

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 내 코디 페이지




[그림 2-31] 내 코디 페이지

- 리뷰 등록 페이지

HOME
CLOSET
CODY
REVIEW

Review



활동 시간:

연도-월-일

--:--

~

--:--


활동 장소:

한줄평:

확인

새 리뷰 작성하기

[그림 2-31] 리뷰 등록 페이지

| | | | |
|--|-------------------------|-------------|-------------|
|  <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div> | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

2.2.5 API 서버

1. 제공해야 하는 API 설계 및 구현 방법 계획

핵심 모델인 User, Clothes, ClothesSet, ClothesSetReview를 바탕으로 다양한 API에 대한 설계를 수행하였다. [그림 2-32]처럼 각각의 API에 대한 상세 설계 내용 기술을 진행하였다. 자세한 정보는 해당 링크에서 확인 가능하다. (<https://bit.ly/2JQTE3f>)

모든 옷 정보 반환

Request

Request Header

- **Authorization** : 사용자 인증 정보

Path Parameters

X

Query Parameters

- **limit** : 최대로 받을 유저의 갯수
→ default = 100
- **offset** : 데이터를 받아오기 시작할 행의 번호
→ default = 0
- **upper_category** : 해당하는 대분류의 옷만 받아오고 싶을 때 사용
- **lower_category** : 해당하는 소분류의 옷만 받아오고 싶을 때 사용
- **me** : True면 현재 사용자의 옷만 반환
- **ordering** : 정렬할 기준 필드 지정 - (**id**, **created_at**)
→ ex) **created_at** 오름차순 정렬 : `/clothes?ordering=created_at`
→ ex) **created_at** 내림차순 정렬 : `/clothes?ordering=-created_at`

Body Parameters

X

Response

Success Code

- **200** : OK, 성공

Other Response Codes

- **401** : Unauthorized, 사용자 인증 정보가 잘못된 경우


Response Header

- **Content-Type** : `application/json`

Response Body

- 해당하는 **Clothes** array
 - **id**, **upper_category**, **lower_category**, **image_url**, **alias** 로 구성

[그림 2-32] 모든 옷 정보 반환 API 상세 설계 내용

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

2. 기본적인 CRUD(Create, Retrieve, Update, Delete) 엔드포인트 구현

[그림 2-33]과 같이 엔드포인트 URL들의 각각의 CRUD 엔드포인트를 구현하였다.

```
{
  "users": "http://127.0.0.1:8000/users/",
  "clothes": "http://127.0.0.1:8000/clothes/",
  "clothes-sets": "http://127.0.0.1:8000/clothes-sets/",
  "clothes-set-reviews": "http://127.0.0.1:8000/clothes-set-reviews/"
}
```

[그림 2-33] CLUD 엔드포인트 구현

3. 추가적인 엔드포인트 구현


- 현재 사용자의 인증 정보를 바탕으로 현재 유저 정보를 반환하는 me 엔드포인트를 구현하였다.

```
GET /users/me/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 197,
  "username": "minsu",
  "nickname": "",
  "gender": "",
  "birthday": null
}
```

[그림 2-34] me 엔드포인트

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 이미지 데이터를 받아 분석한 결과를 반환하는 inference 엔드포인트를 구현하였다.


```
{
  "image_url": "https://otte-bucket.s3.ap-northeast-2.amazonaws.com/clothes/temp/clothes_1586344565954.png",
  "upper_category": "상의",
  "lower_category": "하드티"
}
```

[그림 2-35] inference 엔드포인트 구현

- 주어진 날씨와 유사한 날씨의 추천 옷 카테고리를 반환하는 today_category 엔드포인트를 구현하였다.

```
{
  "하의": "슬랙스",
  "원피스": "원피스",
  "아우터": "코트",
  "치마": "미니스커트",
  "상의": "반팔티셔츠"
}
```

[그림 2-36] today_category 엔드포인트 구현

| | | | |
|---|-------------------------|-------------|-------------|
|  <div> <p>국민대학교</p> <p>소프트웨어학부</p> <p>캡스톤 디자인 I</p> </div> | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

- 검색된 주소지의 id와 전체 주소를 반환하는 location_search 엔드포인트를 구현하였다.

```
{
  "count": 4,
  "next": 4,
  "results": [
    {
      "id": "126",
      "location": "서울특별시 성북구 정릉제1동"
    },
    {
      "id": "127",
      "location": "서울특별시 성북구 정릉제2동"
    },
    {
      "id": "128",
      "location": "서울특별시 성북구 정릉제3동"
    },
    {
      "id": "129",
      "location": "서울특별시 성북구 정릉제4동"
    }
  ]
}
```

[그림 2-37] location_search 엔드포인트 구현

4. 테스트 코드 작성 및 유닛 테스트 진행

구현한 엔드포인트들을 테스트하는 유닛 테스트 코드를 작성했으며 해당 코드로 테스트를 진행하였다.

Ran 56 tests in 53.996s

OK

Destroying test database for alias 'default'...

[그림 2-37] 유닛 테스트

3 수정된 연구내용 및 추진 방향

3.1 수정사항

1) 코디의 리뷰 단계


사용자가 입었던 코디에 대한 리뷰를 '시원했다', '추웠다', '따뜻했다', '더웠다'의 4단계로 구분하였지만 날씨에 적당한 옷을 입었을 때 '따뜻했다'와 '시원했다'의 구분이 모호하였다. 따라서 '추웠다'를 1단계, '적당했다'를 3단계, '더웠다'를 5단계로 두고 총 5단계로 코디의 리뷰 단계를 구체화하였다.



[그림 3-1] 코디 리뷰 작성 페이지

2) 옷 카테고리 분석

기존에는 사용자가 자신의 옷을 등록할 때, 옷 사진을 업로드한 후 버튼을 눌러야 옷 카테고리의 분석이 진행되었다. 하지만 사용자의 편리성과 옷 등록에 소요되는 시간 단축을 위해서 옷 사진을 업로드함과 동시에 자동으로 분석이 진행되도록 수정하였다.

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

3) 날씨 API 정확성


날씨 API로 데이터를 받아 올 때, 날씨 예측의 정확도를 고려해야 했다. 따라서 정확도가 높은 기상청 API를 채택하였다. 하지만 기상청 API에서 제공하는 날씨 데이터는 몇몇 시간대에 응답이 비어있는 경우가 있다. 따라서 확실하게 데이터를 받아올 수 있는 시간으로만 기상청 API를 호출하기로 하였다. 확실하게 데이터를 받아올 수 있는 시간은 오전 2시부터 3시간 간격인 2시, 5시, 8시, 11시, 14시, 17시, 20시, 23시이다.

4) 24시간 이전의 날씨

기상청 API는 24시간 이전의 날씨 데이터들을 제공하지 않는다. 또한 다른 날씨 API들은 정확성이 떨어지거나 유료로 사용해야 되는 등의 문제가 있다. 따라서 기상청 API를 이용하여, 위의 '3) 날씨 API 정확성'에서 다룬 '확실하게 데이터를 받아올 수 있는 시간대'마다 스크립트를 실행하여 날씨 정보를 받아 오기로 하였다. 받아 온 데이터는 데이터베이스에 저장한다. 데이터베이스에 저장된 날씨 정보는 7일 간 유지하며 이후에는 삭제되도록 한다.

5) 리뷰 작성 기간

날씨 정보는 데이터베이스에 7일 간 저장되며 이후에는 삭제되도록 한다. 따라서 사용자가 본인의 코디 리뷰를 남길 수 있는 날짜는 현재로부터 7일 전까지만 가능하도록 제한한다. 따라서 리뷰를 남길 때 해당 코디를 입은 날짜를 7일보다 이전으로 선택한다면 날짜를 다시 선택하도록 경고문을 띄우도록 한다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

6) 현재 날씨에 맞는 코디 제안 및 옷 추천

사람은 기온 이외에 습도, 바람, 일사 등 다양한 조건에 따라 같은 기온에도 다르게 느끼게 된다. 따라서 현재 날씨에 맞는 코디 제안 및 옷을 추천할 때 최저와 최고 체감온도를 기준으로 한다. 하지만 체감온도가 정확하게 일치하지 않아도 약간의 오차 구간 내에서는 옷차림이 크게 변하지 않는다. 그러므로 우리는 최저 체감온도 ± 2 , 그리고 최고 체감온도 ± 2 의 구간을 주어 현재 날씨에 맞는 코디를 제안하고 옷을 추천하기로 하였다.


4 향후 추진계획

4.1 향후 계획의 세부 내용

4.1.1 클라이언트(Single Page Application)

1) API 서버와 통신

현재 클라이언트 쪽 코드에는 API 서버와 통신하는 부분이 포함되어 있지 않고, 정적인 데이터들로 고정된 작업을 수행하는 코드만 포함되어 있다. API 서버에서 실제 데이터를 주고 받기 위해서 Javascript의 axios 라이브러리를 이용하여, API 서버와 통신이 이상이 없는지, 데이터는 제대로 오고 가는지, API 서버 쪽에 수정이 필요한 부분이 있는지 확인할 필요가 있다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

2) 반응형 웹페이지

현재 프론트엔드 코드는 데스크탑 페이지에 맞게 구현되고 있다. 모바일로 접속을 시도할 시 이를 인식하고 웹페이지 UI를 반응형으로 바꾸도록 수정할 필요가 있다.

4.1.2 API 서버

1) 오늘의 날씨 엔드포인트 제공


클라이언트에서 현재 날씨에 대한 정보를 받아와 사용자에게 보여주는 페이지에서 해당 부분을 위한 엔드포인트가 없어, 클라이언트가 직접 기상청 API로 요청을 보내야 하는 상황이다. 이에 따라 현재 날씨 정보를 반환하는 엔드포인트를 API 서버에 새로 만들어 클라이언트는 서버와의 통신만 신경 쓸 수 있게 해야 한다.

2) Apache 웹서버와 연동 후 EC2 인스턴스에 배포

현재 Apache와 연동하지 않고 단순히 Django가 제공하는 runserver 명령을 이용해 서버를 실행시키면 접속이 잘되지만, Django의 공식문서에 의하면 이는 보안이나 성능 상의 문제로 Production용 서버로 사용할 수 없다. 따라서 WSGI(Web Server Gateway Interface)를 이용하여 Apache 웹서버와 연동하여 serving할 수 있도록 해야한다.

<https://docs.djangoproject.com/en/3.0/ref/django-admin/#django-admin-runserver>

[참고 4-1] Django 공식문서

| | | | |
|---|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

4.1.3 개선이 필요한 기능

1) 24시간 이전의 날씨 받아오기 기능

현재 기상청에서 제공하는 동네예보 서비스는 요청 시간 기준 24시간 전까지의 데이터만 제공하는 API이다. 따라서, 24시간 이전의 데이터를 받아오려면 다른 방법이 필요한 상황이다. 이에 따라 요청이 왔을 때 기상청 API로 정보를 요청하는 것이 아니라, 특정 시간에 스크립트를 실행하는 형식으로 날씨 정보를 데이터베이스에 7일 간 저장하도록 하고, 이를 활용하도록 한다.


2) 리뷰 작성 기간 제한 기능

날씨 정보는 데이터베이스에 7일 간 저장되고 이후 삭제되므로 사용자가 본인의 코디 리뷰를 남길 수 있는 기간을 제한해야 한다. 따라서 현재로부터 7일 전까지만 리뷰 작성이 가능하도록 한다. 리뷰를 남길 때 해당 코드를 입은 날짜를 7일 보다 이전으로 선택한다면 날짜를 다시 선택하도록 경고문을 띄우도록 한다.

4.1.4 추가가 필요한 기능

1) 간편한 옷 등록 / 코디 등록 기능

현재는 옷을 등록하려면 일일이 사진을 찍어야 하는 번거로움이 있다. 따라서 사용자의 편의성을 위해서 옷을 입고 전신 사진을 찍어 올리면 이를 여러 옷으로 인식하고 각각 분석하여 사용자의 옷장에 자동으로 등록해주는 기능이 필요하다. 이는 앞서 계획한 개발 목표를 빠르게 달성할 경우, 시간적 여유가 된다면 개발하도록 한다.

| | | | |
|--|-------------------------|-------------|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | 옷때? (OTTE?) | |
| | 팀 명 | 옷마이갓 | |
| | Confidential Restricted | Version 1.6 | 2020-APR-20 |

5 고충 및 건의사항

날씨 정보를 받아오는 것이 힘들었습니다. 정확도가 높은 기상청 API를 사용하기로 했지만 기상청 API는 현재 기준 24시 이전의 날씨 정보를 제공하지 않았고 다른 날씨 API들은 유료화가 되어있거나 정확도가 낮았습니다. 기상청의 '과거 날씨' 페이지를 크롤링하여 데이터를 수집하려고도 하였지만 해당 페이지는 사용자의 코디 제안 및 옷 카테고리 추천을 하는 기준인 '체감온도'를 제공하지 않아서 불가능했습니다. 따라서 기상청 API를 통해 특정 시간에 스크립트를 실행하여 일주일 치의 날씨 정보를 저장하고 이를 활용하기로 하였습니다.

코로나-19로 인해서 팀원들과 직접 만나기 힘든 점과 캡스톤 디자인 평가를 온라인 상에서 진행하는 것이 아쉬웠습니다.