

프로젝트명: Weather Wears You

1. 개요

1. 주제

ChatGPT와 기상청 API를 활용한 지정일 날씨에 따른 의상 추천 웹서비스

- 키워드 입력을 통한 이용자의 선호도 반영
- 날짜 입력으로 특정일의 날씨 정보를 제공
- 선호도 정보를 토대로 쇼핑몰의 상품 정보 제공

2. 목표: 제안목표 (수정 시 변경 사유 포함)

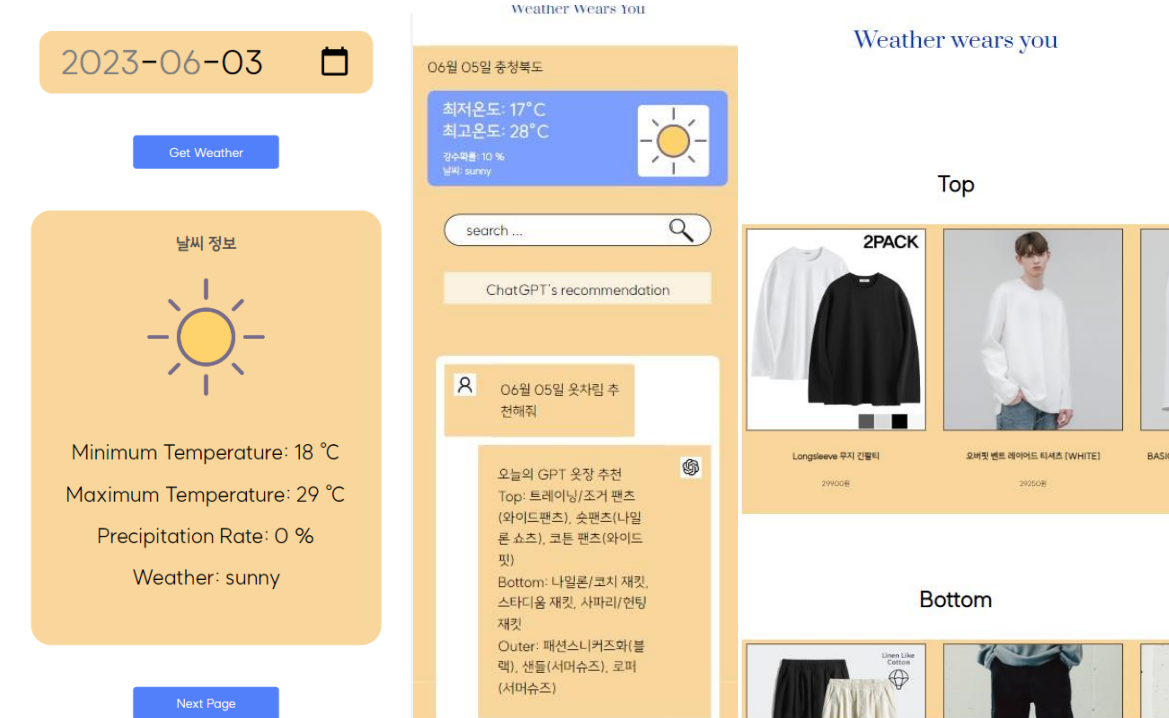
본 프로젝트의 제안 목표는 웹서비스의 사용자들에게 날씨 정보를 제공하여 편의성을 향상시키고, 날씨정보와 회원정보를 기반으로 의상을 추천하며, 이를 기반으로 실제 상품을 추천하는 서비스를 제공하는 것입니다. 현재 존재하는 수많은 패션 추천 서비스들은 이용자의 선호도나 유행을 기반으로 패션 아이템을 제안하고 있지만, 저희 서비스는 이런 기능을 한 단계 더 발전시키고자 합니다. "Weather Wears You"는 고객의 스타일 선호도를 반영하는 동시에, 기상청 API를 이용하여 입력된 날짜의 예상 날씨를 분석, 해당 날씨에 적합한 의상을 추천합니다.

서비스를 이용하는 이용자들은 특정 날짜의 예상 날씨에 맞춘 적절한 의상 조합을 받아볼 수 있을 뿐만 아니라, 이를 바탕으로 쇼핑몰에서 판매중인 상품 정보까지 제공받을 수 있습니다. 따라서 이용자는 자신의 스타일과 예상 날씨를 고려한 패션 아이템을 효과적으로 찾을 수 있을 것입니다.

이와 같은 서비스는 바쁜 현대인들에게 편리성을 제공하고자 하는 본 프로젝트의 목표를 충실히 이행합니다. 그리고 기상 예측의 불확실성이 커진 최근 상황에서, 이 서비스는 실질적인 가치를 제공하게 될 것입니다. 이용자가 불필요한 스트레스 없이 자신의 일상을 계획하고, 자신만의 패션을 완성시킬 수 있도록 도와줄 것입니다.

3. 성과: 수행결과, 산출물, 주요성과 요약

→ 과제 최종 결과물



4. 기여사항: 역할에 따른 개발 범위, 주요이슈, 문제해결, 성과명시

① 최이정

- 팀장(PM) / Backend 개발 / github 관리
- Spring Security 와 MySql 을 통해 회원관리 DB 및 로그인과 회원가입 기능을 관리하는 User API 구현:
 - Spring Security 를 이용하여 MySql 에 User(Username, User ID, User Email, Password Token, Gender)의 정보를 담은 DB 생성하였습니다.
 - User DB 에 저장된 회원을 조회하여 Bearer authentication 을 통해 access 토큰을 인증하는 방식으로 로그인 기능을 구현하였습니다. 이때, 토큰과 함께 회원의 gender 를 함께 전송하였습니다. 해당 gender 정보는 이후 chatGPT 로부터 추천 의상을 얻는 질의를 전송할 때 이용됩니다.
- external API 구현:

→ 기상청 open API 를 통해 3 일 이내 날씨정보를 제공해주는 단기예보 및 3 일에서 10 일까지의 날씨정보를 제공해주는 중기예보를 통합하여, 목표 날짜에 맞는 날씨정보를 제공하는 Weather Controller 를 구현하였습니다.

→ chatGPT open API 에게 질의를 전송하고 답변을 받는 기능을 수행하는 chatGPT Controller 와 Weather Controller 간의 연동을 구현하여 external API 를 완성하였습니다.

→ 크롤링을 통해 저장한 무신사의 상품정보를 ChatGPT 의 답변을 기반으로 추출하기 위해, ChatGPT 에 전송하는 질의 보완 작업을 통해 답변 형식 및 결과를 수정하는 작업을 수행하였습니다.

- **Backend 와 Frontend 간 연동:**

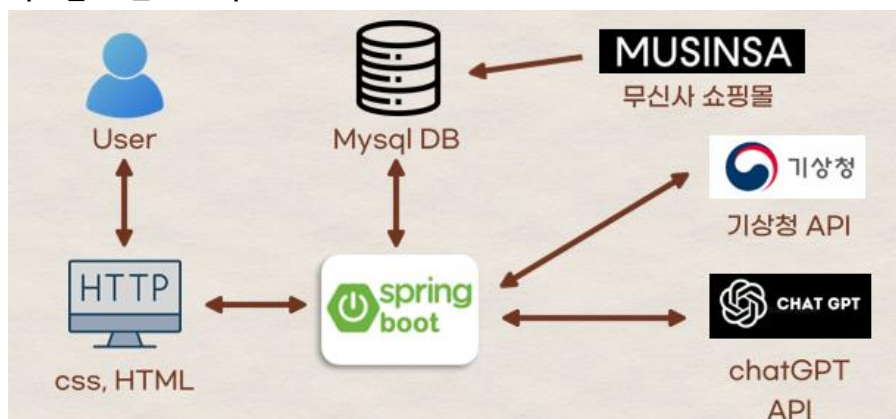
→ Backend 상의 User API 와 external API 가 제공하는 답변을 전송하기 위하여 Frontend 에 Backend 답변을 호출할 수 있는 코드를 추가하였습니다.

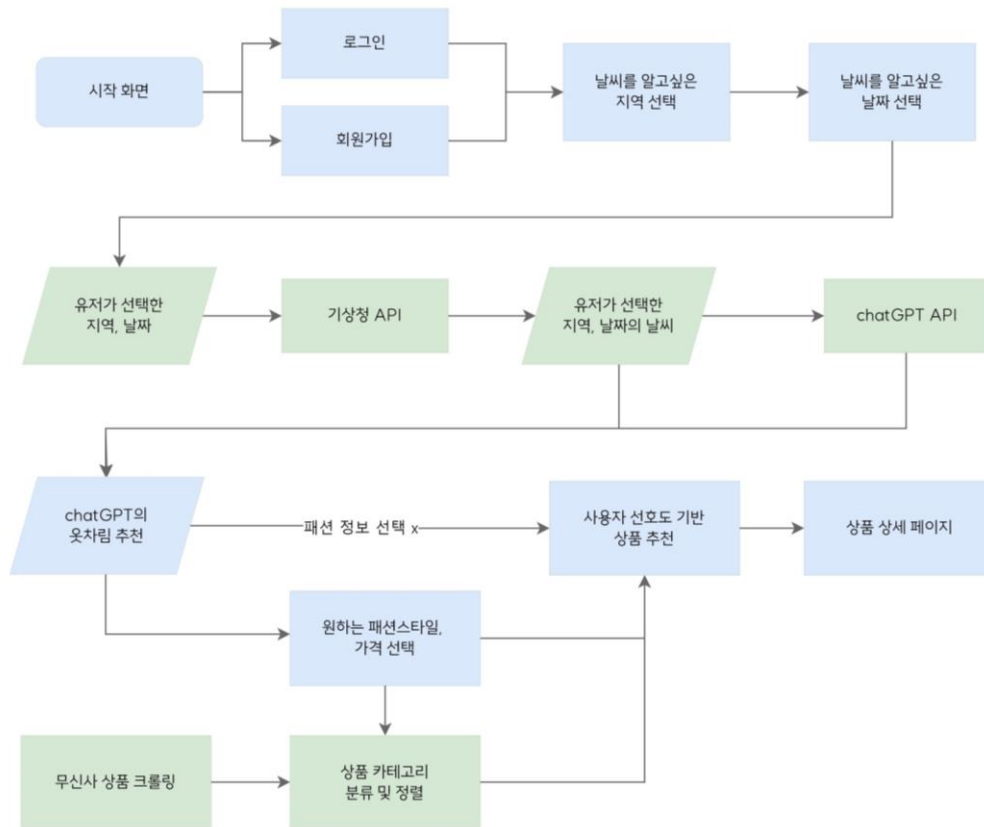
→ Frontend 의 http 서버를 생성하고 Backend 상에 CORS filter 를 추가하여 Backend 와 Frontend 간 연동 작업중 발생하는 CORS 문제를 해결하였습니다.

→ 우리나라 기상청 open API 의 SSL 인증서가 jvm 의 신뢰하는 인증기관 인증서 목록에 등록이 되어 있지 않아 weather controller 에서 오류가 발생하였습니다. 기상청 open API 의 인증서를 프로젝트를 구현중인 버전의 java program file 에 설치해주어 문제를 해결하였습니다.

2. 개발내용

1. 시스템 또는 SW 구조





2. 개발내용 범위 및 개발내용: 주요구성요소, 설계고려사항, 특징점(프로젝트 독창성)등 포함

sprint1: 2023.04.06. - 2023.04.14

<공통>

- 계획안 발표 준비

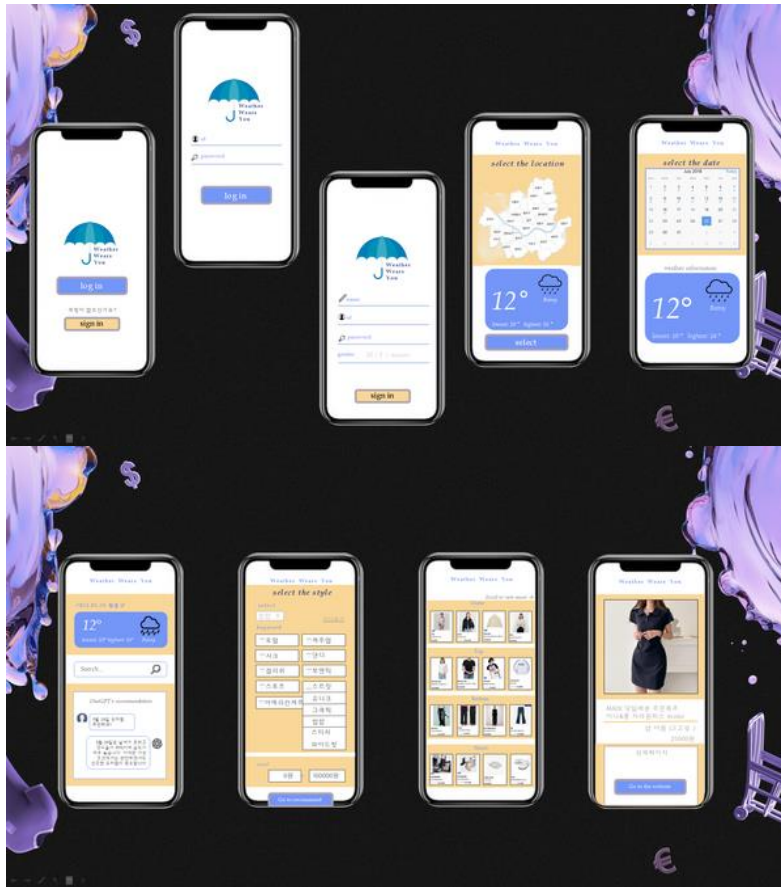
<Front-end>

- ChatGPT 질의 제작

ChatGPT 가 파싱하기 쉽도록 일관성 있는 답변을 하도록 사용자의 성별, 지역, 날짜, 날씨를 포함하는 질의 제작

Give me a word(don't describe with adjectives or colors) for each category (outer, top, bottom, and shoes with 20 examples) in a comma-separated list for a [female] in Seoul on [april 10] with probability of precipitation [90]% , [cloudy] weather, lowest temperature [5] degree celsius and highest temperature [10] degree celsius. Give me only the list, don't tell me additional notes.

- 화면기획 구성



- 스타일 키워드 선정

포멀	단정한	격식	출근룩	미니멀룩	
캐주얼	편안한	꾸안꾸	캠퍼스룩	화사한	
시크	레이어드	모노톤	레더	올블랙	
댄디	데이트룩	남친룩	화사한	와이드핏	
걸리시	귀여운	여친룩	화사한	하이틴룩	레이어드
로맨틱	데이트룩	하객룩	소개팅	상견례	모노톤
스포츠	스포티	편안한	고프고여		
스트릿	유니크	그래픽	힙합	스티치	와이드핏

<Back-end>

- Item 관리 API CRUD

Item

Create: DB에 새로운 상품 등록(크롤링으로)

Read: 회원 선호도, 가격 정보, chatGPT 결괏값을 반영한 상품 정보 받기

Update: 상품 정보 업데이트(제고 변경, 가격 변경)

Delete: DB에서 상품 정보 삭제

- product list 구성: id, 상품 이름, 가격, 이미지 링크 등의 정보를 json file의 형태로 반환
- Item 관리 API spec 정리

Item

(1) `GET/items` 회원 선호도, 가격 정보, chatGPT 답변에 기반한 상품 정보 받기

chatGPT에서 스타일 정보로 물어보지 않고 우리가 직접 분류 시 쿼리 파라미터에 style preference가 들어가야 함

Query parameters: { "priceRange": { "min": "integer", "max": "integer" }, "category": { "top": "string", "bottom": "string", "outer": "string", "shoes": "string" } }

Response: [{ "itemId": "string", "name": "string", "imageUrl": "string", "price": "integer", "category": "string", "itemLink": "string" }]
-> item마다 각자 출력

(2) `PUT/items/:itemId` 상품 정보 업데이트

Request body: { "stock": "integer", "price": "integer" }

Response: { "message": "상품 정보가 성공적으로 업데이트 되었습니다." }

(3) `DELETE /items/:itemId` DB에서 상품 정보 삭제

Response: { "message": "상품 정보가 성공적으로 삭제 되었습니다." }

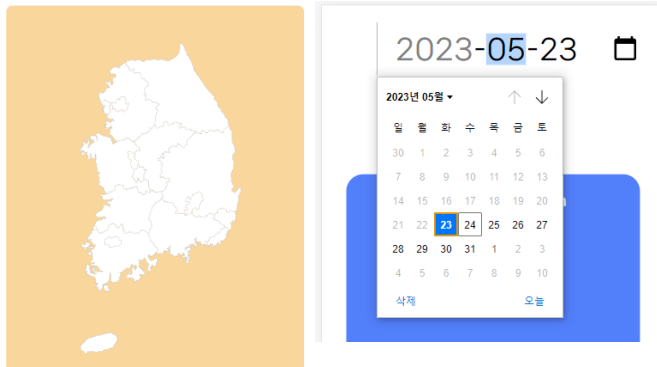
sprint2: 2023.04.13. - 2023.04.30.

<Front-end>

- project 생성
- 마크업(화면 간 이동 구현)
- 지도, 캘린더 화면 제작

Weather Wears You

지역을 선택하세요



<Back-end>

- ChatGPT openapi, 기상청 openapi 사용 방법 research 및 구현
- 기상청 controller

```
@GetMapping("/weather/{cityName}")
public LinkedHashMap<String, Object> getWeatherForDate(@PathVariable String cityName,
                                                         @RequestParam String targetDate)
```

날씨를 받고자 하는 지역과 날짜를 넘겨준다.

```
if (diff >= 0 && diff <= 2) {
    result = kmaClientVilageFcst.getWeatherForSpecificDate(cityName, targetDate);
} else if (diff >= 3 && diff <= 10) {
    LinkedHashMap<String, Integer> tempResult = kmaClientMidTemp.getWeatherForSpecificDate(cityName, targetDate);
    LinkedHashMap<String, String> weatherResult = kmaClientMidWeather.getWeatherForSpecificDate(cityName, targetDate);

    result.put("minTemp", tempResult.get("minTemp"));
    result.put("maxTemp", tempResult.get("maxTemp"));
    result.put("weather", weatherResult.get("weather"));
    result.put("precipitationRate", weatherResult.get("PrecipitationRate"));
} else {
    throw new IllegalArgumentException("Target date must be within 10 days from today.");
}
```

타겟 날짜와 현재 날짜의 차이에 따라 단기 예보에서 받아올지 중기 예보에서 받아올지 결정 후 날씨를 받아온다.

2. chatGPT

```
@PostMapping(path = "/" )
public ResponseEntity<byte[]> chat(@ModelAttribute FormInputDTO dto) {
```

dto에 담긴 사용자의 성별, 선호 스타일 태그, 그리고 날씨 정보를 입력받는다.

```
String prompt = String.format("The available options for itemtypes for each category are the following. I want you to parse through all options and choose the items that best fits %s in the weather on %s with probability of precipitation %s%%, %s weather, lowest temperature %s degree celsius and highest temperature %s degree celsius. (If the chosen itemtype has an additional keyword in the brackets behind the itemtype, it is optional to choose one of them with the itemtype.)" +
    "In the response, the additional keyword should be placed behind each itemtype with brackets. " +
    "Just give me the list of items without the additional notes or apologies.",
    words[0], styleText, words[1], words[2], words[3], words[4], words[5]);
return new CompletionRequest(model: "text-davinci-003", prompt, temperature: 0.7, max_tokens: 500);
```

받은 정보를 토대로 질의를 구성해 chatGPT에 넘겨준 후 답변을 받는다.

- spring batch & security research
- 크롤링 방안 결정 및 sample 생성

```
private static String PATH = "C:\\work\\WeatherWearsYou\\chromedriver.exe"; // WebDriver path

1개 사용 위치   ▲ HongYeSun
public static WebDriver getChromeDriver() {
    if (ObjectUtils.isEmpty(System.getProperty("webdriver.chrome.driver"))) {
        System.setProperty("webdriver.chrome.driver", PATH);
    }
}
```

크롤링은 크롬 드라이버와 selenium을 이용한 동적 크롤링 방식으로 결정

sprint3: 2023.05.01 - 2023.05.17

<Front-end>

- 임시데이터 입히기



<Back-end>

- 크롤링 데이터 생성

category	item_id	gender	img_link	item_type	name	price	style
4	404474	0	https://image.msscdn.net...	긴소매 티셔츠	Longsleeve 무지 긴팔티	29900	#긴팔티#레이어드#티셔츠...
4	659554	0	https://image.msscdn.net...	니트/스웨터	[세트] 하프 폴라 니트 티...	35100	#모크넥#목폴라#반폴라#...
4	1031260	0	https://image.msscdn.net...	피케/카라 티셔츠	23S/S 오버핏 피케티셔...	39900	#반팔PK티셔츠#카라티셔...
4	1557508	2	https://image.msscdn.net...	맨투맨/스웨트셔츠	베츠 어센틱 맨투맨 그레이	27600	#크루넥#맨투맨#스웨트셔...
4	1704897	1	https://image.msscdn.net...	기타 상의	우먼즈 코튼 브라 탑 [플렉]	20290	#브라탑#스포츠브라#무신...
4	1882578	0	https://image.msscdn.net...	피케/카라 티셔츠	스티브 반팔 카라티 / WH...	35910	#반팔티#피케티#카라티#...
4	2038497	2	https://image.msscdn.net...	맨투맨/스웨트셔츠	[SS Ver.]해비 코튼 오버...	36000	#오버핏#렉비티#긴팔티셔...

- item 관리 API 생성 및 DB 구현
- 기상청 및 ChatGPT openapi 연동
- ec2 instance 생성

sprint4: 2023.05.18 - 2023.05.31

<Front-end>

- UI/UX 최종 수정

- ☒ select 화면 다음 성별(남 여 유니섹스) 선택창 추가
- ☒ 무신사에 있는 키워드로 수정 (고프고어 없애고 띄어쓰기 해보기)
- ☒ 로딩 화면 추가 (chatGPT, 상품 로딩창)
- ☒ 상품-아마자 연결
- ☐ 날씨 이미지 연결
- ☒ 날씨-아마자
- ☒ chat 페이지 지역명-영어> 한글
- ☒ chat 페이지 길아 수정
- ☒ map-제주도 깨짐
- ☒ calendar 디자인 논의, 중앙정렬
- ☒ calendar 본문 글자 카우고 나머지 줄아가
- ☒ chat 다음페이지 넘어가는거 수정
- ☒ chat 영어한글
- ☐ 폰트 통일 (로그인까지) 깔끔하게 딱딱 두껍게
- ☒ 폰트 가움임x
- ☒ 로그인 틀렸을 때
- ☐ 지도에 지역 표기
- ☒ 로그인 회원가입 색깔 - gender랑 통일
- ☒ 상세페이지 새로 만들거
- ☒ 스크롤 페이지 간격 카우거

- back 서버와 http 서버 연결

<Back-end>

- 크롤링 카테고리 결정 및 최종 DB 생성

```
public static void doCrawlingByCategoryID(Integer categoryID, Integer productCnt, ItemRepository itemRepository) throws InterruptedException {  
    WebDriver driver = setDriver.getChromeDriver();  
    String url = "https://www.musinsa.com/app/";
```

카테고리 ID와 각 카테고리 당 몇 개의 상품을 가져올지를 인자로 받아 무신사 홈페이지를 순회하며 상품을 받아온다.

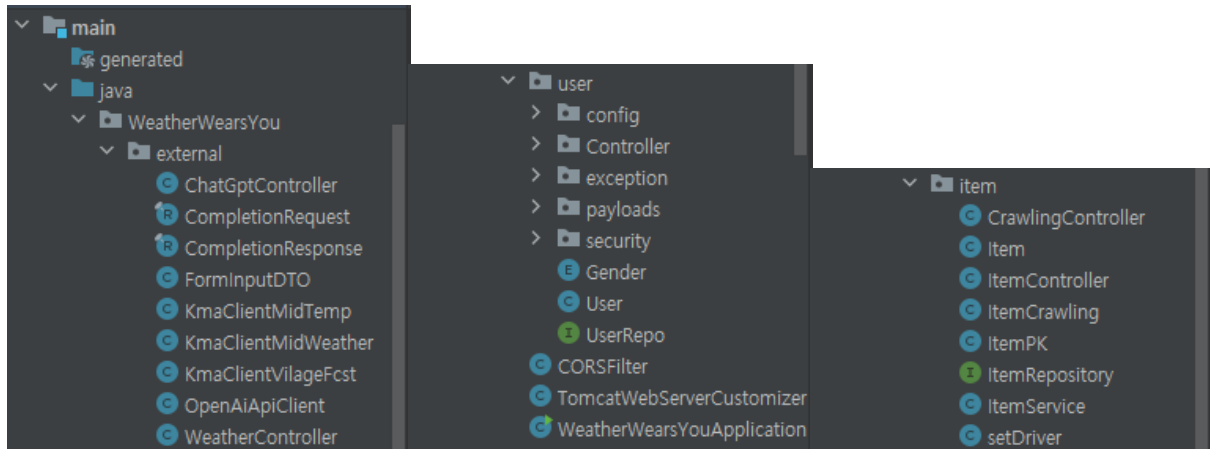
- spring security를 이용한 회원관리 생성(CRUD)

```
@PostMapping("/signin")  
public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {
```

```
@PostMapping("/signup")  
public ResponseEntity<?> registerUser(@Valid @RequestBody SignUpRequest signUpRequest) {
```

- 사용자의 로그인과 회원가입 controller 생성

웹 서비스 프로젝트 소스 최종 구조



3. 이슈 및 해결방안

- UI/UX 수정

기상청 open API의 정보 제공 방식으로 인하여 기존의 서울지도 이미지를 등록했던 것에서 전국 지도 이미지를 등록하는 것으로 변경하였습니다. 기상청 open API를 활용하기 위한 research를 진행한 결과 도나 시 단위보다 구체적인 지역에 대해서는 3일 뒤까지만 기상정보를 제공한다는 것을 확인하였습니다. 따라서 유저가 보다 포괄적인 범위에서 지역을 선택할 수 있도록 지역 선택을 위한 지도 이미지를 전국단위로 변경시켜 주었습니다.

- 스타일 키워드 변경

인터넷 쇼핑몰(무신사)이 제공하는 스타일 키워드와의 통일감을 위하여 자체적으로 선정한 스타일 키워드 대신 인터넷 쇼핑몰에서의 키워드를 참고하여 적용하였습니다.

- 초기 화면기획에서는 캘린더를 단순히 한 화면에 display 하는 방식이었습니다. 다만 해당 방법은 연도와 달을 선택하는 데에 불편함이 있는 방식이라는 생각이 들어 날짜를 선택할 때에 팝업으로 달력이 display 되어 날짜를 선택할 수 있는 화면으로 수정하였습니다. 이 방법을 통해 유저는 자신이 선택한 날짜를 더 명확히 확인할 수 있고 날짜를 선택할 때에도 더 편리하게 선택할 수 있게 되었습니다.

- 우리나라 기상청 openAPI의 SSL 인증서가 jvm의 신뢰하는 인증 기관 인증서 목록에 등록이 되어있지 않았기 때문에 지속적으로 기상청 openapi를 이용하는 controller에서 "Java PKIX path building failed"와 같은 오류가 발생하였습니다. 따라서 기상청 open API의 인증서를 프로젝트를 구현중인 버전의 java program file에 설치해줌으로써 문제를 해결할 수 있었습니다.

- Backend 에서 구현한 API 를 Frontend 에 연동시켜주는 과정에서 CORS 권한 문제가 발생하였습니다. 해당 문제는 SOP 때문에 발생한 것으로, 두 서버의 origin 이 다르기 때문에 발생한 것이었습니다. 하여 Backend 코드의 main 함수에 WebMvcConfigurer 를 추가해주고, 각 컨트롤러에 frontend 서버의 주소를 담은 Cross Origin 어노테이션을 추가해주었습니다. 다만, 이 방법을 통해서도 문제가 해결되지않아 지속적으로 여러 가지 시도를 해 본 결과 Frontend 서버가 제대로 된 port 를 가진 서버로 통신을 하고 있지 않았던 점이 문제였다는 것을 알게 되었습니다. 따라서 frontend 프로젝트에 테스트용 http server 를 설치해주어 연동시켜 준 결과, 문제를 해결 할 수 있었습니다.

- DB 에 크롤링을 통해 얻은 데이터를 테스트하며 저장하는 과정에서 test 를 성공적으로 마친 후에 데이터가 디스크에 저장되지 않고 휘발되는 문제가 지속적으로 발생하였습니다. 해당 문제는 DataJpaTest 어노테이션을 활용하면서 자동적으로 DB 가 테스트 전의 상황으로 롤백되어 발생하는 문제였습니다. 어플리케이션을 구동하여 정상적으로 크롤링하여 DB 에 성공적으로 데이터들을 저장할 수 있었습니다.

- Frontend 의 지도를 구현하는 과정에서 현재 선택한 지역을 취소하고 새로운 지역을 선택하였을 때 색칠된 구역이 두 개로 중복되는 문제가 지속적으로 발생하였습니다. 디버깅을 통해 javascript 의 자동 실행 함수인 ready 함수 속 onclick 이벤트가 중복 발생 중이라는 사실을 찾아냈고 이를 수정해 해당 문제를 해결할 수 있었습니다.

- Frontend 에 chatGPT 와 상품 정보를 받아올 때까지 시간이 오래 걸리는 사실을 고려하여 로딩 화면을 추가해 사용자에게 정상적으로 작동 중이라는 사실을 알리도록 화면을 추가 구성하였습니다.

- ChatGPT API가 null response를 주는 일이 주기적으로 발생하였습니다. 이는 계정의 free trial 을 다 써서 api key 가 만료됐거나 public git 에 key 를 업로드하면 보안 문제로 OpenAI 에서 api key 를 비활성화시켰기 때문이었습니다. 다른 계정으로 key 를 새로 생성하거나 key 를 새로 발급받아 해당 문제를 해결하였습니다.

- ChatGPT 는 상의, 하의, 외투, 신발 총 네 가지의 항목마다 세 개의 카테고리를 추천해주고 있습니다. 그리고 각 카테고리마다 선택적으로 스타일 키워드 해시태그를 추천해줍니다. 따라서 chatGPT 의 선택에 의해 카테고리에 구체적 스타일 키워드가 있을 수도, 없을 수도 있습니다. 크롤링으로 DB 에 상품을 저장할 때에는 각 카테고리 별로 최근 1 년간의 판매 순위를 매겨 rank 형태로 순위가 같이 저장되게 됩니다. 그리고 조건에 맞는 상품을 찾아올 때에는 조건에 맞는 상품들을 모두 넣은 배열에서 판매 순위로 정렬 후 제일 높은 10 개를 뽑는 방식입니다. 그러나 이렇게 되면 상세 스타일 해시태그가 없는 카테고리는 판매 순 1 위부터 10 위가 배열에 들어오게 되고, 결과에는 하나의 카테고리만 보여지는 문제가 있었습니다. 따라서 다양성을 위해 각 카테고리 별로 무조건 3-4 개씩 상품이 보이도록 하였습니다.

- ChatGPT 답변의 일관성 부족 및 ChatGPT 답변과 인터넷 쇼핑몰 카테고리 매칭에 대한 해결책을 강구하였습니다. 초기 모델은 막연히 날씨정보, 성별, 그리고 스타일 키워드를 포함한 질의를 영어로 하여 영어로 답변을 얻었다면, 수정을 거친 모델에서는 질의에 기존 정보는 모두 포함하되, 답변으로 제공할 수 있는 아이템들에 선택 범위를 포함시켰습니다. 기존의 방법은 영어로 되어 있고 범위가 광범위하여, 번역 문제 및 쇼핑몰에서 제공하는 아이템들을 효과적으로 필터링 할 수 없다는 한계를 가지고 있었습니다. 또한 매번 일관적인 답변을 제공해주지 못한다는 문제가 있었습니다. 따라서, 질의에 ChatGPT 가 제공해 줄 답변 형태를 포함시켜 매번 같은 형태의 답변을 받을 수 있도록 만들어주었습니다. 또한, 데이터를 제공받을 인터넷 쇼핑몰(무신사)의 카테고리 분류기준과 세부스타일을 참고하여 해당 기준들 내에서만 답변을 할 수 있도록 리스트를 만들어 질의에 포함시켰습니다. 그 결과, 정해진 범위 내에서 일정한 형식을 가진 한국어 답변을 받을 수 있었습니다.

4. 성과

1. 목표치: 기능, 성능(정량, 정성)목표

- **보안:** SprintSecurity 를 이용한 회원 가입 및 로그인 기능 구현
- **크롤링:** 무신사 홈페이지에 대한 크롬 드라이버와 selenium 을 이용한 동적 크롤링 수행
- **데이터베이스 관리:**
 - 회원관리 DB 생성 및 관리
 - 무신사에서 크롤링을 통해 수집한 아이템 관리
 - ChatGPT API 로부터 받은 답변의 파싱을 통해 얻은 키워드를 기반으로 아이템관리 DB 로부터 조건에 맞는 아이템 추출
- **프론트엔드 화면구성 및 백엔드와의 연동:**
 - 시작화면 및 로그인 화면 구현, 백엔드의 user controller 와 연동
 - 캘린더 및 날짜 화면 구현
 - 날씨정보 및 ChatGPT 의 답변 제공을 위한 화면 구현
 - 백엔드의 openAPI 활용을 위한 ChatGPT controller 과 연동
 - ChatGPT API 에 보낼 질의 보완 및 아이템 추출을 위한 정보 수집에 이용될 gender 선택과 스타일 및 가격 선택 화면 구현
 - 아이템 관리 DB 로부터 추출된 아이템 display 를 위한 scroll page 구현
 - 백엔드의 Item controller 와의 연동

- OpenAPI 활용

- 기상청 openAPI 활용을 위한 weather controller 생성
- ChatGPT openAPI 활용을 위한 chatGPT controller 생성

2. 수행결과: 목표대비 성과, 성과미달 시 사유, 원인분석

제안발표시 목표로 했던 개발 범위를 100% 완성할 수 있었습니다.

개발을 위하여 구현에 이용된 코드는 다음과 같습니다.

- Frontend:

- 로그인 화면 구현(login.html): 로그인 기능은 Bearer authentication 을 통해 access 토큰을 인증하는 방식으로 로그인 기능을 구현하였습니다. 이 과정에서 로그인에 성공하는 경우에만 회원관리 DB로부터 회원의 gender 에 대한 정보도 전송 받아 chatGPT 에 전송하는 질의에 gender 에 대한 정보도 포함시킬 수 있도록 하였습니다. 이후 화면은 map.html 로 넘어가게 됩니다.

```
document.getElementById('login').addEventListener('click', function(event) {
    event.preventDefault();

    // Extract info from form
    const id = document.getElementById('id').value;
    const password = document.getElementById('password').value;

    // Send a POST request to the API
    fetch('http://localhost:8080/api/auth/signin', { // change here
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            usernameOrEmail: id,
            password: password,
        }),
    })
    .then(response => response.json())
    .then(data => {
        console.log('Response:', data);
        if (data.accessToken) {
            alert('Login successful!');
            sessionStorage.removeItem('gender');
            sessionStorage.setItem('gender', data.gender);

            window.location.href = "map.html";
        } else {
            alert('Login failed: ' + data.message);
        }
    })
    .catch((error) => {
        console.error('Error:', error);
    });
});
```

→ 회원가입 화면 구현(sign.html): 신규회원은 username, email, password, gender 를 입력하여 회원가입을 하고, 해당 정보는 backend 의 회원관리 DB 에 저장됩니다. 회원가입에 성공하는 경우에는 map.html 화면으로 넘어갑니다. 다만, 이미 존재하는 회원이거나, 회원가입 형식에 맞지 않는 입력을 하면 에러메세지가 송출됩니다.

```
document.getElementById('signup-form').addEventListener('submit', function(event) {
    event.preventDefault();

    // Fetch form to apply custom Bootstrap validation
    var form = document.getElementById("signup-form");

    if (form.checkValidity() === false) {
        event.stopPropagation()
    } else {
        // Extract info from form
        const formData = new FormData(form);
        const username = formData.get('username');
        const email = formData.get('email');
        const password = formData.get('password');
        const gender = formData.get('gender');

        // Send a POST request to the API
        fetch('http://localhost:8080/api/auth/signup', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                username: username,
                email: email,
                password: password,
                gender: gender,
            }),
        })

        .then(response => response.json())
        .then(data => {
            if (data.success) {
                sessionStorage.setItem('gender', gender);
                alert('Sign up successfull!');
                window.location.href = "map.html";
            } else {

```

→ 캘린더 화면 구현(calender.html): 회원이 선택한 날짜에 대한 정보를 수집합니다.

```
</header>
<div id="top">
    <p><section id="intro">날짜를 선택하세요</section></p>
    <input type="date" id="date-picker" min="" max="" onchange="checkDateAvailability()">
    <button onclick="getWeather()">Get Weather</button>

    <div id="weather-box">
        <h2>날씨 정보</h2>
        <img id="weather-image" src="" alt="Weather Image" style="display: none; width: 300px; height: 300px; margin-top: 50px; margin-bottom: 30px;">
        <div id="weather-info"></div>
    </div>

    <button onclick="navigateToNextPage()">Next Page</button>

    <script>
        var minDate = new Date();
        var maxDate = new Date();
        maxDate.setDate(maxDate.getDate() + 9);
        document.getElementById("date-picker").min = minDate.toISOString().slice(0, 10);
        document.getElementById("date-picker").max = maxDate.toISOString().slice(0, 10);

        function checkDateAvailability() {
            var selectedDate = new Date(document.getElementById("date-picker").value);
            var today = new Date();
            if (selectedDate > maxDate) {
                document.getElementById("date-picker").value = "";
                alert("Please select a date between " + (today.getMonth() + 1) + "/" + today.getDate() + "/" + today.getFullYear() + " and " + (maxDate.getMonth() + 1) + "/" + maxDate.getDate() + "/" + maxDate.getFullYear());
            }
        }
    </script>
</div>
```

→ 백엔드의 weathercontroller 와의 연동: 회원이 선택한 날짜 정보와 map.html 에서 선택된 지역정보를 기반으로 weather controller 에 접속하여 해당 조건에 맞는 날씨 정보를 제공받습니다.

```
function getWeather() {
    var selectedDate = new Date(document.getElementById("date-picker").value);
    var city = sessionStorage.getItem('location');
    var url = `http://localhost:8080/weather/${encodeURIComponent(city)}?targetDate=${selectedDate.toISOString().slice(0, 10).replace(/-/g, "")}`;
    fetch(url)
        .then(response => {
            if(response.ok) {
                return response.json();
            }
            throw new Error("Network response was not ok.");
        })
        .then(data => {
            console.log(data);
            var minTemperature = data.minTemp;
            var maxTemperature = data.maxTemp;
            var precipitationRate = data.precipitationRate;
            var weather = data.weather;

            if(sessionStorage.getItem('date')){
                sessionStorage.removeItem('date');
            }
            var storedDate=(selectedDate.getFullYear().toString()+"00"+selectedDate.getMonth()+1).toString().slice(-2)+"00"+(selectedDate.getDate()).toString().slice(-2);
            sessionStorage.setItem('date', storedDate);

            var weatherInfo = document.getElementById("weather-info");
            weatherInfo.innerHTML = "<p>Minimum Temperature: " + minTemperature + " °C</p><p>Maximum Temperature: " + maxTemperature + " °C</p>";

            sessionStorage.setItem("maxTemp", maxTemperature);
            sessionStorage.setItem("minTemp", minTemperature);
            sessionStorage.setItem("precipitationRate", precipitationRate);
            sessionStorage.setItem("weather", weather);
        });
}
```

→ 지도 화면 구현(calendar.html): 회원이 선택한 지역에 대한 정보를 수집합니다. 지역을 선택할 때마다 선택된 지역의 색이 바뀌어 선택되지 않은 구역과 구별될 수 있게 만들어주었습니다. 또한, 해당 회원이 선택한 이전 지역정보를 저장하여, 이전에 선택하였던 지역을 다시 선택할 수 있는 서비스를 제공하였습니다.

```
function check(){
    if(!sessionStorage.getItem("location")){
        alert("지역을 선택해주세요.");
        url="#";
    }
    else{
        var url="calendar.html"
        location.href=url;
    }
}

function go_branch(city_do) {
    var Arr = Array("sejong", "chungnam", "jeju", "gyeongnam", "gyeongbuk",
        "jeonbuk", "chungbuk", "gangwon", "gyeonggi", "jeonnam",
        "ulsan", "busan", "daegu", "daejeon", "incheon", "seoul",
        "gwangju");

    var strArr = Array("세종특별자치시", "충청남도", "제주특별자치도", "경상남도", "경상북도",
        "전라북도", "충청북도", "강원도", "경기도", "전라남도", "울산광역시", "부산광역시",
        "대구광역시", "대전광역시", "인천광역시", "서울특별시", "광주광역시");

    //alert(city_do);
    var idx = Arr.indexOf(city_do);
    //alert(strArr[idx]);
}
```

```

//console.log(city_do)
var re=confirm(strArr[idx]+" 선택하시겠습니까?");

if(re==true){
    alert("선택되었습니다.");
    var mapCondition =strArr[idx];

    if(sessionStorage.getItem('location')){
        sessionStorage.removeItem('location');
    }
    sessionStorage.setItem('location', city_do);
}

681 <div class="dg-map-area">
682 <div class="dg-map dg-map-in">
683 <svg x="0" y="0" width="180%" height="200%"
viewBox="0 0 800 1200" >
684 <!-- Add the polygon element. -->
685 <polyline id="chungnam" onclick="go_branch
('chungnam')" stroke-miterlimit="1" points="204.28,
468.979 203.321,470.128 202.363,470.32 201.788,
479.519 201.597,480.094 201.405,484.118 201.213,485.
077 199.68,488.334 199.489,488.718 199.105,488.91
198.147,490.826 196.614,496.19 196.422,497.148 196.
614,498.298 196.997,498.49 197.189,498.49 197.572,
499.065 198.339,499.64 198.53,499.832 199.297,500.
023 199.68,500.215 200.063,500.598 200.255,500.981
200.639,501.173 201.213,501.556 201.788,501.748
202.171,502.131 202.555,502.131 203.13,502.131 203.
896,502.515 203.321,502.898 202.938,503.089 202.747,
503.281 202.363,503.665 201.788,504.623 201.98,
505.198 201.597,506.539 202.171,506.922 203.705,509.
414 204.28,510.755 206.004,511.33 206.004,510.755
206.58,510.372 206.771,510.563 207.346,510.947 208.
113,511.522 208.304,511.905 210.604,517.271 211.179,
519.377 211.754,519.186 211.945,518.611 212.521,
518.038 212.712,517.271 213.479,516.505 213.67,516.
313 213.67,514.972 214.054,514.396 214.245,514.
013 214.82,513.247 215.012,513.247 215.395,513.

```

→ chatGPT 응답 display 화면(chat.html): 이전의 calender.html, map.html, login.html 또는 signin.html에서 수집된 날짜, 지역, 성별 정보를 기반으로 ChatGPT controller에 접속하여 ChatGPT 에게 위 정보가 포함된 질의를 전송하고, 그에 맞는 의상조합을 제공해주는 응답을 받을 수 있도록 하였습니다.

```

function getGPTChat() {
    // Assuming gender and targetDate are available
    var gender = sessionStorage.getItem("gender");
    var targetDate = sessionStorage.getItem("date");
    var city = sessionStorage.getItem("location");

    var weather = sessionStorage.getItem("weather");
    var weatherImage = document.getElementById("weather-image");
    switch (weather) {
        case "sunny":
            weatherImage.src = "weather_img/sunny.png";
            weatherImage.style.display = "block";
            break;
        case "rainy":
            weatherImage.src = "weather_img/rainy.png";
            weatherImage.style.display = "block";
            break;
        case "cloudy":
            weatherImage.src = "weather_img/cloudyy.png";
            weatherImage.style.display = "block";
            break;
        case "blurry":
            weatherImage.src = "weather_img/blurry1.png";
            weatherImage.style.display = "block";
            break;
    }
}

```


→ 백엔드의 chatGPT controller 와의 연동

```
fetch("http://localhost:8080", {
  method: "POST",
  body: formData
})
.then(response => response.json())
.then(data => {
  const ele=document.getElementById('boxLoading');
  ele.remove();
  div1.style.display='block';
  div2.style.display='block';
  div3.style.display='block';
  div4.style.display='block';
  div5.style.display='block';
  var chat = "오늘의 GPT 옷장 추천<br>";
  if(sessionStorage.getItem("minTemp") > 24) {
    delete data.Outer;
  }
  for (const [key, value] of Object.entries(data)) {
    chat += key + ": " + value.join(", ") + "<br>";
  }

  document.getElementById("gptchat").innerHTML = chat;
})
.catch(error => {
  console.error('Error:', error);
});
```

→ 로그인 시 참조되는 gender 에서 사용자가 새로 선택한 gender 로 갱신(gender.html): 회원이 다시 성별을 선택할 수 있는 페이지를 만들어, 자신이 입을 의상 조합에 대한 추천이 아닌 다른 사람을 위해 추천해줄 수 있는 의상 조합을 받아 볼 수 있는 서비스를 제공하였습니다. 이때, 기존의 로그인 및 회원가입 화면에서 입력되었던 성별에 대한 정보는 newgender 로 갱신됩니다.

```
<script type="text/javascript">
function saveGender(event) {
  sessionStorage.removeItem("newgender");
  const genderNodeList = document.getElementsByName("gender");
  // console.log(document.getElementsByName("gender"))
  var i = 0;
  genderNodeList.forEach((node) => {
    if (node.checked) {
      sessionStorage.setItem("newgender", node.id);
      console.log(node.id);
      return 0;
      // document.getElementById("result").innerText = node.value;
    }
  });
  if (!sessionStorage.getItem("newgender")) {
    alert("성별 선택 없이 상품을 추천합니다.");
  }
}
```

→ 스타일 및 가격 범위 입력을 위한 화면 구현(select.html): 스타일 키워드를 미리 생성하여 사용자가 선택을 하면 ChatGPT openAPI 에 전송될 질의에 추가적으로 포함될 수 있도록 만들어주었습니다. 또한, 가격 범위를 입력 받아, 크롤링한 상품들이 해당 조건에 맞게 추출될 수 있도록 하였습니다.

```
function handleOnChange(e) {
  // 선택된 데이터의 텍스트값 가져오기
  const text = e.options[e.selectedIndex].text;

  console.log(e.name);
  if (
    sessionStorage.getItem("bigcategory") ||
    sessionStorage.getItem("smallcategory")
  ) {
    sessionStorage.removeItem("bigcategory");
    sessionStorage.removeItem("smallcategory");
  }
  sessionStorage.setItem("bigcategory", e.name);
  sessionStorage.setItem("smallcategory", text);
  // 선택한 텍스트 출력
  document.getElementById("result").innerText = text;
}

function maxcost(c) {
  if (isNaN(c)) {
    alert("숫자만 입력 가능합니다.");
    return;
  }
  console.log(c);
  if (c < sessionStorage.getItem("mincost")) {
    alert(sessionStorage.getItem("mincost") + "값보다 작을 수 없습니다.");
    sessionStorage.removeItem("maxcost");
    return;
  }
  if (sessionStorage.getItem("maxcost")) {
    sessionStorage.removeItem("maxcost");
  }
  sessionStorage.setItem("maxcost", c);
}

function mincost(c) {
  if (isNaN(c)) {
    alert("숫자만 입력 가능합니다.");
    return;
  }
}
```

→ 아이템 관리 DB 로부터 추출된 아이템 display 를 위한 scroll page 구현 및 백엔드의 Item controller 와의 연동(scrollpage.html): 새롭게 갱신된 성별 정보와 select.html 에서 저장한 스타일 키워드, 기존의 지역 및 날씨 정보를 포함한 질의를 ChatGPT controller 를 통해 전송하여, 추천 상품에 대한 응답을 받아옵니다. 이후 해당 답변과 가격조건을 기반으로 아이템이 저장된 DB 로부터 상품을 추출하여, display 해 줄 수 있게 만들어주었습니다. 단, 10 도 이상의 온도에서는 주로 외투를 입지 않는다는 점을 감안하여, ChatGPT controller 로부터 outer 에 대한 응답을 받지 않도록 하기에, 해당 코드에서도 outer 에 대한 상품이 선택되지 않은 경우에는 outer 를 화면에 display 하지 않습니다. 최종적으로 사용자가 scroll page 에서 아이템을 선택하면, 해당 아이템이 있는 무신사 링크로 이동하게 됩니다.

```

function getItems() {

    let loader = `상품을 불러오는 중입니다.<br>잠시만 기다려 주세요.`;
    document.getElementById('boxLoading').innerHTML = loader;
    const div1=document.getElementById('section_0');
    const div2=document.getElementById('section_1');
    const div3=document.getElementById('section_2');
    const div4=document.getElementById('section_3');

    div1.style.display='none';
    div2.style.display='none';
    div3.style.display='none';
    div4.style.display='none';

    // Assuming gender and targetDate are available
    var city = sessionStorage.getItem("location");
    var gender = sessionStorage.getItem("newgender"); // Replace with actual value
    var targetDate = sessionStorage.getItem("date"); // Replace with actual value
    var style =
        sessionStorage.getItem("bigcategory") +
        ", " +
        sessionStorage.getItem("smallcategory");
    var weather = sessionStorage.getItem("weather");
    var precipitationRate = sessionStorage.getItem("precipitationRate");
    var minTemp = sessionStorage.getItem("minTemp");
    var maxTemp = sessionStorage.getItem("maxTemp");

    // Form data
    var formData = new FormData();
    formData.append("city", city);
    formData.append("gender", gender);
    formData.append("targetDate", targetDate);
    formData.append("style", style);
    formData.append("weather", weather);
    formData.append("precipitationRate", precipitationRate);
    formData.append("minTemp", minTemp);
    formData.append("maxTemp", maxTemp);

    // Fetch request and form URL
    var mincost = sessionStorage.getItem("mincost");
    var maxcost = sessionStorage.getItem("maxcost");
    fetch("http://localhost:8080", {
        method: "POST",
        body: formData,
    })
    .then((response) => response.json())
    .then((data) => {
        const ele=document.getElementById('boxLoading');
        ele.remove();
        if(sessionStorage.getItem("minTemp") < 25) {
            div1.style.display='block';
        }
        div2.style.display='block';
        div3.style.display='block';
        div4.style.display='block';
        itemURL = `http://localhost:8080/items?chatGPTResponse=${JSON.stringify(data)}`;
        if (mincost != 0) {
            itemURL += "&minPrice=" + mincost;
        }
        if (maxcost != 10000000) {
            itemURL += "&maxPrice=" + maxcost;
        }
        itemURL += "&gender=" + gender;

        console.log(itemURL);
        return fetch(itemURL);
    })
}

```

```

.then((response) => response.json())
.then((data) => {
    var prodId;
    var index = [0, 0, 0, 0];
    var frontURL = "https://www.musinsa.com/app/goods/";
    for (var i = 0; i < data.count; i++) {
        switch (data.data[i].category) {
            case 0:
                prodId = "Top" + index[0];
                index[0]++;
                break;
            case 1:
                prodId = "Bottom" + index[1];
                index[1]++;
                break;
            case 2:
                prodId = "Outer" + index[2];
                index[2]++;
                break;
            case 3:
                prodId = "Shoes" + index[3];
                index[3]++;
                break;
        }
    }
}

```

- OpenAPI 활용

→ 기상청 openAPI 활용을 위한 weather controller 생성: 중기기온예보, 중기일기예보, 단기예보 3 가지로 분리되어 제공되는 기상청 openAPI 의 날씨 및 기온 정보를 수집하기 위해 weather controller 를 생성해주었다. 기상청 open API 는 다양한 정보들을 제공해주지만, 그 중에서도 최고온도, 최저온도, 날씨정보, 강수량을 추출하였다. 사용자가 원하는 날짜에 맞는 날씨정보를 제공해줄 수 있도록 목표로 하는 날짜가 오늘로부터 며칠 떨어져 있는지 계산 하는 수식을 생성하여 3 일 미만 떨어져 있는 경우에는 단기예보를 이용하고, 3 일 이상 떨어져 있는 경우에는 중기예보를 이용할 수 있도록 하였다.

```

@GetMapping("/weather/{cityName}")
public LinkedHashMap<String, Object> getWeatherForDate(@PathVariable String cityName,
                                                         @RequestParam String targetDate)
    throws UnsupportedOperationException, ParseException {
    LinkedHashMap<String, Object> result = new LinkedHashMap<>();

    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyyMMdd");
    Date today = new Date();
    Date target = sdf.parse(targetDate);
    Long diff = (target.getTime() - today.getTime()) / (24 * 60 * 60 * 1000) + 1;

    if (diff >= 0 && diff <= 2) {
        result = kmaClientVilageFcst.getWeatherForSpecificDate(cityName, targetDate);
    } else if (diff >= 3 && diff <= 10) {
        LinkedHashMap<String, Integer> tempResult = kmaClientMidTemp.getWeatherForSpecificDate(cityName, targetDate);
        LinkedHashMap<String, String> weatherResult = kmaClientMidWeather.getWeatherForSpecificDate(cityName, targetDate);

        result.put("minTemp", tempResult.get("minTemp"));
        result.put("maxTemp", tempResult.get("maxTemp"));
        result.put("weather", weatherResult.get("weather"));
        result.put("precipitationRate", weatherResult.get("PrecipitationRate"));
    } else {
        throw new IllegalArgumentException("Target date must be within 10 days from today.");
    }

    return result;
}

```

→ ChatGPT openAPI 활용을 위한 chatGPT controller 생성: chatGPT 에 회원의 성별, 스타일, 지정일의 날씨정보가 담긴 질의를 전송하고, 그에 따른 추천 의상이 담긴 답변을 받는 역할을 수행하는 controller 입니다. 우선, api key 오류나 인터넷 불안정으로 인한 오류가 발생하는 것을 대비하기 위하여 질의를 전송하거나 답변을 받는데 오류가 나면, 최대 10 번까지 질의를 재전송하여, 올바른 답변을 받아볼 수 있도록 해주었습니다. chatGPT 로부터 전송 받은 답변은 모두 Top, Bottom, Outer, Shoes 에 따라 “#”로 분리되어있는 일정한 형식을 가지고 있기 때문에, 이에 따라 parsing 해주어 아이템들을 분리합니다. 또한, 최고온도가 10 도 이상의 기온일 때에는 주로 outer 를 입지 않는다는 판단에 따라 기존 답변에서 outer 를 제거해주는 형태로 답변을 수정해줍니다.

```
private String chatWithGpt3(String[] message, String style) throws Exception {
    for(int i = 0; i < MAX_RETRIES; i++) {
        try {
            var completion = CompletionRequest.defaultWith(message, style);
            var postBodyJson = jsonMapper.writeValueAsString(completion);
            var responseBody = client.postToOpenAiApi(postBodyJson, OpenAiService.GPT_3);
            var completionResponse = jsonMapper.readValue(new String(responseBody, StandardCharsets.UTF_8), CompletionResponse.class);
            return completionResponse.firstAnswer().orElseThrow();
        } catch (Exception e) {
            System.out.println("Error occurred: " + e.getMessage());
            if (i == MAX_RETRIES - 1) {
                throw e; // re-throw the exception if on last retry
            }
        }
    }
    throw new Exception("Failed to get a response from the API after " + MAX_RETRIES + " attempts");
}

@PostMapping(path = "/")
public ResponseEntity<byte[]> chat(@ModelAttribute FormInputDTO dto) {
    HttpHeaders responseHeaders = new HttpHeaders();
    responseHeaders.setContentType(MediaType.valueOf("application/json;charset=UTF-8"));

    try {
        String city = dto.getCity(); // Get the city information from the frontend
        String gender = dto.getGender(); // Get the gender information from the frontend
        String targetDate = dto.getTargetDate(); // Get the target date information from the frontend
        String modifiedPrompt;
        String maxTemp = null;
        if(dto.getWeather() == null) {
            LinkedHashMap<String, Object> weatherData = weatherController.getWeatherForDate(city, targetDate);
            maxTemp = Integer.toString((Integer) weatherData.get("maxTemp"));
            modifiedPrompt = String.format("%s,%s,%s,%s,%s,%s",
                gender, targetDate,
                weatherData.get("precipitationRate"),
                weatherData.get("weather"),
                weatherData.get("minTemp"),
                maxTemp);
        } else {
            maxTemp = dto.getMaxTemp();
            modifiedPrompt = String.format("%s,%s,%s,%s,%s,%s",
                gender, targetDate,
                dto.getPrecipitationRate(),
                dto.getWeather(),
                dto.getMinTemp(),
                maxTemp);
        }
    }
}
```

```

String response;
if (dto.getStyle() == null) {
    response = chatWithGpt3(modifiedPrompt.split( regex: "#"), style: null);
} else {
    response = chatWithGpt3(modifiedPrompt.split( regex: "#"), dto.getStyle());
}
String[] categories = response.split( regex: "\\r?\\n");

categories = Arrays.copyOfRange(categories, from: 2, categories.length);

LinkedHashMap<String, List<String>> results = new LinkedHashMap<>();
String[] categoriesKeys = {"Top", "Bottom", "Outer", "Shoes"};

for (int i = 0; i < categories.length; i++) {
    String[] items = categories[i].substring( beginIndex: categories[i].indexOf(":") + 2).trim().split( regex: "#");
    List<String> translatedItems = new ArrayList<>();
    for (String item : items) {
        String translatedItem = TranslationUtil.translateEngToKor(item);

        if (!translatedItem.matches( regex: "[a-zA-Z ]+") || item != "") {
            translatedItems.add(translatedItem);
        }
    }
    results.put(categoriesKeys[i], translatedItems);
}

// Remove 'Outer' if max temperature is >= 10
if (maxTemp != null && Double.parseDouble(maxTemp) >= 10) {
    results.remove( key: "Outer");
}

```

- 보안: SpringSecurity 를 이용한 회원 가입 및 로그인 기능 구현

로그인 기능 구현: User DB 에 저장된 회원을 조회하여 Bearer authentication 을 통해 access 토큰을 인증하는 방식으로 로그인 기능을 구현하였습니다. 이때, 토큰과 함께 회원의 gender 를 함께 전송하였습니다. 해당 gender 정보는 이후 chatGPT 로부터 추천 의상을 얻는 질의를 전송할 때 이용됩니다. 'WeatherWearsYou'가 회원관리 DB 를 생성하는 이유는 보안이 보다는 회원정보를 활용하여 chatGPT 로부터 받는 답변의 정확도를 높이는 것이 목적이기 때문에, 토큰이 갱신되지 않아도 되도록 구현하였습니다.

```

@PostMapping("/signin")
public ResponseEntity<> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {

    Authentication authentication = authenticationManager.authenticate(
        new UsernamePasswordAuthenticationToken(loginRequest.getUsernameOrEmail(), loginRequest.getPassword()));

    SecurityContextHolder.getContext().setAuthentication(authentication);

    // Get the authenticated user's details
    UserPrincipal principal = (UserPrincipal) authentication.getPrincipal();

    // Find the user in the database to get the gender
    User user = userRepository.findById(principal.getId())
        .orElseThrow(() -> new UsernameNotFoundException("User not found with id : " + principal.getId()));

    // Get the user's gender
    String gender = user.getGender().name();

    // Generate the JWT
    String jwt = tokenProvider.generateToken(authentication);

    // Create a new response with the JWT and the gender
    JwtAuthenticationResponse response = new JwtAuthenticationResponse(jwt);
    response.setGender(gender);

    return ResponseEntity.ok(response);
}

```

회원가입 기능 구현: Spring Security 를 이용하여 MySql 에 User(Username, User ID, User Email, Password Token, Gender)의 정보를 담은 DB 생성하였습니다. 이미 회원정보를 가지고 있는 경우에는 에러메시지를 송출하여 재가입을 막도록 하였습니다.

```
@PostMapping("/signup")
public ResponseEntity<> registerUser(@Valid @RequestBody SignUpRequest signUpRequest) {
    if (userRepository.existsByUsername(signUpRequest.getUsername())) {
        return new ResponseEntity(new ApiResponse( success: false, message: "Username is already taken!"), HttpStatus.BAD_REQUEST);
    }

    if (userRepository.existsByEmail(signUpRequest.getEmail())) {
        return new ResponseEntity(new ApiResponse( success: false, message: "Email Address already in use!"), HttpStatus.BAD_REQUEST);
    }

    // Creating user's account
    User user = new User(signUpRequest.getUsername(), signUpRequest.getEmail(), signUpRequest.getPassword(), signUpRequest.getGender());

    user.setPassword(passwordEncoder.encode(user.getPassword()));

    User result = userRepository.save(user);

    URI location = ServletUriComponentsBuilder.fromCurrentContextPath().path("/api/users/{username}")
        .buildAndExpand(result.getUsername()).toUri();

    return ResponseEntity.created(location).body(new ApiResponse( success: true, message: "User registered successfully"));
}
```

- 크롤링: 무신사 홈페이지에 대한 크롬 드라이버와 selenium 을 이용한 동적 크롤링 수행 후 mySql 을 통해 DB 에 저장해주었습니다. 이때 scrollpage.html 에서 무신사링크로 이동하기 위해 각 아이템의 링크를 저장하였으며, chatGPT 의 답변 속 아이템들에 맞는 상품을 추출하기 위하여, 상품을 주로 이용하는 회원의 성별, 상품명, 상품에 대한 해시태그를 저장하였습니다. 또한, select.html 에서 수집된 가격정보를 기반으로 상품을 추출하기 위하여 상품의 가격정보도 함께 크롤링 하였습니다.

```
public static void doCrawlingByCategoryID(Integer categoryID, Integer productCnt, ItemRepository itemRepository) throws InterruptedException {
    WebDriver driver = setDriver.getChromeDriver();
    String url = "https://www.musinsa.com/app/";
```

- 데이터베이스 관리:

→ 무신사에서 크롤링을 통해 수집한 아이템 관리


category	item_id	gender	img_link	item_type	name	price	style
4	404474	0	https://image.msscdn.net...	긴소매 티셔츠	Longsleeve 무지 긴팔티	29900	#긴팔티#레이어드#티셔츠...
4	659554	0	https://image.msscdn.net...	니트/스웨터	[세트] 하프 폴라 니트 티...	35100	#오크릭#목폴라#반폴라#...
4	1031260	0	https://image.msscdn.net...	피케/카라 티셔츠	235/S 오버핏 피케티셔...	39900	#반팔pk#티셔츠#카라티셔...
4	1557508	2	https://image.msscdn.net...	맨투맨/스웨트셔츠	배츠 어센틱 맨투맨 그레이	27600	#크루넥#맨투맨#스웨트셔...
4	1704897	1	https://image.msscdn.net...	기타 상의	우먼즈 코튼 브라 탑 [블랙]	20290	#브라탑#스포츠브라#무신...
4	1882578	0	https://image.msscdn.net...	피케/카라 티셔츠	스티브 반팔 카라티 / WH...	35910	#반팔티#피케티#카라티#...
4	2038497	2	https://image.msscdn.net...	맨투맨/스웨트셔츠	[SS Ver.]해비 코튼 오버...	36000	#오버핏#락비티#긴팔티셔...

→ ChatGPT API 로부터 받은 답변의 파싱을 통해 얻은 키워드를 기반으로 아이템관리 DB 로부터 조건에 맞는 아이템 추출: chatGPT 답변을 parsing 하여, 키워드를 추출하고 해당 키워드를 해시태그로 가지고 있는 아이템들을 선택할 수 있게끔 해주었습니다. 이외에도, gender.html 에서 저장된 gender 정보와 select.html 에서 저장된 가격정보를 filter 로 이용하여 다시 한번 상품을 선택하는 작업을 수행하였습니다.



```
public class ItemController {  
    5 usages  
    private final ItemService itemService;  
  
    1 usage  ⚡ ejeong0303  
    @Autowired  
    public ItemController(ItemService itemService) { this.itemService = itemService; }  
  
    1 usage  ⚡ HongYeSun +1  
    @GetMapping  
    public ResponseEntity<LinkedHashMap<String, Object>> getItemsByFilter(@RequestParam String chatGPTResponse,  
                                                                           @RequestParam(required = false) Integer minPrice,  
                                                                           @RequestParam(required = false) Integer maxPrice,  
                                                                           @RequestParam String gender) throws IOException {  
  
        List<Item> items;  
        Integer gen;  
        LinkedHashMap<String, Object> result = new LinkedHashMap<>();  
        switch(gender) {  
            case "male" :  
                gen = MALE;  
                break;  
            case "female" :  
                gen = FEMALE;  
                break;  
            case "unisex" :  
                gen = UNISEX;  
                break;  
            default : gen = UNISEX;  
        }  
    }  
  
    if (minPrice != null && maxPrice != null) {  
        items = itemService.getItemsByChatGPTResponseAndPriceRange(chatGPTResponse, minPrice, maxPrice, gen);  
    } else if (minPrice != null && maxPrice == null) {  
        items = itemService.getItemsByChatGPTResponseAndPriceRange(chatGPTResponse, minPrice, maxPrice: 10000000, gen);  
    } else if (minPrice == null && maxPrice != null) {  
        items = itemService.getItemsByChatGPTResponseAndPriceRange(chatGPTResponse, minPrice: -1, maxPrice, gen);  
    } else {  
        items = itemService.getItemsByChatGPTResponse(chatGPTResponse, gen);  
    }  
    result.put("data", items);  
    result.put("count", items.size());  
  
    return ResponseEntity.ok().body(result);  
}
```


3. 프로젝트 산출물



시작 페이지

 <input type="button" value="Login"/> Don't have an account? <input type="button" value="Sign in"/>	유저가 웹을 열게 되면 마주치게 되는 첫 화면으로, 날씨를 이용한 앱이라는 것을 알아볼 수 있도록 제작한 Weather Wears You 로고와 로그인 화면으로 이어지는 파란 버튼, 회원 가입 화면으로 이어지는 노란 버튼으로 이루어져 있다.
---	--


로그인/회원가입 페이지

 ID soj0227@naver.com Password ***** <input type="button" value="Login"/>	 name soralee@test.com ID password ***** prefer <input type="button" value="M"/> <input type="button" value="F"/> <input type="button" value="Unisex"/> <input type="button" value="Sign in"/>	아이디와 비밀번호, 그리고 추가적으로 회원 가입 시에는 서비스에서 사용할 이름을 적을 수 있는 로그인/회원가입 화면이다. 아이디는 이메일 형식이어야 하며 password에는 한글을 사용할 수 없다. 회원 관리를 위해 MySQL DB를 사용하였다. 회원가입시 저장된 회원의 성별정보는 해당 회원이 chatGPT로부터 의상 추천을 받을 때에 활용된다.
---	--	--

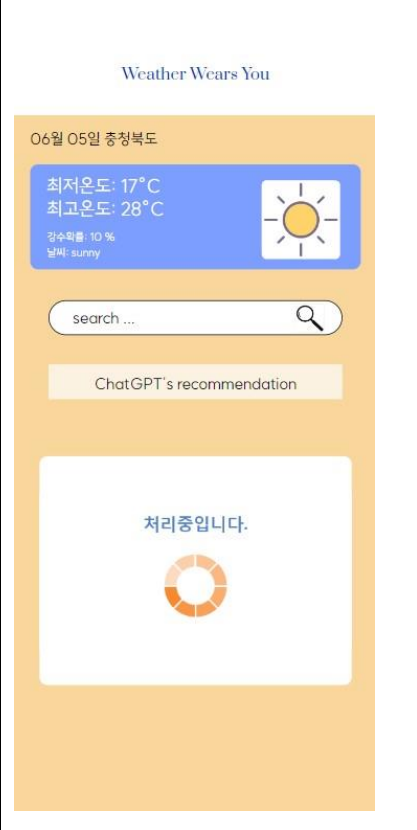
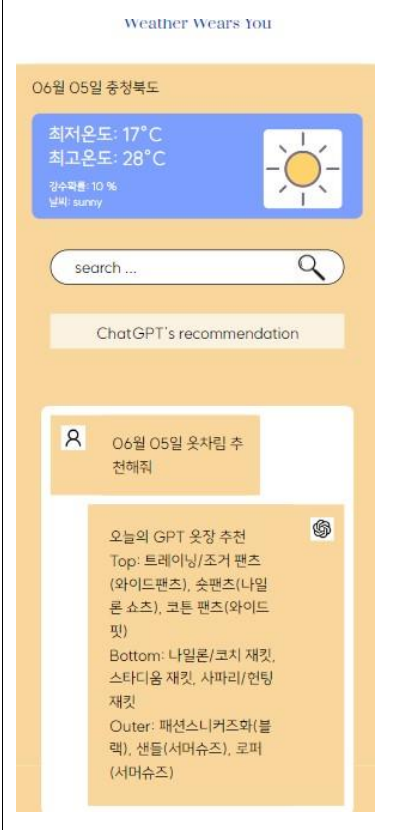
지역 선택 페이지

<p>Weather Wears You</p> <p>지역을 선택하세요</p>  <p>select</p> <p>저번에 선택한 지역은 강원도입니다. 다시 선택하시겠습니까?</p> <p>continue</p>	<p>Weather Wears You</p> <p>지역을 선택하세요</p>  <p>select</p> <p>저번에 선택한 지역은 강원도입니다. 다시 선택하시겠습니까?</p> <p>continue</p>	<p>옷을 입고 갈 지역을 선택하는 화면이다.</p> <p>먼저, 내가 이전에 선택한 지역 정보가 존재한다면 좌측 그림 하단과 같이 이전에 선택한 지역 정보를 그대로 이용할 수 있도록 하는 서비스를 고안하였다.</p> <p>지도는 svg 형태로 그렸으며 d3.js를 활용하였다.</p> <p>만약 충청북도를 클릭할 경우 "충청북도를 선택하시겠습니까?" 라는 알림창이 뜨게 되며 선택할 경우 해당 지역이 색칠되고 정보가 저장되게 된다.</p>
---	---	--

날짜 선택 페이지

<p>Weather wears you</p> <p>날짜를 선택하세요</p> <p>2023-06-일</p> <p>Get Weather</p> <p>날씨 정보</p> <p>Next Page</p>	<p>Weather wears you</p> <p>날짜를 선택하세요</p> <p>2023-06-05</p> <p>Get Weather</p> <p>날씨 정보</p>  <p>최저 기온: 17 °C 최고 기온: 28 °C 강수 확률: 10 % 날씨: 맑음</p> <p>Next Page</p>	<p>기상청 api를 활용해 앞서 선택한 지역, 그리고 이 페이지에서 선택한 날짜 모두 고려한 날씨 정보를 알 수 있는 페이지이다.</p> <p>내가 선택한 날씨 정보를 한눈에 볼 수 있으며 만약 생각해둔 날짜에 날씨가 마음에 들지 않는 경우 다른 날짜를 선택해 날씨를 확인해보며 일정을 수정할 수 있도록 고안한 화면이다.</p>
---	--	--

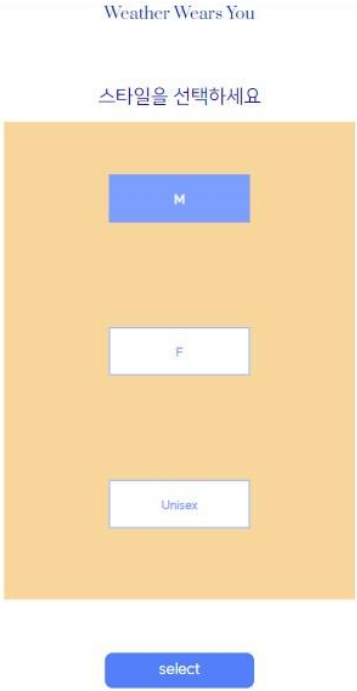
chat GPT 기반 옷 추천 페이지

		<p>chat GPT가 앞서 기상청에서 받아온 정보에 기반을 두어 옷차림을 추천해주는 화면이다.</p> <p>상단엔 간단한 날씨 정보가 적혀있으며 그 밑 검색창을 클릭하면 다음 페이지로 넘어가게 된다.</p> <p>gpt로부터 답변을 받아오는데 시간이 걸리기 때문에 답변이 뜨기 전까지 좌측 화면처럼 로딩 중이라는 것을 알 수 있도록 설계하였다.</p> <p>gpt는 상, 하의, 신발, 아우터를 추천해주며 최저 기온이 10도 이상일 경우는 아우터를 추천하지 않는다.</p>
--	---	---


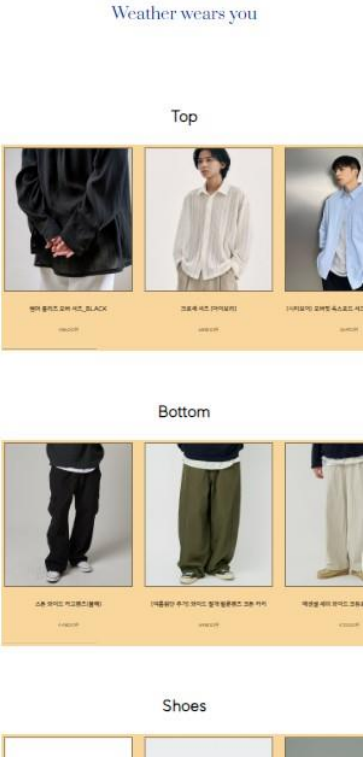
스타일 선택 페이지

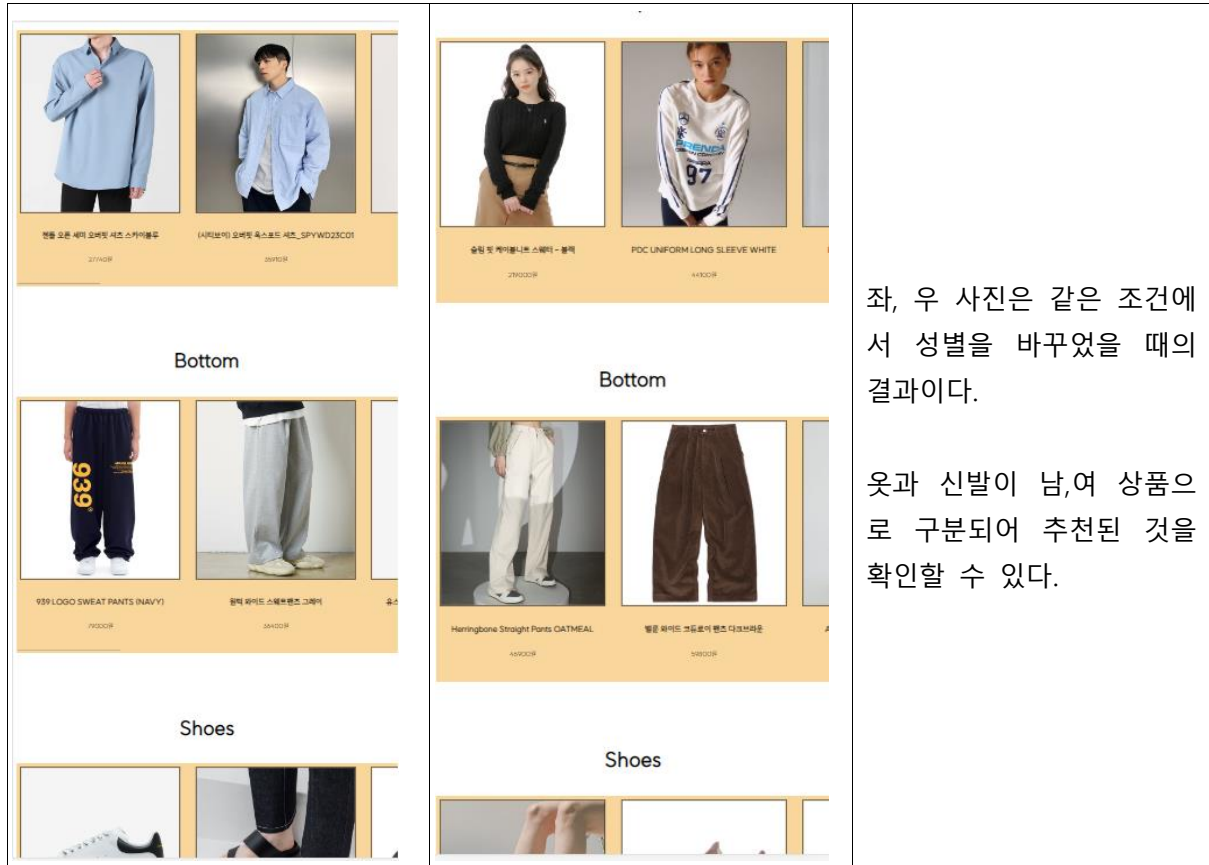
		<p>내가 입고 나가고 싶은 스타일을 선택하는 페이지이다.</p> <p>좌측 화면은 크게 8가지로 분류된 카테고리들을 보여주며 각 카테고리를 클릭하면 2~5개 정도의 세부 카테고리를 보여주게 된다.</p> <p>카테고리를 선택하고, 가격 정보를 선택한 다음 다음 페이지로 넘어가게 되는데, 이 정보들이 필요가 없다면 건너뛰기를 선택하면 된다.</p> <p>가격정보를 쓰지 않을 경우 최소 0원, 최대 1000만원으로 선택된다.</p>
---	--	--

성별 선택 페이지

	<p>내 성별이나 옷을 선물하고 싶은 사람의 성별, 혹은 스타일을 추가적으로 선택하는 화면이다.</p> <p>남성, 여성, 유니섹스로 구분을 하였다. 내가 여자더라도 남자 옷이나 유니섹스 옷을 선택할 수 있어 패션 선택의 폭을 넓히고자 고안한 페이지이다.</p>
--	--

상품 상세 추천 페이지

		<p>무신사로부터 크롤링해온 상품 정보를 추천하는 페이지이다.</p> <p>상품 정렬까지 시간이 조금 소요되므로 로딩 중이라는 페이지를 추가 구현하였다.</p> <p>chat gpt의 답변과 내 스타일을 모두 고려한 상품이 정렬 완료된다면 상의, 하의, 신발, 아우터 순으로 5~10개 상품을 보여주게 된다.</p> <p>여기서 상품 이미지를 클릭하게 되면 무신사 상세 페이지로 바로 넘어가게 된다.</p>
---	---	---



4. 멘토 평가의견 (근거자료 첨부)

"모두 처음 하시는 프로젝트라 걱정이 많았는데요. 학기 중 바쁜 와중에 완성된 결과물까지 만들어내셔서 정말 고생이 많았다고 생각합니다. 특히, 처음부터 계획했던 크롤링을 통한 상품 등록, chatGPT 를 이용한 추천시스템, 사용하기 간편한 UI 모두 잘 구현되어 높게 평가합니다."

정도한	Q, R, C, D, E				
2					
	A	B	C	D	E
31	정도한:				
32	"모두 처음 하시는 프로젝트라 걱정이 많았는데요. 학기 중 바쁜 와중에 완				
33	성된 결과물까지 만들어내셔서 정말				
34	고생이 많았다고 생각합니다. 특히, 처				
35	음부터 계획했던 크롤링을 통한 상품				
36	등록, chatGPT를 이용한 추천시스템,				
37	사용하기 간편한 UI 모두 잘 구현되어				
38	높게 평가합니다."				
39	제가 여행중이라 밖이라서 간단하게				
40	만 적었습니다~				
41	오후 2:56				
42	네 정말 감사합니다!				
43	오후 2:56				
44					
45	정도한:				
46	네 고생 많았어요~				
47	오후 2:57				

5. 검증

1. 검증결과: (성능, 품질 검증) 시험방법, 시험결과치

sbs 뉴스에서 제공된 날씨와 기온별 옷차림을 참고하여 해당 자료에 맞지 않는 상품들을 선별하는 작업을 수행하였습니다. 날씨는 sunny, blurry, cloudy, rainy 로, 기온 범위는 아래 sbs 뉴스에서 제공되는 정보를 참고하여 5℃ 이하, 6~9℃, 10~11℃, 12~16℃, 17~19℃, 20~22℃, 23~26℃, 27℃ 이상일때로 나누어 개별 test 각 기온과 날씨 별로 20 번씩 수행하였습니다.

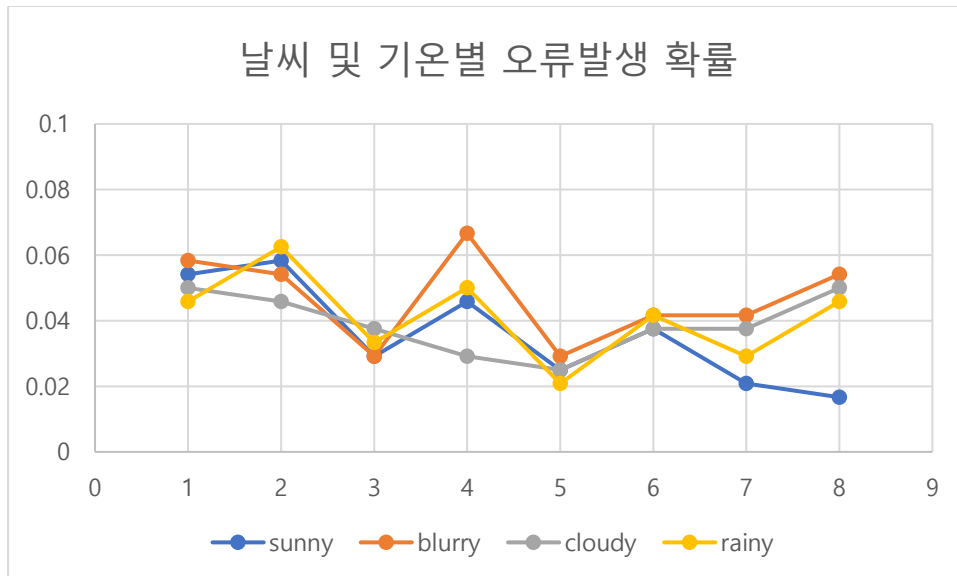


날씨와 기온을 변경하며 개별 test 를 20 번씩 수행한 결과, ChatGPT 로 부터 top, bottom, outer, shoes 에 맞는 아이템을 3 개씩 추천 받을 수 있었고, 총 각각 240 개의 아이템에 대한 정보를 얻을 수 있었습니다. 날씨에 맞지 않는 상품의 개수를 항목별 총 아이템의 개수인 240 으로 나누어 오류발생 확률을 구할 수 있었습니다. 오류발생 확률을 표로 나타낸 결과는 다음과 같습니다.

(°c)	~5	6~9	10~11	12~16	17~19	20~22	23~26	27~30
sunny	0.054	0.058	0.029	0.046	0.025	0.038	0.021	0.017
blurry	0.058	0.054	0.029	0.066	0.029	0.042	0.042	0.054
cloudy	0.05	0.046	0.038	0.030	0.025	0.038	0.038	0.05
rainy	0.046	0.063	0.033	0.05	0.021	0.042	0.030	0.046

또한, 스타일 적합성을 확인하기 위하여, 특징적인 스타일 키워드들을 선정하여 무신사에서 제공해주는 코디북의 상품들과의 비교를 수행하였습니다. '걸리시-여친룩', '포멀-단정한', '시크-모노톤'을 대표 키워드로 선정하여, test 를 진행하였습니다.

2. 분석: 시험결과 분석방법, 분석결과, 자체평가의견



위의 표를 기반으로 그래프를 그려 분석한 결과, 'WeatherWearsYou'가 날씨와 기온에 따른 옷 추천에 있어서 비교적 높은 정확도를 가지고 있음을 확인할 수 있었습니다. 모든 항목에 있어서 오류발생확률이 10% 미만이라는 결과는 저희 서비스가 매우 안정적이고 효과적으로 동작할 수 있다는 것을 의미합니다.

또한, 기온과 날씨에 따른 오류 발생 확률에 큰 차이가 없다는 점은, 저희가 생성한 서비스가 다양한 환경과 조건에 잘 적용되고 있음을 보여줍니다. 이는 저희가 이용하고 있는 알고리즘이 날씨와 기온이라는 다양한 변수를 모두 잘 고려하며, 각각의 상황에 맞는 적절한 추천을 제공하고 있음을 나타냅니다.

이러한 평가를 바탕으로, 'WeatherWearsYou'는 사용자에게 실시간 날씨와 기온에 따른 적절한 옷차림을 추천해 주기 위한 좋은 성능을 가지고 있다고 할 수 있을 것 같습니다. 앞으로도 이러한 높은 성능을 유지하며, 필요한 경우 개선사항을 탐색하고 적용하여 사용자의 만족도를 더욱 높일 것입니다.

또한, 무신사 코디북의 상품들과 'WeatherWearsYou'의 상품들을 비교해본 결과 유사한 상품들이 제시되는 것을 확인할 수 있었고, 특히 같은 해시태그를 가진 상품들이 추출되는 것을 확인할 수 있었습니다. 검증 결과는 다음과 같습니다.

검증 - 스타일 적합성(거리시-여친룩)

무신사

스타일: 거리시-여친룩

Weather wears you

Top

Bottom

Shoes

검증 - 스타일 적합성(포멀-단정한)

무신사

스타일: 포멀-단정한

Weather wears you

Top

Bottom

Shoes

검증 - 스타일 적합성(시크-모노톤)

무신사

스타일: 시크-모노톤

Weather wears you

Top

Bottom

Shoes

6. 개발내용

- 이해결 이슈, 보완사항, 향후계획 등

'WeatherWearsYou' 프로젝트를 수행하면서 직면했던 가장 큰 이슈는 원하는 답변 형식을 유지하고 정확도가 높은 답변을 얻을 수 있게끔 chatGPT 에 전송하는 질의를 수정하는 작업이었습니다.

초기 모델은 막연히 날씨정보, 성별, 그리고 스타일 키워드를 포함한 질의를 영어로 하여 영어로 답변을 얻었다면, 수정을 거친 모델에서는 질의에 기존 정보는 모두 포함하되, 답변으로 제공할 수 있는 아이템들에 선택 범위를 포함시켰습니다. 기존의 방법은 영어로 되어 있고 범위가 광범위하여, 번역 문제 및 쇼핑몰에서 제공하는 아이템들을 효과적으로 필터링 할 수 없다는 한계를 가지고 있었습니다. 또한 매번 일관적인 답변을 제공해주지 못한다는 문제가 있었습니다. 따라서, 질의에 ChatGPT 가 제공해 줄 답변 형태를 포함시켜 매번 같은 형태의 답변을 받을 수 있도록 만들어주었습니다. 또한, 데이터를 제공받을 인터넷 쇼핑몰(무신사)의 카테고리 분류기준과 세부스타일을 참고하여 해당 기준들 내에서만 답변을 할 수 있도록 리스트를 만들어 질의에 포함시켰습니다. 그 결과, 정해진 범위 내에서 일정한 형식을 가진 한국어 답변을 받을 수 있었습니다.

다만, 많은 노력을 기울였음에도 불구하고 GPT3 는 저희가 생성한 알고리즘이 아니기 때문에, 답변을 예측하기가 어려웠고 왜 그러한 답변을 제시해 주었는지 또한 파악하기가 어려웠습니다. 하여 답변 형식을 일정하게 유지하는 부분에 있어서는 오류를 모두 해결하였지만, 답변의 정확도에 있어서는 문제를 완전히 해결하기가 어려웠습니다. GPT3 는 성능이 매우 뛰어난 AI 이기 때문에, 오류가 발생하는 확률을 계산해보았을 때 모든 testcase 에 있어서 10% 미만으로 나타났지만, 더 오류를 줄이지 못했다는 아쉬움이 남습니다. 마지막까지 질의 형태를 바꾸고, 한국어로 전송되었던 기존의 아이템 키워드들을 영어로 mapping 시켜 번역 시켜주는 함수를 만들어 질의를 전송하기도 해보았지만, 오류발생확률이 감소하지 않고 지속적으로 유지된다는 것을 확인하고, 이 부분을 저희 서비스의 한계로 인정할 수 밖에 없었습니다.

따라서, 다음번에도 ChatGPT 를 이용한 서비스를 개발한다면, 답변의 형태가 유동적이어도 무관하고 AI 가 주어진 조건에 선택을 하는 질의를 전송하기 보다는 직접 창의적인 답변을 생성할 수 있는 질의를 보내는 방식으로 ChatGPT 를 이용해보고 싶다는 생각을 하게 되었습니다. 'WeatherWearsYou' 역시, 그러한 방향으로 개선한다면, 정확도 부분에서 보완이 된 서비스로 거듭날 수 있을 것 같습니다.

'WeatherWearsYou'는 단독으로 사용되기 보다는 기존의 쇼핑몰들의 부가서비스를 이용된다면 더 의미가 있을 것이라고 생각합니다. 인터넷 쇼핑몰들이 가진 방대한 상품 DB를 활용하며, 기존 회원들이 사용할 수 있도록 한다면 더 다양하고 정확한 추천을 제공해 줄 수 있을 것이라고 생각합니다. 따라서, 기회가 된다면 UI/UX와 답변 정확도를 보완하여, 저희 서비스를 다른 쇼핑몰의 부가서비스로 배포해보고 싶습니다. 그리고 단독으로 서비스를 배포하게 된다면, 사용자 리뷰 및 평점 정보를 제공하는 기준을 추가하여 모바일 앱으로 발전시켜보고자 합니다.