**종합설계PBL 중간 발표 보고서**

프로젝트명: 트북이

(AI기반 노트북 추천 및 맞춤형 노트북 쇼핑몰)

팀명: 브레인스토밍

|  |  |  |
| --- | --- | --- |
| 전공 | 학번 | 이름 |
| 컴퓨터과학과 | 2018010873 | 전병규 |
| 컴퓨터과학과 | 2018010861 | 양소열 |
| 컴퓨터과학과 | 2018010877 | 정윤수 |
| 정보통계학과 | 2018010820 | 이지상 |

**목 차**

**1. 프로젝트 개요**

1.1 프로젝트 배경 및 필요성

1.2 프로젝트 목표와 주요 내용

**2. 시스템 구성**

2.1 프로젝트 구현 환경

2.1.1 결제 API

2.2 기능 구성

2.3 전체적인 UI 구성

**3. 프로젝트 설계 대비 구현 상황**

3.1 프로젝트의 업무 분장 및 수행 일정

3.1.1 업무 분장

3.1.2 수행 일정

3.2 Front-End 구현 상황

3.3 Back-End 구현 상황

**4. 테스트 계획**

**종합설계PBL 중간 발표 보고서**

**브레인스토밍팀(트북이(AI 기반 노트북 추천 및 맞춤형 노트북 쇼핑몰))**

**1. 프로젝트 개요**

**1.1 프로젝트 배경 및 필요성**

* 2021년 기준으로 전 세계 노트북 시장 규모는 약 2,000억 달러로 추산된다.
* 노트북 시장은 지속해서 성장하고 있고, COVID-19 팬데믹으로 인해 원격 근무 및 온라인 수업 등이 증가하면서 노트북 수요가 급증했다.
* 기술의 발전으로 노트북의 성능도 상향 평준화가 되어, 사용자가 선택할 수 있는 노트북의 폭이 넓다.
* 노트북은 한국인이 가장 많이 구매한 전자제품 1위로 꼽히는 제품으로 수요가 많다.
* 노트북을 구매할 때, 제품의 성능, 디자인, 가격, 브랜드 등 다수의 고려해야 할 요소가 존재한다.
* 대부분의 사람은 이러한 요소를 고려하지 않고, 지인과 직원의 소개, 여러 커뮤니티와 유튜브에 의존하며 현명하게 소비할 권리를 보장받지 못하고 있다.
* 메이저 3사의 100만원 상당의 노트북, 기타 해외업체의 50만원 상당의 노트북의 사양은 상이하지만, 소비자들은 메이저 3사의 정보만 알고 있어 편향된 소비를 보이는 경향이 있다.

**1.2 프로젝트의 목표와 주요 내용**

* 사용자의 상황과 성향에 맞는 맞춤형 노트북을 AI를 통해 추천해주고, 사용자의 활동을 분석하며, 미래 쇼핑 트렌드나 선호도를 예측하는 마케팅 전략을 보유하여 시장에서의 차별점을 두려 한다.
* 사용자가 선호하는 노트북에 대한 정보를 수집한다.
* 사용자가 선호하는 노트북에 대한 데이터를 기반으로 개인에게 맞는 최적의 노트북을 추천한다.
* AI로 추천된 노트북뿐 아니라 이외로 자신이 사고 싶은 노트북을 직접 쇼핑몰에서 검색하고 결제가 가능하다.
* 상품의 정보를 AI 기술을 활용하여 자동으로 수집하고 분석하기 때문에 사용자가 노트북에 대한 정보를 추가로 조사할 필요가 없음. 그리고 최신 노트북 정보와 가격 변동에 대한 업데이트가 자동으로 이루어지기 때문에 소비자 입장에서 신뢰할 수 있는 정보가 제공된다.

**2. 시스템 구성**

**2.1 프로젝트 구현 환경**

|  |  |  |
| --- | --- | --- |
| SW | 개발 언어 | HTML, CSS, Javascript, Java, Python |
| 편집기 | Visual studio Code, IntelliJ IDEA |
| DBMS | MySQL |
| 서버 | Docker |
| 디자인 | Figma |
| 협업 | GitHub, 구글 문서, Notion |
| Front-End | HTML, CSS, Javascript, React |
| Back-End | Spring Boot, Hibernate, FastAPI |
| DevOps | Docker, Github Action |
| AI | Kaggle, Tensorflow, Pytorch |
| 결제 API | Iamport API |
| 구동 환경 | OS | MacOS |

**2.1.1 결제 API**

|  |  |
| --- | --- |
| Iamport API | 신용카드, 계좌이체, 가상계좌 등 다양한 결제 수단을 지원하는 API |

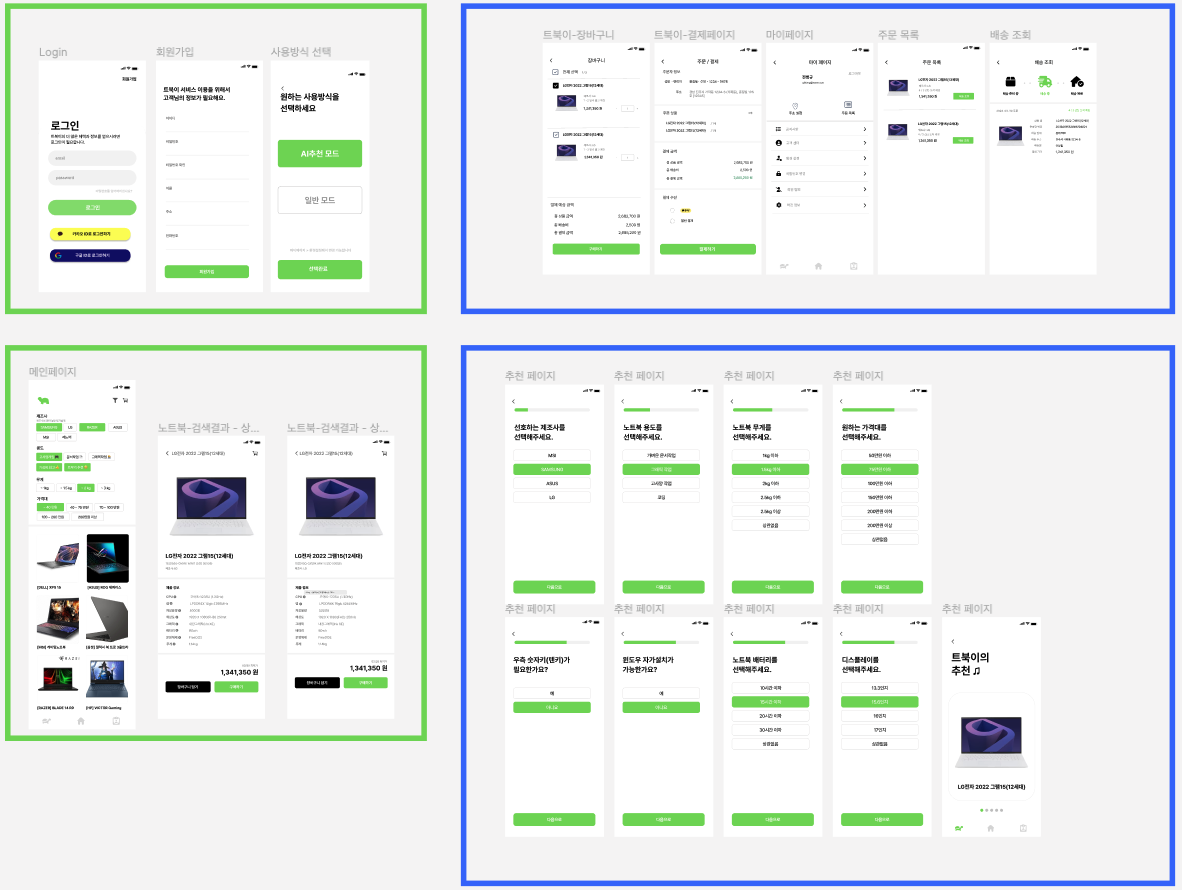
**2.2 기능 요구사항**

|  |  |  |
| --- | --- | --- |
| 로그인 기능 | e-mail과 비밀번호를 입력하여 로그인하는 기능  세션 로그인도 구현하여 사용자가 원하는 로그인을 할 수 있도록 함. | |
| 입력 | e-mail, 비밀번호 |
| 출력 | 로그인 |
| 사용방식 선택 기능 | 사용자가 원하는 사용방식을 선택할 수 있는 기능 | |
| 입력 | AI 추천 모드, 일반 모드 |
| 출력 | AI 추천 모드와 일반모드 중 사용자가 선택한 방식 |
| 필터 기능 | 원하는 필터를 설정하여 필터에 맞는 노트북을 제공 | |
| 입력 | 제조사, 용도, 무게, 가격대 등 원하는 필터 선택 |
| 출력 | 메인화면에 필터에 맞는 노트북 출력 |
| 노트북 상세정보  조회 기능 | 앱에서 제공하는 노트북에 대한 정보를 제공해주는 기능. | |
| 입력 | 노트북 이미지 클릭 |
| 출력 | 해당 노트북의 상세정보 제공 |
| 장바구니 기능 | 사용자가 구매를 희망하는 노트북을 장바구니에 담을 수 있는 기능 | |
| 입력 | 노트북 상세정보 페이지 내 장바구니 담기 버튼 클릭 |
| 출력 | 장바구니 리스트에 추가 |
| 주문 기능 | 사용자가 원하는 노트북을 결제할 수 있도록 주문하는 기능 | |
| 입력 | 장바구니 페이지 내 구매하기 버튼, 상세정보 페이지 내 구매하기 버튼 |
| 출력 | 주문/결제 페이지로 이동 |
| 결제 기능 | 사용자가 원하는 노트북을 결제할 수 있도록 하는 기능 | |
| 입력 | 주문/결제 페이지 내 결제하기 버튼 |
| 출력 | 결제 페이지로 이동 |

**2.2 비기능 요구사항**

|  |  |  |
| --- | --- | --- |
| 성능 요구사항 | 응답시간 | 시스템은 사용자의 요청에 대해 2초 이내에 응답해야 한다. |
| 처리량 | 시스템은 초당 100개의 요청을 처리할 수 있어야 한다. |
| 확장성 | 시스템은 수평 및 수직 확장이 가능해야 하고, 트래픽 증가에 따라 선형적으로 확장될 수 있어야 한다. |
| 메모리  사용량 | 시스템은 효율적으로 메모리를 사용해야 한다. 메모리 누수나 비효율적인 자원 사용을 최소화해야 한다. |
| 로깅 및  모니터링 | 시스템은 로그를 적절히 기록하고 모니터링 해야한다. 성능 지표와 예외를 모니터링하여 시스템 성능을 지속적으로 개선할 수 있어야 한다. |
| 보안 요구사항 | 인증,  권한부여 | 시스템은 사용자를 인증하고 권한을 부여하여 민감한 데이터와 기능에 접근할 수 있는지 여부를 제어해야 한다. |
| 데이터  보호 | 시스템은 데이터의 기밀성, 무결성 및 가용성을 보호해야 한다. 적절한 데이터 암호화 및 액세스 제어 메커니즘을 사용해야 한다. |
| 가용성 요구사항 | 시스템  운영시간 | 시스템은 99.9%의 가용성을 제공해야 한다. 예정된 유지버스 시간을 제외한 경우에도 사용자가 접근할 수 있어야 한다. |
| 장애복구 | 시스템은 장애가 발생한 경우에도 빠른 복구를 제공해야 한다. 데이터 백업 및 복원, 장애 허용적 아키텍처를 고려해야 한다. |
| 유지보수성  요구사항 | 코드  가독성 | 시스템은 읽기 쉽고 이해하기 쉬운 코드로 작성해야 한다. 적절한 주석과 문서화를 통해 코드를 설명해야 한다. |
| 변경  용이성 | 시스템은 새로운 기능 추가나 기존 기능 수정에 대한 변경이 용이해야 한다. 높은 결합도와 낮은 의존성을 가진 모듈화된 아키텍처를 고려해야 한다. |
| 확장적 요구사항 | 수평확장 | 시스템은 증가하는 트래픽에 대해 서버를 수평으로 확장할 수 있어야 한다. 로드밸런싱, 분산 캐싱등을 활용해서 확장성을 개선해야 한다. |
| DB확장성 | 환시스템은 대량의 데이터를 처리하기 위해 데이터베이스를 확장할 수 있어야 한다. 샤딩, 파티셔닝등의 기술을 고려해야 한다. |
| 호환성 요구사항 | 플랫폼  호환성 | 시스템은 다양한 운영 체제, 브라우저, 디바이스에 호환되어야 한다. 크로스 플랫폼 테스트와 호환성 테스트를 수행해야 한다. |
| API  호환성 | 시스템은 변경이나 업그레이드로 인해 기존 API를 손상시키지 않고 호환성을 유지해야한다. |
| 규정준수  요구사항 | 암호화  요구사항 | 시스템은 민감한 데이터를 암호화하여 저장하고 전송해야 한다. 암호화 알고리즘과 키 관리를 포함한 보안 메커니즘을 사용해야 한다. |
| 사생활  보호 | 시스템은 사용자의 개인정보와 사적인 데이터를 적절히 보호해야 한다. 개인정보 처리 방침, 동의 절차 및 사용자 데이터 액세스 제어등을 포함해 개인정보 보호에 대한 규정을 준수해야 한다. |

**2.3 전체적인 UI 구성**



**3. 프로젝트 설계 대비 구현 상황**

**3.1 프로젝트의 업무 분장 및 수행 일정**

**3.1.1 업무 분장**

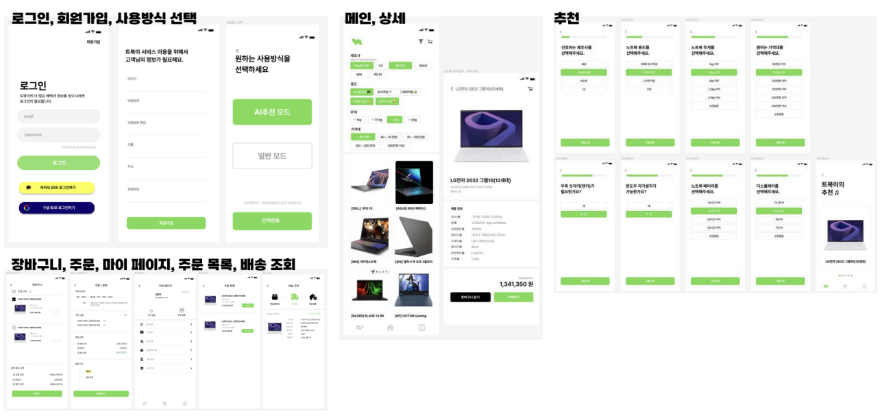
|  |  |  |
| --- | --- | --- |
| 팀 구성 | | 역할 및 책임 |
| 전 병 규 | 팀 장 | · 도메인, Product엔티티 설계  · API 명세서 작성  · Product UI구현  · Product 마이크로 서비스 API 구현  · React와 백엔드간 연동 |
| 양 소 열 | 팀 원 | · User 엔티티 설계  · Spring Security를 통한 인증과 인가 구현  · 카카오, 구글, 네이버 소셜 로그인 구현  · JWT 토큰 발급 기능 구현  · User 마이크로 서비스 API 구현 · 리액트와 백엔드간 인증과 인가 로직 연동  · FastAPI를 통한 실시간 크롤링 서버 구현 |
| 정 윤 수 | 팀 원 | · 주문 엔티티 설계  · 주문 마이크로 서비스 API 구현  · 결제 마이크로 서비스 API 구현  · React 연동  · I’mport API 연동  · Kafka 연동 |
| 이 지 상 | 팀 원 | · 필터링 화면구상  · 노트북 데이터 크롤링 및 스크래핑  · 데이터 전처리  · AI추천알고리즘 구현 및 모델 성능 평가 |

**3.1.2 수행 일정**

|  |  |  |
| --- | --- | --- |
| 종합설계PBL 프로젝트 중간 수행 일정 | | |
|  | 전병규 | 양소열 |
| 5주차  3/27~ | · IntelliJ, Visual studio code, Spring boot 개발 환경 구축  · MSA 아키텍쳐 학습  · JPA 학습 | · IntelliJ, Visual studio code, Spring boot 개발 환경 구축  · MSA 아키텍쳐 학습  · JPA 학습 |
| 6주차  4/3~ | · MSA 아키텍쳐 학습  · 전체 도메인 설계 | · Spring Security 필터와 동작원리 이해  · Spring Security 커스텀 필터 만들어보기(70%) |
| 7주차  4/10~ | · Product 엔티티 설계  · API 명세 작성  · UI 및 기능 설계 | · 기존 Basic Auth 방식의 시스템의 이해  · Session 방식의 로그인 이해 및 Session 방식 로그인 구현 |
| 8주차  4/17~ | · UI 및 기능 설계  · Product 마이크로 서비스 구현(10%) | · Session 로그인의 단점을 보완한 JWT방식 학습  · JWT 인증, 인가 필터 구현  · 회원가입 페이지 제작(React) |
| 9주차  4/24~ | · UI 개발(상품 상세 정보 페이지, 30%) | · OAuth2.0 로그인 방식 학습  · 네이버, 카카오, 구글 로그인 구현  · 리액트와 구글 로그인 연동 테스트(React) |
| 10주차  5/1~ | · UI 개발(상품 상세 정보 페이지, 100%)  · Product 마이크로 서비스 구현 30% | · RestAPI 연동을 위해 OAuth 로그인 로직 수정  · 직접 리소스 API 호출을 통한 액세스 토큰 획득 |
| 11주차  5/8~ | · UI 개발(메인 화면 페이지, 70%)  · Product 마이크로 서비스 구현 70% | · OAuth 로그인 방식에 JWT 토큰 반환 로직 작성  · JWT 토큰 발급 테스트 작성 (30%) |
| 12주차  5/15~ | · UI 개발(메인 화면 페이지, 100%)  · React 변환(10%)  · Kafka 개발 환경 구축 | · MSA 아키텍처에서 인증, 인가 처리 로직구현(10%)  · JWT 토큰 발급 테스트 작성 (100%) |

|  |  |  |
| --- | --- | --- |
| 종합설계PBL 프로젝트 중간 수행 일정 | | |
|  | 정윤수 | 이지상 |
| 5주차  3/27~ | · IntelliJ, Visual studio code, Spring boot 개발 환경 구축  · MSA 아키텍쳐 학습  · JPA 학습 | · 추천 알고리즘에 대한 기본 개념과 원리에 대해 학습  · 사용자 기반(User-based) 추천과 아이템 기반(Item-based) 추천에 대한 차이와 장단점 분석 |
| 6주차  4/3~ | · 임의 회원 마이크로 서비스 구현  · 임의 상품 마이크로 서비스 구현  · Spring Eureka, API Gateway 연결 | · Collaborative Filtering, Content-Based Filtering, Hybrid Recommender 등 추천 알고리즘의 다양한 유형 탐색. |
| 7주차  4/10~ | · UI 및 기능 설계(30%): 장바구니, 주문, 마이 페이지  · 주문 마이크로 서비스 구현(10%) | · 다나와 사이트의 구조와 데이터 패턴을 분석하기 위해 다나와 사이트의 HTML 구조 분석.  · 다나와 사이트에서 제품 목록을 가져오기 위한 URL 패턴과 페이지 구조 파악.  · 제품 이름, 가격, 상세 스펙 등 필요한 데이터  추출하기 위한 HTML 태그와 CSS 선택자 분석. |
| 8주차  4/17~ | · UI 및 기능 설계(100%): 장바구니, 주문, 마이 페이지  · 주문 마이크로 서비스 구현(30%) | · 다나와 사이트에서 필요한 데이터를 추출하기 위한 크롤러 구현.  · Python의 requests 라이브러리를 사용하여 다나와 사이트에 HTTP 요청.  · BeautifulSoup 라이브러리를 사용하여 HTML 페이지를 파싱, 필요한 데이터를 추출하는 코드 작성. |
| 9주차  4/24~ | · UI 개발(30%): 장바구니, 주문, 마이 페이지  · 주문 마이크로 서비스 구현(50%) | · 데이터의 정확성과 일관성을 유지하기 위해 데이터 정제 및 가공 작업 수행.  · 크롤링 및 스크래핑 과정에서 발생하는 에러와 예외 처리에 대한 방안 탐구. |
| 10주차  5/1~ | · UI 개발(80%) - HTML,CSS,JS  · Kafka 개발 환경 구축 - Zookeeper, Kafka Server  · 주문 마이크로 서비스 구현(70%) | · 추천 알고리즘에서 사용할 요구 데이터를 수집하기 위한 페이지 화면 구상.  · 페이지의 디자인과 레이아웃을 고려하여 UI/UX 구상(HTML, CSS, JS 기반) |
| 11주차  5/8~ | · Kafka Connect 구축  · Mysql JDBC Connect 연결  · 주문 마이크로 서비스 구현(80%)  · 결제 마이크로 서비스 구현(10%) | · 수집 가능한 데이터의 특성을 파악하고 EDA 기법으로 분석하는 작업 진행. |
| 12주차  5/15~ | · React 주문 기능 연동  · 결제 마이크로 서비스 구현(25%)  · I’mport API 연동 | · 데이터의 특성과 목적을 고려하여 최적의 추천 알고리즘을 선택하기 위해 각 알고리즘의 적합성 검토 |

**3.1 Front-End 설계 대비 구현 상황 (전체의 90% 구현)**

**3.1.1 UI 설계(100%)**

저희 웹 앱은 로그인, 회원가입, 사용방식 선택, 메인, 노트북 상세, 노트북 추천, 장바구니, 주문, 마이 페이지, 주문 목록, 배송 조회 등의 다양한 화면으로 구성되어 있습니다. 특히, 노트북 추천 기능은 주요 기능으로서 사용자가 직관적으로 원하는 노트북의 세부 사항을 선택할 수 있도록 설계되었습니다.

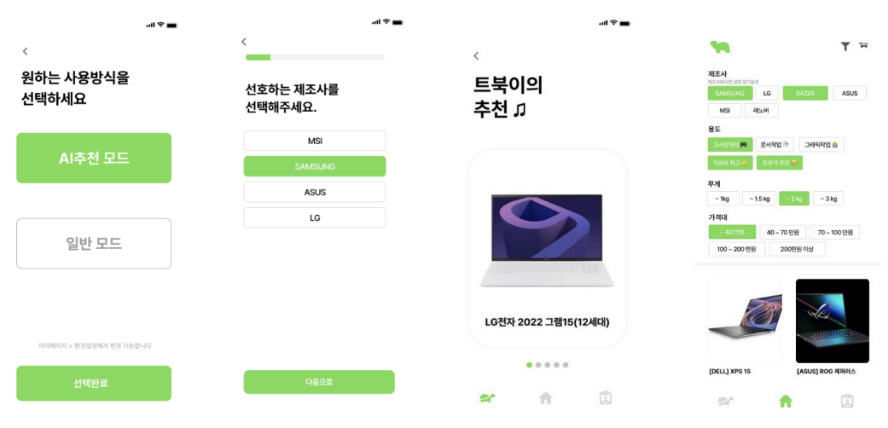
노트북 추천 화면은 슬라이드 형태로 구성되어 있어 한 번의 추천마다 다양한 노트북 선택지를 제공합니다. 각 추천마다 5개의 상품을 보여주어 사용자가 다양한 옵션을 살펴볼 수 있도록 하였습니다. 이를 통해 사용자는 편리하게 원하는 노트북을 선택할 수 있습니다.

로그인 기능은 다양한 형식을 지원하기 위해 소셜 로그인 기능도 포함되어 있습니다. 사용자는 웹에 로그인할 때마다 AI 추천 모드와 일반 모드 중 선택할 수 있습니다. 이를 통해 사용자는 자신의 취향과 선호도에 따라 AI 추천을 받을 수도 있고, 직접 노트북을 탐색할 수도 있습니다.

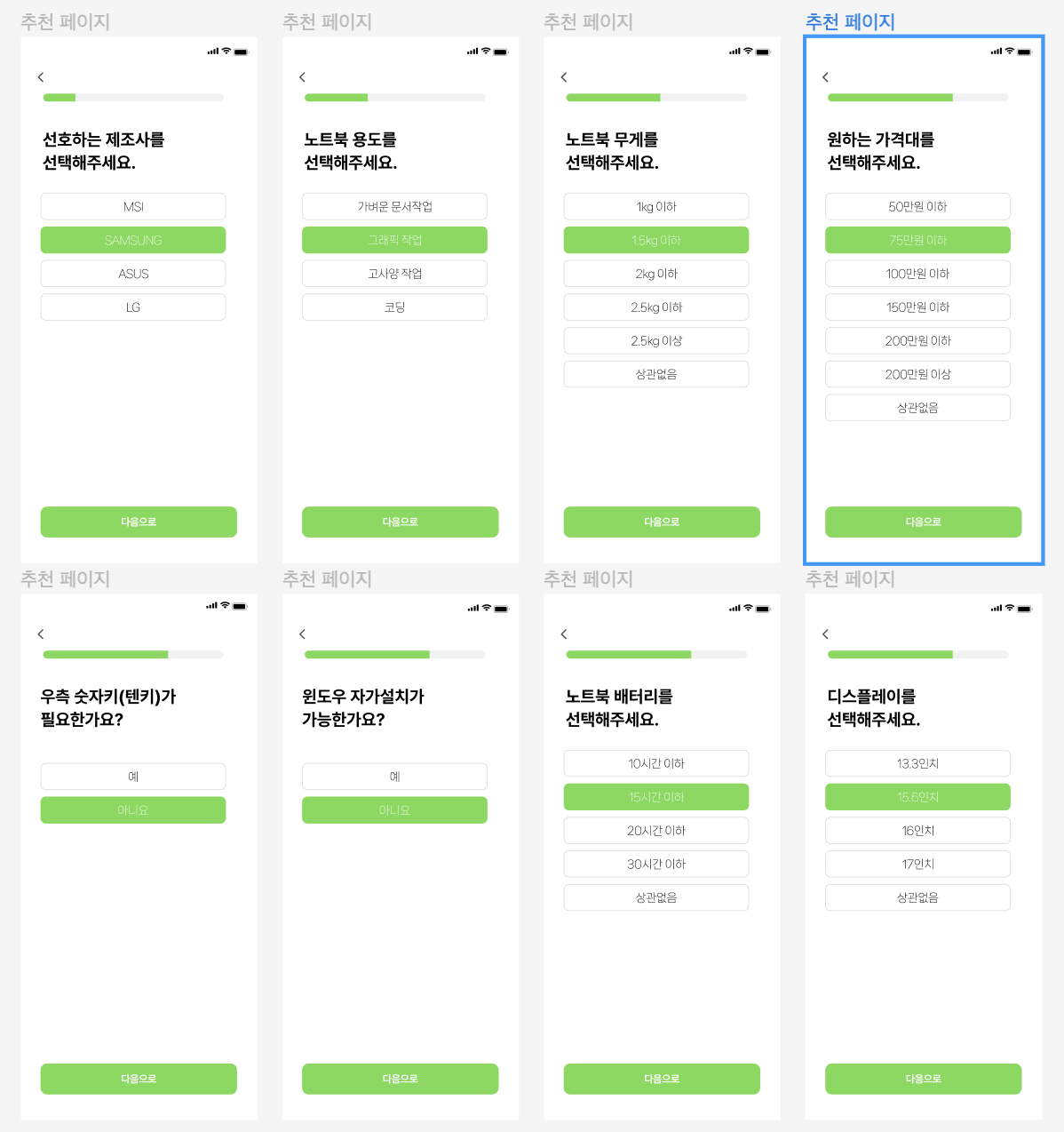
또한, 노트북 추천과 함께 사용자가 실제로 구매할 수 있도록 구매 및 결제, 그리고 배송 기능을 담당하는 화면도 설계되었습니다. 사용자는 노트북 추천을 받은 후에 원하는 상품을 선택하고 구매할 수 있습니다. 결제와 배송 과정은 간편하고 신속하게 처리되어 사용자가 원하는 노트북을 빠르게 소유할 수 있도록 합니다.

이렇게 다양한 화면과 기능을 통해 사용자는 편리하게 로그인하고 웹 앱을 이용할 수 있으며, 노트북 추천과 구매 과정을 원활하게 경험할 수 있습니다. 저희 웹 앱은 사용자에게 멋진 디자인과 직관적인 기능을 제공하여 노트북 구매와 사용에 대한 최상의 경험을 제공합니다.

**3.1.2 UI 개발(100%)**



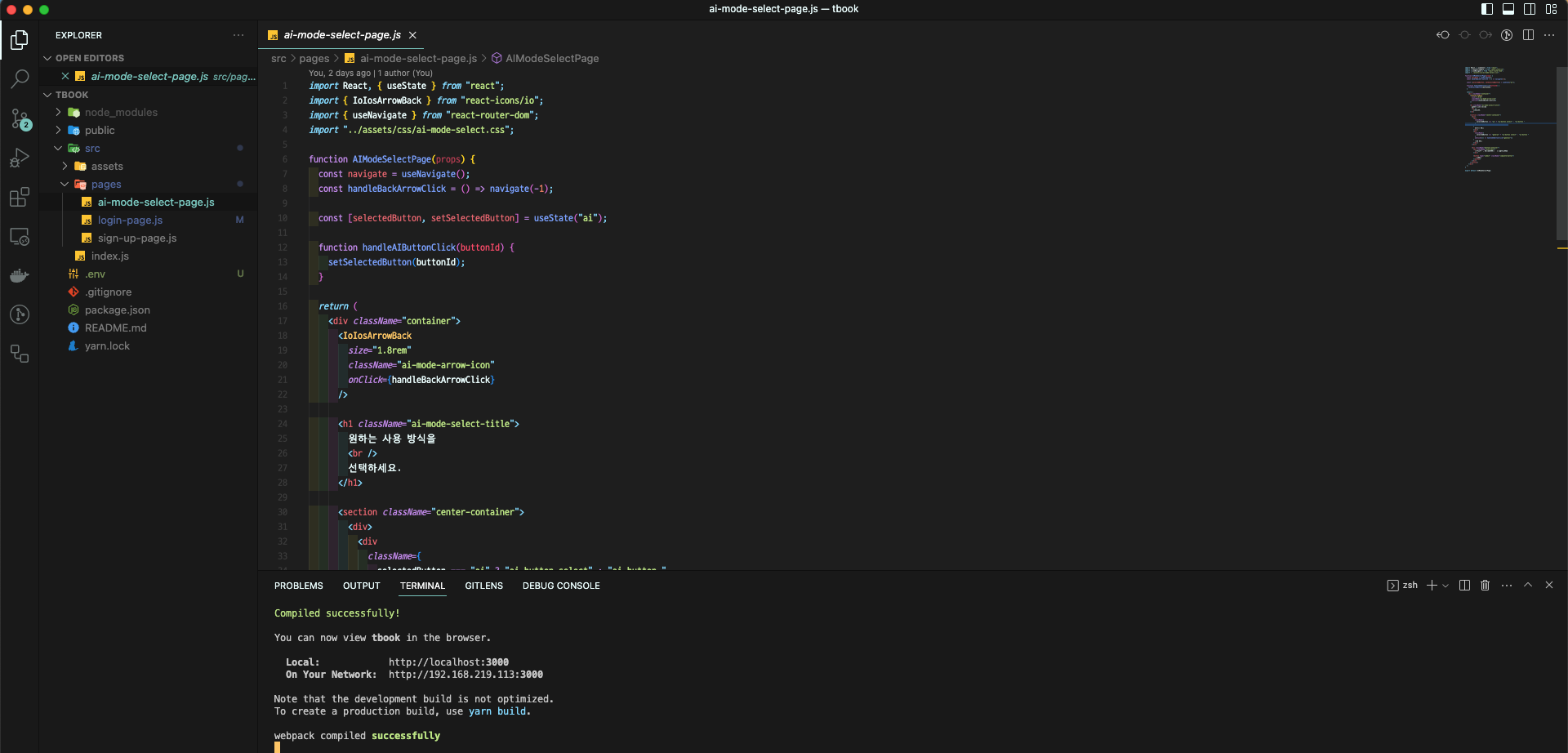
HTML, CSS, JS를 활용하여 웹 앱 형태로 개발되었습니다. 이 웹 앱은 사용자가 데스크탑과 모바일 기기에서 일관된 UI를 경험할 수 있도록 설계되었습니다. 팀원들은 효율적인 개발을 위해 각자 담당한 부분을 섹션으로 나누어 작업하였습니다. 사용자, 주문, 결제, 제품 등과 같은 각 섹션은 개별적으로 화면을 개발하였습니다.



특히, AI 필터링 화면은 사용자의 선택에 따라 진행 사항이 업데이트되는 프로그레스 바를 갖추었습니다. 이 기능을 통해 사용자가 선택을 할 때마다 진행 상황을 시각적으로 확인할 수 있습니다. 사용자가 필터를 선택하고 설정할 때마다 프로그레스 바가 업데이트되어 진행 과정을 보다 효율적으로 파악할 수 있게 되었습니다.

이렇게 웹 앱을 개발함으로써 사용자는 데스크탑과 모바일 기기에서 일관된 UI를 경험할 수 있으며, AI 필터링 기능을 사용할 때도 진행 상황을 시각적으로 확인할 수 있습니다. 멋진 디자인과 함께 사용자에게 좋은 경험을 제공하는 웹 앱이 되도록 노력하고 있습니다.

**3.2.3 React 연동(30%)**

****

기존에 HTML, CSS, JS로 개발된 페이지를 React 컴포넌트로 재구성하는 계획을 가지고 있습니다. 이러한 전환을 통해 사용자에게 더욱 부드러운 동적 페이지를 제공할 수 있으며, SPA(Single Page Application) 구조를 통해 웹 앱에 최적화된 사용자 경험을 제공할 수 있습니다.

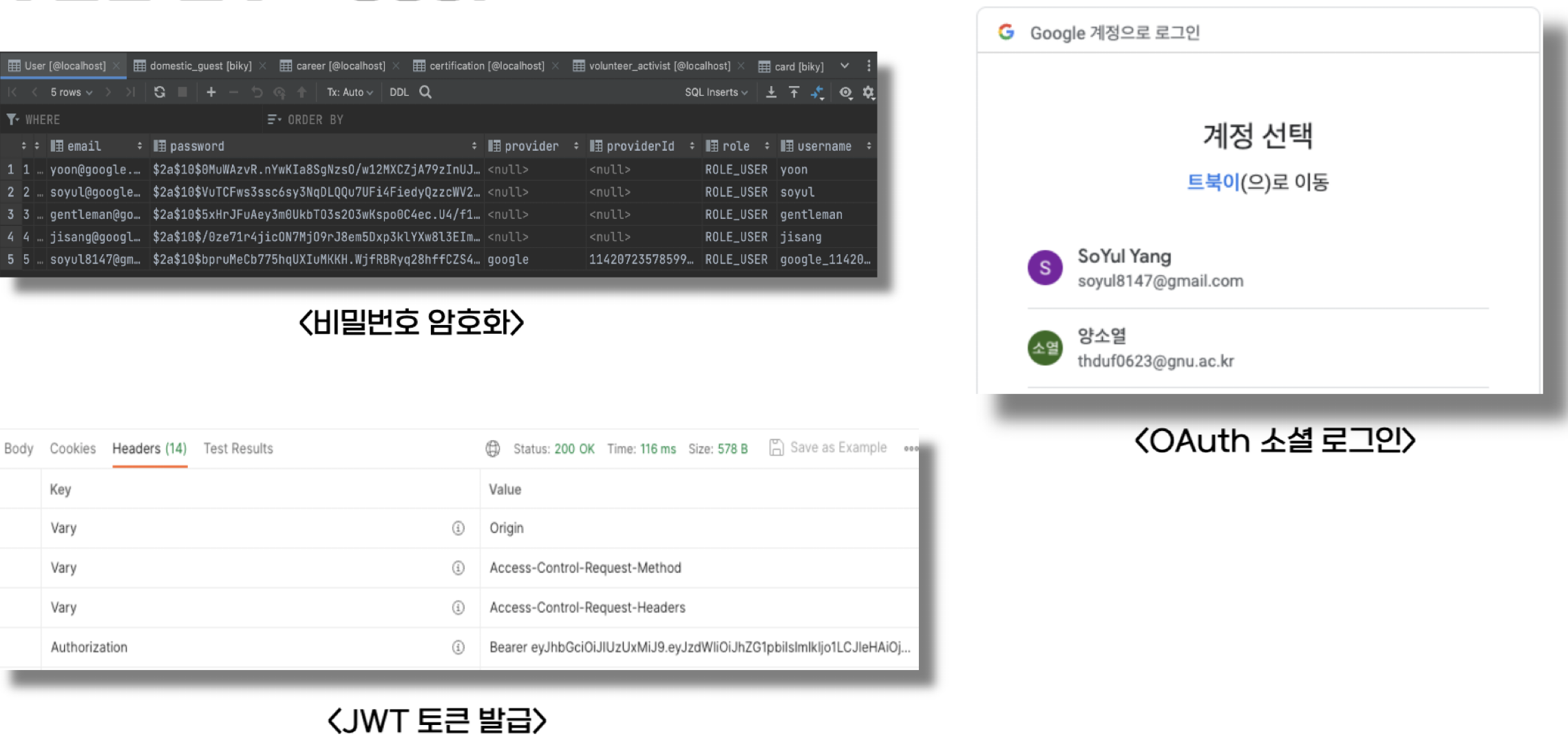
React 컴포넌트를 사용하면 각 부분을 독립적인 모듈로 분리하여 관리할 수 있습니다. 이를 통해 코드의 재사용성을 높이고 개발 과정을 효율화할 수 있습니다. 또한, React의 가상 돔(Virtual DOM)을 활용하여 효율적인 렌더링을 가능하게 하여 성능을 향상시킬 수 있습니다.

추가적으로, axios와의 연동을 통해 프론트엔드와 백엔드 간의 통신을 처리할 계획입니다. axios는 HTTP 클라이언트 라이브러리로, 비동기적인 방식으로 서버와의 데이터 교환을 처리할 수 있습니다. 이를 통해 사용자의 요청에 따라 서버와 실시간으로 데이터를 주고받을 수 있고, 웹 앱의 기능을 확장하고 향상시킬 수 있습니다.

이렇게 React 컴포넌트로의 전환과 axios와의 연동을 통해 더욱 부드럽고 동적인 페이지를 제공하며, 프론트엔드와 백엔드 간의 데이터 통신을 원활하게 처리하는 멋진 웹 앱을 구축할 계획입니다. 사용자에게 탁월한 경험과 기능을 제공하는 웹 앱을 개발하기 위해 노력하고 있습니다.

**3.2 Back-End 구현 상황 (전체의 90% 구현)\**

**3.2.1 User Service 설계(60%)**

****

**회원인증 (100% 구현):**

저희 서비스에서는 스프링 부트에서 BCPasswordEncrypt Bean을 주입받아 패스워드를 암호화하여 DB에 저장하는 기능을 구현하였습니다. 이를 통해 사용자의 패스워드가 안전하게 저장될 수 있도록 보장하고 있습니다. 또한, OAuth 2.0 라이브러리를 활용하여 구글, 카카오, 네이버 로그인 기능을 구현하였습니다. 이를 통해 사용자들은 소셜 로그인을 통해 우리 서비스를 더 쉽게 이용할 수 있습니다.

또한, Spring Security FilterChain을 이용하여 기존에 사용하던 Basic 인증을 비활성화하고 JWT 토큰을 통한 인증 방식으로 구현하였습니다. 로그인에 성공하면 JWT 토큰이 발급되고, 이를 이용하여 인증을 처리합니다. JWT 토큰은 SpringSecurityPersistContext에 저장되어 서버측에서 사용자의 로그인 정보를 안전하게 관리할 수 있도록 구성되어 있습니다.

**RestAPI 변환 (30% 구현):**

이전에 SSR(Server Side Rendering)에서 테스트를 진행한 뒤, CSR(Client Side Rendering)을 위해 REST API 제작이 필요하게 되었습니다. 현재는 OAuth 2.0 라이브러리를 사용하지 않고, 직접 RestTemplate을 이용하여 각 리소스 서버에서 데이터를 불러오는 작업을 진행 중입니다. 이 작업이 완료되면 프론트엔드와 인증 및 인가 작업을 테스트할 계획입니다.

이렇게 암호화된 패스워드 저장, OAuth 2.0 소셜 로그인, JWT 토큰을 이용한 인증, Spring Security FilterChain을 활용한 보안 기능 등을 구현함으로써 우리 서비스는 사용자들에게 안전하고 편리한 인증 및 로그인 경험을 제공합니다. 프론트엔드와의 테스트를 통해 이러한 기능들이 원활하게 동작하는지 확인할 예정입니다.

**3.2.2 Order&Payment Service 설계(60%)**



**MSA 환경 구축 (30% 구현):**

저희는 주문 기능을 구현하기 위해 MSA (마이크로서비스 아키텍처) 환경을 구축하였습니다. 이 환경은 다양한 서비스들을 개발하여 효율적이고 유연한 시스템을 구현하기 위한 토대가 되었습니다. 우리는 주 서버로서 신뢰성과 확장성을 제공하는 Spring Eureka를 도입하였고, API Gateway를 통해 외부와의 통신을 관리하였습니다. 또한, 사용자, 제품, 주문과 같은 핵심 기능을 수행하는 여러 마이크로서비스를 개발하였습니다. 마지막으로, 이러한 마이크로서비스들 사이의 데이터 동기화를 위해 Kafka와의 통합을 구현하였습니다.

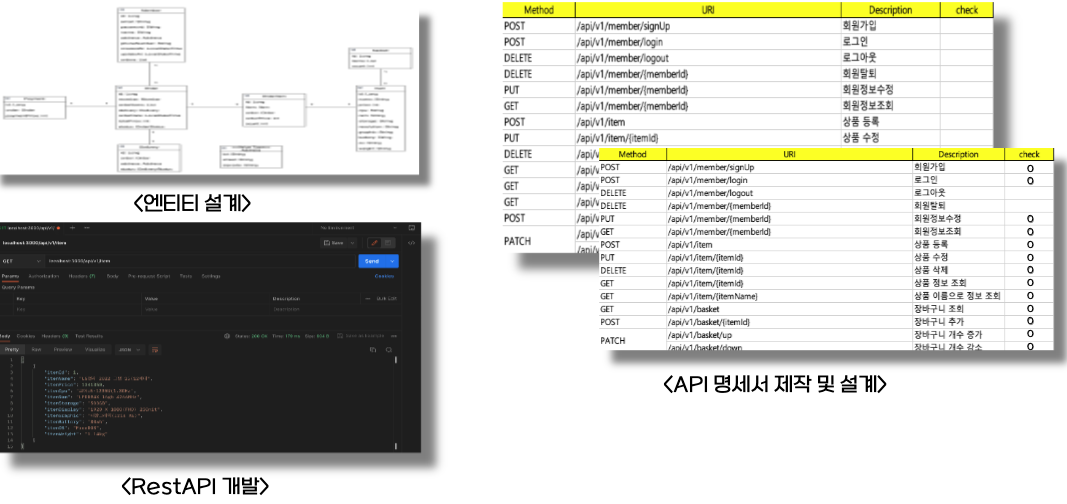
**Kafka 통합 (70% 구현):**

우리는 API 마이크로서비스들 간의 데이터베이스 동기화를 위해 전문적이고 강력한 Kafka를 통합하였습니다. 이를 통해 우리는 데이터의 신속하고 안정적인 전달을 보장할 수 있었습니다. 먼저, 상단 왼쪽에 위치한 ZooKeeper 서버는 안정적인 분산 시스템 운영을 지원하였습니다. 하단 왼쪽에 있는 Confluent 서버는 Kafka 플랫폼의 중심 역할을 수행하며, 상단 오른쪽에 위치한 Kafka 서버는 높은 처리량과 확장성을 제공하였습니다. 마지막으로, 하단 오른쪽에 위치한 컨슈머 서버는 Kafka로부터 데이터를 수신하고 처리하는 역할을 담당하였습니다. 이러한 Kafka 통합은 우리의 시스템이 안정적이고 유연한 데이터 흐름을 유지할 수 있도록 도와주었습니다.

**주문 기능 (70% 구현):**

사용자들이 제품을 주문하는 간편하고 신속한 기능을 제공하기 위해 주문 API를 개발하였습니다. 이를 위해 먼저 우리는 주문 엔티티를 설계하여 주문 정보를 체계적으로 관리할 수 있도록 하였습니다. 또한, 제품 주문 기능을 개발하여 사용자가 제품을 선택하고 주문할 수 있도록 하였습니다. 마지막으로, 제품 잔액 차감을 위한 API를 구현하여 주문이 완료되면 제품 재고를 업데이트하고 잔액을 차감할 수 있도록 하였습니다. 이를 통해 사용자들은 편리하게 제품을 주문할 수 있고, 시스템은 신속하게 주문 처리와 재고 관리를 수행할 수 있게 되었습니다.

**3.2.3 Product Service 설계(60%)**

****

**요구사항 분석 및 환경구축(100%):**

저희 서비스를 구현하기 위해 체계적인 도메인 설계와 엔티티 설계를 수행하였습니다. 도메인 설계를 통해 관계를 명확히 파악할 수 있으며, 각 기능의 엔티티 설계를 통해 필요한 정보를 효율적으로 관리하고 활용할 수 있게 되었습니다. 이를 바탕으로 API 명세서를 작성하여 개발 시 효율성을 극대화하였습니다.

뿐만 아니라 저희는 마이크로서비스 아키텍처를 통해 각 기능을 독립적으로 개발하고 연동할 계획입니다. 마이크로서비스 아키텍처를 채택함으로써 각 기능 간의 결합도를 낮추고 유연한 개발과 확장성을 도모할 수 있습니다. 또한, 마이크로서비스 간의 통신을 위해 표준화된 API를 사용하여 각 기능을 효과적으로 연결할 것입니다.

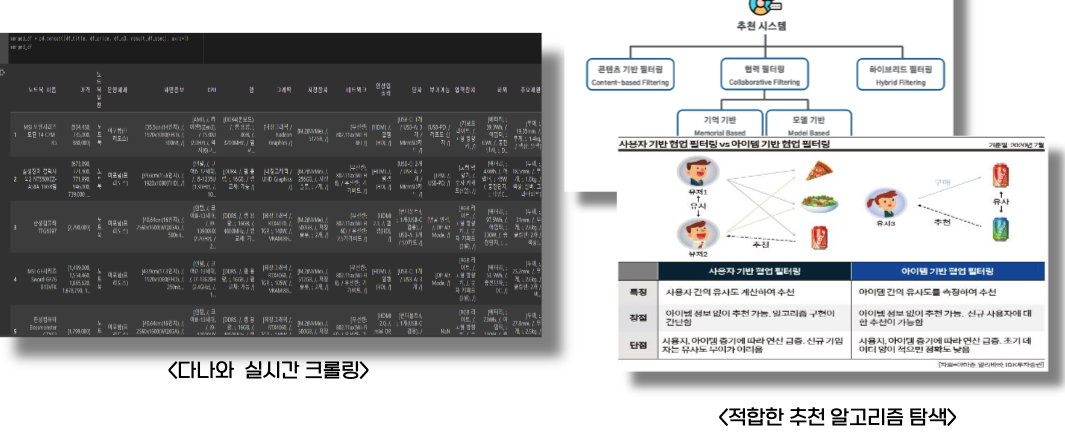
**상품 기능 개발(70%):**

저희 서비스에서는 상품에 대한 상세 정보와 장바구니 기능 등 상품 기능을 위한 API를 개발하였습니다. 이를 통해 고객들이 원하는 상품에 대한 자세한 정보를 얻을 수 있으며, 장바구니에 원하는 상품을 담아 더욱 편리하게 쇼핑을 즐길 수 있습니다.

저희가 구현한 상품 API는 상품 상세 정보 조회 등 다양한 기능을 포함하고 있습니다. 이를 통해 고객들은 다양한 옵션을 활용하여 원하는 상품을 쉽게 찾을 수 있습니다. 또한, 상세 정보 API를 통해 고객들은 상품의 사양, 가격, 재고 등을 확인할 수 있어 구매 결정을 내리는 데 도움을 받을 수 있습니다.

장바구니 기능은 고객들이 원하는 상품을 임시로 저장하고, 필요할 때 언제든지 확인하고 구매할 수 있는 기능입니다. 저희는 장바구니와 주문 기능을 연동하여 고객들이 장바구니에 담은 상품을 주문할 수 있도록 계획하고 있습니다. 이를 통해 고객들은 장바구니에 담은 상품을 한꺼번에 주문하거나 필요에 따라 선택적으로 구매할 수 있게 됩니다.

**3.2.4 Recommend System 설계(60%)**

****

**다나와 사이트 크롤링 및 데이터 정제화 (100% 구현)**

다나와 사이트의 HTML 구조와 데이터 패턴을 분석하여 데이터 추출을 위한 필요한 정보를 파악하였습니다. HTML 태그와 CSS 선택자를 활용하여 제품 이름, 가격, 상세 스펙 등의 데이터를 추출할 수 있는 방법을 이해하였습니다. Python의 requests 라이브러리와 BeautifulSoup 라이브러리를 사용하여 다나와 사이트에 HTTP 요청을 보내고, HTML 페이지를 파싱하여 필요한 데이터를 추출하는 크롤러를 구현하였습니다. 이를 통해 다나와 사이트에서 필요한 데이터를 추출하였습니다. 추출한 데이터를 정제하여 일관된 형식으로 가공하고, 필요한 정보를 추출하여 저장하는 작업을 수행하였습니다. 이를 통해 추출한 데이터의 정확성과 일관성을 유지하였습니다.

**추천 알고리즘 개념 이해 및 구현 (30% 구현)**

추천 알고리즘의 개념과 주요 접근 방식인 사용자 기반(User-based) 추천과 아이템 기반(Item-based) 추천을 이해하였습니다. 또한, 협업 필터링, 콘텐츠 기반 필터링, 하이브리드 추천 시스템 등 다양한 알고리즘 유형을 학습하였습니다.

추천 알고리즘 선택과 데이터 수집 계획: 추천 알고리즘 선택을 위해 데이터의 특성을 분석하고, 사용자에게 필요한 정보를 수집하기 위한 페이지 화면을 구성하였습니다. 또한, EDA(탐색적 데이터 분석)를 통해 데이터의 특성을 파악하여 최적의 추천 알고리즘을 비교 및 분석하였습니다.

**4. 테스트 계획**

|  |  |  |
| --- | --- | --- |
| 기능 | 체크리스트 | 통과 여부(PASS, FAIL) |
| 로그인 기능 | 올바른 아이디와 비밀번호로 로그인 되는지 확인 |  |
| 잘못된 아이디나 비밀번호로 로그인 시도 시 오류 처리되는지 확인 |  |
| 세션 로그인, 일반 회원 로그인 등 로그인 상태 유지 기능이 제대로 작동하는지 확인 |  |
| 사용방식 선택 기능 | 사용자가 원하는 사용 방식을 선택하고 저장되는지 확인 |  |
| 선택한 사용 방식에 따라 해당 기능들이 제대로 활성화 되는지 확인 |  |
| 필터 기능 | 사용자가 필터 옵션을 선택하면 원하는 결과를 가져오는지 확인 |  |
| 다양한 필터 조합에 대해 예상한 결과가 나오는지 확인 |  |
| 노트북 상세정보 조회 기능 | 특정 노트북 상세정보를 조회할 때 필요한 정보들이 정확히 표시되는지 확인 |  |
| 올바르지 않은 상품 조회 시 오류처리되는지 확인 |  |
| 장바구니 기능 | 상품을 장바구니에 추가하는 기능이 제대로 작동하는지 확인 |  |
| 장바구니에 있는 상품을 제거하는 기능이 정상적으로 작동하는지 확인 |  |
| 주문 기능 | 장바구니에 담긴 상품을 주문하여 주문 내역이 정확히 생성되는지 확인 |  |
| 주문한 상품의 재고가 감소하는지 확인 |  |
| 결제 기능 | 결제 수단을 선택하고 결제 정보를 입력하여 정상적으로 결제가 이루어지는지 확인 |  |
| 올바르지 않은 결제 정보를 입력할 경우 오류 처리되는지 확인 |  |