B팀 개발 완료 보고서

프로젝트 명	쇼핑몰 웹 개발 – Ripo(리포)	팀원	소연희, 천지호
프로젝트 URL	https://ripo-tau.vercel.app/	작업기간	2025.05.08. - 2025.05.23.
개발 환경 및 활용 기술	개발 환경 : window, chrome(해상도1920), Visual Studio Code, Figma, GitHub, Vercel 활용 기술 : HTML, Sass, React, Node.js, XAMPP, MySQL, PHP, Swiper Slide, MUI, motion, sweetalert2, Local Storage, Session Storage		

작업 내역

기획 & 데이터 수집

- 팀원과 회의를 통해 쇼핑몰의 주제를 '디자인문구'로 선정
- 판매할 아이템을 구상하고, 대카테고리→중카테고리→소카테고리 구조로 세분화하여 정리
- 소카테고리에 포함할 상품 목록을 선정하고 구체화
- 쇼핑몰 콘셉트에 맞춰 디자인 방향, 컬러 및 폰트 등 시각적 요소 선정

화면 설계 및 업무 분배

- 목적 : 매일 기록하는 습관과 소소한 취향을 반영해, 자연스럽게 소비로 이어지는 감성적이고 귀여운 쇼핑 공간 제공
- 방향성 : 카테고리별 상품 리스트 및 상세 정보를 제공하고 회원가입, 로그인, 검색, 장바구니, 주문 내역을 통해 사용자 맞춤 기능 제공
- 01 홈 : 사용자가 처음 접속하는 화면으로, 슬라이드를 활용해 상품 콘텐츠 정보 전달, 빠르게 정보를 확인할 수 있도록 카테고리 탭 구성, 새로 나온 상품들을 한눈에 볼 수 있도록 슬라이드 형식으로 상품 소개
- 02 검색 : 정보들을 쉽게 검색할 수 있고, 추천 키워드를 제시하여 사용자 편의성 높임
- 03 카테고리 : 서버에 저장된 정보를 불러와 사용자가 원하는 카테고리를 선택하여 탐색할 수 있도록 구성
- 04 상품 정보 : 선택한 카테고리에 해당하는 상품들을 리스트 형식으로 제공하며, 상품 선택 시, 상세 페이지에서 이미지, 설명, 가격 등의 정보 확인, 장바구니 담기 및 구매 기능도 함께 제공
- 05 장바구니 : 상세 페이지에서 담은 상품을 저장하고, 선택한 상품만 골라 결제할 수 있도록 구성, 체크 박스를 통한 선택 결제, 개별 삭제, 총 금액 자동 계산 등의 기능 제공
- 06 마이페이지 : 로그인 기능 제공, 주문 내역 확인, 자주 묻는 질문 및 공지사항 등 사용자 편의를 위한 기능 구성
- 업무 분배 : 팀원 각자 부족했던 영역을 고려해 다양한 파트를 경험할 수 있도록 유연하게 역할을 분배하고 조정

디자인

- 화면 설계한 와이어프레임을 토대로 이미지, 글꼴, 색상 등 적용
- 그 후 팀원들과 상의 후 부족한 점을 보완하여 수정 작업 진행
- 2025년 5월 08일 ~ 5월 13일까지 기획, 화면설계, 디자인 작업 진행

화면구현

- 기획 및 디자인 작업을 마친 후, 5월14일부터 ~ 5월21일까지 개발

- **00 스플래쉬**

  1) 스플래쉬

  ① setTimeout을 통해 정해둔 시간동안 화면에 나타났다 사라지도록 구현

  ② 접속 경로가 home이고 localStorage에 visited로 저장된 값이 없다면 첫 접속으로 인식하여 스플래쉬 페이지로 이동하여 localStorage에 visited 저장

- **01 홈**

  1) 헤더

  ① 헤더는 상황에 따라 로고, 뒤로 가기 버튼, 닫기 버튼, 아무것도 없는 네 가지로 나뉘며 각 페이지에 맞게 구성

  2) 바텀 메뉴

① MUI 플러그인을 활용하여 주요 메뉴로의 이동을 돕기 위해 하단 메뉴를 구성하였으며, 클릭 시 색상 변경을 통해 현재 페이지를 인지할 수 있도록 작업
② 헤더와 바텀 메뉴가 필요 없는 페이지의 경우 hiddenPaths 변수에 경로를 정의하거나 '/' 기준으로 split하여 포함 여부를 확인한 뒤, 해당 경로가 포함되어 있을 경우 컴포넌트가 렌더링되지 않도록 분기 처리

3) 섹션 1 - 메인 슬라이드

① Swiper Slide를 활용하여 랜덤으로 4개의 데이터를 가져와 화면 상단에 표시
② 하루 동안 동일한 컨텐츠가 유지되도록 localStorage에 데이터를 저장하고, 24시간이 지난 후 새로운 데이터로 갱신되도록 구현
③ 클릭 시 해당 상품의 상세 페이지로 이동하기 위해 카테고리 데이터를 불러온 뒤, 상품 데이터의 cat_id와 카테고리 데이터의 id가 일치할 경우 cat_name을 추출하여 'product/cat_name' 경로로 이동하도록 구현

4) 섹션 2 - 소카테고리 바로가기

① 주요 소카테고리를 선정한 뒤, imgUrl과 name을 포함한 배열을 생성하고 클릭 시 'product/name' 경로로 이동하도록 구현. 상세페이지 이동 방식은 '홈 – 메인 슬라이드'와 동일

5) 섹션 3 – 신상품 리스트

① Swiper Slide를 활용하여 상품 데이터를 뒤에서 4개씩 두 번 slice하여, 두 줄의 swiper가 화면에 나타나도록 구현
② 각 항목에는 상품 첫 번째 이미지, 상품명, 가격이 표시되며 클릭 시 해당 상품의 상세페이지로 이동. 상세페이지 이동 방식은 '홈 – 메인 슬라이드'와 동일

- **02 검색**

1) 검색 내 메인 페이지

① 사용자가 검색어를 입력하면 searchInput의 값이 업데이트되며, 값이 공백이 아닐 경우 wordClick 함수가 실행되어 검색 디테일 페이지로 이동하도록 구현
② 몇 가지 검색어를 선정하여 keyword 변수에 배열 형태로 저장한 뒤 화면에 표시하였으며 클릭 시 해당 검색어로 검색이 가능하도록 구현
③ Swiper Slide를 활용하여 localStorage에 저장된 최근 본 상품 데이터를 화면에 표시하였으며, localStorage 데이터와 카테고리 데이터를 모두 불러온 후 렌더링 여부를 결정하기 위해 isReady 상태값을 활용. 클릭 시 해당 상세페이지로 이동하며 이동 방식은 '홈 – 메인 슬라이드'와 동일

2) 상세 페이지

① 검색어를 입력하면 해당 단어가 p태그에 노출되고, 검색된 결과의 총 length값이 함께 보여지도록 작업
② 검색어가 상품 이름(p_name)에 포함된 경우 결과값에 포함시켜, filteredData에 저장되어 최종적으로 검색한 단어에 대한 데이터를 관리하는 searchResult 변수에 저장
③ 결과 리스트가 홀수일 경우, 한 줄에 두 개의 컬럼이 정렬되도록 push를 이용해 빈 박스를 추가하여 화면에 균형 있게 표시되도록 구현
④ 탑 버튼 클릭 시 화면 최상단으로 이동할 수 있도록 window.scrollTo({ top: 0, behavior: 'smooth' })"을 사용해 부드럽게 스크롤 되도록 구현

- **03 카테고리**

1-1) 카테고리 리스트 (프론트)

① 서버에 저장된 카테고리 데이터의 level 값을 이용하여 대/중/소 카테고리 구분
② level=0인 대카테고리를 map으로 렌더링하고, 해당 대카테고리의 id와 동일한 parent_id를 가진 level=1인 중카테고리 데이터를 CategorySub 컴포넌트에 전달
③ CategorySub는 전달받은 중카테고리 데이터를 기준으로, 위와 같은 방식으로 level=2인 소카테고리 데이터를 경로와 함께 CategoryItem 컴포넌트로 전달
④ 클릭 시 해당 소카테고리명이 포함된 경로로 이동

1-2) 카테고리 리스트 (서버)

① 프론트에서 GET 요청 시 getData( ) 함수 실행
② getData( ) 함수는 SQL의 category 테이블 전체 데이터를 조회
③ 조회된 데이터는 배열 형태로 가져온 뒤 JSON 형태로 인코딩하여 프론트에 응답

2) 전체 상품 보기 버튼

① 클릭 시 상품 리스트 페이지로 이동. 카테고리와 상관없이 모든 상품 데이터가 화면에 표시
② 상품 리스트 페이지 및 상세 페이지는 경로에 type 값이 필수이므로 'all'값을 전달하며, 각 컴포넌트에서 해당 값을 기준으로 별도 처리 필요

- **04 상품 정보**

1) 상품 리스트 (프론트)

① 상단에 현재 소카테고리명과 상품 개수(length)를 표시

② CardItem 컴포넌트를 활용하여 파라미터로 전달받은 type 값에 따라 해당 카테고리의 상품 리스트를 카드 형태로 출력. 클릭 시 상세 페이지로 이동

③ 전체 카테고리 데이터를 기반으로 유효성 검사하고, 존재하지 않는 경우 notFound 상태값을 true로 설정하여 안내 화면 출력

④ 전체 상품 데이터를 불러온 후, type이 'all'일 경우 전체 출력, 그렇지 않으면 해당 카테고리 ID와 일치하는 상품만 필터링하여 출력

⑤ 로딩 상태를 처리하여 사용자 경험 개선

⑥ 검색 상세 페이지와 마찬가지로 상품 개수가 홀수일 경우, 레이아웃 균형 유지를 위해 빈 박스 추가

⑦ 탑 버튼을 통해 사용자 편의 제공

2) 상세 페이지 (프론트)

① 선택한 상품의 상세 정보를 제공하며, 장바구니 추가 및 구매 기능을 이용할 수 있음

② 상단에는 상품 대표 이미지들이 자동 슬라이드 형식으로 출력되며, 이미지가 3장 이상일 경우 무한 반복. 기존 이미지는 쉼표(,)로 구분된 문자열을 배열로 변환하여 사용

③ 슬라이드 아래에는 카테고리 경로(대 > 중 > 소), 상품명, 가격이 표시되고 하단에 배송정보도 함께 노출됨. 가격은 쉼표가 포함된 숫자 형태로 출력됨

④ 탭은 상세정보, 문의 두 가지의 메뉴로 구성

- 상세정보 탭: 상품 상세 설명이 HTML 형태로 출력되며 렌더링 최적화를 위해 React.memo 적용
- 문의 탭: 안내 메시지와 함께 자주 묻는 질문 및 마이페이지로 이동할 수 있는 두 개의 버튼이 표시됨

⑤ 하단 구매 바는 기본 상태에선 축소되어 있으며, 클릭 시 확장되어 상품 정보, 수량 조절 UI, 장바구니/구매 버튼이 노출됨

- 장바구니: 클릭 시 서버에 저장되고 저장 완료 스낵바 메시지 표시
- 구매: 클릭 시 구매 페이지로 이동하며 state로 상품 정보 전달

⑥ 로딩 상태를 처리하여 사용자 경험 개선

⑦ 존재하지 않는 상품이거나 잘못된 카테고리로 접근 시 안내 메시지를 출력하여 예외 처리

⑧ 페이지 진입 시 해당 상품 정보가 localStorage에 저장 (최대 8개, 중복 제거 및 최신순 정렬)

3) 상품 리스트 & 상세 페이지 (서버)

① 카테고리 조회와 동일하게 GET 요청 시 getData( ) 함수 실행

② getData( ) 함수는 product 테이블의 전체 데이터를 조회하며, id 기준으로 오름차순 정렬하여 프론트로 반환

- **05 장바구니**

1-1) 장바구니 목록 가져오기 (프론트)

① sessionStorage에 저장된 mem_id값을 기반으로, 해당 사용자와 일치하는 상품 목록을 가져오기 위해 서버에 get으로 요청

② mem_Id 를 파라미터로 전달하여 응답받은 데이터를 setCartList에 저장하고, 동시에 localStorage에도 저장, 이후 상품이 삭제되거나 추가될 때 localStorage를 갱신하여 바로 반영되도록 구성

1-2) 장바구니 목록 저장하기 (서버)

- POST

① 프론트에서 POST 요청 시 postData( ) 함수 실행

② 프론트로부터 받은 JSON 데이터를 디코딩하고, 값이 없을 경우 실패 메시지를 JSON 형태로 반환

③ 전달받은 데이터를 변수로 분리하여 INSERT 쿼리를 통해 각각 cart 테이블에 저장될 항목(mem_id, p_id, p_name, p_price, p_ea, p_thumb, cat_id)에 매핑

④ 쿼리 실행 결과에 따라 메시지 반환

- PUT

① 프론트에서 PUT 요청 시 putData( ) 함수 실행

② POST 요청과 유사하게 JSON 데이터를 수신하지만, 사용자 ID(mem_id)와 상품 ID(p_id)가 필수이며, 해당 값이 없을 경우 에러 메시지를 반환

③ 전달받은 데이터를 변수로 분리한 뒤, UPDATE 쿼리를 통해 cart 테이블에서 해당 mem_id와 p_id를 가진 항목의 p_price, p_ea 값을 수정

④ 쿼리 실행 결과에 따라 메시지 반환

2-1) 상품 선택 및 삭제 (프론트)

① 상품 상세 페이지에서 장바구니에 담은 상품들을 확인할 수 있으며, 개별 선택 후 구매가 가능하도록 구현

② checkItems 상태를 활용하여 각 상품의 선택 여부를 관리하고, every( ) 메소드를 사용하여 모든 상품이 선택되었는지 확인한 후, 전체 선택 여부를 판별하도록 작업

③ 장바구니에 상품이 추가되거나 삭제될 경우 cartList가 변경되며, 체크 상태 값을 초기화되도록 작업

④ 상품 삭제는 상품 우측에 있는 닫기 버튼이나 해당 상품을 선택하여 선택 삭제 버튼으로 삭제 가능하도록 구현

2-2) 상품 삭제 (서버)

① 상품 선택 삭제

- POST 메소드를 사용하여 JSON 형식의 데이터를 서버로 전송
- 서버에서는 JSON 데이터를 PHP 배열로 변환하여 아이디 값들을 배열에 넣고 아이디 값이 없으면 빈 배열로 처리
- ids가 비어있지 않고 배열이 확인되면, 배열 안에 있는 해당 아이디 값을 정수 int로 변환하고 쉼표로 문자열을 만들어 해당 아이디에 포함되는 장바구니 목록을 삭제하도록 처리

② 상품 개별 삭제

- 개별 상품 id만 서버에 전달하여 해당 상품 하나만 삭제되도록 구성

3) 선택한 상품 정보 확인(가격 및 수량)

① checkItems를 기준으로 선택된 상품들을 필터링하여 수량과 가격을 각각 합산한 후, 화면에 보여주고 선택된 항목이 없을 경우 기본값인 0으로 처리되도록 작업

4) 추천 상품 보여주기

① 서버에 GET 요청을 보내 상품 리스트를 가져오고, 해당 데이터를 setAddList에 저장하여 이 리스트 중 최근 등록된 8개 상품을 slice로 잘라 화면에 표시

② 장바구니에 담긴 상품과, 추천 상품을 선택했을 때 디테일 페이지로 넘어가게 하기 위해 카테고리 정보를 GET으로 요청

③ 응답받은 카테고리 데이터 중, 각 상품의 cat_id와 일치하는 카테고리명을 찾아 setCartCtgrList와 setProductCtgrList에 저장하여 상품 클릭 시 해당 상세 페이지로 이동할 수 있도록 구현

5) 구매하기 버튼 클릭 시 데이터 전달

① 구매하기 버튼 클릭 시 checkItems를 기준으로 선택된 상품 목록을 필터링

② 선택된 상품의 정보를 결제 페이지로 데이터를 전달하기 위해 '/pay'페이지로 이동될 때, state로 선택한 상품, 전체 가격, 배송비, 총 합을 넘겨 결제 페이지에서 값을 받아 사용할 수 있도록 작업

- **06 결제**

1) 결제 정보 가져오기

① 상세 페이지와 장바구니에서 구매하기 버튼 클릭 시, location.state를 통해 결제 페이지로 데이터를 전달받아 사용

- 전달받은 payData를 기준으로 두 가지 상황에 따라 처리

② 장바구니에서 데이터가 넘어온 경우 (payData.items 존재)

- 상품 리스트를 배열 형태로 받아와 product에 저장
- 상품 가격과 수량을 reduce( )로 계산하여 totalPrice, productCount에 저장
- 배송비(delivery)와 총 주문금액(totalOrder)은 전달된 값을 기준으로 설정

③ 상품 상세 페이지에서 데이터가 넘어온 경우 (payData.p_id 존재)

- 단일 상품 정보를 product에 저장
- 상품 가격(price)과 수량(ea)을 계산하여 총 금액과 배송비 설정

2) 주문자 내역 불러오기

① sessionStorage에 저장된 mem_Id 값을 기반으로 사용자 정보를 서버에서 불러옴

② 응답받은 데이터를 data에 담아두어 setFormData의 이름과 전화번호를 초기값으로 설정하여 주문자 정보 입력칸에 채워지도록 구현

3) 주문번호 생성 및 결제 버튼 클릭

① 주문번호 생성 (orderId)

- 현재 날짜와 시간을 기준으로 주문번호를 생성
- 날짜나 시간이 한 자리 숫자일 경우, 해당 문자 앞에 0을 채워 숫자의 길이를 두자리 수로 맞추기 위해 n.toString( ).padStart(len, '0');를 사용하여 작업
- 오늘 날짜와 현재 시간을 가지고 와 YYYYMMDD-HHMMSSmmm의 형식으로 출력되어 식별 가능하도록 구성

② 결제 버튼 클릭 시 처리
- 결제 버튼 클릭 시, product 데이터를 map( )으로 순회하여 주문 데이터 배열(orderData)로 변환
- 해당 상품을 저장하기 위해, POST로 서버에 요청하여 상품 데이터를 전송
- 만일, 장바구니에서 넘어온 상품일 경우, 해당 상품을 제거하기 위해 localStorage에서 해당 상품의 id값을 찾아 제거하고 localStorage를 업데이트
- 동시에 cart_delete.php에도 상품 삭제 요청한 후, 결제 완료 페이지로 이동

4) 결제완료

① 결제 완료 후, 사용자가 방금 결제한 주문 내역을 확인할 수 있도록 정보 제공
② "계속 쇼핑하기" 등의 버튼을 함께 제공하여 UX 향상과 사용자 편의성을 고려하여 구성

- **07 마이페이지**

1-1) 회원가입 (프론트)

① 입력폼
- 사용자로부터 이름, 아이디, 비밀번호, 전화번호, 이메일 정보를 입력받도록 구성

② 약관동의
- 전체 동의 및 개별 동의 기능을 구현하고 팝업을 통한 약관 상세보기 후 확인 버튼 클릭 시 체크 상태가 자동으로 반영되게 작업

③ 폼 유효성 및 제출 처리
- 모든 필수 정보가 입력되었고, 필수 약관 동의가 완료된 경우에만 회원가입 버튼 활성화
- POST로 form-data방식으로 전송하여 회원가입이 완료되면 완료되었다는 팝업창으로 알리고 로그인 페이지로 이동

1-2) 회원가입 (서버)

① POST 요청을 받아 member 테이블의 아이디, 이름, 비밀번호, 이메일, 전화번호 칼럼에 회원 정보 저장
② 특정 회원 조회시 GET 요청 후 데이터 조회 가능하도록 작업

2-1) 로그인 (프론트)

① 로그인 버튼을 눌렀을 때, login 함수가 실행
② 해당 함수에서는 FormData에 사용자 입력 데이터를 담아 login.php 에 POST 전송
③ 응답에서 success가 true일 경우 sessionStorage에 사용자 id를 저장하고 홈으로 이동
④ 로그인 실패 시 아이디 또는 비밀번호가 맞지 않다는 팝업창 알림

2-2) 로그인 (서버)

① POST로 요청된 id와 pw를 $id, $pw 변수에 저장하고 입력한 아이디를 기준으로 DB에서 회원정보를 조회
② 해당 아이디가 존재하면 $row에 회원 정보 저장
③ DB에서 가져온 비밀번호와 사용자가 입력한 $pw가 일치하는지 비교하고 사용자의 아이디와 이름을 세션에 저장
④ 프론트에서 이후 페이지 이동 시, 세션 값을 이용하여 로그인 상태를 유지할 수 있도록 작업

3-1) 로그아웃 (프론트)

① 로그아웃 클릭 시, 로그아웃을 진행하겠다는 팝업이 표시되고 확인 버튼을 클릭하게 되면 logout.php로 요청 전송된 후 세션 초기화 및 sessionStorage 삭제
② sessionStorage.clear( )로 프론트 저장 정보 삭제
③ 팝업의 취소 버튼 클릭 시, 팝업이 닫힘

3-2) 로그아웃 (서버)

① session_unset( )으로 세션 변수 초기화한 후, session_destroy( )로 세션 파일 제거 및 세션 종료

4-1) 마이페이지 (프론트)

① 회원정보 조회
- sessionStorage에 저장된 mem_id가 없다면 로그인 페이지로 이동
- 서버에서 회원 정보를 응답하면 setViewUser를 통해 상태에 저장
- viewUser에 값이 있을 때만, 화면에 출력
- 해당 값으로 상단에 사용자의 이름 및 ID를 표시

4-2) 마이페이지 (서버)

① 세션에 member_id와 member_name이 존재하는 경우 JSON 형태로 사용자 정보 반환
② 프론트에서 로그인 상태인지 확인하고 사용자 이름 및 ID를 출력하는 데 활용

5-1) 주문내역 (프론트)

① 주문내역 리스트
- 결제완료한 주문 내역 리스트를 불러오기 위해 mem_id를 파리미터로 전달하여, 응답받은 데이터를 setOrderList에 저장

	- orderList를 map( )을 활용하여 화면에 노출하고, 주문 일자는 저장된 데이터 중 created_at에서 날짜 부분만 slice(0,10)으로 추철해 상단에 출력 - 하루에 여러 개를 주문한 경우, 날짜가 중복 노출되지 않도록 이전 항목과 날짜가 다를 때만 날짜 영역을 표시 - 각 주문 상품은 컴포넌트를 활용하여 이미지, 상품명, 수량, 가격 등 정보를 출력 - 주문 내역이 없을 경우, InfoMessage 컴포넌트를 통해 주문 내역이 없다는 것을 표시하여 사용자에게 인식시킴 ② 주문내역 클릭 시 - 주문한 상품을 클릭했을 때, 해당 상품으로 이동시키기 위해 카테고리 전체 목록을 API를 통해 받아옴 - 이후 주문내역에 있는 상품의 아이디와 카테고리 아이디 값이 같을 경우 ctgyList 배열에 저장 - 해당 상품을 클릭했을 때, 카테고리 아이디값을 찾아 상세 페이지로 넘어갈 수 있도록 작업 **5-2) 주문내역 (서버)** ① 주문내역 저장 (POST) - JSON 형태의 데이터를 받아 json_decode( )로 PHP 배열로 변환 - orders 테이블에 데이터를 삽입 - 주문번호, 아이디, 상품관련 정보들, 카테고리 아이디, 현재 시간등을 저장 ② 주문내역 조회 (GET) - mem_id값을 기반으로 회원의 주문 내역을 조회 - orders 테이블에서 mem_id에 해당하는 데이터를 역순으로 정렬해서 가지고 옴 - fetch_all(MYSQLI_ASSOC)를 사용하여 연관 배열 형태로 전체 데이터를 가져오고, json_encode( )로 응답 **6) 자주 묻는 질문** ① 총 6개의 질문과 답변이 아코디언 형태로 구성되어 있으며, 각 항목은 클릭 시 열리고 다시 클릭하면 닫히는 토글 방식 ② 한 번에 하나의 질문만 펼칠 수 있도록 인덱스를 기준으로 상태를 관리(openIndex) **7) 공지사항** ① 자주 묻는 질문과 마찬가지로 아코디언 형태로 구성되어 있으며, 클릭 시 내용이 펼쳐짐 ② 각 항목에 **공지 유형(일반공지, 이벤트 등)**을 표시하여, 정보 전달력과 사용자 경험 향상 **8) 전화연결** ① 전화걸기 버튼 클릭 시, 사용자의 브라우저 정보를 얻어 모바일 기기일 경우 전화 기능을 활성화 시키고, 모바일 기기가 아닌 경우 alert으로 웹 브라우저 환경일 경우 직접 고객센터로 전화하라는 문구를 띄워 알림 **테스트** • 개발 작업을 마친 후, 5월22일부터 ~ 5월23일까지 배포 후 테스트 (중간 통합 제외) • 서로 작업한 파일들을 깃허브와 파일질라를 통해 통합 후 Vercel 배포 • 공통 컴포넌트 작업 및 개별 작업이 일정 수준 이상 완료될 때마다 중간 통합을 진행하였으며, 총 7차 통합까지 진행 • 중간 통합 후 작업 내용을 상호 검토하고, 수정 사항을 적용하는 과정을 반복하며 프로젝트의 완성도 높임 **배포** • https://ripo-tau.vercel.app/
산출물 내역	기획, 화면설계, 디자인, PPT 작업파일 pdf 추출 후 해당 폴더에 저장 • B팀-semi프로젝트-리포/semi프로젝트_B팀_01기획,화면설계.pdf • B팀-semi프로젝트-리포/semi프로젝트_B팀_02디자인.pdf • B팀-semi프로젝트-리포/semi프로젝트_B팀_03PPT.pdf
장·단점 및 특이사항	-