B팀 개발 완료 보고서

프로젝트 명	정보전달 앱 개발 - 떠나봅서	팀원	소연희, 안지현, 천지호, 황수빈, 이용욱
프로젝트 URL	https://jeju-trip-eosin.vercel.app/	작업기간	2025.04.02. - 2025.04.23.
개발 환경 및 활용 기술	개발 환경 : window, chrome(해상도1920), Visual Studio Code, MongoDB, Express, GitHub, Vercel 활용 기술 : Figma, HTML, Sass, React, Node.js, 공공데이터 API, 카카오/네이버/구글 로그인 API, Swiper Slide, react-date-range, date-fns, JSON, Local Storage, Session Storage		
작업 내역	**기획** • 팀원들과의 회의를 통해 대 카테고리를 여행으로 선정 • 국내에서 자주 찾는 여행지인 제주도로 위치 결정 • 제주 여행 앱에 어떤 내용을 담을지 선정하기 위해, 팀원별로 다양한 여행 관련 어플을 분석하여 카테고리 정리 • 분석한 내용들을 토대로 내용 취합 후 카테고리 선정 • 제주 여행 앱과 어울리는 디자인 컨셉, 컬러 및 폰트 선정 **데이터 수집** • 선정된 카테고리를 기반으로 공공데이터포털에서 오픈 API 서치 후 데이터 신청 • 오픈 API 데이터들을 팀원들과 함께 볼 수 있게 파일 정리 후 공유 **화면 설계 및 업무 분배** • 목적 : 제주의 관광지, 맛집, 축제, 쇼핑 관련 API를 활용하여 카테고리별 정보 제공 • 방향성 : 여행 계획과 체크리스트 기능으로 여행 준비가 가능하고 커뮤니티를 통한 여행 팁, 후기, 추천 정보 전달 • 00 검색 : API 정보들을 쉽게 검색할 수 있고, 인기 검색어, 인기 태그를 보여줌으로서 검색에 용이하게 구성 • 01 홈 : 접속 시 가장 먼저 노출되는 화면으로, 제공되는 정보를 간략히 소개하기 위해 카테고리별 하나의 콘텐츠를 가지고 슬라이드 형식으로 구성, 빠르게 정보를 확인할 수 있도록 카테고리 탭 구성, 관광지/맛집을 간단히 소개하는 파트, 커뮤니티의 떠나팁 부분의 이미지를 가지고 포토스팟을 소개해주는 파트, 관광객 수 통계를 월별로 보여주어 어느 때에 관광객 방문이 많은지 파악할 수 있는 파트 • 02 트립 : API에서 정보를 불러와 관광지, 맛집, 축제/행사, 쇼핑 카테고리의 정보를 제공, 원하는 장소를 저장할 수 있는 좋아요 기능 구성 • 03 커뮤니티 : 사용자들이 자유롭게 대화를 나눌 수 있고, 질문, 맛집, 장소 등을 공유할 수 있는 파트, 관리자만 올릴 수 있는 떠나팁 카테고리로 사용자들에게 포토스팟을 따로 공유해주는 파트, 게시물에 올린 이미지들을 모아둔 갤러리 구성, 게시물 좋아요와 댓글을 달 수 있는 기능 구성 • 04 내 여행 : 제주 날씨를 보여주고, 저장한 일정들을 리스트로 보여주고 트립에서 둘러본 정보들을 토대로 일정을 추가로 만들 수 있는 기능 구성, 사용자들에게 여행 일정을 추천해주는 추천 일정 기능 • 05 마이페이지 : 여행 일정에 필요한 체크리스트 작성 기능 제공, 좋아요한 콘텐츠나 커뮤니티 게시물을 확인할 수 있는 페이지 구성, 내가 작성한 글을 확인할 수 있는 나의 활동 페이지 구성, 등록한 여행 일정을 확인할 수 있고 추천 일정을 확인할 수 있는 기능, 사용자들을 위한 고객센터 자주 묻는 질문 및 전화 파트 구성, 로그인 로그아웃 기능 • 업무 분배 : 작업 파트는 팀원 간 균등하게 나누되, 랜덤하게 분배한 후 필요한 경우 각자 관심 분야나 경험을 반영하여 유연하게 조정 **디자인** • 화면 설계한 와이어프레임을 토대로 이미지, 글꼴, 색상 등 적용 • 그 후 팀원들과 상의 후 부족한 점을 보완하여 수정 작업 진행 • 2025년 3월 20일 ~ 3월 31일까지 기획, 화면설계, 디자인 작업 진행		

React 화면구현

- 기획 및 디자인 작업을 마친 후, 4월1일부터~ 4월18일까지 개발

- **00 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 API로 불러온 인기 검색어, 인기 태그는 클릭을 통해 검색이 가능하며, 띄어쓰기가 포함된 단어는 공백 없이 수정하여 작업

  2) 디테일 페이지

  ① 검색어를 입력하면 해당 단어가 h3 태그에 노출되고, 검색된 결과의 총 length값이 함께 보여지도록 작업

  ② 검색어는 콘텐츠의 제목, 주소, 태그에 포함된 경우 결과값에 포함시켜, filteredData에 저장되어 관광지, 음식점, 쇼핑, 축제 카테고리별로 분류하여 작업

  ③ 결과 리스트는 한 번에 30개씩 표시되며, 화면 하단에 도달하면 다음 리스트가 자동으로 로드되도록 구현

- **01 홈**

  1) 헤더

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

  ② 검색 아이콘과 리스트(버거 메뉴) 아이콘을 통해 어느 페이지에서든 검색과 메뉴 접근이 가능하도록 작업

  2) 버거 메뉴

  ① 각 메뉴에 해당하는 페이지로 이동해야 하므로 NavLink를 통해 이동시킴

  ② 버스정보의 경우 외부 사이트 URL을 새 탭에서 열 수 있도록 window.open을 사용

  ③ 헤더에 리스트(버거 메뉴) 버튼이 있기 때문에 헤더에 버거 컴포넌트를 불러와 버거가 열고 닫힘을 알기 위해 useState로 열린 상태를 관리

  3) 바텀 메뉴

  ① 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 변수 안에 제거해야 할 경로를 적어 해당 경로가 포함되어있을 때는 실행되지 않게 작업함

  4) 섹션1 메인 슬라이드

  ① Swiper Slide를 사용하여 관광지, 맛집, 축제, 쇼핑 카테고리 중 하나를 랜덤으로 뽑아 하루 동안 화면에 보여줄 데이터를 localStorage에 저장하고, 그 데이터를 노출시키도록 구현

  ② 하루가 지나게 되면 localStorage에 저장된 값을 삭제하고 새로운 데이터를 불러오도록 작업

  ③ 슬라이드 클릭 시 트립 디테일 페이지로 이동하게 작업

  5) 섹션2 탭 메뉴

  ① 트립 카테고리인 tour, food, festival, shopping 부분을 빠르게 이동할 수 있도록 탭으로 구성

  ② 해당 카테고리 선택 시 트립 리스트 페이지로 이동할 수 있도록 작업

  6) 섹션3 날씨

  ① API를 활용해 현재 기온, 최저 최고 기온, 바람 정보 제공

  ② 날씨 상태를 한눈에 알아볼 수 있도록, 예를 들어 맑음일 경우엔 맑은 날씨에 해당하는 이미지가 보여지도록 작업

  7) 섹션4 관광지

  ① 메인 슬라이드처럼 하루 동안 보여줄 데이터를 localStorage에 저장하고, 24시간이 지나면 새로운 데이터로 교체되어 새로운 콘텐츠가 노출되도록 구성

  ② 전체보기 버튼을 클릭하면 관광지 리스트 페이지로 이동 가능하게 작업

  ③ 콘텐츠 클릭 시 펼쳐지며, 해당 콘텐츠를 한 번 더 클릭하면 콘텐츠 디테일 페이지로 이동

  8) 섹션5 맛집

  ① 관광지 섹션과 마찬가지로 하루 동안 보여줄 데이터를 localStorage에 저장하고, 24시간이 지나면 새로운 콘텐츠로 교체되도록 작업

  ② 전체보기 버튼을 사용하여 맛집 리스트 페이지로 이동할 수 있게 작업

  ③ 콘텐츠 클릭 시 디테일 페이지로 이동

9) 섹션6 포토

① 커뮤니티의 떠나팁 카테고리에 업로드된 이미지를 서버에서 불러와 Swiper Slide로 구성

② 이미지 클릭 시 해당 게시물의 디테일 페이지로 이동하고, 전체보기 버튼 클릭 시 갤러리 페이지로 이동 가능하도록 작업

③ 이미지나 전체보기 버튼을 클릭 했을 때, 로그인 상태의 경우에만 접근 가능하며, 로그인하지 않은 경우에는 팝업창을 띄워 로그인하도록 유도

10) 섹션7 관광객 수

① 2024년 기준의 관광객 수 API 데이터를 불러와 첫 화면 기본값을 현재 월로 설정하여 해당 월에 맞게 관광객 수가 보여지도록 구성

② 인풋 박스를 클릭하면 팝업창이 나타나며, 사용자가 원하는 월을 선택할 수 있게 구현

③ 선택한 월이 true이거나 관광객 수의 length값이 0 이상일 경우, 데이터에서 '구분연월'을 찾아 해당 데이터를 가져와 숫자를 보여주도록 작업

④ 팝업창에서 원하는 월을 선택하지 않았을 경우, 팝업의 완료 버튼이 활성화되지 않고 닫기 버튼으로 창을 닫을 수 있도록 구성

⑤ 숫자가 올라가는 애니메이션은 motion.dev 플러그인을 사용해 관광객 수치를 자연스럽게 보여지도록 작업

11) 하단 탑 버튼

① 탑 버튼 클릭 시 화면 최상단으로 이동할 수 있도록 `window.scrollTo({ top: 0, behavior: 'smooth' })`" 을 사용해 부드럽게 스크롤되도록 구현

- **02 트립**

1) 트립 내 메인 페이지

① Swiper Slide를 활용해 각 카테고리당 5개의 콘텐츠를 노출하도록 구성

② 홈 화면과 마찬가지로 하루 동안 보여줄 동일한 콘텐츠가 유지되도록 데이터를 localStorage에 저장하고, 24시간이 지난 후 새로운 데이터로 갱신되도록 구현

2) 리스트 페이지

① 카테고리별 대표 이미지를 3장씩 저장하여 하루 동안 동일한 이미지가 보여지도록 localStorage에 저장

② 사용자는 필터를 사용하여 리스트의 순서를 오름차순, 내림차순, 좋아요순으로 확인할 수 있도록 작업

③ 콘텐츠 리스트를 보여주기 위해 해당 카테고리에 맞는 데이터를 호출하며, 로딩 상태를 처리하여 사용자 경험 개선

④ 로딩 시간 단축을 위해 처음 보여지는 리스트의 개수를 30개로 지정한 후, 화면의 하단에 도달했을 때 콘텐츠가 자동으로 로드되는 무한 스크롤 방식으로 구현

⑤ 리스트 우측의 좋아요 버튼을 통해 사용자가 좋아요를 누른 콘텐츠를 확인할 수 있으며, 사용자들이 해당 콘텐츠에 좋아요 누른 총 수를 확인할 수 있도록 작업

⑥ 좋아요 수는 필터의 "좋아요순" 정렬 기준으로 활용

3) 디테일 페이지

① 선택한 콘텐츠의 상세 정보를 제공하고 해당 콘텐츠의 주소값을 기준으로 주변 콘텐츠를 추천하는 방식으로 작업

② 전화번호나 주소가 없을 경우를 대비해 대체 텍스트를 지정하였고, 전화번호는 지역번호가 없는 경우 자동 추가 및 하이픈(-) 형식을 정리하여 일관성 있게 처리

③ API에서 제공되는 태그는 #으로 구분되어 있어, #을 쉼표로 통일시킨 후 공백과 빈 문자열 제거 작업을 거쳐 태그 목록을 가공

④ 주변 콘텐츠 추천은 주소값을 00시 00읍,면,동 단위까지 잘라내어 같은 지역의 콘텐츠만 추출하고, 콘텐츠 타입이 tour일 경우 tourlist 데이터에서 가져오도록 설정

⑤ 리스트 페이지에서 필터를 적용시킨 후 디테일 페이지로 넘어왔을 때 하단에 이전 이후 버튼 클릭 시 필터 조건이 유지된 콘텐츠들이 순서대로 보여지도록 구현

⑥ 좋아요 기능은 로그인한 사용자만 사용할 수 있으며, 로그인하지 않았을 경우 로그인 안내 팝업이 노출시켜 로그인 유도

⑦ 로그인 후 좋아요 버튼을 클릭하면 현재 상태의 반대값으로 토글되어 서버에 상태를 업데이트되고, 해당 상태가 저장되도록 처리

- **03 커뮤니티**

1) 게시물 업로드

① + 아이콘 클릭 시 새 게시물 작성 페이지로 이동

② 작성 페이지에서 주제 선택이 가능하고, 선택한 주제는 selectedItem으로 상위에 전달하여

localStorage에 저장
③ 게시글 업로드 시 주제, 타이틀, 내용, 날짜 등 post 서버로 전송
④ 이미지 업로드는 최대 4장까지 가능하며, 선택한 이미지는 selectedImages에 저장되어 미리보기로 보여지도록 작업
⑤ FormData를 사용하여 다중 이미지를 전송하고, 서버에 Imgur 업로드 후 URL 반환
⑥ 게시물 업로드 후 리스트에 반영시킨 후, 게시물 클릭 시 디테일 페이지로 이동되면서 localStorage에 게시물 저장 후 디테일 페이지에 적용되도록 작업
⑦ loggedInUserId.id === post.userId 로 작성자 여부 확인하여, 작성자인 경우 수정, 삭제, 댓글 삭제가 가능하도록 작업

2) 게시물 좋아요 및 댓글

① 게시물 리스트 페이지와 디테일 페이지 모두 하트 클릭 시 좋아요 기능이 작동되며, 하트의 색상과 숫자 카운트가 변경되도록 작업
② 좋아요 상태를 localStorage.post에 저장하고 서버에 전송되어, 사용자가 좋아요 누른 게시글 목록 조회가 가능
③ 리스트 페이지에서 좋아요 누른 후 디테일 페이지로 넘어가게 되면 하트의 색상이 변경되지 않아 localStorage.post의 hasVote값을 참조하여 디테일에도 적용되게 작업
④ 댓글 작성 시 서버에 저장되며, 댓글 목록은 최신순으로 정렬되도록 설정
⑤ 댓글 작성자는 삭제 버튼이 노출되며, 게시물 삭제 시 해당 게시물의 좋아요, 댓글도 삭제되도록 작업

3) 갤러리

① axios.get으로 이미지 데이터를 요청하여 post.imageUrls[0] 첫 번째 이미지만 가져와 표시되도록 작업
② 이미지 클릭 시 해당 게시물을 localStorage에 저장한 후 디테일 페이지로 이동되도록 설정

- **04 내 여행**

1) 내 여행 일정 만들기

① 여행 기간 선택 및 관리
- react-date-range 라이브러리를 사용하여 사용자가 여행 시작일과 종료일을 선택할 수 있도록 구현
- 선택된 날짜는 "yyyy.MM.dd" 포맷으로 가공하여 localStorage에 저장하여 화면에 보여주도록 함

② 티켓 컴포넌트 생성 및 티켓 타이틀 수정 기능
- 날짜 배열을 기준으로 각 날짜에 해당하는 티켓 컴포넌트가 렌더링 되도록 작업
- 티켓은 날짜와 등록된 장소 리스트를 표시하며, 추가로 장소를 등록하거나 수정할 수 있도록 구현
- 서버에서 받아온 기본 타이틀을 기반으로 초기화하며, 사용자가 티켓 타이틀을 클릭 시 input으로 전환되어 제목을 수정할 수 있도록 구현
- 이때 수정은 onChange로 상태를 업데이트하고, onBlur 이벤트로 입력 완료 처리함

③ 여행 기간 수정 및 저장 기능
- 여행 날짜 배열을 수정할 수 있도록 하고, 수정 시 화면에 즉시 반영되도록 구현
- 전체 일정 저장하기 버튼 클릭 시 변경된 값을 넣기 위해 타이틀, 날짜, 캘린더 각각 기능별로 useEffect를 분리하고 의존성 배열을 지정하여 변경될 때만 필요한 로직이 실행되도록 작업

④ 장소 추가 기능
- 티켓별 장소 추가 버튼 클릭 시 지역별, 좋아요 두가지 탭을 사용하여 사용자가 원하는 탭에서 장소 리스트를 확인하고 추가할 수 있도록 구현
- 사용자가 리스트를 선택했을 때, 체크 되었는지 확인하기 위한 화면 구현은 선택된 리스트 항목 중 현재 항목과 일치한 ID가 있는지 확인하기 위해 boolean 값을 반환하도록 작업

2) 내 여행 일정 불러오기 (수정, 추가, 삭제 기능)

① 수정, 추가, 삭제 기능 공통 설정
- DB 연결 및 컬렉션을 지정하는 함수 dataCtrl을 생성하여 중복 코드 없이 DB 작업을 수행할 수 있도록 설정
- 사용자 데이터를 여러 곳에서 조회해야 하는 상황을 고려하여, getUser(userId) 함수를 별도 분리
- 이로 인해 중복된 사용자 조회 로직을 제거하고 코드 재사용성 높임

② 여행 일정 리스트 조회 (GET /user/:userId)
- 특정 사용자 ID에 해당하는 모든 여행 일정 목록을 반환하는 API 구축
- 사용자 데이터가 존재하지 않는 경우, 빈 배열을 반환하여 예외 처리 가능하게 작업

③ 여행 일정 디테일 조회 (GET /user/:userId/:planId)
- 사용자 ID와 일정 ID를 함께 받아 해당하는 단일 일정 데이터를 반환하는 기능 구현
- 데이터가 없거나 일치하는 일정이 없는 경우, 빈 배열을 반환하여 안전하게 처리

④ 여행 일정 추가 (POST /)

- 이미 등록된 사용자가 새 일정을 추가할 경우, 해당 사용자의 allList 배열에 새 일정을 $push하여 삽입되도록 작업
- 새로운 사용자라면 새로운 문서를 생성하고 allList에 초기값으로 일정 등록되도록 작업
- 추가 작업이 완료된 후, 항상 전체 일정 목록을 반환하여 동기화 될 수 있도록 처리

⑤ 기존 일정 수정 (PUT /)
- 사용자와 일치하는 문서에서 일정 ID를 기준으로 수정하는 기능 구현
- 기존 코드에서 수정 완료된 후 변경된 내용을 바로 반영시키기 위해 해당 인덱스를 교체하고, 배열 전체를 $set으로 업데이트하여 최신 데이터가 저장되도록 작업

⑥ 일정 삭제 (DELETE /del?id=xxx&userId=yyy)
- 요청받은 userId를 찾은 후, allList 배열에서 id가 일치하는 일정을 $pull 연산을 통해 삭제하는 구조로 구현
- 삭제 이후에는 수정된 데이터를 바로 반환하여 화면에 빠르게 갱신시키도록 처리

3) 여행 추천 페이지

① 여행 기간 선택 (박/일)
- mui 라이브러리의 Select 컴포넌트를 활용하여 사용자에게 1박2일, 2박3일 등 옵션 제공
- 선택한 값은 dates 상태에 저장되며, 이후 추천 일정 조회에 사용되도록 함

② 추천 일정 매핑
- 여행 추천 데이터를 기반으로 필요한 데이터만 뽑아 dataMap 객체로 정리
- 사용자가 선택한 여행 기간에 따라 dataMat에서 추천 일정을 가져와 화면에 출력
- 추천 일정 항목은 PickPlanItem 컴포넌트를 통해 세부적으로 보여주도록 구성

③ Day 버튼 (스트롤 이동)
- 여행 일수에 맞게 Day1, Day2, Day3 등 버튼을 생성하고 상단에 배치
- ref를 배열로 선언하여 각 Day별 DOM 요소를 개별 저장하여 버튼 클릭 시 해당 인덱스에 맞는 요소로 이동되게 작업

④ 일정 가져오기 (추천 일정 전체 복사)
- 사용자가 추천 일정을 사용하고 싶은 경우, 전체 일정 가져가기 버튼을 클릭하여 복사할 수 있도록 구현
- 비로그인 상태라면 로그인이 필요한 서비스라는 팝업을 띄워 로그인하도록 유도
- 로그인 상태라면 추천 일정을 내 여행 계획으로 복사한 뒤, PlannerDetail 페이지로 이동되게 작업
- 새로 복사된 일정에 랜덤 ID를 생성하고, 고유 제목(ex. "추천 일정 1")을 붙여 저장되도록 작업

- **05 마이페이지**

*로그인·로그아웃은 소셜 로그인 연동으로 진행하여 카카오, 네이버, 구글 활용

1-1) 로그인 (서버)

① 애플리케이션 등록 및 환경설정
- 카카오, 네이버, 구글의 공통사항은 "Client ID 발급 (카카오는 Client Secret 불필요), 서비스 URL과 Redirect URI 일치 필수 (/ 유무 주의), Redirect URI 불일치 시 인증 실패 발생" 이 3가지로 파악
- 카카오: Kakao Developers에서 사이트 도메인과 Redirect URI 설정한 후 OpenID Connect 활성화 필요
- 네이버: 개발자 센터에서 서비스 URL, Callback URL 설정한 후, 하나의 서비스 URL만 등록이 가능하여 개발 및 배포 환경에 따라 수정이 필요함
- 구글: Google Developer Console을 통해 승인된 JavaScript 원본, 리디렉션 URI 설정

② 서버 개발 (kakao.js, naver.js, google.js)
- Express 서버에서 소셜 로그인 기능을 구현
- 프론트에서 받은 인가 코드를 이용하여 토큰을 요청하고 사용자 정보를 조회하여 DB에 저장하는 방식으로 처리
- 토큰 요청 URL: 카카오 https://kauth.kakao.com/oauth/token, 네이버 https://nid.naver.com/oauth2.0/token, 구글 https://oauth2.googleapis.com/token
카카오 토큰 요청 시 카카오에서 요구하는 정확한 key 이름(grant_type, client_id 등)을 사용하여 작업
- 사용자 정보 요청 URL: 카카오 https://kapi.kakao.com/v2/user/me, 네이버 https://openapi.naver.com/v1/nid/me, 구글 https://www.googleapis.com/oauth2/v2/userinfo
네이버 로그인 시 사용자 정보를 불러오기 위해 응답 구조를 res.data.response의 형태로 수정하여 작업
- 공통 사항: 사용자 정보 요청 시 access_token을 헤더에 포함시키고, 카카오 id는 숫자 타입이므로, 저장 시 String() 변환하여 타입 일치 처리

③ DB 저장 처리

- 사용자 ID 기준으로 DB 조회
- 존재하지 않는 경우 insertOne으로 신규 추가
- 사용자 정보 + 토큰을 포함한 응답 반환

④ Vercel 배포 및 환경변수 등록
- 깃허브에 서버 코드 업로드 후 Vercel을 통해 배포
- 배포 환경에 맞춰 각 소셜 플랫폼의 서비스 URL 및 Redirect URL 수정
- .env 파일에 등록한 key를 Vercel 프로젝트 설정에 추가하여 적용

1-2) 로그인 (프론트)

① 애플리케이션 등록 및 환경설정
- 서버와 동일하게 프론트도 소셜 로그인 연동을 위해 각 플랫폼에서 앱 설정을 적용
- 개발 및 배포 환경 전환 시 서비스 URL과 Redirect URI를 함께 수정

② 로그인 기능 개발
- 버거 메뉴 또는 마이페이지 내 버튼을 통해 로그인 페이지 진입할 수 있도록 작업
- 소셜 로그인 버튼 클릭 시 각 소셜 로그인 함수 실행
- 각 플랫폼 인증 URL 생성 후 window.location.href로 이동하고 공통 요청 파라미터 값인 client_id, redirect_uri, response_type 등 설정 후 네이버, 구글은 추가로 state 값 설정
- 각 플랫폼 별 리디렉션 처리 컴포넌트는 URL 파라미터를 읽어와 서버에 요청하고 서버로부터 받은 토큰, 유저 정보를 sessionStorage에 저장하고, 로그인한 플랫폼도 함께 저장하여 로그아웃 시 사용 완료 후 홈으로 이동하도록 작업
- 로그인 하지 않고 둘러보기 버튼 클릭 시 홈으로 페이지 이동하도록 작업

③ Vercel 배포 및 환경변수 등록
- 깃허브에 서버 코드 업로드 후 Vercel을 통해 배포
- 배포 환경에 맞춰 각 플랫폼 설정 페이지에서도 서비스 URL 및 Redirect URL 수정
- .env 파일에 등록한 key를 Vercel 프로젝트 설정에 추가하여 적용

2-1) 로그아웃 (서버)

① naver.js에서 클라이언트 ID, 시크릿, 프론트로부터 전달받은 토큰을 이용해 네이버에 토큰 삭제 요청

② 요청 URL: https://nid.naver.com/oauth2.0/token, 요청 방식: GET, 요청 파라미터: grant_type=delete, client_id, client_secret, access_token
*service_provider 파라미터는 외부 서비스 연동 로그인 시만 필요하여 생략 가능.

2-2) 로그아웃 (프론트)

① 로그아웃 버튼(버거 메뉴, 마이페이지) 클릭 시, 로그아웃 확인 팝업 및 완료 팝업 생성 후 메인으로 이동하도록 작업

② 마이페이지에 노출된 사용자 정보는 세션 정리 후 제거

③ 각 플랫폼 공통 기능
- 세션 스토리지 사용자 정보 삭제
- 로그인 플랫폼 별 분기 처리
- 로그아웃 완료 후 팝업 호출을 위한 콜백 지원
- 플랫폼별 요구사항에 맞게 외부 URL 호출 또는 토큰 만료 처리

④ 각 플랫폼별 로그아웃 처리
- 카카오: Kakao Developers 설정에서 Logout Redirect URI 등록 필요, access_token 확인 후, https://kauth.kakao.com/oauth/logout + client_id + redirect_uri로 리디렉션
- 네이버: 저장된 access_token을 서버(/naver/logout)로 전달하여 토큰 만료 처리, 세션 초기화 후 콜백(onAfterLogout) 실행
- 구글: 세션 초기화만 수행, 콜백(onAfterLogout)을 통한 추가 처리 지원

⑤ 실제 사용 방법
- My.jsx, Burger.jsx에서 handleLogoutConfirm() 함수 호출하여 세션 내 provider 값을 기준으로 로그아웃 분기 처리
- 카카오는 리디렉션 로그아웃으로 콜백 사용 불가, 네이버/구글은 콜백 통해 완료 팝업 후 메인 이동 처리

3) 마이페이지

① 비로그인 시, 일부 페이지 접근 제한
- sessionStorage에 저장된 토큰값을 기반으로 로그인 여부를 확인하여 isLoggedIn 변수에 저장
- 추천 일정, QnA, 전화 걸기를 제외한 메뉴 클릭 시 isLoggedIn의 값이 true일 경우에만 해당 페이지로 이동되고, 그렇지 않을 경우 로그인 요청 팝업을 띄워 로그인 유도

② 현재 사용자가 추가한 여행 수 표시
- 로그인 시 서버에 user.id를 기준으로 /plan GET 요청을 보내 응답 데이터의

개수(res.data.length)를 myTrip 상태에 저장
- '나의 여행 보기' 메뉴 우측에 여행 수를 표시하며, 여행이 없을 경우 해당 메뉴 클릭 시 여행 추가 요청 팝업을 띄움

4-1) 체크리스트 (서버)

① 1명의 사용자는 1개의 문서를 가지게 된 후, 그 문서 안에 allList 배열로 여러 개의 체크리스트를 보관
② 사용자의 모든 체크리스트 조회
- GET "/user/:userId" 특정 사용자(userId)의 allList 배열을 체크리스트 배열로 반환
③특정 체크리스트 항목 조회
- GET "/user/:userId/:checkId" 특정 사용자(userId)의 특정 체크리스트(checkId)를 체크리스트 객체로 반환
④ 새 체크리스트 추가
- POST / 특정 체크리스트를 새 객체로 덮어쓰기
- 데이터가 존재하면 allList에 push / 데이터가 없으면 새 문서 생성
⑤ 기존 체크리스트 추가
- PUT / 새로운 체크리스트를 사용자의 allList에 추가
- checkId 기준으로 해당 체크리스트 찾아 전체 교체 / 수정 후 allList 저장
⑥ 기존 체크리스트 삭제
- PUT / del 특정 체크리스트 삭제
- checkId가 일치하지 않는 리스트만 남겨 새로운 allList 저장
⑦ 여행 일정(plan)과 연결된 체크리스트 삭제
- PUT /plan/del 여행 일정 삭제 시 연결된 체크리스트도 함께 삭제
- planId가 일치하지 않는 리스트만 남겨 새로운 allList 저장

4-2) 체크리스트 (프론트)

[리스트]
① Session Storage에 저장된 userId로 요청하여 데이터 조회
② 리스트 추가 버튼 클릭 시 GetTripPopup 팝업 노출
③ 팝업 select box는 체크리스트가 연결되지 않은 여행 타이틀만 표시하고 선택없이 확인 버튼 클릭 시 알림 팝업으로 표시
④ 여행을 선택하거나 여행 없이 체크리스트 추가 가능
⑤ 체크리스트 삭제를 하기 위해서는 스와이프 형식으로 삭제 버튼을 제공하고 버튼 클릭 시 확인 팝업이 나타나 완료됐다는 팝업이 노출되도록 작업
⑥ 삭제 완료 후 checkData 상태 갱신 및 삭제 완료 팝업 표시
⑦ 리스트 항목 클릭 시 ListItem 컴포넌트 내 navigate를 사용하여 디테일 페이지로 이동
[디테일 페이지]
① 새 체크리스트와 기존 체크리스트 구분 조건은 Zustand의 isEdit 값으로 구분하여 작업
② 새 체크리스트일 경우 state로 보내주는 데이터에 'isEdit: true' 포함.
③ 편집 모드: 편집 버튼 클릭 시 enterEditMode()를 호출하여 편집 모드 진입, 완료 버튼 클릭 시 변경 사항이 있으면 저장 팝업 오픈과 exitEditMode()를 호출하고 변경사항이 없으면 exitEditMode()만 호출하도록 작업
④ 항목 체크: CheckItem 컴포넌트에서 체크박스 클릭 시, 리스트 내부 항목의 checked 값 토글
⑤ 항목 추가: 하단 AddCheckItem 컴포넌트 노출 (isEdit === true일 때만), { name: 입력값, checked: false } 형태로 리스트에 추가
⑥ 항목 삭제: 스와이프로 삭제 버튼 노출 시킨 후 버튼 클릭 시 삭제 대상 저장 후 팝업 생성, 확인 버튼 클릭 시 화면에서 항목 삭제
⑦ 저장: 새 체크리스트는 서버의 /check 경로에 POST로 요청, 기존 체크리스트는 서버의 /check 경로에 PUT 으로 요청 후 originData에 갱신되도록 작업
⑧ 페이지 나가기: 닫기 버튼 클릭 시 변경사항이 있는지 확인한 후 데이터가 저장되지 않는다는 경고 팝업으로 알려준 후 팝업 내 확인 버튼 클릭 시 원래 데이터로 복구 및 편집 모드를 종료하도록 작업

5) 나의활동 (작성한 게시물 및 댓글 목록)

① 서버로부터 데이터 가져오기
- 세션에 저장된 사용자 정보 중 ID를 추출하여 userId 변수에 저장
- 게시물은 useEffect 내에 userId를 이용하여 /post 경로에 GET(/user/:userId) 요청 후 응답 데이터를 postData 상태에 저장
- 댓글은 /reply 경로에 GET 요청 후 데이터를 cmtOriginData에 저장
- 댓글 데이터에서 postId만 추출한 후 Set을 사용하여 중복 제거 작업

	- 중복 제거한 데이터를 기준으로 각 게시물 상세 데이터를 요청하여 Promist.all을 통해 여러 작업을 동시에 처리 - 모든 요청이 완료된 후 게시물 상세 데이터와 댓글 데이터를 병합하여 cmtData로 저장 ② 화면 렌더링 및 상세페이지 이동 - renderContent() 함수에서 탭 메뉴 선택값에 따라 데이터 분기 렌더링 진행 (ex. case 0: 게시물(postData), case 1: 댓글(cmtData)) - 데이터는 createdAt 기준으로 최신순으로 정렬하고 데이터가 없을 시 NoWritePost 또는 NoWriteReply 컴포넌트를 출력하여 데이터 없음을 표시 - 각 콘텐츠 클릭 시 localStorage에 저장한 후, 좋아요 여부 확인을 위해 /like 경로로 GET 요청 후 현재 postId 포함 여부를 체크한 후 페이지 이동 ③ 출력 형식 - 게시물 : 대표 이미지와 작성일(yyyy.MM.dd) 포맷으로 출력되도록 작업 - 댓글: 작성일을 yyyy.MM.dd hh:mm 포맷으로 출력되도록 작업 **6) 좋아요 페이지** ① 사용자가 누른 좋아요 데이터를 가져오기 위해 장소 카테고리별 데이터를 요청 및 통합 ② 커뮤니티 게시물의 좋아요는 서버에 데이터를 요청하여 사용자별 좋아요한 게시물 조회 ③ 장소, 게시물 컴포넌트를 활용해 탭 전환이 가능하며, 데이터가 없을 경우 없다는 안내 문구가 뜨도록 작업 **7) QnA** ① useState로 열려있는 질문 인덱스를 관리하여 질문 클릭 시, 해당 항목의 답변을 열고 재 클릭 시 닫히도록 구현 **8) 전화연결** ① onClick 이벤트를 추가하여 팝업을 생성한 후 통화 버튼 클릭 시 함수가 실행되도록 작업 ② 사용자의 브라우저 정보를 얻어 모바일 기기일 경우 전화 기능을 활성화 시키고, 모바일 기기가 아닌 경우 alert으로 웹 브라우저 환경일 경우 통화 연결이 불가하다는 문구를 띄워 알림 • **06 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 **테스트** • 개발 작업을 마친 후, 4월21일부터~ 4월23일까지 배포 후 테스트 (중간 통합 후 테스트 제외) • 서로 작업한 파일들을 깃허브를 통해 통합 후 Vercel 배포 • 공통 컴포넌트 작업 및 개별 작업이 일정 수준 이상 완료될 때마다 중간 통합을 진행하였으며, 총 6차 통합까지 진행 • 중간 통합 후 작업 내용을 상호 검토하고, 수정 사항을 적용하는 과정을 반복하며 프로젝트의 완성도 높임 **배포** • https://jeju-trip-eosin.vercel.app/
산출물 내역	기획, 화면설계, 디자인, PPT 작업파일 pdf 추출 후 해당 폴더에 저장 • ./jeju-trip/work/2차프로젝트_C팀_01기획,화면설계.pdf • ./jeju-trip/work/2차프로젝트_C팀_02디자인.pdf • ./jeju-trip/work/2차프로젝트_C팀_03PPT.pdf
장·단점 및 특이사항	-