

PORTFOLIO

아기용품 쇼핑몰

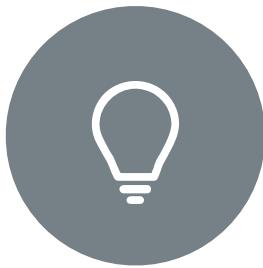
김 관 수
kgs940823@gmail.com
010.2166.8175

Contents



Introduce

팀원 소개
개발 환경



Requirement

Site Map
요구사항 정의
Mock up
결과 사진



DBMS System

ERD
테이블 명세서



Code

주요 개발 기능

Introduce

팀원 소개

김 관 수

 kgs940823@gmail.com

FronEnd

공통(Header, Nav, Footer)
메인페이지
상품 목록 조회
상품 검색 / 결과
회원가입
회원정보 수정
아이디 찾기 / 결과
비밀번호 찾기

BackEnd

상품 목록 / 결과 페이지
상품 정렬
회원가입 페이지
아이디 찾기 페이지

신 희 선

 shieun8@gmail.com

FronEnd

마이페이지
장바구니
주문 내역
주문 상세내역
상품 상세페이지
결제 페이지
로그인

BackEnd

장바구니
상품 상세페이지
주문내역
결제 페이지

장 옥 기

 reinaprc@gmail.com

FronEnd

나의 게시물 목록
공지사항
Q&A
게시글 작성

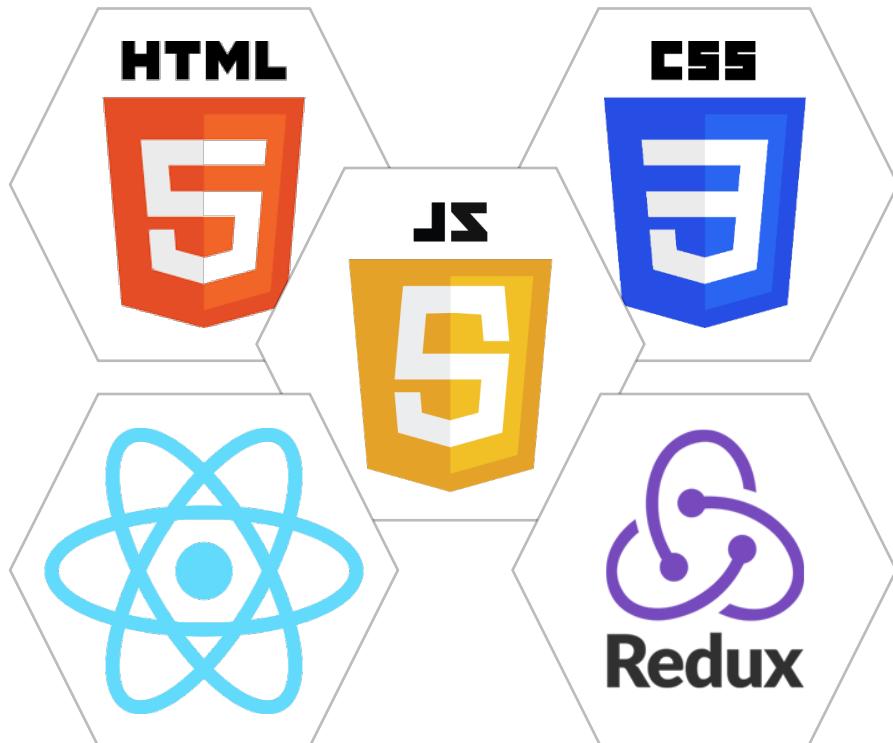
BackEnd

공지사항
로그인

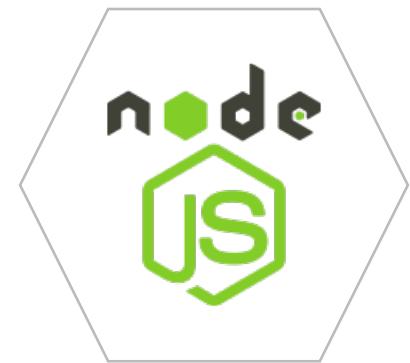
Introduce

개발 환경

FrontEnd



BackEnd

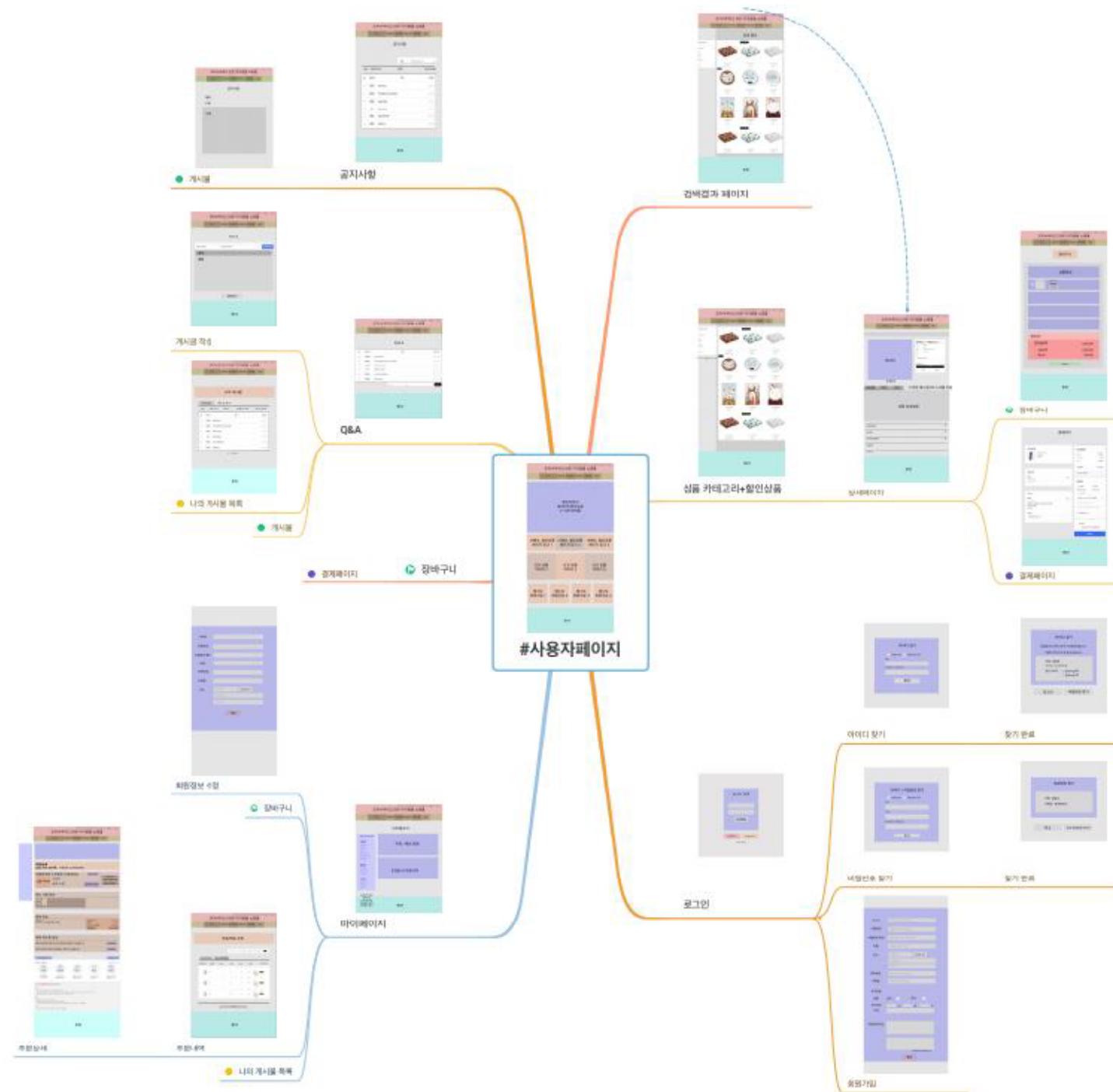


Database



Requirement

Site Map



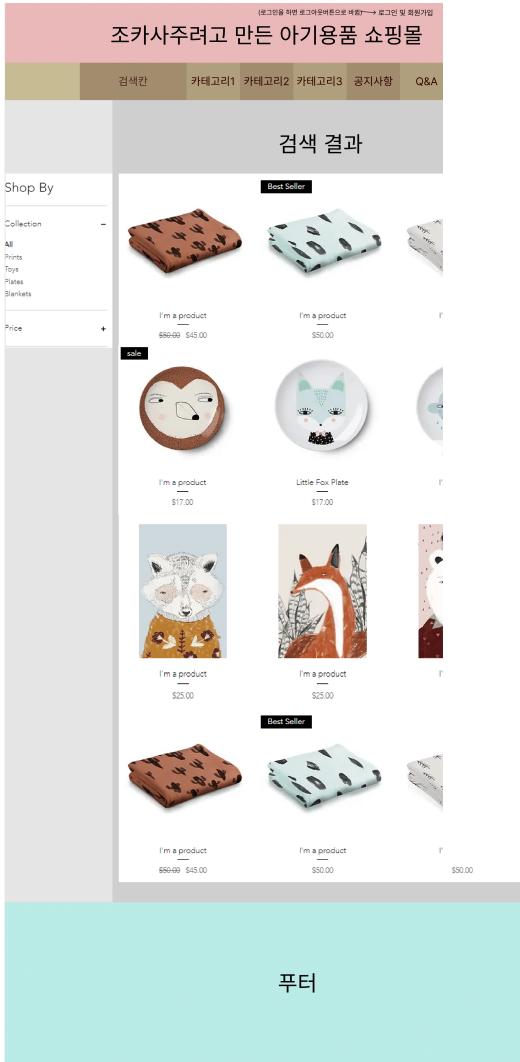
Requirement

요구사항 정의(WBS)

5조 Little BEE (아기용품 쇼핑몰 요구사항 명세서)								Enter Project Range					
								Start Date	End Date				
								22-11-14	22-12-16				
Zoom (enter 1 for Daily, 7 for Weekly)--->								1				#	#
1depth	2depth	3depth	4depth	상세 업무	담당	시작일	종료일	기간	진척률	상태			
메인 페이지				공통 - { header (로그인, 마이페이지, 장바구니), footer,	김관수	2022-11-14	2022-11-20	5	100 %	완료			
	카테고리별 상품목록			세부 카테고리,	김관수	2022-11-21	2022-11-27	5	100 %	완료			
		상품별 상세페이지		제품 이미지 3~4개(Image slide), 옵션/수량 선택, 바로결제, 장바구	김관수	2022-11-28	2022-12-05	6	100 %	완료			
			장바구니	상품 선택/구매/삭제를 위한 CheckBox,	신희선			0	100 %	완료			
			결제페이지	상품명 클릭시 상세페이지 이동, 주문자/배송정보 수정버튼	김관수	2022-12-06	2022-12-12	5	100 %	완료			
마이페이지				주문/배송현황, 최근 주문내역, 더보기 버튼 클릭 시 주문내역 이동,	신희선	2022-11-14	2022-11-18	5	100 %	완료			
		장바구니		상품 선택/구매/삭제를 위한 CheckBox,	신희선	2022-11-19	2022-11-25	5	100 %	완료			
			회원정보 수정	입력값 검사, 우편번호찾기API, 수정 완료 시 alert, 뒤로가기	강태현	2022-11-14	2022-11-17	4	100 %	완료			
			주문내역	주문내역/취소,교환,반품	신희선	2022-11-28	2022-12-06	7	100 %	완료			
			주문상세	주문내역 삭제, 장바구니에 담기(추가 및 이동여부 alert), 배송조회,	신희선	2022-12-07	2022-12-14	6	100 %	완료			
로그인 페이지				로그인, 아이디 비밀번호 찾기, 회원가입, 관리자 로그인(버튼만)	강태현	2022-11-17	2022-11-18	2	100 %	완료			
			회원가입	회원가입 필수입력값, 입력값 검사 설정, 우편번호API,	강태현	2022-11-21	2022-11-24	4	100 %	완료			
			아이디 찾기	이메일로 찾기, 휴대폰번호로 찾기	강태현	2022-11-25	2022-11-28	2	100 %	완료			
			찾기 완료	일치하는 아이디에서 뒤 4자리 가리고 띄움,	강태현	2022-11-28	2022-12-01	4	100 %	완료			
			비밀번호 찾기	이메일로 찾기, 휴대폰번호로 찾기	강태현	2022-12-02	2022-12-02	1	100 %	완료			
			찾기 완료	취소버튼, 입력된 이메일/휴대폰번호로 임시 비밀번호 전송	강태현	2022-12-05	2022-12-06	2	100 %	완료			
공지사항				제목 / 카테고리 / 날짜 필터 선택, 검색어 입력	장옥기	2022-11-14	2022-11-16	3	100 %	완료			
		공지 게시글		공지 내용 표시	장옥기	2022-11-17	2022-11-20	2	100 %	완료			
Q&A				제목/작성자/내용 필터 선택, 검색어 입력, 글쓰기 버튼	장옥기	2022-11-21	2022-11-24	4	100 %	완료			
		Q&A 게시글		QnA 작성 내용 표시	장옥기	2022-11-25	2022-11-27	1	100 %	완료			
		나의 게시글 목록		내가 쓴 QnA, 내가 쓴 후기, QnA 목록 선택 시 게시글 페이지로 이동	장옥기	2022-11-28	2022-11-30	3	100 %	완료			
			게시글 작성	문의 유형 선택, 구매내역 상품들을 팝업창으로 표시 및 선택, 이전,	장옥기	2022-12-01	2022-12-09	7	100 %	완료			
검색결과 페이지				검색칸, 검색버튼, 검색 결과 개수 표시	김관수	2022-12-13	2022-12-16	4	100 %	완료			
장바구니				상품 선택/구매/삭제를 위한 CheckBox,	신희선			0	0 %	대기			
		결제페이지		상품명 클릭시 상세페이지 이동, 주문자/배송정보 수정버튼	김관수			0	0 %	대기			

Requirement 화면 설계(Mockup)

✓ Figma를 사용하여 Mockup 작업 진행



This sequence of three screenshots illustrates the checkout process:

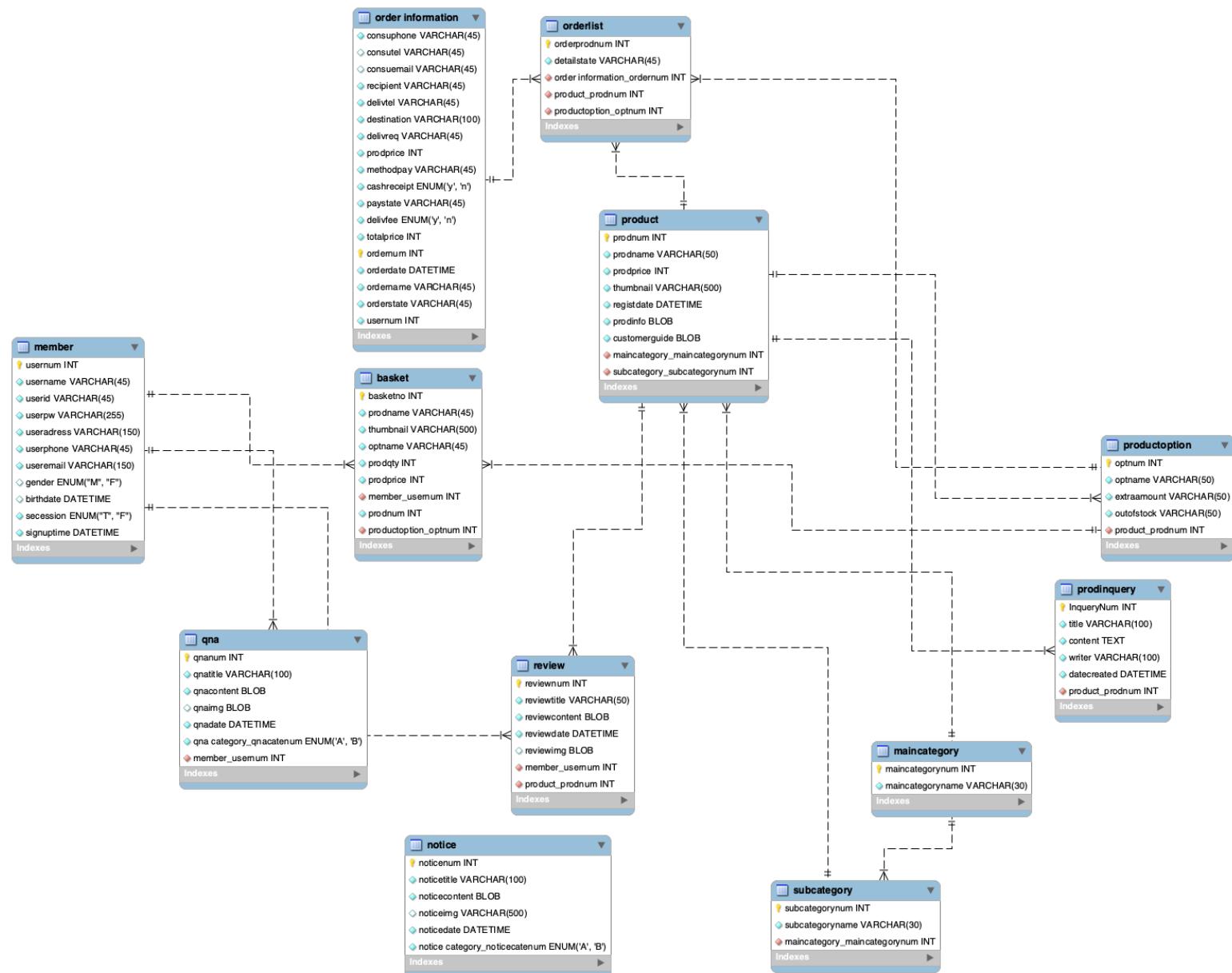
- Step 1: 결제하기** (Payment) - Shows a summary of the purchase: Daily Facial Soap (18,000원). It includes a note about payment methods and a QR code.
- Step 2: 조카사주려고 만든 아기용품 쇼핑몰** (Baby Product Shopping Mall) - Shows the product details: Daily Facial Soap (18,000원), shipping information (송길동), and payment method (Credit Card / Debit Card).
- Step 3: 결제 정보** (Payment Information) - Shows the payment method selected: Credit Card / Debit Card, and the total amount: 18,000원.

A registration form with fields for:

- 아이디**: 아이디(을)를 입력해주세요.
- 비밀번호**: 비밀번호를 입력해주세요.
- 비밀번호 확인**: 비밀번호를 다시 입력해주세요.
- 이름**: 이름을 입력해주세요.
- 주소**: 우편번호 (우편번호 찾기), 도로명주소, 상세주소.
- 전화번호**: 전화번호를 입력해주세요.
- 이메일**: 이메일을 입력해주세요.
- 추가정보**: 성별 (남자, 여자), 생년월일 (년, 월, 일), 지역.
- 개인정보수집**: A large text area for privacy policy, with a checkbox at the bottom right labeled "확인".

DBMS System

ERD(Entity Relationship Diagram)



DBMS System

테이블 명세서

테이블 이름		member					
테이블 설명		사용자					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	usernum	int	NOT NULL	PRI	auto_increment		사용자번호
2	username	varchar(45)	NOT NULL				이름
3	userid	varchar(45)	NOT NULL				아이디
4	userpw	varchar(255)	NOT NULL				비밀번호
5	useraddress	varchar(150)	NOT NULL				주소
6	userphone	varchar(45)	NOT NULL				전화번호
7	useremail	varchar(150)	NOT NULL				이메일
8	gender	enum('M','F')	NULL				성별
9	birthdate	datetime	NULL				생년월일
10	secession	enum('T','F')	NOT NULL				탈퇴여부
11	signuptime	datetime	NOT NULL				가입일시

DBMS System

테이블 명세서

테이블 이름		product					
테이블 설명		상품					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	prodnum	int	NOT NULL	PRI	auto_increment		상품 번호
2	prodname	varchar(50)	NOT NULL				상품명
3	prodprice	int	NOT NULL				상품금액
4	thumbnail	varchar(500)	NOT NULL				썸네일이미지
5	registdate	datetime	NOT NULL				상품 등록날짜
6	prodinfo	blob	NOT NULL				상품 상세 내용 이미지
7	customerguide	blob	NOT NULL				상세페이지 내용(배송,반품)
8	maincategory_maincategorynu	int	NOT NULL	MUL			
9	subcategory_subcategorynum	int	NOT NULL	MUL			

테이블 이름		prodinquiry					
테이블 설명		상세 페이지 안의 상품 문의					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	InqueryNum	int	NOT NULL	PRI	auto_increment		문의 번호
2	title	varchar(100)	NOT NULL				문의 제목
3	content	text	NOT NULL				문의 내용
4	writer	varchar(100)	NOT NULL				작성자
5	datecreated	datetime	NOT NULL				작성일
6	product_prodnum	int	NOT NULL	MUL			

테이블 이름		productoption					
테이블 설명		상품 옵션					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	optnum	int	NOT NULL	PRI	auto_increment		옵션 번호
2	optname	varchar(50)	NOT NULL				옵션 이름
3	extraamount	varchar(50)	NOT NULL				추가 금액
4	outofstock	varchar(50)	NOT NULL				품절 여부
5	product_prodnum	int	NOT NULL	MUL			

DBMS System

테이블 명세서

테이블 이름		maincategory					
테이블 설명		상품 카테고리(대분류)					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	maincategorynum	int	NOT NULL	PRI	auto_increment		카테고리 번호(대분류)
2	maincategoryname	varchar(30)	NOT NULL				카테고리 이름(대분류)

테이블 이름		subcategory					
테이블 설명		상품 카테고리(소분류)					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	subcategorynum	int	NOT NULL	PRI	auto_increment		카테고리 번호(소분류)
2	subcategoryname	varchar(30)	NOT NULL				카테고리 이름(소분류)
3	maincategory_maincategorynu	int	NOT NULL	MUL			

DBMS System

테이블 명세서

테이블 이름		basket					
테이블 설명		장바구니 데이터					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	basketno	int	NOT NULL	PRI	auto_increment		
2	prodname	varchar(45)	NOT NULL				상품명
3	thumbnail	varchar(500)	NOT NULL				썸네일이미지
4	optname	varchar(45)	NOT NULL				옵션이름
5	prodqty	int	NOT NULL				상품수량
6	prodprice	int	NOT NULL				상품금액(수량*금액)
7	member_usernum	int	NOT NULL	MUL			사용자번호
8	prodnum	int	NOT NULL				상품번호
9	productoption_optnum	int	NOT NULL	MUL			상품옵션

테이블 이름		orderlist					
테이블 설명		주문 상품 리스트					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	orderprodnum	int	NOT NULL	PRI	auto_increment		
2	detailstate	varchar(45)	NOT NULL				상품별 상태
3	order information_ordernum	int	NOT NULL	MUL			주문번호
4	product_prodnum	int	NOT NULL	MUL			상품번호
5	productoption_optnum	int	NOT NULL	MUL			상품옵션

DBMS System

테이블 명세서

테이블 이름		order information					
테이블 설명		주문정보 및 결제정보					
No	필드명	데이터타입	널허용	키	옵션	기본값	설명
1	consuphone	varchar(45)	NOT NULL				주문자 휴대폰번호
2	consutel	varchar(45)	NULL				주문자 전화번호
3	consuemail	varchar(45)	NULL				주문자 이메일
4	recipient	varchar(45)	NOT NULL				수령인
5	delivtel	varchar(45)	NOT NULL				수령인 연락처
6	destination	varchar(100)	NOT NULL				배송주소지
7	delivreq	varchar(45)	NOT NULL				배송요청사항
8	prodprice	int	NOT NULL				결제금액(배송비미포함)
9	methodpay	varchar(45)	NOT NULL				결제수단
10	cashreceipt	enum('y','n')	NOT NULL				현금영수증 신청 여부
11	paystate	varchar(45)	NOT NULL				결제상태
12	delivfee	enum('y','n')	NOT NULL				배송비 여부
13	totalprice	int	NOT NULL				총 결제금액(배송비포함)
14	ordernum	int	NOT NULL	PRI	auto_increment		주문번호
15	orderdate	datetime	NOT NULL				주문일자
16	ordername	varchar(45)	NOT NULL				주문명
17	orderstate	varchar(45)	NOT NULL				주문상태
18	usernum	int	NOT NULL				사용자번호

DBMS System

테이블 명세서

테이블 이름		review						
테이블 설명		상품에 대한 리뷰를 저장하는 테이블						
No	필드명	데이터타입	널허용	키	옵션	기본값	설명	
1	reviewnum	int	NOT NULL	PRI	auto_increment		글 번호	
2	reviewtitle	varchar(50)	NOT NULL				리뷰 제목	
3	reviewcontent	blob	NOT NULL				리뷰 내용	
4	reviewdate	datetime	NOT NULL				리뷰 작성 날짜	
5	reviewimg	blob	NULL				리뷰 이미지	
6	member_usernum	int	NOT NULL	MUL			유저 번호	
7	product_prodnum	int	NOT NULL	MUL			상품 번호	

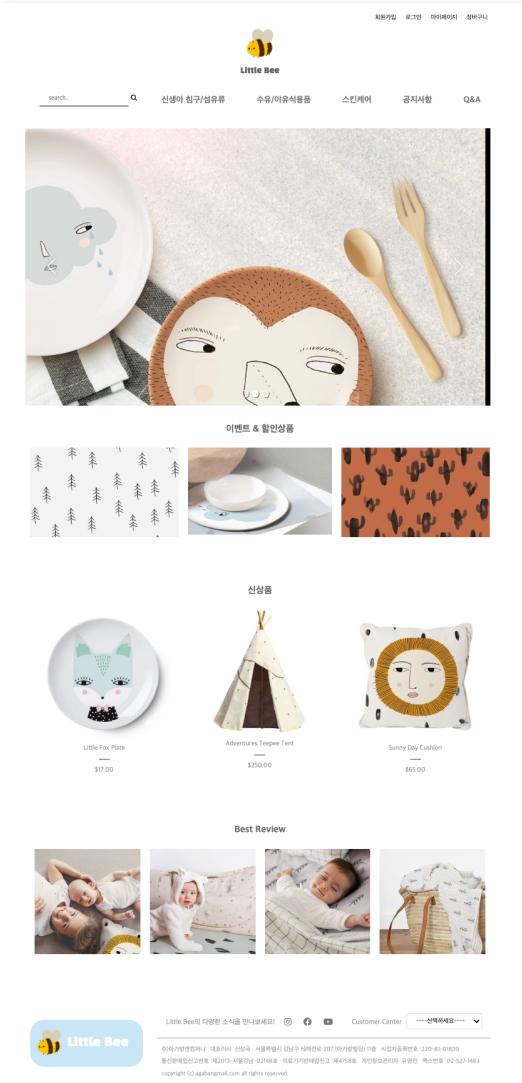
테이블 이름		notice						
테이블 설명		공지사항 내용을 저장하는 테이블						
No	필드명	데이터타입	널허용	키	옵션	기본값	설명	
1	noticenum	int	NOT NULL	PRI	auto_increment		공지글 번호	
2	noticetitle	varchar(100)	NOT NULL				공지 제목	
3	noticecontent	blob	NOT NULL				공지 내용	
4	noticeimg	varchar(500)	NULL				공지 이미지	
5	noticedate	datetime	NOT NULL				공지 작성 날짜	
6	notice category_noticecatenum	enum('A','B')	NOT NULL				공지 카테고리	

테이블 이름		qna						
테이블 설명		qna 내용을 저장하는 테이블						
No	필드명	데이터타입	널허용	키	옵션	기본값	설명	
1	qnanum	int	NOT NULL	PRI	auto_increment		Q&A 글 번호	
2	qnatitle	varchar(100)	NOT NULL				Q&A 제목	
3	qnacontent	blob	NOT NULL				Q&A 내용	
4	qnaimg	blob	NULL				Q&A 이미지	
5	qnadate	datetime	NOT NULL				Q&A 작성 날짜	
6	qna category_qnacatenum	enum('A','B')	NOT NULL				Q&A 카테고리	
7	member_usernum	int	NOT NULL	MUL			유저 번호	

Code

주요 개발 기능_메인 페이지

PAGE



FrontEnd

```

1  const Main = memo(() => {
2    const navigate = useNavigate();
3
4    const onClickEventProduct = useCallback((e) => {
5      navigate('/')
6    })
7    const images = [
8      {url: MainSlider1},
9      {url: MainSlider2},
10     {url: MainSlider3}
11   ]
12   const subImg = [
13     {img: MainSub1},
14     {img: MainSub2},
15     {img: MainSub3}
16   ]
17   const newProductImg = [
18     {img: newProduct1, title: 'Little Fox Plate', price: '$17.00'},
19     {img: newProduct2, title: 'Adventures Teepee Tent', price: '$250.00'},
20     {img: newProduct3, title: 'Sunny Day Cushion', price: '$65.00'}
21   ]
22   const ReviewImg = [
23     {img: Review1},
24     {img: Review2},
25     {img: Review3},
26     {img: Review4}
27   ]

```

이미지 및 이미지 관련 내용을 임의의 배열에 저장

```

28   return (
29     <MainContainer>
30       <NavLink>
31         <ImageSlider
32           width="980px"
33           height="591px"
34           images={images}
35           showBullets={true}
36           showMaws={true}
37           autoPlay={true}
38           autoPlayDelay={2.0}
39           loop={true}
40           style={{
41             margin: 'auto',
42             objectFit: 'cover',
43           }}
44         />
45       <NavLink>
46         <h2>이벤트 & 할인상품</h2>
47         <EventProduct onClick={onClickEventProduct}>
48           {subImg.map((v, i) => {
49             return (
50               <li key={i}>
51                 <a href={`#${i}`}>
52                   <img src={v.img} />
53                 </a>
54               </li>
55             )
56           )}
57         </EventProduct>
58         <h2>신상품</h2>
59         <NewProduct>
60           {newProductImg.map((v, i) => {
61             return (
62               <li key={i}>
63                 <a href={`#${i}`}>
64                   <img src={v.img} title={v.title} />
65                   <p className='title'>{v.title}</p>
66                   <div className='line'></div>
67                   <p className='price'>{v.price}</p>
68                 </a>
69               </li>
70             )
71           )}
72         </NewProduct>
73         <h2>Best Review</h2>
74         <ReviewContainer>
75           {ReviewImg.map((v, i) => {
76             return (
77               <li key={i}>
78                 <a href={`#${i}`}>
79                   <img src={v.img} />
80                 </a>
81               </li>
82             )
83           )}
84         </ReviewContainer>
85       </MainContainer>
86     );
87   );

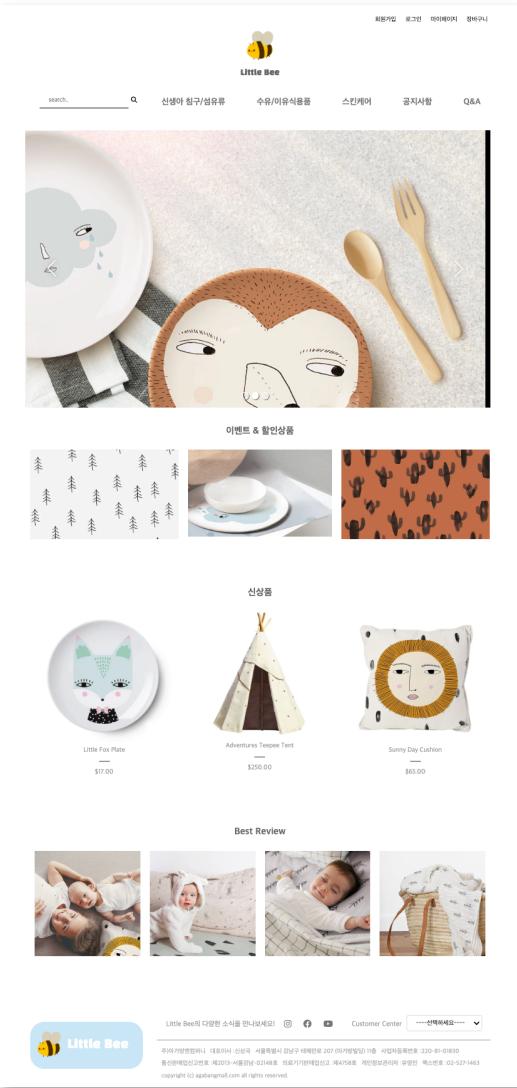
```

배열에 담은 이미지 및 내용 출력

Code

주요 개발 기능_메인 페이지

PAGE



FrontEnd

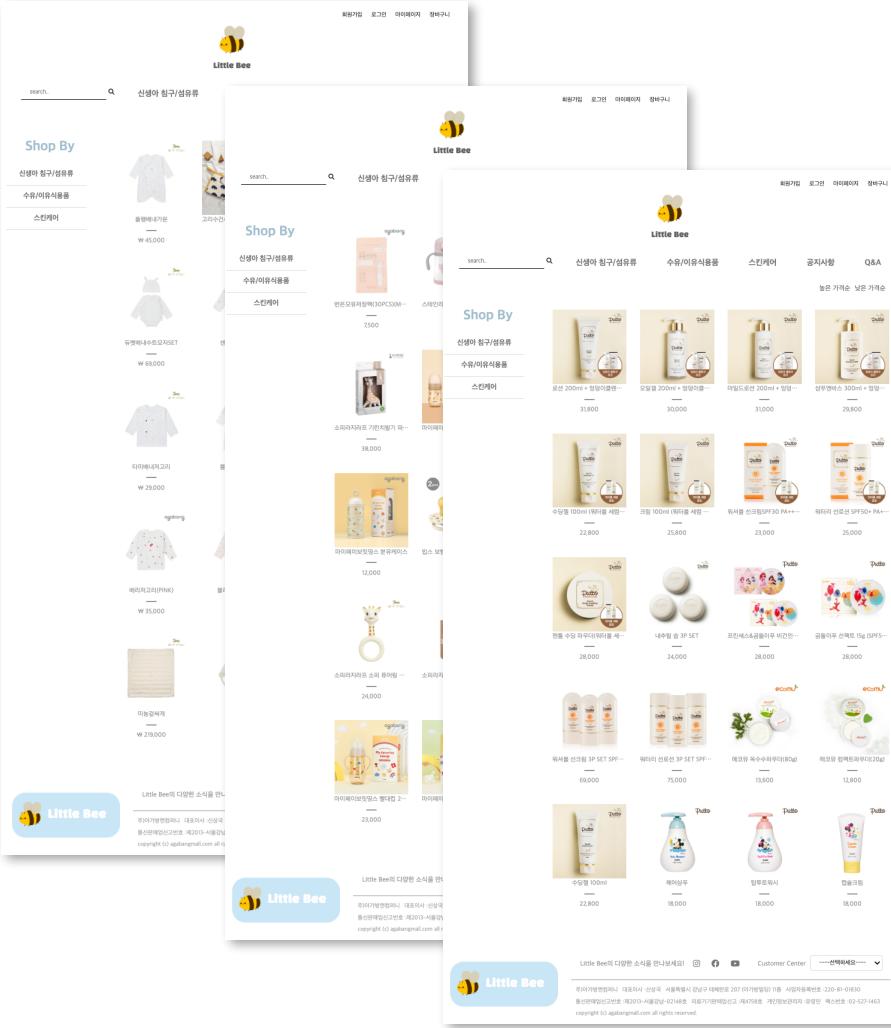
```
47 const NewProduct = styled.ul`  
48   list-style: none;  
49   display: flex;  
50   margin: auto;  
51   max-width: 980px;  
52  
53   li {  
54     width: 33.3%;  
55  
56     padding: 10px;  
57     box-sizing: border-box;  
58  
59     &:first-child {  
60       padding-left: 0;  
61     }  
62     &:last-child {  
63       padding-right: 0;  
64     }  
65  
66     & a {  
67       text-decoration: none;  
68  
69       & img {  
70         width: 100%;  
71       }  
72       & p {  
73         font-size: 14px;  
74         text-align: center;  
75         color: #999;  
76         transition: 0.3s;  
77     }  
78     & .title {  
79       &:hover {  
80         color: #eee;  
81       }  
82     }  
83     & .line {  
84       width: 20px;  
85       border: 1px solid #999;  
86       margin: 15px auto;  
87     }  
88     & a {  
89       & img {  
90         width: 100%;  
91         margin-bottom: 50px;  
92       }  
93     }  
94   };  
95  
96 const ReviewContainer = styled.ul`  
97   max-width: 980px;  
98   display: flex;  
99   margin: auto;  
100  li {  
101    width: 25%;  
102    padding: 10px;  
103    a {  
104      /* padding: 10px 0 10px 10px; */  
105      box-sizing: border-box;  
106      img {  
107        width: 100%;  
108      }  
109    }  
110  };  
111  
112  `;  
113  
114 export {MainContainer, EventProduct, NewProduct, ReviewContainer};
```

styled-components를 사용하여 CSS 코드 작성

Code

주요 개발 기능_상품 목록 조회(Main Category)

PAGE



MainCategory 선택 시 상품 전체 목록 조회(화면 최초 마운트 시 목록 조회)

FrontEnd

1. Product

```

1 const location = useLocation();
2 const CategoryLoc = location.pathname.substring(10, 12);
3 // console.log(CategoryLoc);

4
5 // 최초 마운트시 목록조회
6 useEffect(() => {
7   dispatch(getList({
8     category: CategoryLoc,
9   })).then(()=>{
10     setInit(true);
11   })
12 }, [init]);

```

dispatch로 Slice의 getList 호출



2. ProductSlice

```

1 /**
2  * 리스트 조회를 위한 비동기 함수 */
3 export const getList = createAsyncThunk('ProductSlice/getList', async (payload, { rejectWithValue }) => {
4   let result = null;
5
6   try {
7     const response = await axios.get(`${API_URL}/${payload?.category}`);
8     result = response.data;
9   } catch (err) {
10   console.group('ProductSlice.getList');
11   console.error(err);
12   console.groupEnd();
13   result = rejectWithValue(err.response);
14 }
15
16 return result;
17 });

```

Slice에서 BackEnd로 데이터 요청

```

1 import styled from "styled-components";
2
3 const ProductsContainer = styled.div` 
4   width: 980px;
5   display: flex;
6   justify-content: center;
7   margin: auto;
8
9   &.categoryBox {
10   list-style: none;
11   width: 20%;
12   margin-top: 30px;
13
14   & h1 {
15     font-size: 30px;
16     text-align: center;
17     color: #abecce;
18     font-weight: bold;
19     margin-bottom: 20px;
20   }
21
22   &.link {
23     display: block;
24     width: 179px;
25     height: 48px;
26     background-color: #fff;
27     border-bottom: 1px solid #ddd;
28     line-height: 48px;
29     text-align: center;
30     font-weight: 300;
31     color: #777;
32     text-decoration: none;
33
34     &:hover {
35       background-color: #aaa;
36       color: #fff;
37     }
38   }
39   &.title {
40     font-weight: 700;
41   }
42
43   &.menuItem {
44     width: 200px;
45   &.sub {
46     list-style: none;
47     margin: 0;
48     padding: 0;
49     z-index: 1000;
50     max-height: 0;
51     overflow: hidden;
52     transition: max-height 180ms ease-out;
53   }
54 }
55
56   .route {
57     margin-bottom: 30px;
58   }
59
60
61
62
63 export default ProductsContainer;

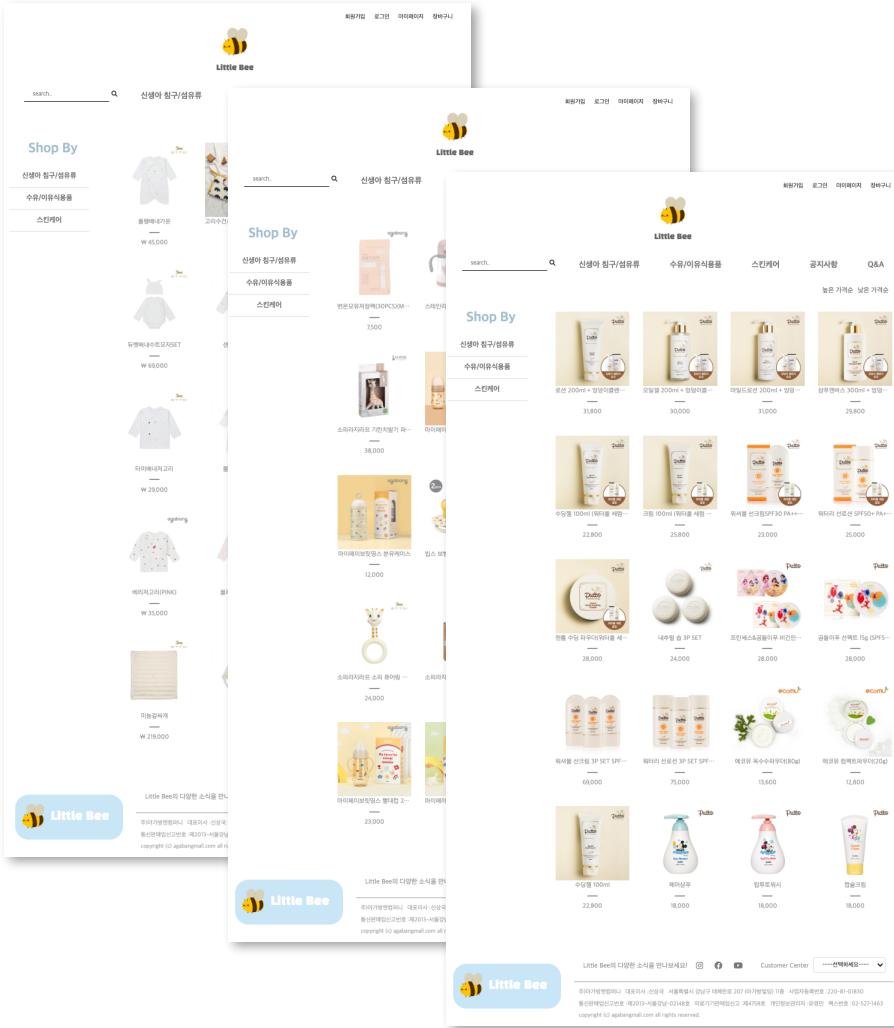
```

상품 목록 CSS 코드

Code

주요 개발 기능_상품 목록 조회(Main Category)

PAGE



MainCategory 선택 시 상품 전체 목록 조회(화면 최초 마운트 시 목록 조회)

BackEnd

3. ProductController

```

1  /** 전체목록 조회 --> Read(SELECT) */
2  router.get(`:${url}/:maincategory_maincategorynum`, async (req, res, next) => {
3    const { maincategory_maincategorynum: category } = req.params;
4    console.log(req.params);
5
6    // 페이지수, 표시 내용 파라미터
7    const { page=1, rows=20 } = req.query;
8
9    // 검색어를 Mybatis에 전달하기 위한 객체로 구성
10   const params = {};
11
12   // 데이터 조회
13   let json = null;
14   let pageInfo = null;
15
16   try {
17     // 전체데이터 수 얻기
18     const totalCount = await ProductService.getCount(params);
19     pageInfo = pagenation(totalCount, page, rows);
20
21     params.offset = pageInfo.offset;
22     params.listCount = pageInfo.listCount;
23     json = await ProductService.getList({ category: category });
24   } catch (err) {
25     return next(err);
26   }
27
28   res.sendResult({ data: json, pagination: pageInfo });
29 });

```

Slice에서 전달받은 데이터를 Service의 getList로 전달

4. ProductService

```

1  /** 상품 목록 데이터 조회 */
2  async getList(params) {
3    let dbcon = null;
4    let data = null;
5
6    try {
7      dbcon = await DBPool.getConnection();
8
9      let sql = mybatisMapper.getStatement('ProductMapper', 'selectList', params);
10     let [result] = await dbcon.query(sql);
11
12     if (result.length === 0) {
13       throw new RuntimeException('조회된 데이터가 없습니다.');
14     }
15
16     data = result;
17     console.log(data);
18   } catch (err) {
19     throw err;
20   } finally {
21     if (dbcon) {dbcon.release();}
22   }
23
24   return data;
25 }

```

parms를 통해 전달받은 값을 Mapper의 selectList로 전달

5. ProductMapper(SQL)

```

1  <!— 디중행 조회를 위한 기능 정의 —>
2  <select id="selectList">
3    SELECT prodnum, prodname, prodprice, thumbnail, subcategory_subcategorynum FROM product WHERE maincategory_maincategorynum=#{category} ORDER BY prodnum ASC
4
5  <!— 페이지 구현을 위한 LIMIT절 추가 —>
6  <if test="listCount > 0">
7    LIMIT ${offset}, ${listCount}
8  </if>
9 </select>

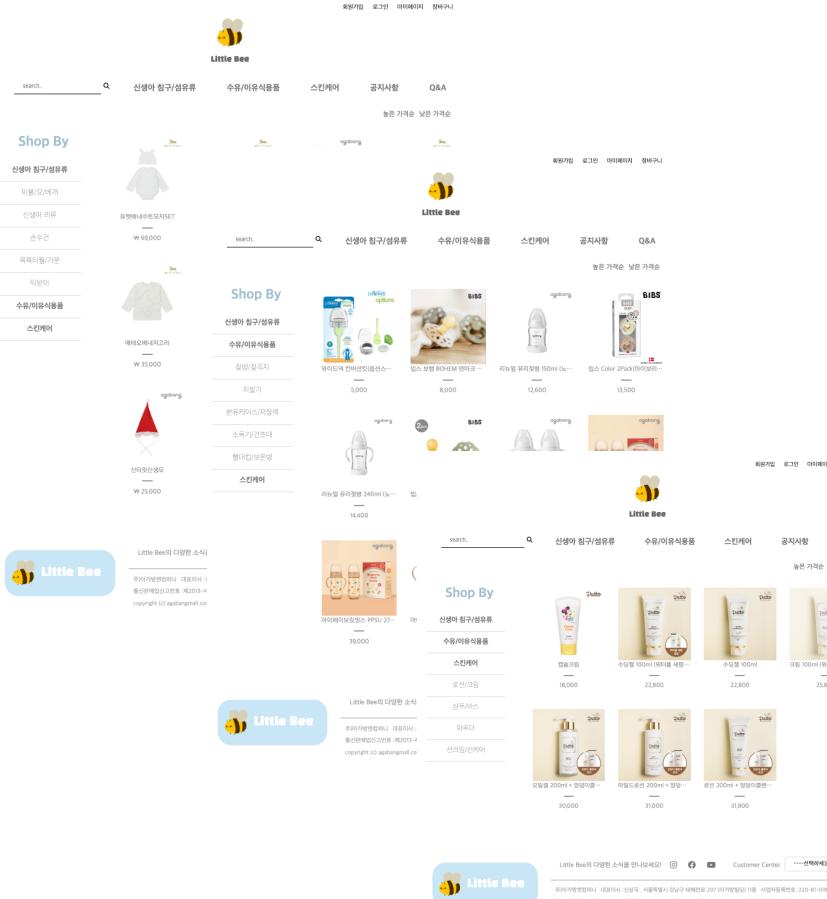
```

Service로부터 전달 받은 category값에 해당되는 데이터를 조회
(조회 후 FrontEnd에서 결과 데이터를 받아 화면 출력)

Code

주요 개발 기능_상품 목록 조회(Sub Category)

PAGE



FrontEnd

● ● ● 1. ProductNav

```

1  <ProductsContainer>
2    <div className='categoryBox'>
3      <ul className='categoryBoxSub'>
4        <li className='collapse-content'>Shop By</li>
5        <li className='menuItem'>
6          <a href="#" onClick=(onMenuItemClick)>신생아 험구/섬유류</a>
7          <ul className='sub'>
8            <li><NavLink to='/products/10/00' className='link'>이불/도마/매트</NavLink></li>
9            <li><NavLink to='/products/10/21' className='link'>신생아 의류</NavLink></li>
10           <li><NavLink to='/products/10/22' className='link'>수유관</NavLink></li>
11           <li><NavLink to='/products/10/23' className='link'>수유컵/기ottle</NavLink></li>
12           <li><NavLink to='/products/10/24' className='link'>식판비寤</NavLink></li>
13         </ul>
14       </li>
15       <li className='menuItem'>
16         <a href="#" onClick=(onMenuItemClick)>수유/미유식용품</a>
17         <ul className='sub'>
18           <li><NavLink to='/products/11/10' data-category="30" className='link'>젖병/젖꼭지</NavLink></li>
19           <li><NavLink to='/products/11/11' data-category="31" className='link'>치党校</NavLink></li>
20           <li><NavLink to='/products/11/32' data-category="32" className='link'>분유케이스/저장백</NavLink></li>
21           <li><NavLink to='/products/11/33' data-category="33" className='link'>소독기/건조기</NavLink></li>
22           <li><NavLink to='/products/11/34' data-category="34" className='link'>빨대기/보온병</NavLink></li>
23         </ul>
24       </li>
25       <li className='menuItem'>
26         <a href="#" onClick=(onMenuItemClick)>스킨케어</a>
27         <ul className='sub'>
28           <li><NavLink to='/products/12/40' data-category="40" className='link'>로션/크림</NavLink></li>
29           <li><NavLink to='/products/12/41' data-category="41" className='link'>샴푸/バス</NavLink></li>
30           <li><NavLink to='/products/12/42' data-category="42" className='link'>파우더</NavLink></li>
31           <li><NavLink to='/products/12/43' data-category="43" className='link'>선크림/선팩</NavLink></li>
32         </ul>
33       </li>
34     </ul>
35   </div>
36   <RouteContainer>
37     <Routes>
38       <Route path="11/*" element=<FoodProducts /> />
39       <Route path="10/*" element=<BeddingCloth /> />
40       <Route path="12/*" element=<SkinCare /> />
41       <Route path="/" element=<SearchResult /> />
42     </Routes>
43   </RouteContainer>
44 </ProductsContainer>

```

subCategory를 누르면 해당 주소로 이동

● ● ● 2. ProductSlice

```

1  /** subCategory에 따른 데이터 출력 */
2  export const getSubCategory = createAsyncThunk('ProductSlice/getSubCategory', async (payload, { rejectWithValue }) => {
3    let result = null;
4
5    try {
6      const response = await axios.get(`${API_URL}/${payload.category}/${payload.subcategory}`, {
7        query: payload?.query || ''
8      });
9      result = response.data;
10    } catch (err) {
11      console.group('ProductSlice.getSubCategory');
12      console.error(err);
13      console.groupEnd();
14      result = rejectWithValue(err.response);
15    }
16    return result;
17  })
18 }

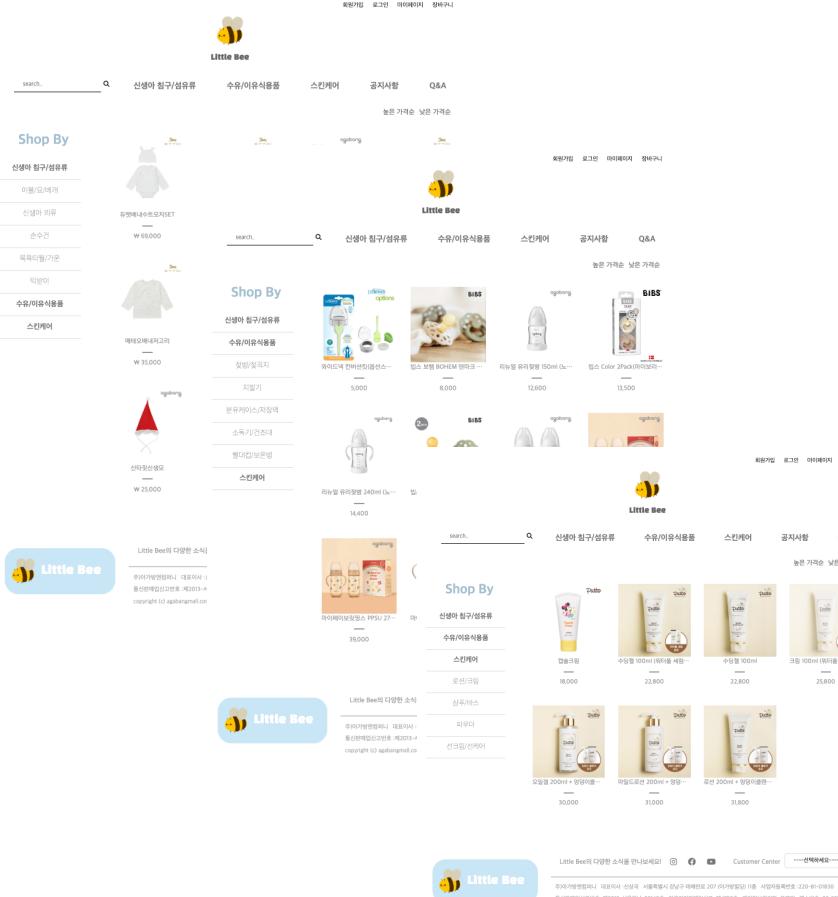
```

Slice에서 BackEnd로 데이터 요청

Code

주요 개발 기능_상품 목록 조회(Sub Category)

PAGE



BackEnd

3. ProductController

```

1  /** SubCategory 조회 */
2  router.get(`/${url}/:maincategory_maincategorynum/:subcategory_subcategorynum`, async (req, res, next) => {
3      const { maincategory_maincategorynum: category, subcategory_subcategorynum: subcategory } = req.params;
4
5      // 데이터 조회
6      let json = null;
7
8      try {
9
10         json = await ProductService.getSubCategory({
11             params,
12             category: category,
13             subcategory: subcategory
14         });
15         console.log(params);
16     } catch (err) {
17         return next(err);
18     }
19
20     res.sendResult({ data: json });
21 })

```

Slice에서 전달 받은 Main, Sub Category 값을 Service의 getSubCategory로 전달

4. ProductNav

```

1  /* SubCategory 조회 */
2  async getSubCategory(params) {
3      let dbcon = null;
4      let data = null;
5
6      try {
7          dbcon = await DBPool.getConnection();
8
9          let sql = mybatisMapper.getStatement('ProductMapper', 'subCategory', params);
10
11         if (result.length === 0) {
12             throw new RuntimeException('조회된 데이터가 없습니다.');
13         }
14
15         data = result;
16     } catch (err) {
17         throw err;
18     }
19
20     finally {
21         if (dbcon) {dbcon.release();}
22     }
23
24     return data;
25 }

```

params를 통해 전달 받은 값을 Mapper의 subCategory로 전달

5. ProductSlice

```

1  <!-- 소분류 상품 -->
2  <select id="subCategory">
3      SELECT prodnum, prodname, prodprice, thumbnail FROM product WHERE maincategory_maincategorynum=#{category} AND subcategory_subcategorynum=#{subcategory} ORDER BY prodprice ASC
4  </select>

```

전달 받은 Main, Sub Category의 값과 일치하는 데이터를 조회

6. Product

```

1  let subB_C = '';
2
3  if (Array.isArray(data)) {
4      subB_C = data?.filter((e) => e.subcategory_subcategorynum == subcategory);
5  }

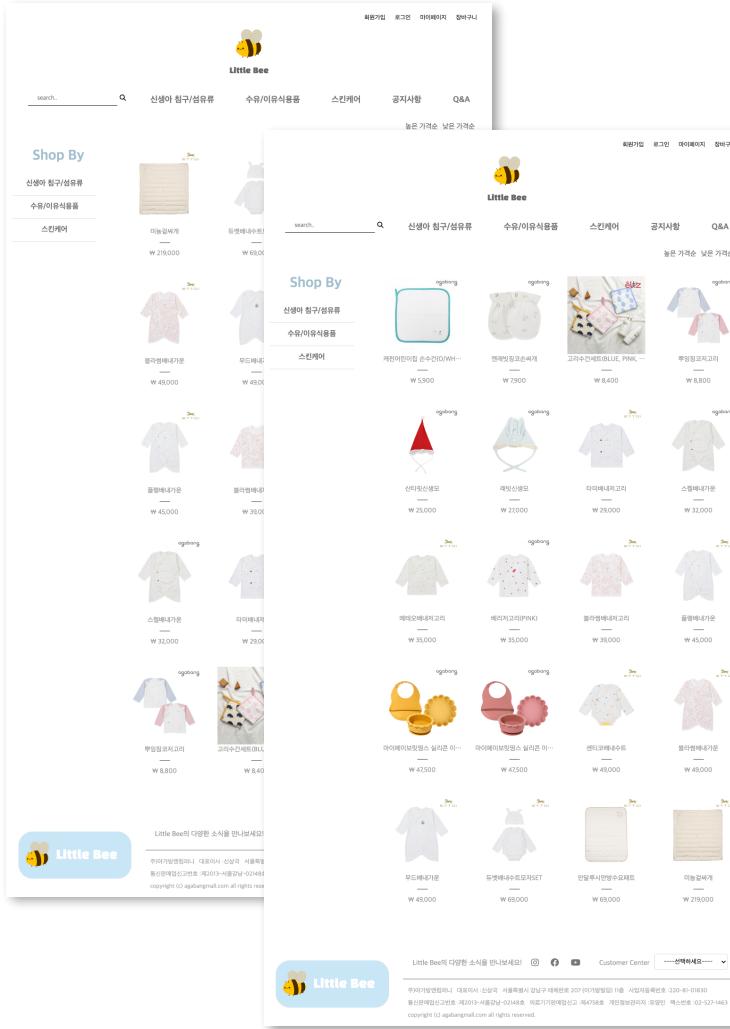
```

(FrontEnd)백엔드에서 받은 데이터 중 현재 주소의 번호와 DB에 저장되어있는 번호가 일치한다면 데이터를 subB_C에 저장
(subB_C를 활용하여 화면 출력)

Code

주요 개발 기능_상품 목록 조회(상품 정렬)

PAGE



FrontEnd

```

1 const dispatch = useDispatch();
2 const {value} = useQueryString();
3
4 const location = useLocation();
5 const CategoryLoc = location.pathname.substring(10, 12);
6
7 const {*} : subcategory = useParams();
8 console.log(subcategory);
9
10 const onMenuItemClick = useCallback((e) => {
11   e.preventDefault();
12   const current = e.currentTarget;
13
14   const content = current.parentElement.querySelector('.sub');
15
16   if (content.style.maxHeight) {
17     content.style.maxHeight = null;
18   } else {
19     content.style.maxHeight = content.scrollHeight + 'px';
20   }
21 }, []);
22
23 useEffect(() => {
24   // console.log("value값 바뀜 => " + value)
25
26   dispatch(getSort({
27     category: CategoryLoc,
28     value: value
29   }));
30 }, [CategoryLoc, value]);
31
32 return (
33   <div>
34     <ProductSort>
35       <NavLink to={`/products/${subcategory}?value=desc`} className='sortBtn'>높은 가격순</NavLink>
36       <NavLink to={`/products/${subcategory}?value=asc`} className='sortBtn'>낮은 가격순</NavLink>
37     </ProductSort>
38   </div>
39 )
40
41 
```

현재 주소의 Category값과 valule값을 Slice로 전달

```

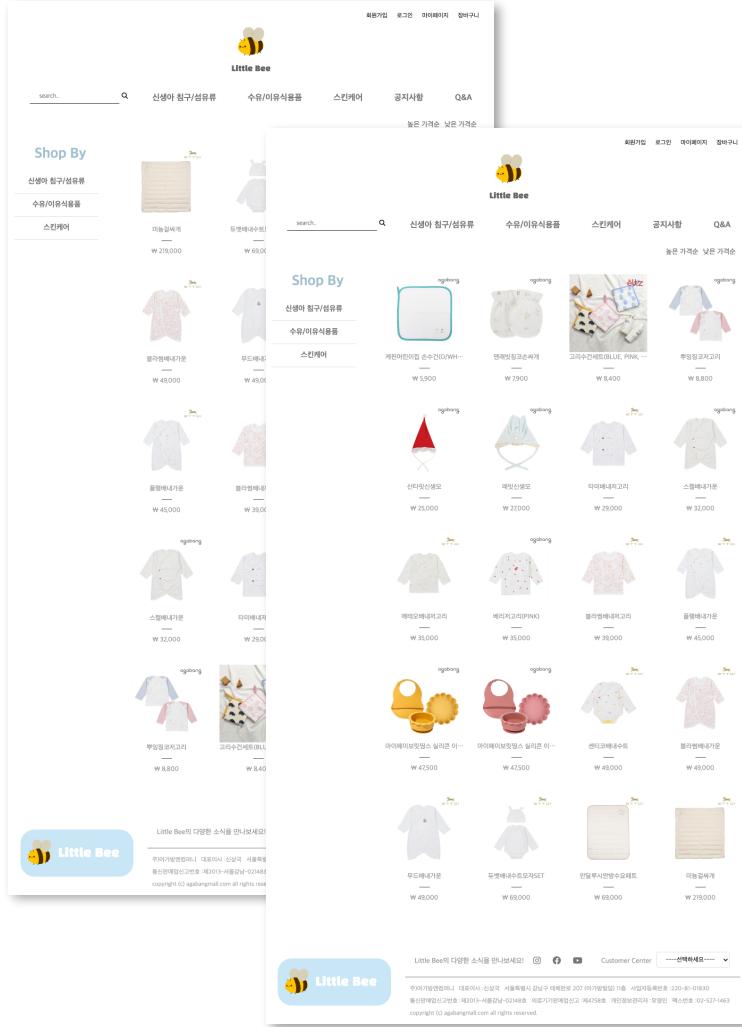
1 /* 리스트 정렬을 위한 비동기함수 */
2 export const getSort = createAsyncThunk('ProductSlice/getSort', async (payload, { rejectWithValue }) => {
3   let result = null;
4   console.log(payload);
5
6   try {
7     const response = await axios.get(`${API_URL}/${payload?.category}/sort`, {
8       params: {
9         // query: payload?.category || '',
10         value: payload?.value || ''
11       }
12     });
13     result = response.data;
14     // console.log(response);
15   } catch (err) {
16     console.group('ProductSlice.getSort');
17     console.error(err);
18     console.groupEnd();
19     result = rejectWithValue(err.response);
20   }
21
22   return result;
23 });
24 
```

Slice에서 BackEnd로 데이터 요청

Code

주요 개발 기능_상품 목록 조회(상품 정렬)

PAGE



BackEnd

```

1  /* 정렬 */
2  router.get(`${url}/:maincategory_maincategorynum/sort`, async (req, res, next) => {
3    const { value } = req.query;
4    const { maincategory_maincategorynum: category } = req.params;
5    console.log(`GET 파라미터 category - ${category}`);
6    console.log(`GET 파라미터 value - ${value}`);
7
8    // 데이터 조회
9    let json = null;
10
11   try {
12     json = await ProductService.getSort({ category: category || null, value: value });
13   } catch (err) {
14     return next(err);
15   }
16
17   res.sendResult({ data: json });
18 });

```

Slice에서 전달 받은 값을 Service의 getSort로 전달

```

1  /** 상품 정렬 */
2  async getSort(params) {
3    let dbcon = null;
4    let data = null;
5
6    try {
7      dbcon = await DBPool.getConnection();
8
9      console.log(params);
10     let sql = mybatisMapper.getStatement('ProductMapper', 'sortItems', params);
11     let [result] = await dbcon.query(sql);
12
13     if (result.length === 0) {
14       throw new RuntimeException('조회된 데이터가 없습니다.');
15     }
16
17     data = result;
18   } catch (err) {
19     throw err;
20   } finally {
21     if (dbcon) {dbcon.release();}
22   }
23
24  }

```

params를 통해 전달 받은 값을 Mapper의 sortItems로 전달

```

1  <!-- 상품 정렬 기능 정의 -->
2  <select id="sortItems">
3    SELECT prodnum, prodname, prodprice, thumbnail, subcategory_subcategorynum FROM product
4
5    <where>
6      <if test="category != null and category != ''">
7        maincategory_maincategorynum=${category}
8      </if>
9    </where>
10
11   ORDER BY prodprice ${value}
12
13 </select>

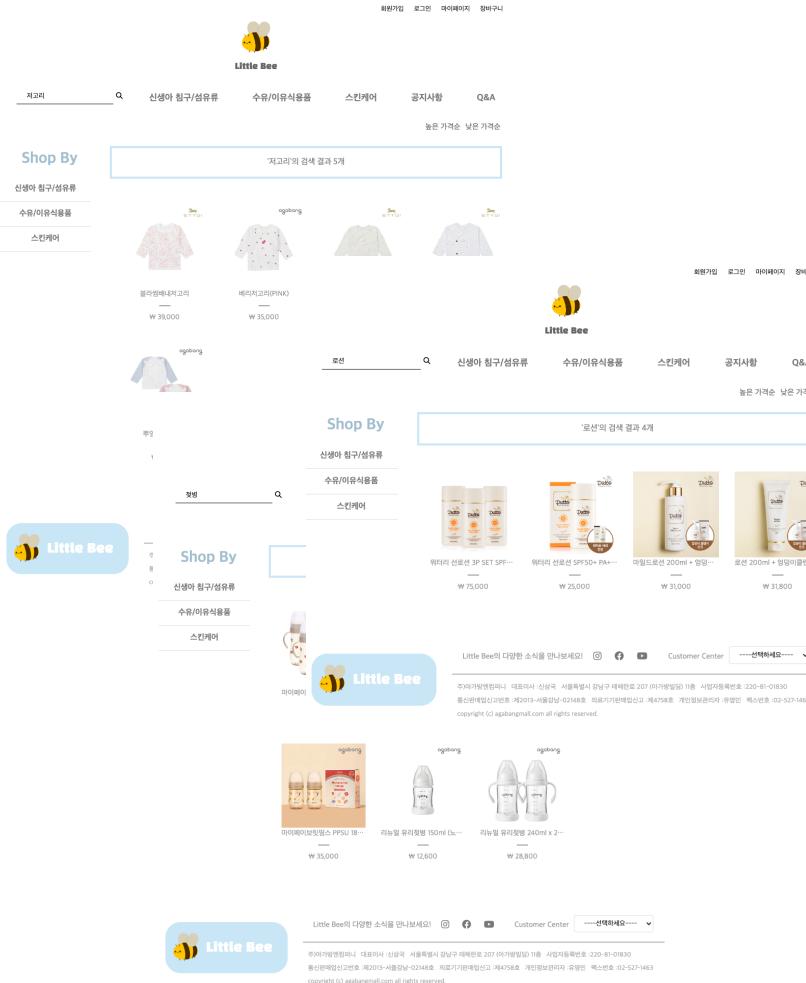
```

전달 받은 category와 value값으로 상품을 정렬
(FrontEnd에서 정렬된 데이터 결과를 받아 화면 출력)

Code

주요 개발 기능_상품 목록 조회(상품 검색)

PAGE



FrontEnd

```
1 const navigate = useNavigate();
2 const { query } = useQueryString();
3 const dispatch = useDispatch();
4
5 const onSearchResult = useCallback((e) => {
6   e.preventDefault();
7
8   const query = e.currentTarget.query.value;
9
10  if (query == '' || null || undefined) {
11    window.alert("검색어를 입력하세요!");
12
13    return;
14  }
15
16  dispatch(getSearchList({
17    query: query
18  }));
19  let redirectUrl = '/search?query=${query}`;
20  navigate(redirectUrl);
21 }, [navigate, query]);
```

검색어를 Slice로 전달
(검색어가 없을 시 alert창 표시)

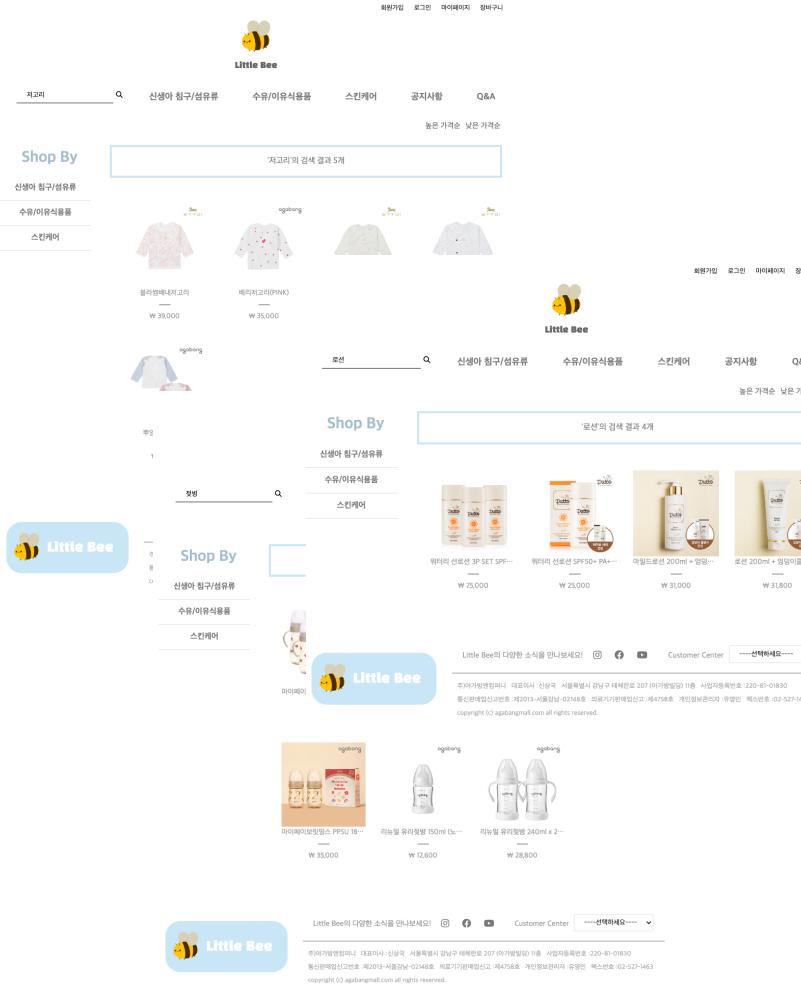
```
1 /** 검색 리스트 조회를 위한 비동기 함수 */
2 export const getSearchList = createAsyncThunk('ProductSlice/getSearchList', async (payload, { rejectWithValue }) => {
3   let result = null;
4   console.log(payload);
5   try {
6     const response = await axios.get('/search', {
7       params: {
8         query: payload?.query || ''
9       }
10    });
11    result = response.data;
12    console.log(result);
13
14  } catch (err) {
15    console.group('ProductSlice.getSearchList');
16    console.error(err);
17    console.groupEnd();
18    result = rejectWithValue(err.response);
19  }
20
21  return result;
22});
```

Slice에서 BackEnd로 데이터 요청

Code

주요 개발 기능_상품 목록 조회(상품 검색)

PAGE



BackEnd

```

1  /** 검색 결과 목록 조회 --> Read(SELECT) */
2  router.get('/search', async (req, res, next) => {
3      // 검색어 파라미터
4      const { query } = req.query;
5
6      // 검색어를 MyBatis에 전달하기 위한 객체로 구성
7      const params = {};
8      if (query) {
9          params.prodname = query;
10     }
11     else {
12         window.alert("검색 결과가 존재하지 않습니다.");
13     }
14     // 데이터 조회
15     let json = null;
16
17     try {
18         json = await ProductService.getSearchList(params);
19     } catch (err) {
20         return next(err);
21     }
22
23
24     res.sendResult({ data: json });
25 };
26
27

```

Slice에서 전달 받은 검색어를 Service의 getSearchList로 전달

```

1  /** 상품 검색 목록 데이터 조회 */
2  async getSearchList(params) {
3      let dbcon = null;
4      let data = null;
5
6      try {
7          dbcon = await DBPool.getConnection();
8
9          let sql = mybatisMapper.getStatement('ProductMapper', 'searchList', params);
10         console.log(params)
11         let [result] = await dbcon.query(sql);
12
13
14         if (result.length === 0) {
15             throw new RuntimeException('조회된 데이터가 없습니다.');
16         }
17
18         data = result;
19         console.log(data);
20     } catch (err) {
21         throw err;
22     } finally {
23         if (dbcon) {dbcon.release();}
24     }
25
26

```

params로 전달 받은 값을 Mapper의 searchList로 전달

```

1  <!-- 검색 결과 조회를 위한 기능 정의 -->
2  <select id="searchList">
3      SELECT prodnum, prodname, prodprice, thumbnail, subcategory_subcategorynum FROM product
4
5      <where>
6          <if test="prodname != null and prodname != ''">
7              prodname LIKE concat('%',#{prodname},'%')
8          </if>
9      </where>
10
11      ORDER BY prodnum DESC
12  </select>

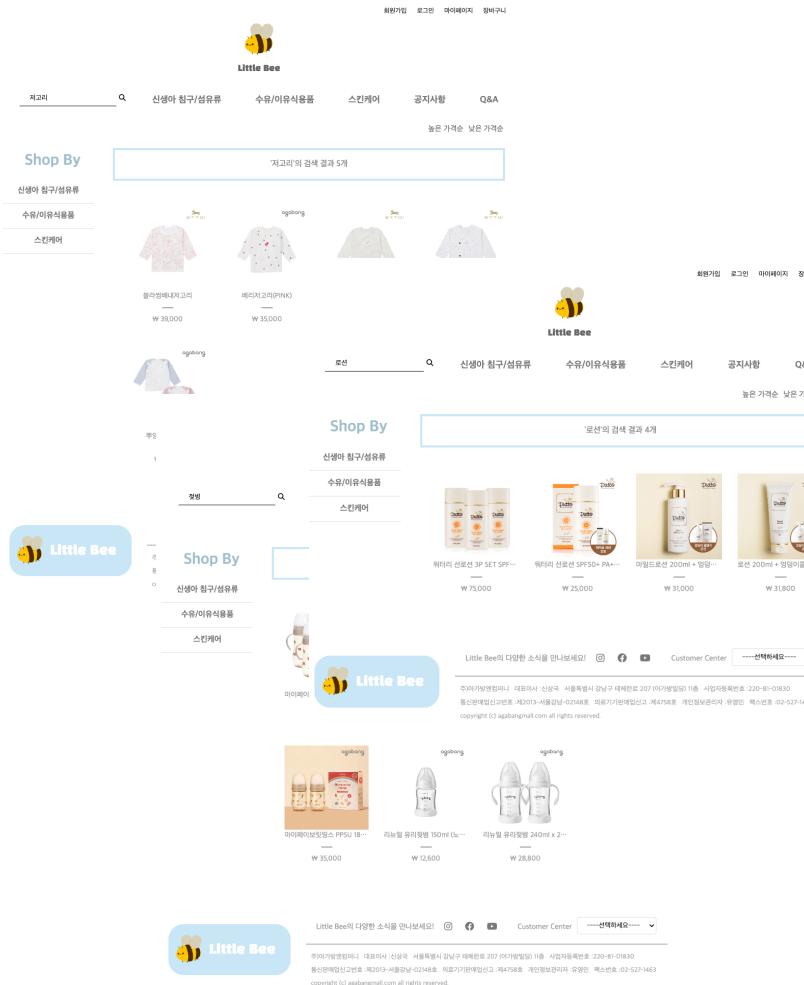
```

전달 받은 검색어와 일치하는 상품을 조회

Code

주요 개발 기능_상품 목록 조회(상품 검색 결과)

PAGE



FrontEnd

```
1  const SearchResult = memo(() => {
2    /* 데이터 캐싱을 위한 상태값 */
3    const [init, setInit] = useState(false);
4
5    /* QueryString 변수 받기 */
6    const { query } = useQueryString();
7
8    /* 리덕스 관련 코드 */
9    const dispatch = useDispatch();
10   const { data } = useSelector((state) => state.ProductSlice);
11
12   /* 최초 마운트시 리덕스를 통해 목록을 조회 */
13   // 화면 새로고침에 대한 상태값이 변경된다면 데이터를 새로 로드
14   useEffect(() => {
15     dispatch(getSearchList({
16       query: query
17     })).then(()=>{
18       setInit(true);
19     })
20   }, [query, init]);
21   console.log(data);
22
23   let length = '';
24   if (Array.isArray(data)) {
25     length = data.length;
26     console.log(data);
27     console.log(Array.isArray(data));
28   }
29
30   return (
31     <div>
32       <SearchArea>
33         <h3>'{query}'의 검색 결과 {length}개</h3>
34       </SearchArea>
35       <ProductListContainer>
36         {init && data?.map((v, i) => {
37           return (
38             <NavLink key={i} to={`/goods/10/${v.prodnum}`}>
39               <li key={i}>
40                 <img src={v.thumbnail} alt=''/>
41                 <p className='title'{>{v.prodname}</p>
42                 <div className='line'></div>
43                 <p className='price'>W {v.prodprice.toLocaleString()}</p>
44               </li>
45             </NavLink>
46           )
47         )}
48       </ProductListContainer>
49     </div>
50   );
51 );
```

Mapper에서 SQL로 조회한 검색어에 대한 결과를 출력

검색어와 검색어가 포함되어 있는 상품의 개수 표시
(개수 표시 부분 제외 상품 목록 조회와 CSS 동일)

Code

주요 개발 기능_회원가입(아이디 중복 검사)

PAGE



The screenshot shows the sign-up form on the Little Bee website. The '아이디 확인' button is highlighted with a red box.

회원가입

회원가입 로그인 마이페이지 장바구니

Little Bee

search...

신생아 침구/섬유류 수유/이유식용품 스크린케어 공지사항 Q&A

회원가입

이름 이름

아이디 아이디 **아이디확인**

비밀번호 비밀번호

비밀번호확인 비밀번호확인

전화번호 전화번호

주소 우편번호 검색 **우편번호찾기**

(상세주소)

이메일 이메일

추가정보

성별 여자 남자

생년월일 연도 월 일

개인정보수집에 동의합니다. [자세히 보기](#)

가입 취소

Little Bee의 다양한 소식을 만나보세요! Customer Center ---선택하세요---

Little Bee

Little Bee의 다양한 소식을 만나보세요! Customer Center ---선택하세요---

주)아방영합미니 대표이사 : 신상국 서울특별시 강남구 테헤란로 207 (아방빌딩) 11층 사업자등록번호 : 220-81-01830
통신판매업신고번호 : 제2013-서울강남-02148호 의료기기판매업신고 제4758호 개인정보관리자 : 유험민 평소번호 : 02-527-1463
copyright (C) agabangmail.com all rights reserved.

FrontEnd

1. Sign up

```
const regex = RegexHelper.getInstance();

try {
    /** 아이디 검사 */
    regex.value(document.querySelector(".user_id"), "아이디를 입력하세요.");
    regex.minLength(document.querySelector(".user_id"), 5, "아이디는 5글자이상 입력해야합니다.");
    regex.maxLength(document.querySelector(".user_id"), 10, "아이디는 10글자까지 입력 가능합니다.");
    regex.engNum(document.querySelector(".user_id"), "아이디는 영문 및 숫자만 가능합니다.");
} catch (e) {
    alert(e.message);
    console.error(e);
    return;
}

dispatch(getList({
    userid: userId.value
}).then(({payload})=> {
    const result = payload.data[0];
    console.log(result);
    if(result.COUNT == 0) {
        window.alert('사용 가능한 아이디입니다.');
    } else {
        window.alert('이미 존재하는 아이디입니다. 다시 입력해주세요.');
    }
}), [data]);
```

아이디에 대한 유효성 검사 후 중복 확인을 위해 입력된 아이디를 Slice로 전달



2. UserSlice

```
/** 다중행 데이터 조회를 위한 비동기 함수 */
export const getList = createAsyncThunk("UserSlice/getList", async (payload, { rejectWithValue }) => {
    let result = null;
    let params = null;

    console.log(payload);

    try {
        const response = await axios.get(URL, {
            params: {
                userid: payload.userid
            }
        });
        result = response.data;
    } catch (err) {
        console.group("UserSlice.getList");
        console.error(err);
        console.groupEnd();
        result = rejectWithValue(err.response);
    }
    return result;
});
```

BackEnd로 데이터 요청

Code

주요 개발 기능_회원가입(아이디 중복 검사)

PAGE



The screenshot shows the Little Bee website's user registration page. At the top, there is a navigation bar with links for 회원가입, 로그인, 마이페이지, and 장바구니. Below the navigation is a logo of a bee and a search bar. The main form is titled '회원가입' and contains fields for 이름 (Name), 아이디 (ID), 비밀번호 (Password), 비밀번호확인 (Password Confirmation), 전화번호 (Phone Number), 주소 (Address), 이메일 (Email), and 추가정보 (Additional Information). The '아이디 확인' button is highlighted with a red box.

BackEnd

3. UserController

```
1  /** 전체목록 조회 --> Read(SELECT) */
2  router.get(url, async (req, res, next) => {
3      const { userid } = req.query;
4
5      console.log(req.query.userid);
6
7      // 데이터 조회
8      let json = null;
9
10     try {
11         json = await UserService.getList({
12             userid: userid
13         });
14     } catch (err) {
15         return next(err);
16     }
17
18     res.sendResult({ data: json });
19 });

Slice에서 전달 받은 아이디를 Service의 getList로 전달
```

4. UserService

```
1  /** 목록 데이터 조회 */
2  async getList(params) {
3      let dbcon = null;
4      let data = null;
5
6      try {
7          dbcon = await DBPool.getConnection();
8
9          let sql = mybatisMapper.getStatement('UserMapper', 'selectList', params);
10         let [result] = await dbcon.query(sql);
11
12         if (result.length === 0) {
13             throw new RuntimeException('조회된 데이터가 없습니다.');
14         }
15
16         data = result;
17     } catch (err) {
18         throw err;
19     } finally {
20         if (dbcon) {dbcon.release();}
21     }
22     return data;
23 }
```

params를 통해 들어온 값을 Mapper의 selectList로 전달

5. UserMapper(SQL)

```
1  <!-- 다중행 조회를 위한 기능 정의 -->
2  <select id="selectList">
3      SELECT COUNT(*) `COUNT` FROM member WHERE userid=#{userid}
4  </select>
```

DB에 같은 아이디가 있는지 확인 후 갯수 조회
(COUNT = 1 이면 이미 사용중인 아이디가 있다는 의미)

Code

주요 개발 기능_회원가입

PAGE

The screenshot shows the Little Bee website's sign-up form. At the top, there's a navigation bar with links for 회원가입, 로그인, 마이페이지, and 장바구니. Below the navigation is a search bar and a logo for 'Little Bee'. The main content area has a title '회원가입' (Sign Up) with three colored dots above it. The form contains fields for 이름 (Name), 아이디 (ID), 비밀번호 (Password), 비밀번호확인 (Confirm Password), 전화번호 (Phone Number), 주소 (Address), 우편번호 (Postal Code), 성별 (Gender), 생년월일 (Birth Date), and 이메일 (Email). There are also gender selection buttons (여자, 남자), birth date input fields, and checkboxes for terms and conditions. At the bottom are two buttons: '가입' (Join) and '취소' (Cancel), with the '가입' button highlighted by a red box.

1. Sign up

```

1  try {
2      /**이름검사 */
3      regex.value(document.querySelector(".user_name"), "이름을 입력하세요");
4
5      /** 아이디 검사 */
6      regex.value(document.querySelector(".user_id"), "아이디를 입력하세요.");
7
8      /**비밀번호검사 */
9      regex.value(document.querySelector(".user_pw"), "비밀번호를 입력하세요");
10
11     regex.englishSpecial(document.querySelector(".user_pw"), "비밀번호는 특수문자( !@#$%^+= 기능 ) / 문자 / 숫자 포함 형태의 8~15자리 이내 입력해주세요. ");
12
13     /**비밀번호같은지검사 */
14     regex.value(document.querySelector(".confirm_pw"), "비밀번호 확인란을 입력하세요");
15     regex.compareTo(
16         document.querySelector("#confirm_pw"),
17         document.querySelector("#user_pw"),
18         "비밀번호를 다시 입력하세요."
19     );
20
21     /**연락처검사 */
22     regex.value(document.querySelector(".phone_num"), "전화번호를 입력하세요");
23     regex.cellphone(document.querySelector(".phone_num"), "전화번호를 다시 입력하세요");
24
25     /**이메일검사 */
26
27     regex.value(document.querySelector("#email"), "이메일을 입력하세요");
28     regex.email(document.querySelector("#email"), "이메일을 다시 입력해주세요");
29
30     regex.agree(current.agree, "개인정보 수집 동의란에 체크해주세요.");
31 } catch (e) {
32     alert(e.message);
33     console.error(e);
34     return;
35 }
36
37 dispatch(postitem({
38     username: userName.value,
39     userid: userId.value,
40     userpw: userPw.value,
41     userphone: userPhoneNum.value,
42     useraddress: roadNameAddress,
43     useremail: userEmail.value,
44     gender: genderChoice,
45     birthdate: userBirth.value
46 })) .then(({ payload, error }) => {
47     if (error) {
48         window.alert(payload.data.rmsg);
49         return;
50     }
51     window.alert("회원가입이 완료되었습니다.");
52     navigate('/');
53 })
54 }, []);

```

전체 Input요소에 대한 입력값 검사 후 데이터 추가를 위해
입력된 데이터 Slice로 전송

FrontEnd

2. UserSlice

```

1  /** 데이터 저장을 위한 비동기 함수 */
2  export const postItem = createAsyncThunk(
3      "UserSlice/postItem",
4      async (payload, { rejectWithValue }) => {
5          let result = null;
6
7          // console.log(payload);
8
9          try {
10              const response = await axios.post(URL, payload);
11              result = response.data;
12              console.log(result);
13          } catch (err) {
14              console.group("UserSlice.postItem");
15              console.error(err);
16              console.groupEnd();
17              result = rejectWithValue(err.response);
18          }
19
20          return result;
21      }
22  );

```

Slice에서 BackEnd로 데이터 요청

Code

주요 개발 기능_회원가입

PAGE

The screenshot shows the Little Bee website's user registration page. At the top, there is a navigation bar with links for 회원가입, 로그인, 마이페이지, and 장바구니. Below the navigation is a search bar and a logo for 'Little Bee'. The main content area has a title '회원가입' and several input fields for user information: 이름 (Name), 아이디 (ID), 비밀번호 (Password), 비밀번호확인 (Confirm Password), 전화번호 (Phone Number), 주소 (Address), 우편번호 (Postal Code), 이메일 (Email), and 추가정보 (Additional Information). There are also gender selection buttons (여자, 남자) and a birthdate input field. At the bottom, there is a checkbox for terms and conditions and two buttons: '가입' (Join) and '취소' (Cancel).

3. UserController

```

1  /** 회원가입 --> Create(INSERT) */
2  router.post(url, async (req, res, next) => {
3      // 파라미터 받기
4      const { username, userid, userpw, useraddress, userphone, useremail, gender, birthdate } = req.body
5
6      try {
7          regexHelper.value(username, '이름을 입력해주세요.');
8          regexHelper.value(userid, '아이디를 입력해주세요.');
9          regexHelper.value(userpw, '비밀번호를 입력해주세요.');
10         regexHelper.value(useraddress, '주소를 입력해주세요.');
11         regexHelper.value(userphone, '번호를 입력해주세요.');
12         regexHelper.value(useremail, '이메일을 입력해주세요.');
13     } catch (err) {
14         return next(err);
15     }
16
17     // 데이터 저장
18     let json = null;
19
20     try {
21         json = await UserService.addItem({
22             username: username,
23             userid: userid,
24             userpw: userpw,
25             useraddress: useraddress,
26             userphone: userphone,
27             useremail: useremail,
28             gender: gender || null,
29             birthdate: birthdate || null
30         });
31     } catch (err) {
32         return next(err);
33     }
34
35     res.sendResult({ data: json });
36 });

```

Slice에서 전달 받은 데이터에 대해 백엔드에서 한 번 더
유효성 검사 실행 후 Service의 addItem으로 전달

BackEnd

4. UserService

```

1  /** 데이터를 추가하고 추가된 결과를 조회하여 리턴 */
2  async addItem(params) {
3      let dbcon = null;
4      let data = null;
5
6      try {
7          dbcon = await DBPool.getConnection();
8
9          let sql = mybatisMapper.getStatement('UserMapper', 'insertItem', params);
10         let [insertId, affectedRows] = await dbcon.query(sql);
11
12         if (affectedRows === 0) {
13             throw new RuntimeException('저장된 데이터가 없습니다.');
14         }
15
16         // 새로 저장된 데이터의 PK값을 활용하여 다시 조회
17         sql = mybatisMapper.getStatement('UserMapper', 'selectItem', {username: insertId});
18         let [result] = await dbcon.query(sql);
19
20         if (result.length === 0) {
21             throw new RuntimeException('조회된 데이터가 없습니다.');
22         }
23
24         data = result[0];
25     } catch (err) {
26         throw err;
27     } finally {
28         if (dbcon) {dbcon.release();}
29     }
30 }
31

```

params를 통해 전달받은 값을
Mapper의 insertItem으로 전달

5. UserMapper(SQL)

```

1  <!-- 데이터 저장을 위한 기능 정의 -->
2  <insert id="insertItem">
3      INSERT INTO member (username, userid, userpw, useraddress, userphone, useremail, gender, birthdate, secession, signuptime) VALUES (#{username}, #{userid}, #{userpw}, #{useraddress}, #{userphone}, #{useremail}, #{gender}, #{birthdate}, 'T', now())
4  </insert>

```

입력 받은 데이터 저장

Code

주요 개발 기능_유효성 검사

● ● ● RegexHelper

```
1  /**
2  * 정규표현식을 기반으로 입력값에 대한 유효성 검사를 수행하는 클래스.
3  * HTML 문서에서 사용하기 위해 input field에 대한 입력값을 검사한다.
4  */
5  class RegexHelper {
6      static #current = null;
7
8      static getInstance() {
9          if (RegexHelper.#current === null) {
10              RegexHelper.#current = new RegexHelper();
11          }
12
13          return RegexHelper.#current;
14      }
15
16      /**
17      * 값의 존재 여부를 검사한다.
18      * @param {HTMLElement} field 검사할 대상에 대한 <INPUT>요소의 DOM 객체
19      * @param {string} msg     값이 없을 경우 표시할 메세지 내용
20      *
21      * ex) regexHelper.value('#user_id', '아이디를 입력하세요');
22      */
23      value(field, msg) {
24          const content = field.value;
25          if (
26              content == undefined ||
27              content == null ||
28              (typeof content == "string" && content.trim().length === 0)
29          ) {
30              throw new BadRequestException(msg);
31          }
32
33          return true;
34      }
}
```

회원가입, 회원 정보 수정, 로그인 등 유효성 검사가 필요한 곳에 사용
(여러곳에서 사용하기 위해 싱글톤 객체로 작성)

● ● ● ExceptionHelper

```
1  class BadRequestException extends Error {
2      // HTTP 상태코드를 의미하는 멤버변수
3      #statusCode;
4
5      // 입력 요소에 대한 selector
6      #selector;
7
8      // 입력요소를 두 번째 파라미터로 전달받는다.
9      constructor(msg = '잘못된 요청입니다.', selector = null) {
10          super(msg);
11          super.name = 'BadRequestException';
12          this.#statusCode = 400;
13          // 멤버변수에 입력요소를 참조시킨다
14          this.#selector = selector;
15      }
16
17      // 표준화 된 값이기 때문에 setter는 별도로 설정하지 않음
18      get statusCode () {
19          return this.#statusCode;
20      }
21
22      // 입력요소에 대한 getter
23      get selector() {
24          return this.#selector;
25      }
26  }
27
28  export { BadRequestException };
```

RegexHelper에서 검사를 하다가 정해진 형식이 아니라고 판단되면 강제로 에러 발생

Code

주요 개발 기능_회원가입(주소 API)

PAGE

회원가입

회원가입

```
1  /* 다음 주소 API 연동 */
2  const scriptUrl = "/t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js"
3  const open = useDaumPostcodePopup(scriptUrl);
4
5  const handleComplete = (data) => {
6      let fullAddress = data.address;
7      let extraAddress = '';
8      let roadAddr = data.roadAddress;
9
10     if (data.addressType === 'R') {
11         if (data.bname === '') {
12             extraAddress += data.bname;
13         }
14         if (data.buildingName !== '') {
15             extraAddress += (extraAddress !== '' ? ', ' : '') + data.buildingName;
16         }
17         fullAddress += (extraAddress !== '' ? extraAddress : '');
18     }
19     console.log(fullAddress);
20
21     // 우편번호와 주소 정보를 해당 필드에 넣는다.
22     document.getElementById('sampled_postcode').value = data.zonecode;
23     document.getElementById("sampled_roadAddress").value = roadAddr;
24     // document.getElementById("sampled_jibunAddress").value = data.jibunAddress;
25
26     console.log(data.zonecode);
27     console.log(roadAddr);
28     // console.log(data.jibunAddress);
29
30     const handleClick = () => {
31         open({ onComplete: handleComplete });
32     };
33 }
```

우편번호 찾기 (다음 주소 API 연동)

FrontEnd

```
1  const postC = current.post_code.value;
2  console.log(postC);
3  const roadA = current.street_address.value;
4  console.log(roadA);
5  const detailA = current.detail_address.value;
6  console.log(detailA);
7
8  const userName = document.querySelector('.user_name');
9  const userId = document.querySelector('.user_id');
10 const userPw = document.querySelector('.user_pw');
11 const userPhonNum = document.querySelector('.phone_num');
12 const roadNameAddress = ` ${postC} ${roadA}, ${detailA}`;
13 // console.log(roadNameAddress);
14 const userEmail = document.querySelector('.email');
15 const genderChoice = current.gender.value;
16 // console.log(genderChoice);
17 const userBirth = document.querySelector('.birthdate');
```

도로명 주소만 사용(지번주소를 눌러도 도로명 주소 표시)

Code

주요 개발 기능_회원 정보 수정

PAGE

The screenshot shows the Little Bee website's 'My Page' section. At the top, there is a navigation bar with links for '로그아웃', '마이페이지', and '장바구니'. Below the navigation is a search bar and a navigation menu with categories: '신생아 침구/섬유류', '수유/이유식용품', '스킨케어', '공지사항', and 'Q&A'. The main content area is titled '회원 정보 수정' (Member Information Modification). It contains several input fields for updating personal information:

- 주문내역**: Includes a dropdown for '선택하세요 사용자님' and a note about积分 (积分: 10223 P) and等级 (等级: silver).
- 정바구니**: Shows an empty cart.
- 나의 계시글**: Shows an empty list.
- 회원정보 수정**: Contains fields for '이름' (Name), '아이디' (ID), '비밀번호' (Password), '전화번호' (Phone Number), '주소' (Address), '도로명주소' (Road Address), '상세주소' (Detailed Address), and '이메일' (Email). There is also a '우편번호찾기' (Find Zip Code) button next to the address field.

At the bottom of the form are two buttons: '저장' (Save) and '취소' (Cancel). The footer includes a logo for 'Little Bee', social media links, a customer service center, and legal text.

FrontEnd

```
const ProfileEdit = memo(() => {
  const confirm = useCallback(async (e) => {
    e.preventDefault();
  }, []);
  const regex = RegexHelper.getInstance();

  try {
    /* 이름검사 */
    regex.value(document.querySelector(".user_name"), "이름을 입력하세요");
  } catch (e) {
    alert(e.message);
    console.error(e);
    e.selector.focus();
    return;
  }
  const result = await Swal.fire({
    icon: 'success',
    text: '저장되었습니다.',
    showCancelButton: true,
    confirmButtonText: '확인',
  });
  console.debug(result);
  if (result.isConfirmed) {
    navigate('/mymain');
  }
}, [Swal]);
```

```
const navigate = useNavigate();
const onCancel = useCallback((e) => {
  e.preventDefault();
  let referrer = document.referrer;
  navigate('/mymain');
  console.log(referrer);
});
const onChangePassword = useCallback((e) => {
  e.preventDefault();
  navigate('/mymain/profileedit/changepassword');
});
```

입력된 값에 대한 유효성 검사 후 저장이 완료되면 sweet alert로 '저장되었습니다'를 띄운 후 마이페이지로 이동

Code

주요 개발 기능_회원 정보 수정(비밀번호 변경)

PAGE

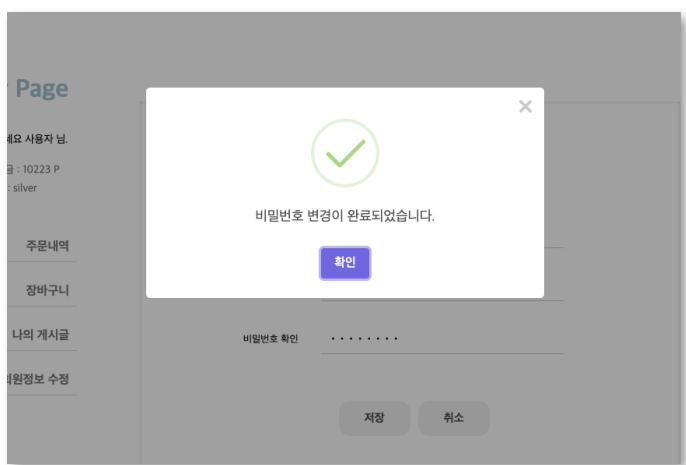
비밀번호 변경

현재 비밀번호

새비밀번호

비밀번호 확인

저장 취소



FrontEnd

```
1 const navigate = useNavigate();
2
3 const confirm = useCallback(
4   async (e) => {
5     e.preventDefault();
6
7     const regex = RegexHelper.getInstance();
8
9     try {
10       /*비밀번호검사 */
11       regex.value(
12         document.querySelector(".currentPassword"),
13         "현재 비밀번호를 입력하세요"
14       );
15       regex.value(
16         document.querySelector(".newPassword"),
17         "새비밀번호를 입력하세요"
18       );
19
20       regex.enumSpecial(
21         document.querySelector(".checkPassword"),
22         "비밀번호는 특수문자 (!@#$%^+= 가능) / 문자 / 숫자 포함 형태의 8~15자리 이내 입력해주세요."
23     );
24
25       regex.value(
26         document.querySelector(".checkPassword"),
27         "비밀번호 확인란을 입력하세요"
28     );
29
30     /** 비밀번호같은지검사 */
31     regex.compareTo(
32       document.querySelector(".newPassword"),
33       document.querySelector(".checkPassword"),
34       "비밀번호가 틀렸습니다. 다시 입력하세요."
35   );
36     regex.compareTo2(
37       document.querySelector(".currentPassword"),
38       document.querySelector(".newPassword"),
39       "이전 비밀번호와 동일한 비밀번호를 입력하였습니다. 다시 입력하세요."
40   );
41   } catch (e) {
42     alert(e.message);
43     console.error(e);
44     e.selector.focus();
45     return;
46   }
47   const result = await Swal.fire({
48     // title: '<strong>'+e.message+'</strong>',
49     icon: "success",
50     text: "비밀번호 변경이 완료되었습니다.",
51     showCloseButton: true,
52     // showCancelButton: true,
53     // focusConfirm: true,
54     confirmButtonText: "확인",
55     // cancelButtonText: '취소'
56   });
57   console.debug(result);
58
59   if (result.isConfirmed) {
60     navigate("/mymain/profileedit");
61   }
62 },
63 [Swal]
64 );
65
66 const onCancel = useCallback((e) => {
67   e.preventDefault();
68
69   let referrer = document.referrer;
70   navigate("/mymain");
71
72   console.log(referrer);
73});
```

1. 저장을 누르면 비밀번호에 대한 유효성 검사 실행

2. 검사를 완료한 후 sweet alert를 사용하여 비밀번호 변경 완료 메세지 출력

3. 완료 메세지의 확인을 누르면 회원 정보 수정 페이지로 이동

Code

주요 개발 기능_아이디 찾기

PAGE

아이디 찾기

이메일로 찾기 휴대전화로 찾기

이름	이름
이메일	이메일

찾기 **취소**

아이디 찾기

이메일로 찾기 휴대전화로 찾기

이름	이름
휴대전화	휴대전화

찾기 **취소**

1. Find Id

```

1  /* 리액스 관련 코드 */
2  const dispatch = useDispatch();
3  const { data, loading, error } = useSelector((state) => state.UserSlice);
4
5  useEffect(() => {
6    const email_radio = document.querySelector(".findemail");
7    email_radio.checked = true;
8  }, []);
9  const navigate = useNavigate();
10 const findBtn = useCallback((e) => {
11   e.preventDefault();
12   const mail = do
13     const phone = d
14     mail.classList.
15     phone.classList
16   }, []);
17 const onEmailChange = do
18   const mail = d
19   const phone = d
20   mail.classList.
21   phone.classList
22 }, []);
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

```

FrontEnd

2. UserSlice

```

1  /* 아이디 찾기를 위한 비동기 함수 */
2  export const getFindId = createAsyncThunk('UserSlice/getFindId', async (payload, { rejectWithValue }) => {
3    let result = null;
4
5    console.log(payload);
6
7    try {
8      const response = await axios.get('/login/findid', {
9        params: {
10          username: payload.username,
11          useremail: payload.useremail
12        }
13      });
14      result = response.data;
15      console.log(response);
16    } catch (err) {
17      console.group('UserSlice.getFindId');
18      console.error(err);
19      console.groupEnd();
20      result = rejectWithValue(err.response);
21    }
22
23    return result;
24  });

```

Slice에서 BackEnd로 데이터 요청

입력된 값에 대한 유효성 검사 후 저장이 완료되면 Slice로 입력된 데이터 전달

Code

주요 개발 기능_아이디 찾기

PAGE

아이디 찾기

이메일로 찾기 휴대전화로 찾기

이름

이메일

아이디 찾기

이메일로 찾기 휴대전화로 찾기

이름

휴대전화

BackEnd

3. UserController

```
1  /** 아이디 찾기 */
2  router.get('/login/findid', async (req, res, next) => {
3      // 파라미터 받기
4      const { username, useremail } = req.query;
5
6      // 유효성 검사
7      try {
8          regexHelper.value(username, '이름을 입력해주세요.');
9          regexHelper.value(useremail, '이메일을 입력해주세요.');
10     } catch (err) {
11         return next(err);
12     }
13
14     // 데이터 조회
15     let json = null;
16
17     try {
18         json = await UserService.getFindId({
19             username,
20             useremail,
21         });
22     } catch (err) {
23         return next(err);
24     }
25     res.sendResult({ data: json });
26     console.log(json);
27 });
```

4. UserService

```
1  /** 아이디 찾기 */
2  async getFindId(params) {
3      let dbcon = null;
4      let data = null;
5
6      try {
7          dbcon = await DBPool.getConnection();
8
9          let sql = mybatisMapper.getStatement('UserMapper', 'findId', params);
10         let [result] = await dbcon.query(sql);
11
12         if (result.length === 0) {
13             throw new RuntimeException('조회된 데이터가 없습니다.');
14         }
15
16         data = result[0];
17     } catch (err) {
18         throw err;
19     } finally {
20         if (dbcon) {dbcon.release();}
21     }
22     return data;
23 }
```

전달 받은 이름 및 이메일에 대해 한 번 더 유효성
검사를 거친 후 Service의 getFindId로 전달

params로 전달 받은 값을 Mapper의 findId로 전달

5. UserMapper(SQL)

```
1  <!-- ID 찾기 -->
2  <select id="findId">
3      SELECT userid, username FROM member WHERE username=#{username} AND useremail=#{useremail}
4  </select>
```

Service로부터 전달 받은 이름과 이메일 두 개가 일치하는 아이디를 조회

Code

주요 개발 기능_아이디 찾기 완료

PAGE

The image contains two screenshots of a web application interface for finding an ID. Both screenshots show a search form with fields for '이름' (Name) and '이메일' (Email). The top screenshot has '이름' set to '홍길동' and '이메일' set to 'gildong12@gmail.com'. The bottom screenshot has '이름' set to '김관수' and '이메일' set to 'ddd@ddd.ddd'. Both screenshots show a modal dialog titled '아이디 찾기 완료' (ID Found) containing the message '홍길동님의 아이디 찾기가 완료되었습니다.' (Hong Gil Dong's ID search is completed) and displaying the found ID ('아이디 : gild*****' or '아이디 : abcd***'). Below the modal are two buttons: '로그인' (Login) and '비밀번호 찾기' (Password Recovery).

BackEnd

```
1 // findId에서 보낸 데이터 받아오기
2 const location = useLocation();
3 const findIdData = location.state.value;
4
5 console.log(findIdData.data.userid);
6 const userId = findIdData.data.userid;
7 const userName = findIdData.data.username;
8
9 // 찾은 아이디 앞의 네자리를 제외 *로 표시
10 function findUserId(str) {
11     const len = str.length;
12     let head = str.substring(0, 4);
13
14     for (let i=4; i<len; i++) {
15         head += '*';
16     }
17     return head;
18 }
19
20 console.log(findUserId(userId));
21
22 const navigate = useNavigate();
23
24 const onLogin = useCallback((e) => {
25     e.preventDefault();
26     navigate("/login");
27 });
28
29 const onFindPw = useCallback((e) => {
30     e.preventDefault();
31     navigate("/login/findpw");
32 });
```

FindId에서 보낸 데이터를 useLocation으로 받아와 앞의 네 자리를 제외한 나머지를 '*' 처리하여 화면에 출력

Code

주요 개발 기능_비밀번호 찾기

PAGE

The screenshot shows a search form titled "비밀번호 찾기". It includes three input fields: "이름" (Name), "아이디" (ID), and "이메일" (Email). Below the inputs are two radio buttons: "이메일로 찾기" (Find by Email) and "휴대전화 번호로 찾기" (Find by Phone Number). At the bottom are "확인" (Check) and "취소" (Cancel) buttons.

The screenshot shows a search form titled "비밀번호 찾기". It includes three input fields: "이름" (Name), "아이디" (ID), and "휴대전화번호" (Phone Number). Below the inputs are two radio buttons: "이메일로 찾기" (Find by Email) and "휴대전화 번호로 찾기" (Find by Phone Number). At the bottom are "확인" (Check) and "취소" (Cancel) buttons.

FrontEnd

A screenshot of a browser window showing the code for handling the "Email" search option. The code uses a `useEffect` hook to set the "email_radio" radio button as checked when the component mounts. It also defines three callback functions: `onEmailChange`, `onPhoneChange`, and `onPhoneNumChange`. These callbacks update the state and classList of the respective radio buttons based on the selected search method.

```
1 useEffect(() => {
2   const email_radio = document.querySelector("#emailFind");
3   email_radio.checked = true;
4 }, []);
5
6 const onEmailChange = useCallback((e) => {
7   const email = document.querySelector(".emailAddress");
8   const phone = document.querySelector(".phoneNum");
9   phone.classList.add("active");
10  email.classList.remove("active");
11 }, []);
12
13 const onPhoneChange = useCallback((e) => {
14   const email = document.querySelector(".emailAddress");
15   const phone = document.querySelector(".phoneNum");
16   email.classList.add("active");
17   phone.classList.remove("active");
18 }, []);
```

Email로 찾기 또는 휴대전화 번호로 찾기 radio버튼

A screenshot of a browser window showing the full FrontEnd code for password finding. The code includes validation logic for both email and phone number inputs, handling user navigation, and error handling. It uses `useCallback` for event listeners and `useNavigate` for navigating between pages like "/findpw/findpwmemailresult" and "/findpw/findpwhoneresult".

```
20 const OnFindPassword = useCallback(async (e) => {
21   e.preventDefault();
22   const email = document.querySelector(".emailAddress");
23   const phone = document.querySelector(".phoneNum");
24   const regex = RegexHelper.getInstance();
25   try {
26     /**이름검사 */
27     regex.value(document.querySelector(".name"), "이름을 입력하세요");
28     regex.kor(
29       document.querySelector(".name"),
30       "이름은 한글만 입력 가능합니다."
31     );
32
33     /** 아이디 검사 */
34     regex.value(document.querySelector(".uid"), "아이디를 입력하세요");
35     regex.engNum(
36       document.querySelector(".uid"),
37       "아이디는 영문 및 숫자만 입력 가능합니다."
38     );
39
40     /**이메일검사*/
41     if (phone.classList.contains("active") == true) {
42       regex.value(
43         document.querySelector(".email"),
44         "이메일을 입력하세요"
45       );
46       regex.email(
47         document.querySelector(".email"),
48         "이메일 형식에 맞춰 입력하세요"
49       );
50     } else {
51       /**이름검사 */
52       regex.value(
53         document.querySelector(".phone"),
54         "전화번호를 입력하세요"
55       );
56       regex.num(
57         document.querySelector(".phone"),
58         "전화번호는 숫자만 입력 가능합니다."
59       );
60       regex.cellphone(
61         document.querySelector(".phone"),
62         "(-)없이 전화번호 11자리를 형식에 맞춰 입력하세요"
63       );
64     }
65   } catch (e) {
66     alert(e.message);
67     console.error(e);
68     return;
69   }
70
71   if (email.classList.contains('active') == false) {
72     navigate("/findpw/findpwmemailresult");
73   } else {
74     navigate("/findpw/findpwhoneresult");
75   }
76
77   const navigate = useNavigate();
78   const onCancel = useCallback((e) => {
79     e.preventDefault();
80
81     alert("취소되었습니다.");
82     navigate("/login");
83   }, [navigate]);
84
85});
```

비밀번호 입력에 대한 유효성 검사 후 해당 페이지로 이동

THANK YOU