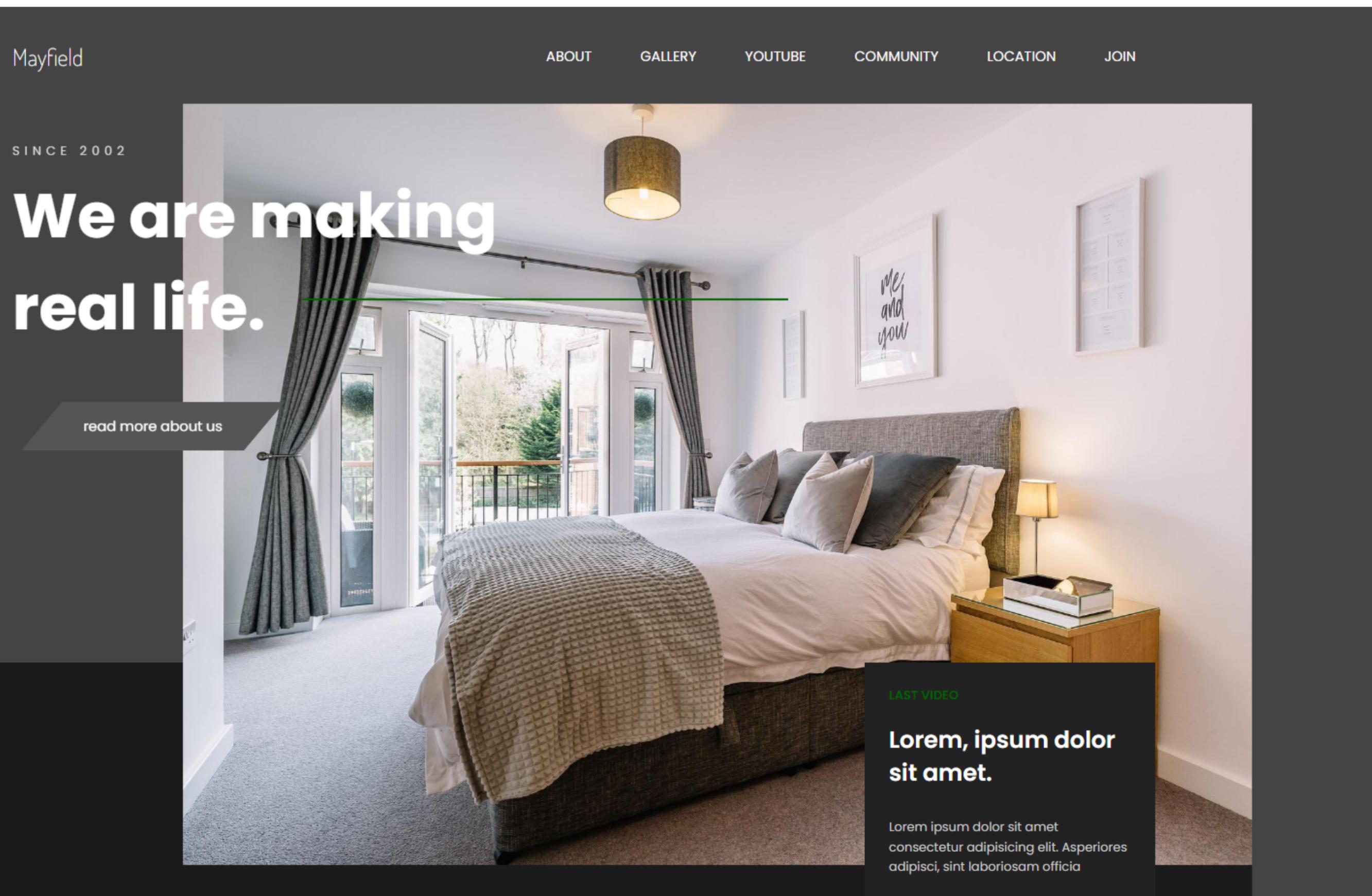


React Portfolio Project

Index



1. 프로젝트 소개
2. 프로젝트 구조
3. 반응형 레이아웃
4. 공통 컴포넌트
5. 리덕스(redux-toolkit)
6. 메인페이지 상세
7. 서브페이지 상세
 - 7-1. about 페이지
 - 7-2. gallery 페이지
 - 7-3. youtube 페이지
 - 7-4. community 페이지
 - 7-5. location 페이지
 - 7-6. join 페이지

1. 프로젝트 소개

Mayfield hotel을 주제로 작업한 포트폴리오 사이트입니다.
깔끔하고 모던한 느낌을 주기위한 어두운 컬러톤과, 네모 각진 느낌으로 작업하였습니다.
기업형 홈페이지에서 주로 사용되는 컨텐츠들로 구성되어있습니다.

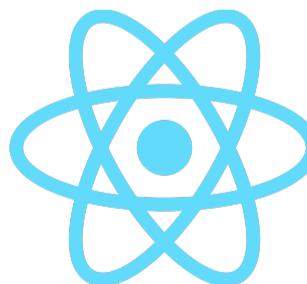
프로젝트 적용 기능



스크롤 이벤트, 마우스 이벤트 등의 다양한 기능구현을 위한 javascript 사용(es6)



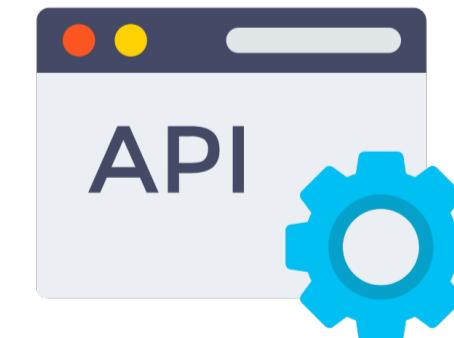
보다 편리하고 효율적으로 css를 작성하기 위한 scss 사용



컴포넌트기반으로 UI 수정과 재사용성이 좋고, 코드 가독성을 높이는 react 사용



컴포넌트를 전역적으로 관리하여 보다 효율적인 데이터 사용이 가능한 redux 사용



Flickr, Youtube, kakao api를 이용하여 데이터를 가져와 처리

2. 프로젝트 구조

 node_modules → 용량이 매우 크기 때문에, .gitignore에 추가하여 git에 올리진 않음

 public

 DB

 img → 이미지 적용 시, process.env.PUBLIC_URL로 경로 찾아 사용

 index.html → 최종 렌더링 결과 화면. index.js에 의해 렌더링된 결과가 표시됨
(index.html 이름을 바꿀 시 오류 발생)

 src

 asset

 components → 컴포넌트는 크게 common, main, sub로 분리해서 관리 (scss도 동일)

 redux → 전역으로 관리할 state를 컴포넌트별로 slice.js로 생성하여 관리

 scss

 App.js → 실제로 화면에 표시되는 내용을 정의

 index.js → HTML 템플릿 및 JavaScript의 컴포넌트를 조합하여 렌더링하고 실제 표시한다.

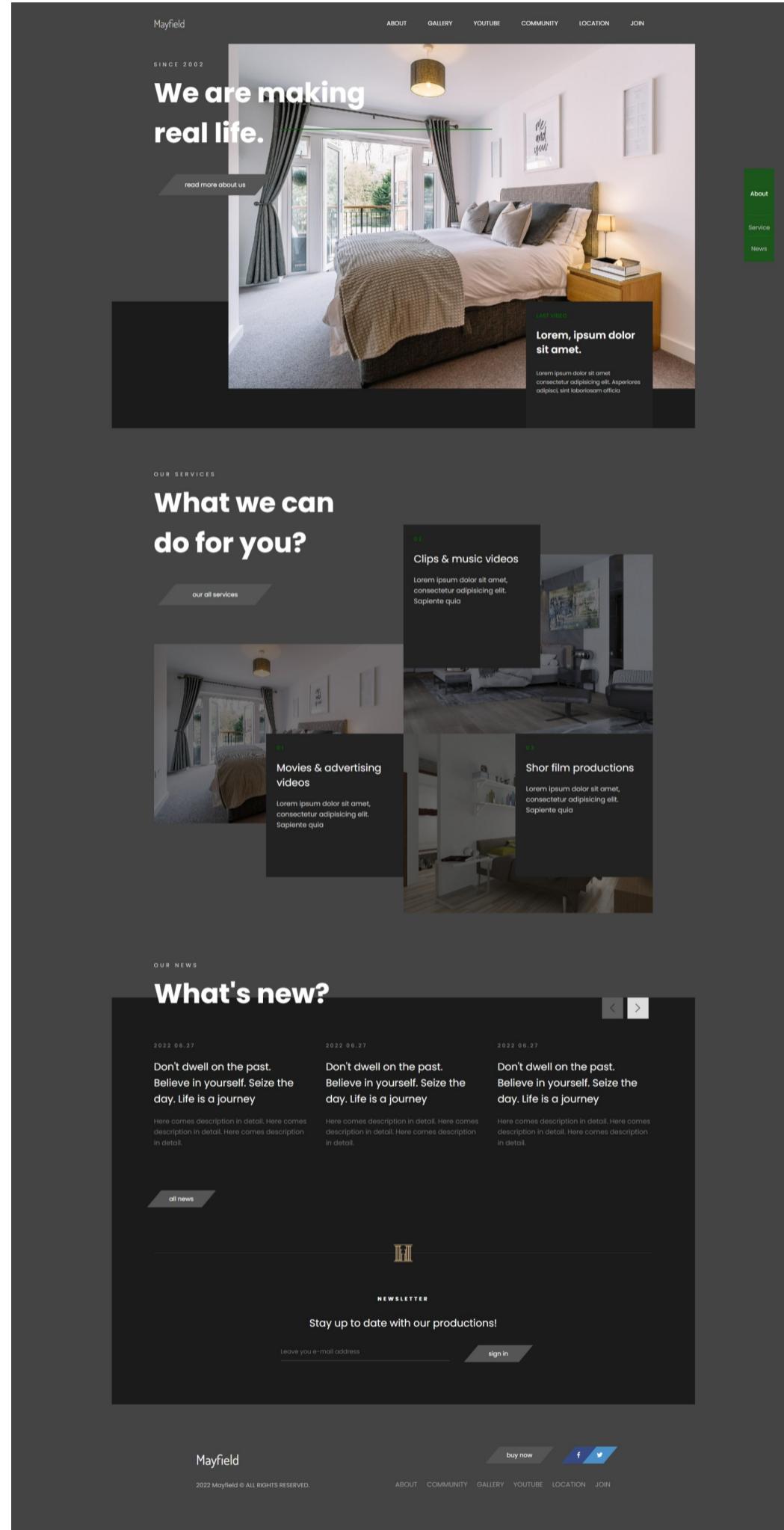
 .gitignore → Git 버전 관리에서 제외할 파일 목록을 지정

 package.json → 개발자가 배포한 패키지에 대해, 다른 사람들이 관리하고 설치하기 쉽게 하기 위한 문서

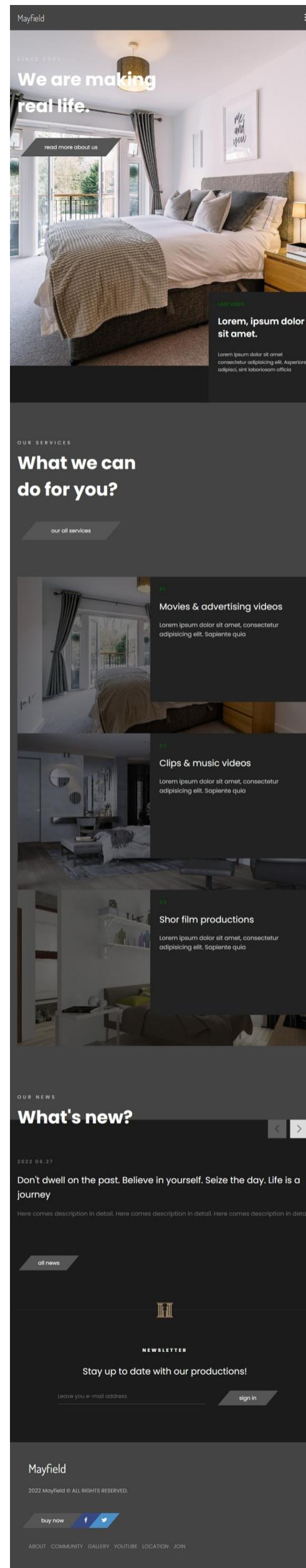
03

3. 반응형 레이아웃

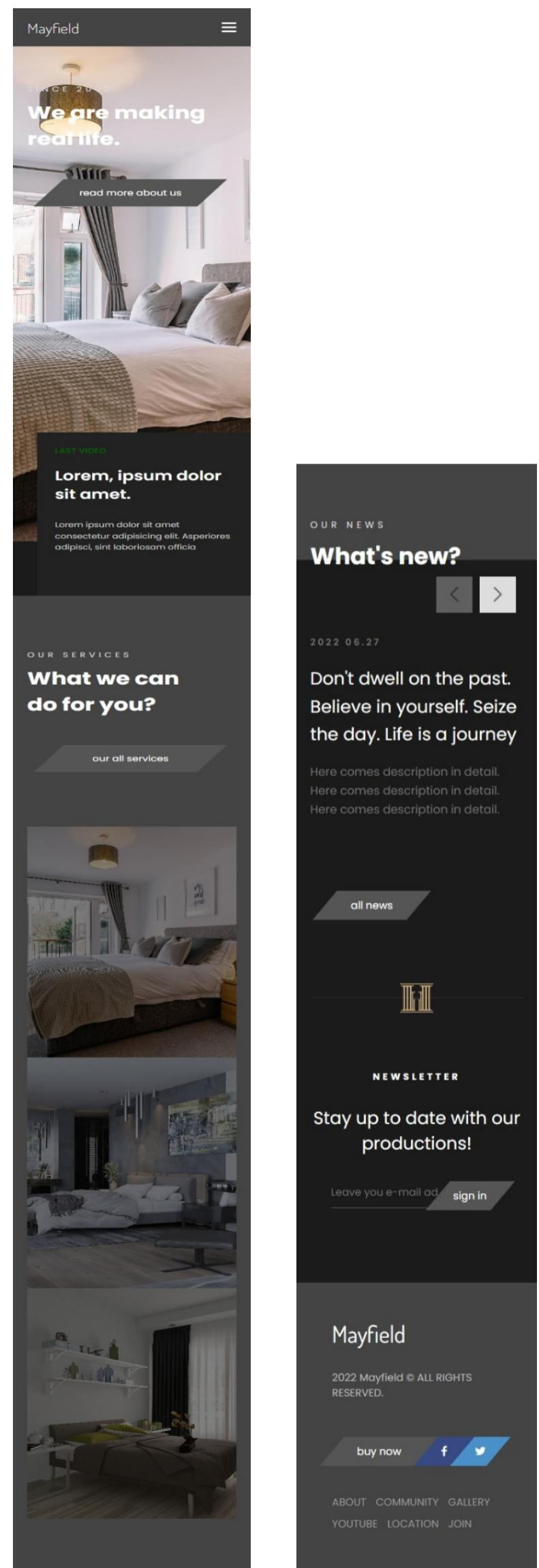
PC



Tablet



Mobile



scss로 디바이스 변수 만들어 사용

style.scss

```
$tablet: 1179px;
$mb: 539px;
```

컴포넌트.scss

```
@media screen and (max-width: $tablet) {
    // 태블릿에서 변경될 스타일 작성
}
```

```
@media screen and (max-width: $mb) {
    // 모바일에서 변경될 스타일 작성
}
```

4. 공통 컴포넌트

Header, Footer, Layout, Menu, Popup

페이지들의 통일성과 유지보수의 용이함을 위해 common으로 분리하여 재사용

 Header와 Footer는 main페이지 일때와, sub 페이지 일때 스타일이 다름으로 분기처리

App.js

```
<Switch>      ————— Switch로 중첩되는 라우터 제거
  <Route exact path="/">
    <Header type={"main"} />           ————— type을 넘겨주어 main, sub 분리
  </Route>
  <Route path="/" render={() => <Header type={"sub"} />} />
</Switch>
```

```
<Switch>
  <Route exact path="/">
    <Footer type={"main"} />
  </Route>
  <Route path="/" render={() => <Footer type={"sub"} />} />
</Switch>
```

각 component에서 type을 받아옴

Header.js

```
function Header({ type }) {
  // header component
}
```

Footer.js

```
function Footer({ type }) {
  // header component
}
```

 Popup - forwardRef, useImperativeHandle을 사용하여 부모컴포넌트에서 자식 컴포넌트의 상태값 변경 가능

Popup.js

```
const Popup = forwardRef(({ children }, ref) => {
  const [Open, setOpen] = useState(false);
});
```

```
useImperativeHandle(ref, () => {
  return {
    open: () => setOpen(true), //팝업열기 함수
    close: () => setOpen(false), //팝업닫기 함수
  };
});
```

해당 컴포넌트에서 만들어지는 함수를
부모컴포넌트에서 사용가능하도록 외부로 반환 가능

5. 리덕스(redux-toolkit)

데이터를 전역적으로 사용하기 위해 리덕스 사용

redux-toolkit을 이용한 상태관리
기존의 redux보다 훨씬 간결하게 사용 가능

- redux
store, action, reducer를 각각 만들어 별도로 관리
- redux-toolkit
slice - action, reducer를 합쳐서 한 파일로 관리

02) index.js

```
const store = configureStore({
  reducer: {
    youtube: youtubeReducer,
  },
});
```

store를 바로 생성해서 slice.js로부터 넘겨받은 데이터를
바로 저장하고 App에 전달

03) app.js

```
const dispatch = useDispatch();
useEffect(() => {
  dispatch(fetchYoutube());
}, []);
```

04) youtube.js

```
const Vids = useSelector(store => store.youtube.data);
```

원하는 컴포넌트에서 자유롭게 store데이터 접근 가능

01) youtubeSlice.js

```
import axios from "axios";
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";

export const fetchYoutube = createAsyncThunk(
  "youtube/requestYoutube",
  async () => {
    const key = "AIzaSyCNEFP7grGD77zUQvYF6Tg93dOjeA-mCjs";
    const playList = "PLKoTiVSIIVm0Dn5bSCc23ly_V6EdZ3k2";
    const num = 4;
    const url = `https://www.googleapis.com/youtube/v3/playlistItems?part=snippet&key=${key}&playlistId=${playList}&maxResults=${num}`;

    const response = await axios.get(url);
    return response.data.items;
  }
);

const youtubeSlice = createSlice({
  name: "youtube",
  initialState: {
    data: [],
    isLoading: false,
  },
  extraReducers: {
    [fetchYoutube.pending]: (state) => {
      state.isLoading = true;
    },
    [fetchYoutube.fulfilled]: (state, action) => {
      state.isLoading = false;
      state.data = action.payload;
    },
    [fetchYoutube.rejected]: (state) => {
      state.isLoading = false;
    },
  },
});

export default youtubeSlice.reducer;
```

axios 요청함수 설정,
리듀서 설정

6. 메인페이지 상세

slide motion

해당 프레임에 마우스 호버시, drag 표시 나타나며
드래그 시 슬라이드 효과 구현됨

localStorage

해당 프로젝트에서는 DB대신 localStorage를 활용하여
데이터를 저장하고, 보여줌

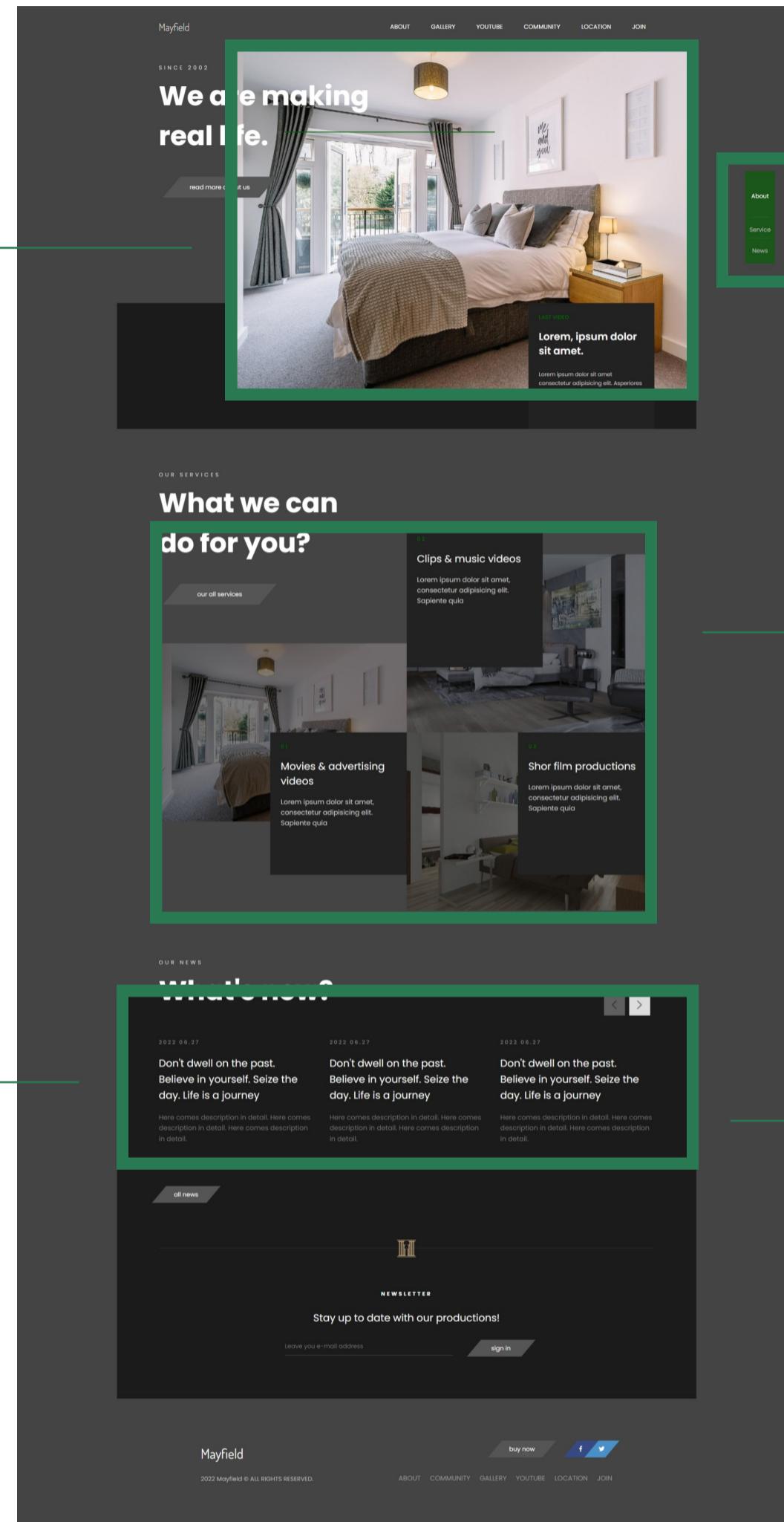
로컬저장소에 state값 물리적으로 저장 (json형태로 반환)

```
const getLocalData = () => {
  const data = localStorage.getItem('post');
  return JSON.parse(data);
};
```

로컬저장소에 초기 데이터가 없을때 오류발생을 방지하기 위해
더미데이터를 대신 출력후 로컬저장소에 다시 저장

```
const getLocalData = () => {
  const data = localStorage.getItem("post");
  const dummyPosts = [
    { title: " ", content: " " }, ..... ,
  ];

  if (data) {
    return JSON.parse(data);
  } else {
    return dummyPosts;
  }
};
```



scroll navigation

각 섹션의 offsetTop값을 구하여
스크롤 효과 적용
해당 위치에 도달하면 class="on"이 붙으면 활성화

redux를 활용하여 데이터 가져오기

메인페이지와, 서브페이지 모두 데이터를 가져와
출력시키기 위해 redux 사용
useSelector로 store에서 가져와 사용

```
const Rooms = useSelector((store) => store.room.data);
```

slide motion

버튼 클릭시 아티클이 한개씩 넘어가는 슬라이드 구현

7. 서브페이지 상세

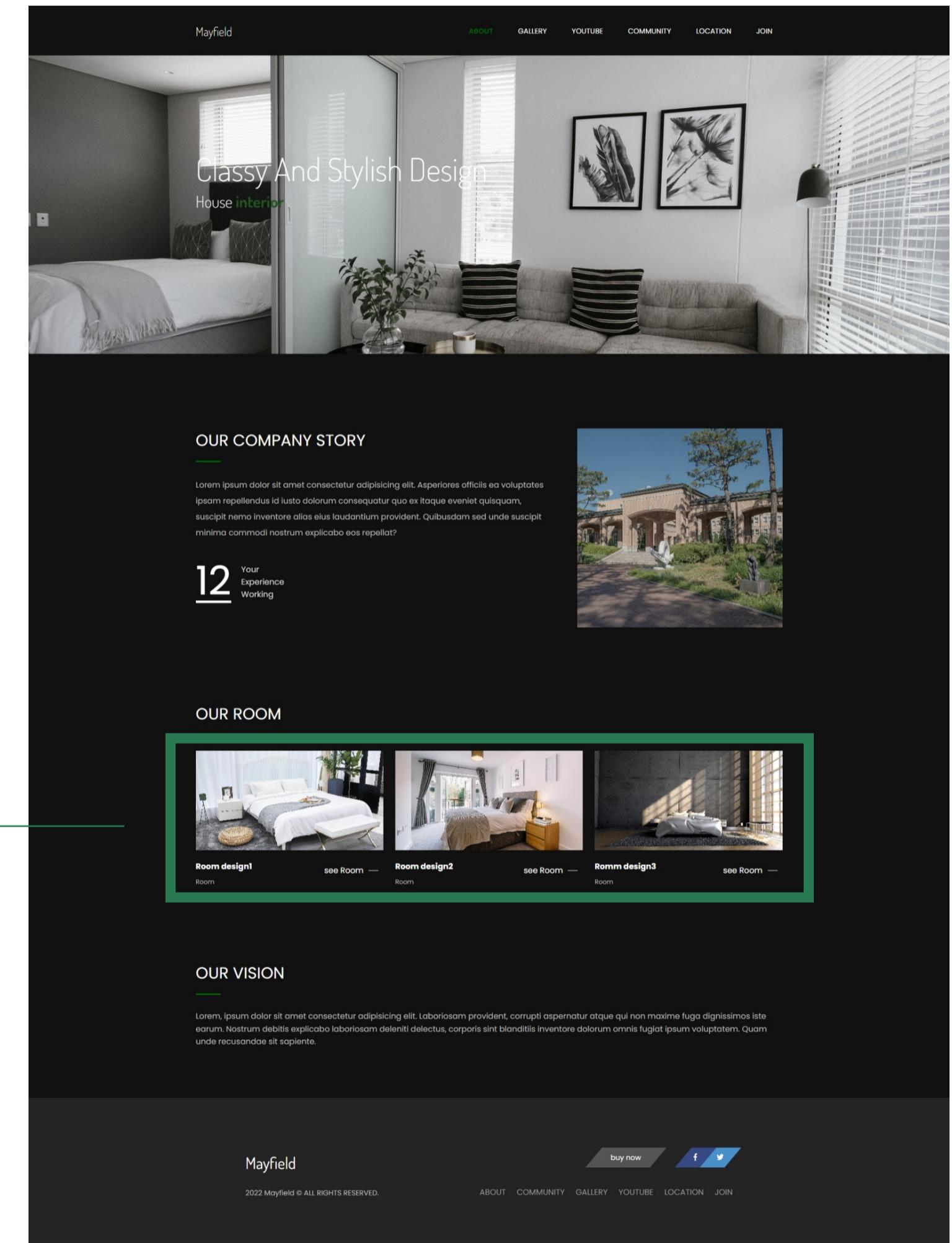
7-1. about page

redux를 활용하여 전역적으로 관리되고 있는
store에 저장되어 있는 room 데이터를 가져와 적용

```
const Rooms = useSelector((store) => store.room.data);
```

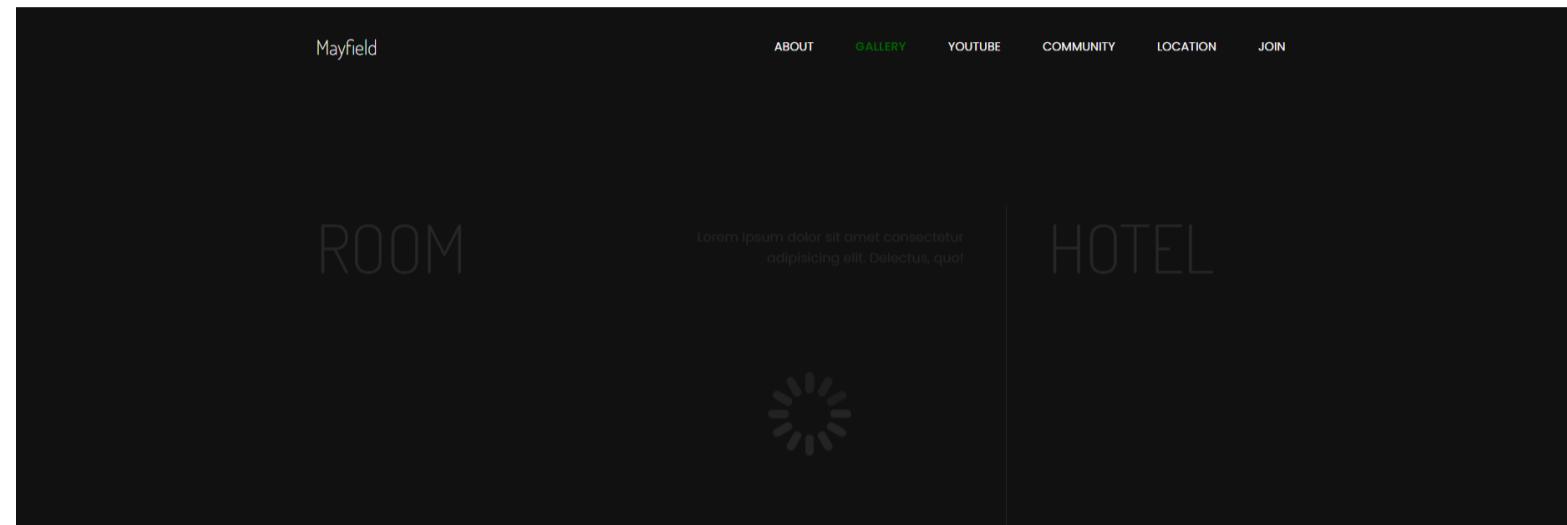
```
<div className="wrap">
  {Rooms.map((room, idx) => {
    return (
      <div className="cont" key={idx}>
        <div className="pic">
          <img src={`${path}/img/${room.pic}`} alt={room.name} />
        </div>
        <strong>{room.name}</strong>
        <p>{room.desc}</p>
        <span>see Room</span>
      </div>
    );
  )}
</div>
```

동적으로 dom생성



7. 서브페이지 상세

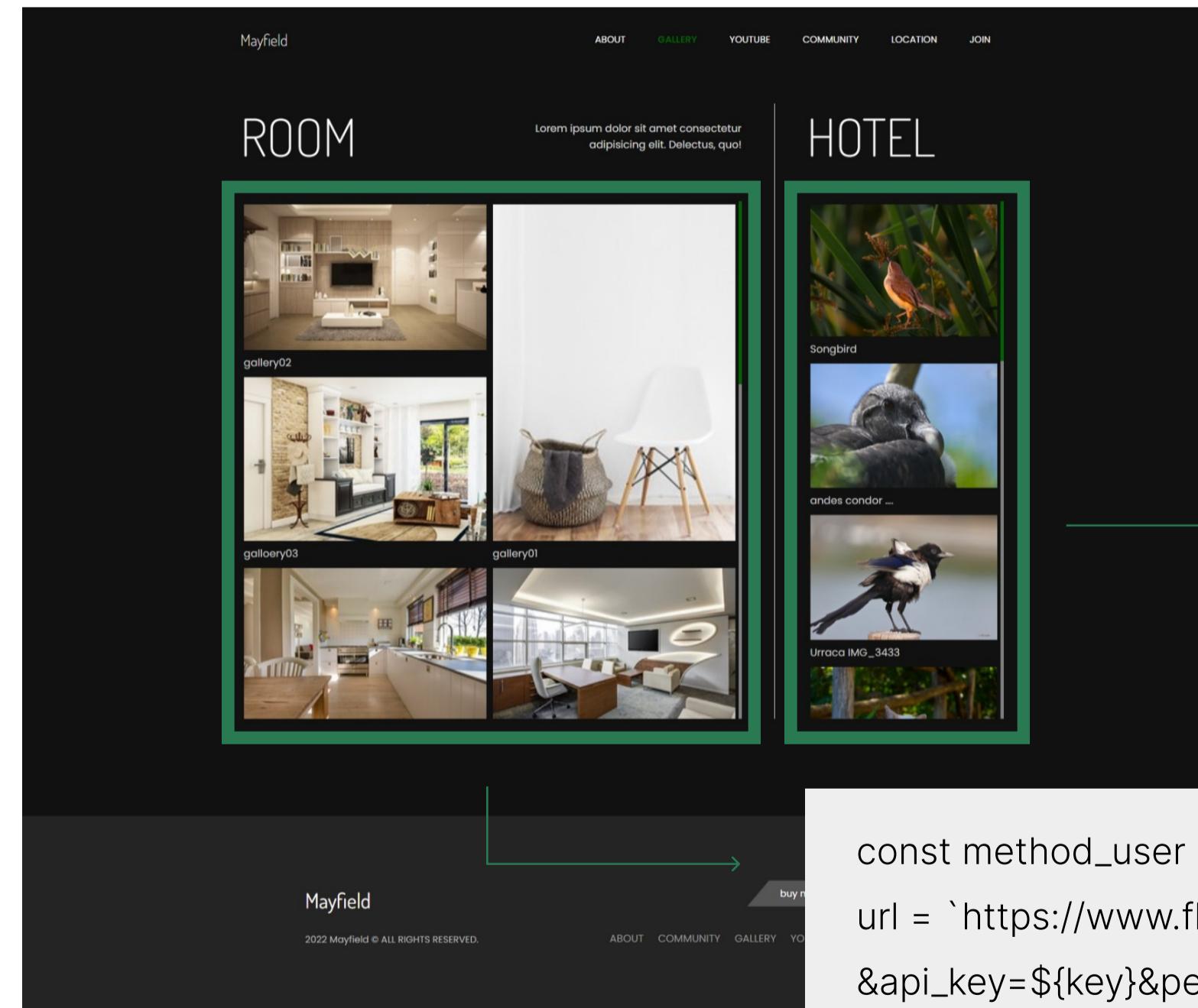
7-2. gallery page



데이터 다 불러오고 나타나게끔 로딩처리

```
setTimeout(() => {
  if (!frame.current && !frame2.current) return;
  frame.current.classList.add("on");
  frame2.current.classList.add("on");
  setLoading(false);
}, 1000); //데이터준비 완료될때까지 지연
```

데이터가 다 불러와지면, frame 나타나고
loading 사라지게 처리



flickr api를 활용한 데이터 처리

돋보기 클릭시 팝업 생성 구현

작성자 클릭시 작성자가 올린 사진목록 나오게끔 메소드 변경 처리

ROOM / HOTEL

각각 다른 api method 호출하여 처리

```
const method_search = "flickr.photos.search"; 검색 메소드 사용
url = `https://www.flickr.com/services/rest/?method=${method_search}
&api_key=${key}&per_page=${opt.count}&nojsoncallback=1&format=json
&tags=${opt.tags}`;
```

```
const method_user = "flickr.people.getPhotos"; 유저 메소드 사용
```

```
url = `https://www.flickr.com/services/rest/?method=${method_user}
&api_key=${key}&per_page=${opt.count}&nojsoncallback=1&format=json&user_id=${opt.user}`;
```

7. 서브페이지 상세

7-3. youtube page

youtube api axios로 불러와 데이터 처리

redux를 활용하여 전역적으로 관리되고 있는
store에 저장되어 있는 youtube 데이터를 가져와 적용

```
const Vids = useSelector((store) => store.youtube.data);
```

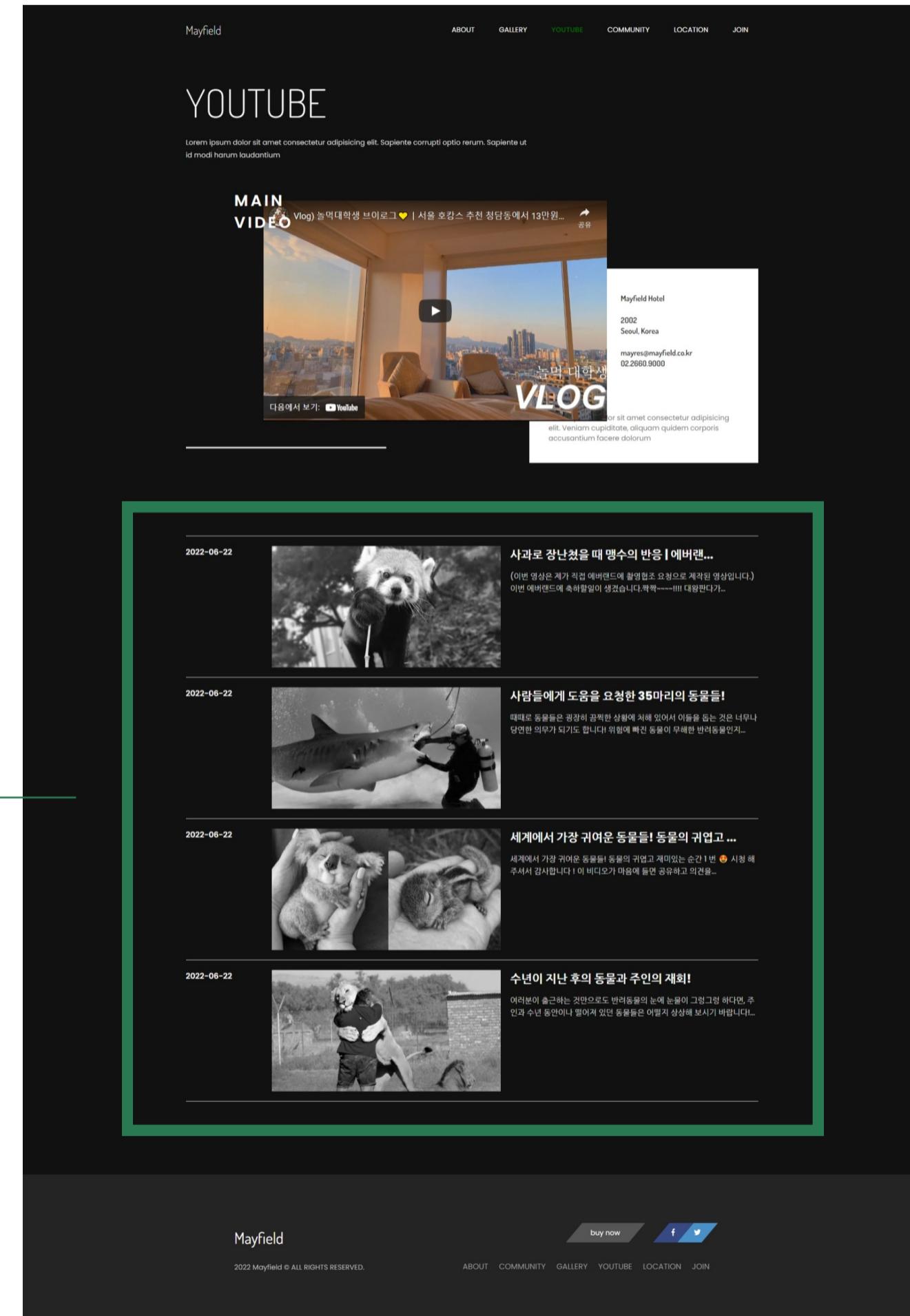
```
<div className="wrap">
  {Vids.map((vid, idx) => {
    const tit = vid.snippet.title;
    const desc = vid.snippet.description;
    const date = vid.snippet.publishedAt;

    return (
      

<span>{date.split("T")[0]}</span>
        <div className="pic" onClick={() => handlePopup(idx)}>
          <img src={vid.snippetthumbnails.high.url} alt={vid.title} />
        </div>
        <div className="txt-wrap">
          <h2>{tit.length > 25 ? tit.substr(0, 25) + "..." : tit}</h2>
          <p>{desc.length > 80 ? desc.substr(0, 80) + "..." : desc}</p>
        </div>

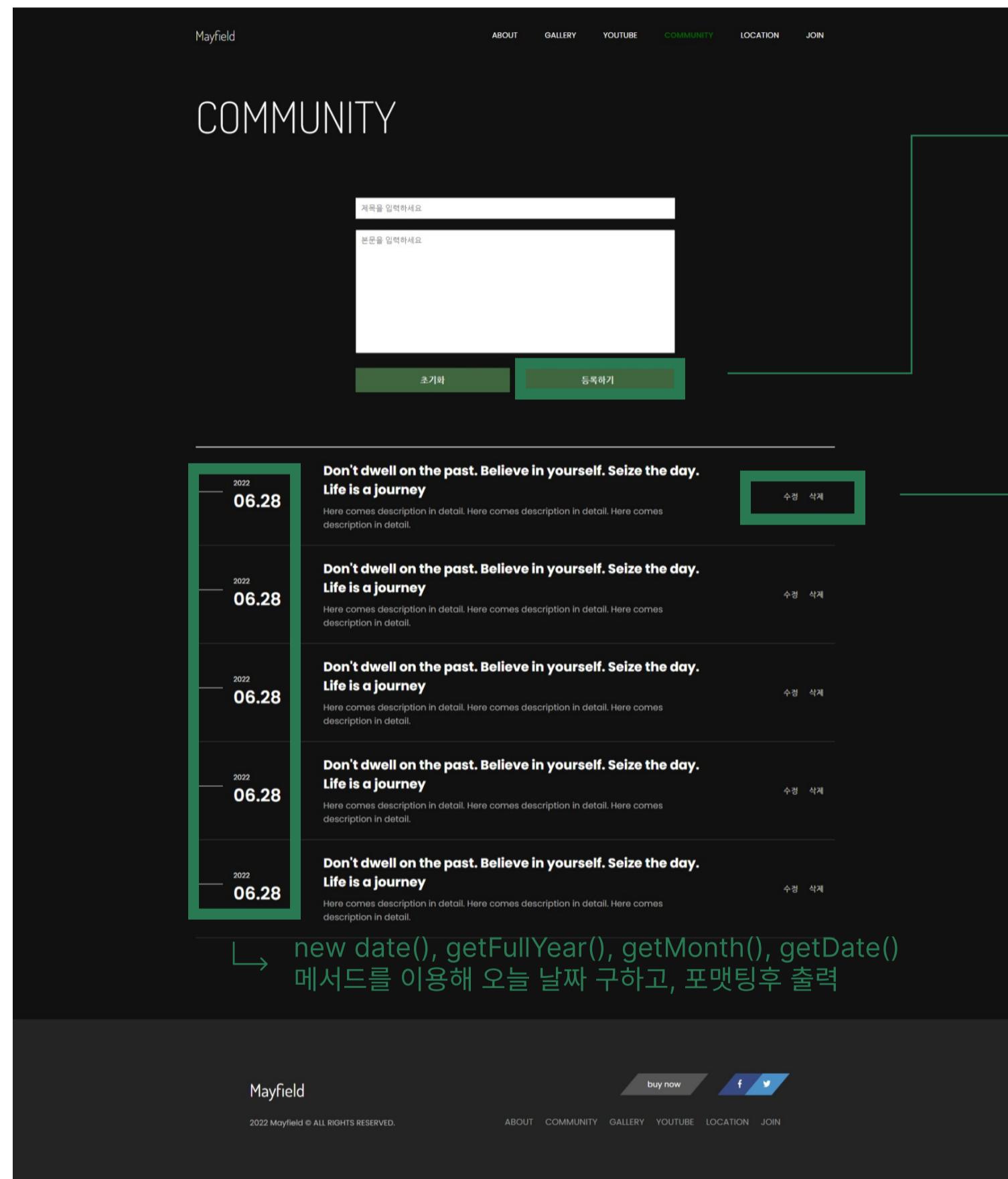

    );
  )})
</div>
```

동적으로 dom생성



7. 서브페이지 상세

7-4. community page



글 저장 함수

```
const createPost = () => {
  if (!input.current.value.trim() || !textarea.current.value.trim()) {
    resetPost();
    return alert("제목과 본문을 모두 입력하세요");
  }
  setPosts([
    { title: input.current.value, content: textarea.current.value },
    ...Posts,
  ]);

  resetPost();
};
```

글 삭제 함수

```
const deletePost = (index) => {
  setPosts(Posts.filter(_ , idx) => index !== idx));
};
```

글 수정 함수

```
const updatePost = (index) => {
  setAllowed(true);
  if (!inputEdit.current.value.trim() || !textareaEdit.current.value.trim()) {
    resetPost();
    return alert("수정할 제목과 본문을 모두 입력하세요");
  }

  setPosts(
    Posts.map((post, idx) => {
      if (idx === index) {
        post.title = inputEdit.current.value;
        post.content = textareaEdit.current.value;
        post.enableUpdate = false;
      }
      return post;
    })
  );
};
```

7. 서브페이지 상세

7-5. location page

kakaomap api를 이용하여 지도 출력

```
const handleResize = () => {
  map_instance.setCenter(Info[Index].latlng);
};

useEffect(()=>{
  window.addEventListener("resize", handleResize);

  return () => window.removeEventListener("resize", handleResize);
}, [Index]);
```

브라우저 리사이즈시 마커 중앙 유지

```
const info = [
  {
    title: "삼성동 코엑스",
    latlng: new kakao.maps.LatLng(37.547369989801844, 126.81897619294178),
    imgSrc: `${process.env.PUBLIC_URL}/img/marker.svg`,
    imgSize: new kakao.maps.Size(40, 40),
    imgPos: { offset: new kakao.maps.Point(20, 30) },
  },
];
```

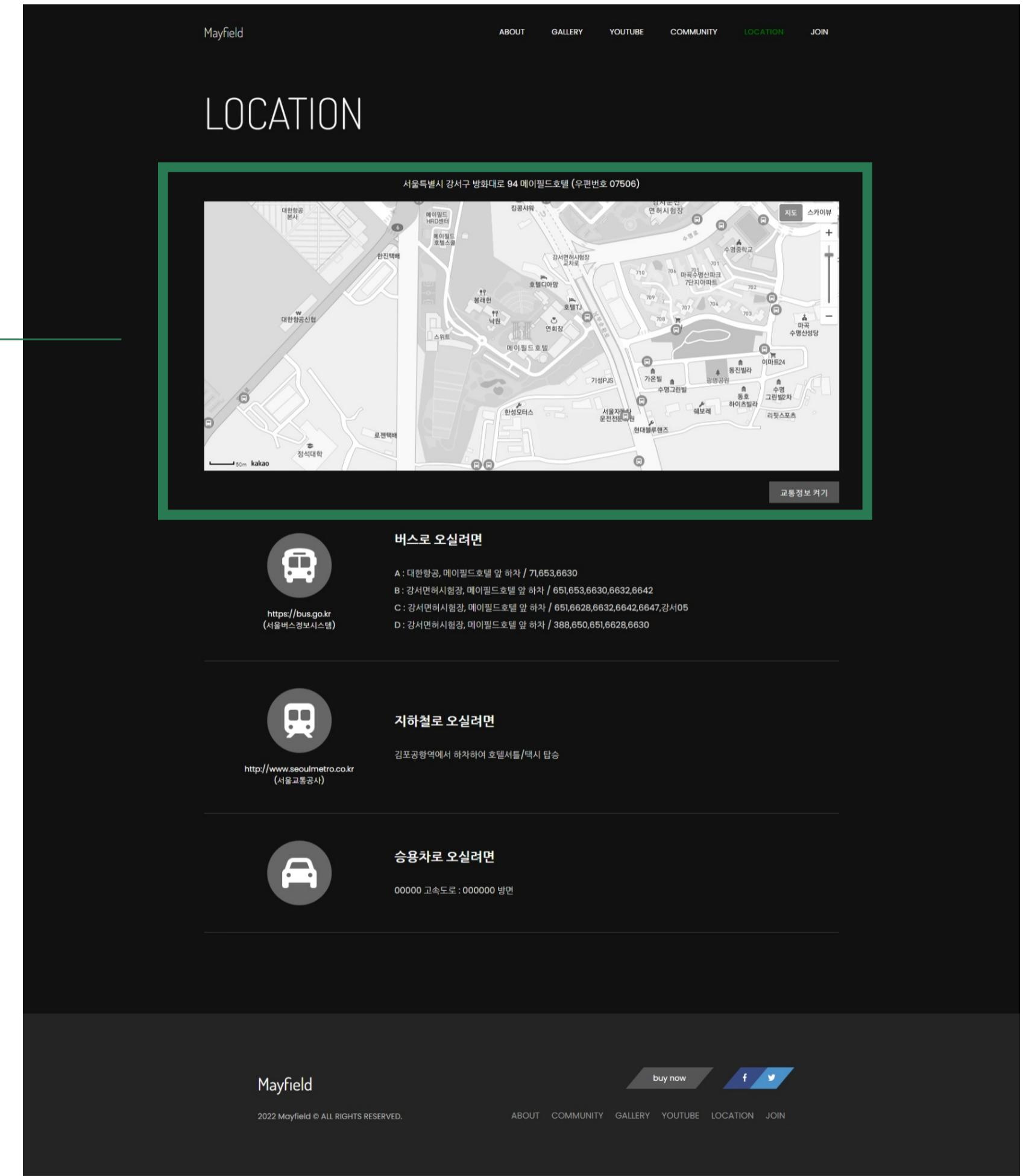
마커 다른이미지로 생성하기

```
const imageSrc = Info[Index].imgSrc;
const imageSize = Info[Index].imgSize;
const imageOption = Info[Index].imgPos;

const markerImage = new kakao.maps.MarkerImage(
  imageSrc,
  imageSize,
  imageOption
);

const markerPosition = Info[Index].latlng;

const marker = new kakao.maps.Marker({
  position: markerPosition,
  image: markerImage,
});
```



7. 서브페이지 상세

7-6. join page

formvalidation 적용
해당 조건에 맞지 않으면 error Message 띄우고 return

Mayfield

ABOUT GALLERY YOUTUBE COMMUNITY LOCATION JOIN

JOIN US

회원정보

USER ID
아이디를 입력해주세요.
아이디를 5글자 이상 입력하세요

PASSWORD
비밀번호를 입력해주세요.
비밀번호는 5글자 이상, 영문, 숫자, 특수문자를 모두 포함하세요

E-MAIL
이메일주소를 입력해주세요.
이메일은 8글자이상 @를 포함해 입력하세요

AREA
지역을 선택해주세요
지역을 선택하세요

GENDER
Male ● Female ●
성별을 선택하세요

COMMENTS
코멘트를 입력해주세요.

남기는 말은 10글자 이상 입력해주세요

개인정보 수집 및 이용동의(필수)

메이필드서울은 아래의 목적으로 개인정보를 수집 및 이용하며, 회원의 개인정보를 안전하게 취급하는데 최선을 다합니다.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio eaque necessitatibus explicabo in labore culpa nulla modi accusamus nesciunt saepe perspiciatis itaque, eveniet sunt unde ullam ducimus voluptates reiciendis inventore pariatur quidem! Commodi delectus, cupiditate inventore unde impedit quis ab nulla non? illum, quasi sed iste temporibus vero voluptate accusantium magni aliquid velit delectus! Quidem modi ipsa adipisci facere omnis, quam aliquam? Qui facere natus, ex sint ipsa, harum vitae dicta aspernatur nesciunt sapiente magni odio, voluptate ab. Dolorum nihil voluptatum voluptatibus voluptas maxime enim ab, quo debitis quod unde est veniam natus nesciunt ex odio labore rem pariatur ducimus! nulla non? illum, quasi

■ 개인정보 수집 및 이용에 동의합니다.
이용에 동의해주세요

입력내용 초기화 가입하기

Mayfield

buy now f t

2022 Mayfield © ALL RIGHTS RESERVED.

ABOUT COMMUNITY GALLERY YOUTUBE LOCATION JOIN

```

const check = (Val) => {
  const errs = {};
  const eng = /[a-zA-Z]/;
  const num = /[0-9]/;
  const spc = /[~!@#$%^&*()_+]/;

  //userid인증처리
  if (Val.userid.length < 5) {
    errs.userid = "아이디를 5글자 이상 입력하세요";
  }

  //password인증처리
  if (
    Val.pwd1.length < 5 ||
    !eng.test(Val.pwd1) ||
    !num.test(Val.pwd1) ||
    !spc.test(Val.pwd1)
  ) {
    errs.pwd1 =
      "비밀번호는 5글자 이상, 영문, 숫자, 특수문자를 모두 포함하세요";
  }
  if (Val.pwd1 !== Val.pwd2 || !Val.pwd2) {
    errs.pwd2 = "두개의 비밀번호를 동일하게 입력하세요";
  }

  //email인증처리
  if (Val.email.length < 8 || !/@/.test(Val.email)) {
    errs.email = "이메일은 8글자이상 @를 포함해 입력하세요";
  }

  if (!Val.gender) {
    errs.gender = "성별을 선택하세요";
  }

  if (Val.comments.length < 20) {
    errs.comments = "남기는 말은 10글자 이상 입력해주세요";
  }

  if (Val.area === "") {
    errs.area = "지역을 선택하세요";
  }

  if (!Val.agree) {
    console.log("체크안함");
    errs.agree = "이용에 동의해주세요";
  }

  return errs;
};

```