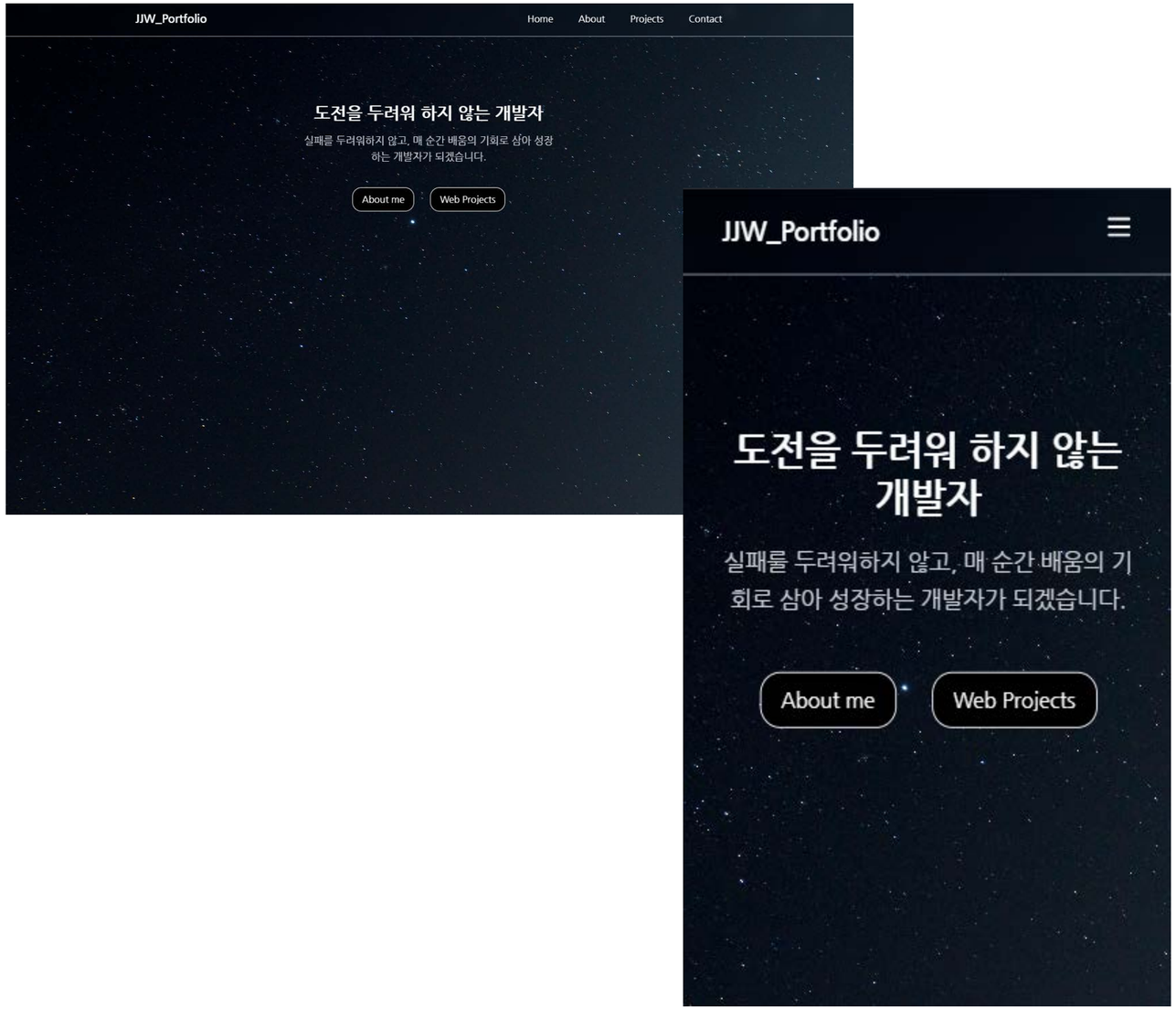
2024.10.28

Portfolio site

개인 포트폴리오 사이트

Portfolio site



프로젝트

- 개인 포트폴리오 사이트

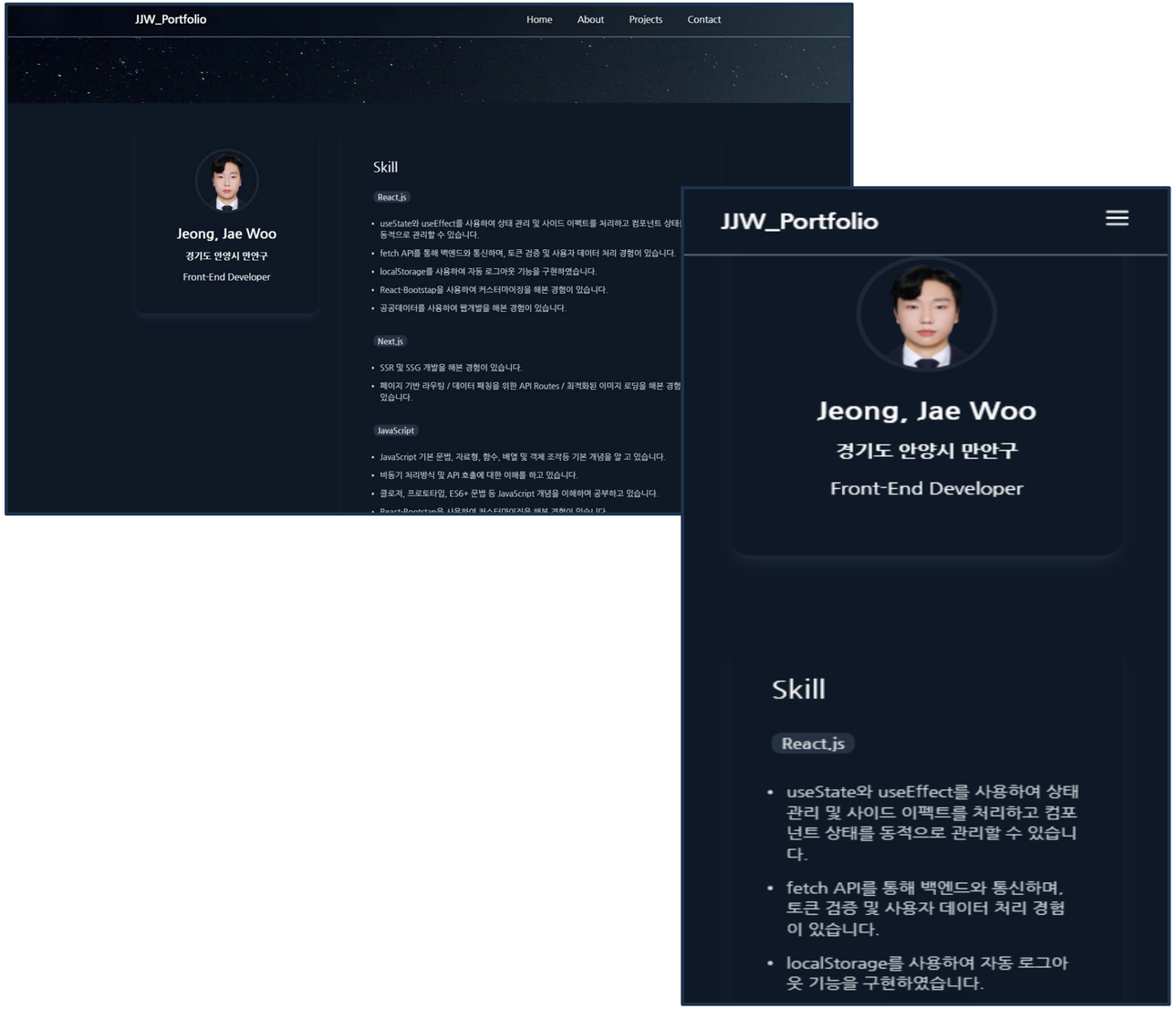
개발 인원 및 기간

- 개발 기간 : 2024.10.14 – 2024.10.16
- 개발 인원 : Front 1명

기술 스택

- Next.js
- Typescript
- Tailwind CSS
- React
- Vercel

Portfolio site

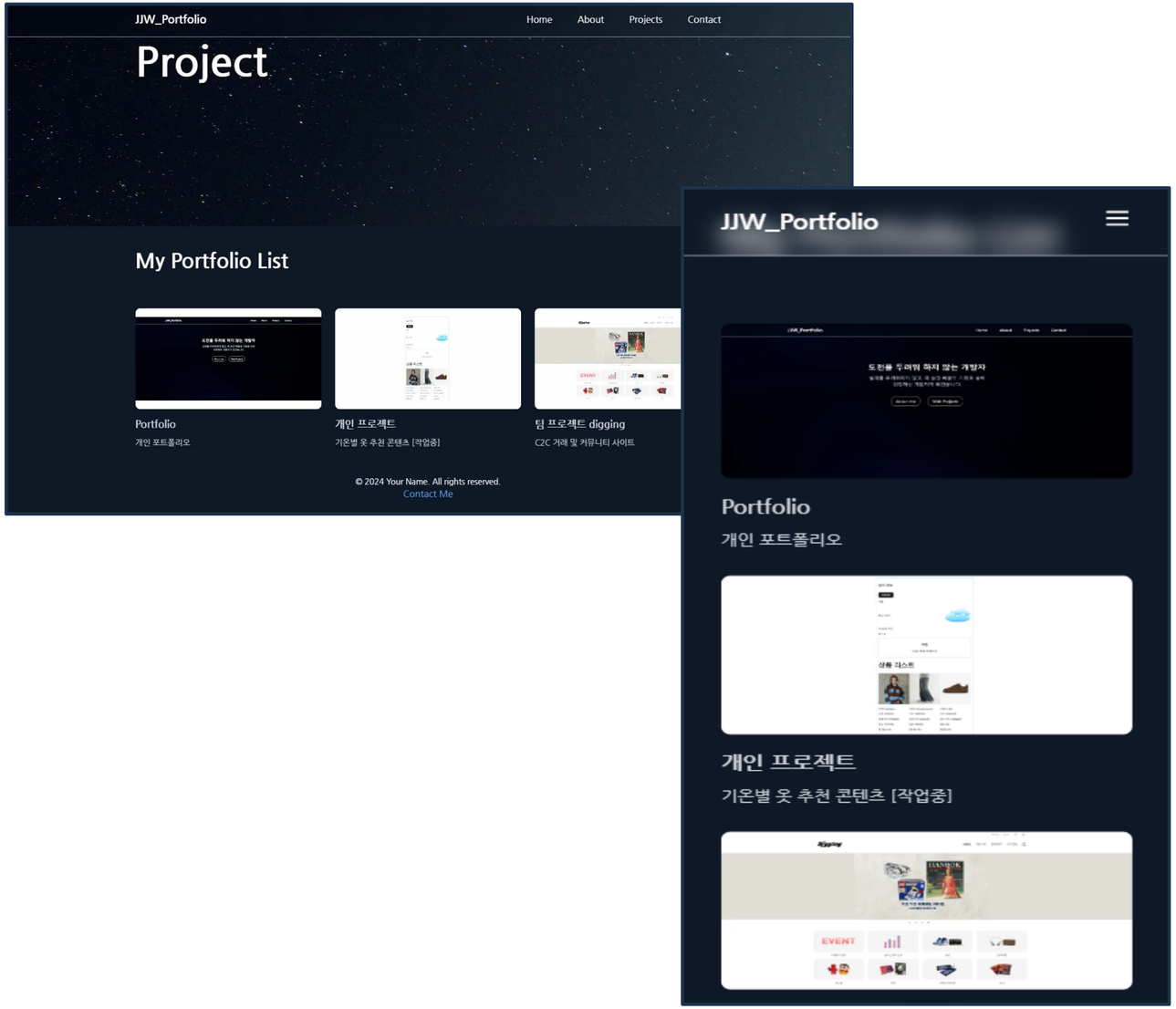


상세 기여 내용

- 초기 로딩 속도를 개선하기 위해 이미지 최적화를 next/image 컴포넌트를 활용하여 lazy loding을 구현

```jsx
import Image from "next/image";

const AboutPage = () => {
  return (
    <div className="bg-customBlue">
      <div className="flex flex-col lg:flex-row min-h-scr
        <div className=" w-full lg:w-80 h-fit rounded-2xl
          <div className="w-28 h-28 rounded-full overflow
            <Image
              src="/image/myphoto1.png"
              alt="myPhoto1"
              width={200}
              height={200}
            />
          </div>
```

Portfolio site

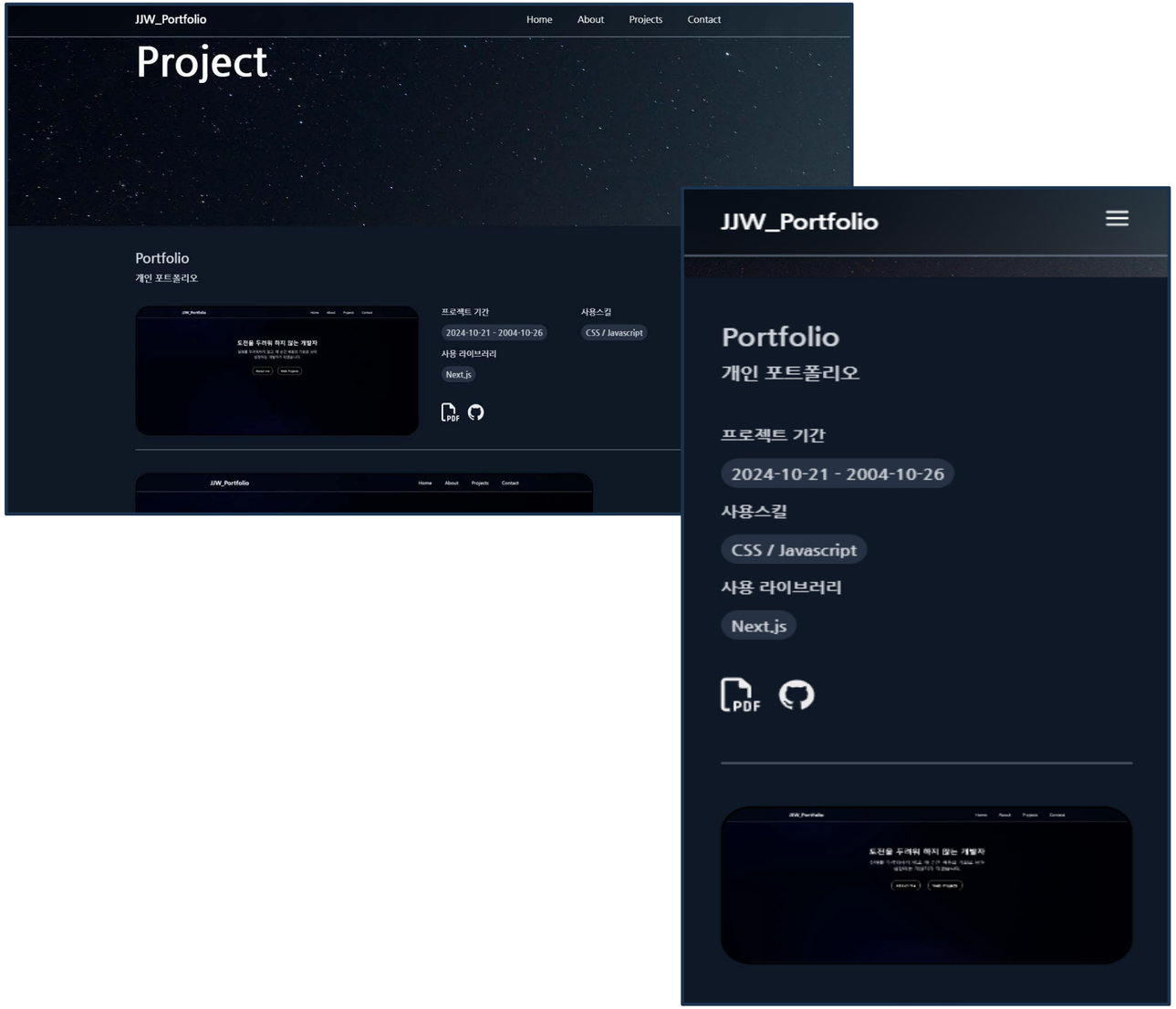


상세 기여 내용

- Projects 배열을 매핑하여 각 프로젝트의 내용을 동적으로 생성하여 코드의 재사용성을 높이고 데이터 변경 시 쉽게 업데이트할 수 있도록 하였습니다.

```
import Link from "next/link";
import Image from "next/image";
import { projects } from "./data/projects";

const ProjectsPage = () => {
  return (
    <div className="bg-customBlue">
      <div className="max-w-5xl  mx-auto px-8 lg:px-0 ">
        <h1 className="text-white text-4xl font-bold py-10">
          My Portfolio List
        </h1>
        <div className="mt-6 grid gap-6 grid-cols-1 md:grid-cols-3 ">
          {projects.map((project, index) => (
            <div key={index}>
              <Link href={`/project/${project.link}`} className="group">
                {/* <div className=" bg-white h-atuo lg:h-44 flex justify-cen

                <div className="bg-white h-auto lg:h-44 flex justify-center
                  <Image
                    src={project.image}
                    alt={project.title}
                    className="object-cover max-w-full h-auto  " //
                    width={500}
                    height={300}
                  />
                </div>
                <div className="pt-3 pointer-events-none">
                  <h2 className="text-lg font-bold  text-gray-300">
                    {project.title}
                  </h2>
                  <p className="text-sm mt-2 text-gray-300">
                    {project.description}
                  </p>
                </div>
              </Link>
            </div>
          ))}
```

Portfolio site

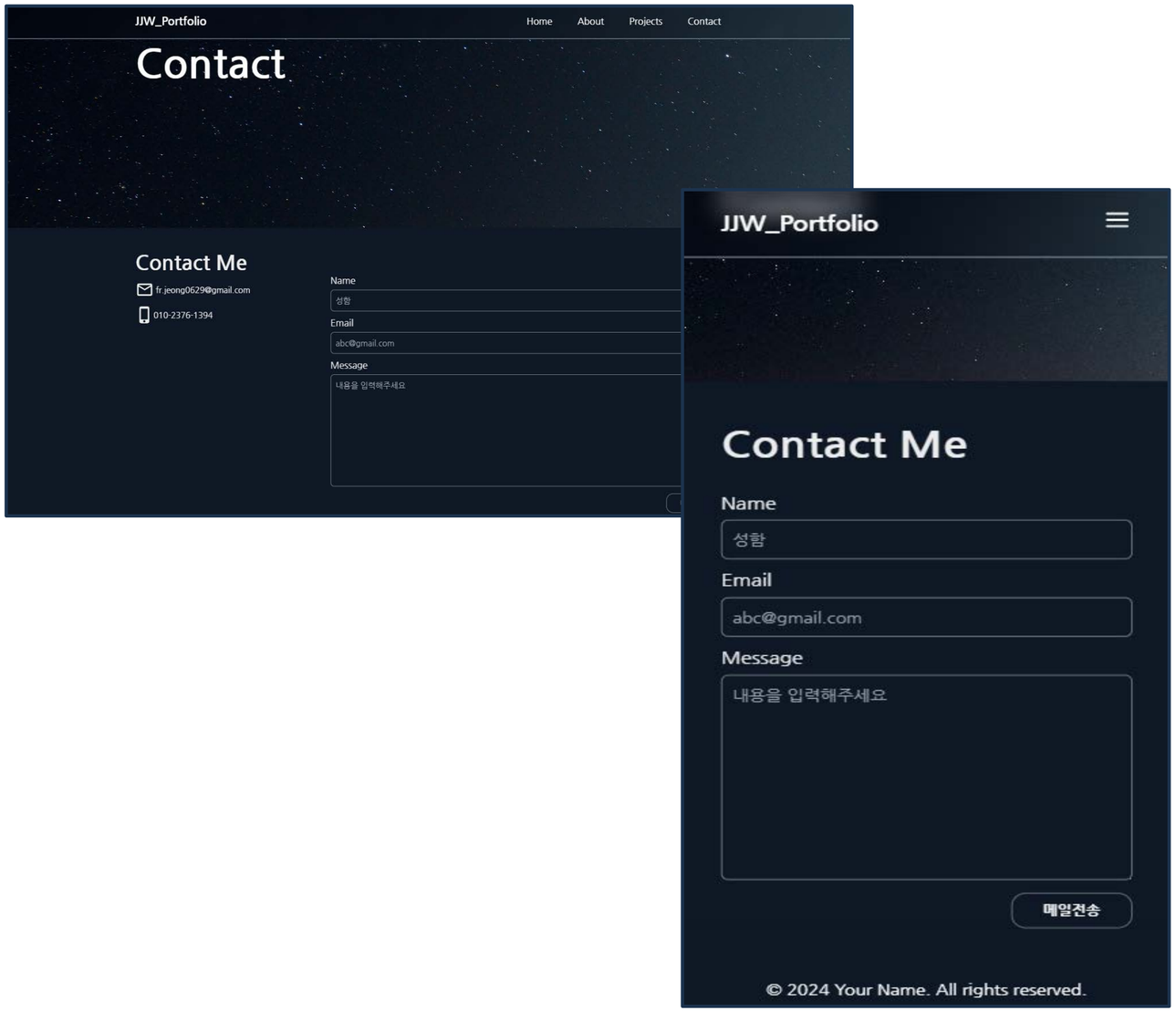


상세 기여 내용

- UsePathname 훅을 사용하여 현재 URL에서 프로젝트 ID를 추출하여, 이를 기반으로 프로젝트 데이터를 가져오는 로직 구현

```tsx
const ProjectDetailPage = () => {
  const pathName = usePathname();
  const projectId = pathName.split("-").pop();
  const id = parseInt(projectId || "0");
  const project = projects[id - 1];

  return (
    <div className="bg-customBlue">
      <div className="max-w-5xl  mx-auto rounded-2xl flex flex-col min-h-screen px-8 lg:px-0  ">
        <div className=" ">
          <p className="text-2xl font-bold text-gray-300 mt-10 mb-2 ">
            {project.title}
          </p>
          <p className="text-base text-gray-300 font-semibold mb-3">
            {project.description}
          </p>
        </div>
        <div className="flex  py-6 border-b-2 border-gray-600 ">
          <div className="  w-1/2 hidden lg:block ">
            <Image
              src={project.image}
              alt={project.title}
              width={500}
              height={400}
              className="rounded-3xl border-2 border-black "
            />
          </div>
          <div className=" flex flex-wrap ml-0 lg:ml-10   ">
            <div className="flex flex-col w-full lg:w-1/2 ">
              <p className="text-sm text-gray-300 font-bold mb-3">
                프로젝트 기간
              </n>
```

Portfolio site



상세 기여 내용

- useState 훅을 사용하여 필드 값을 저장하는 상태객체를 생성
- 각 필드에 대한 유효성 검사를 위한 오류 처리 (errors)
- 기본 폼 제출 동작을 방지하고 이메일 전송 전 입력 필드를 유효성 검사하는 sendEmail 함수 구현
- Emailjs 라이브러리를 사용하여 사용자가 입력한 데이터를 기반으로 이메일 전송하는 기능 구현
- 이메일 전송 시 성공 실패 여부 알림 제공

```
const ContactPage = () => {
  const [formData, setFormdata] = useState<FormData>({
    name: "",
    email: "",
    message: "",
  });

  const [errors, setErrors] = useState<Errors>({});
```

Portfolio site

```tsx
const ContactPage = () => {

  const handleChange = (
    e: React.ChangeEvent<HTMLTextAreaElement | HTMLInputElement>
  ) => {
    const { name, value } = e.target;
    setFormdata({ ...formData, [name]: value });

    setErrors({ ...errors, [name]: "" });
  };

  const sendEmail = (e: FormEvent<HTMLFormElement>) => {
    e.preventDefault();

    const newErrors: { name?: string; email?: string; message?: string } = {};

    // 각 필드에 대해 검증
    if (!formData.name) {
      newErrors.name = "*이름을 입력해 주세요.";
    }
    if (!formData.email) {
      newErrors.email = "*이메일을 입력해 주세요.";
    }
    if (!formData.message) {
      newErrors.message = "*메시지를 입력해 주세요.";
    }

    if (Object.keys(newErrors).length > 0) {
      setErrors(newErrors);
      return;
    }
```

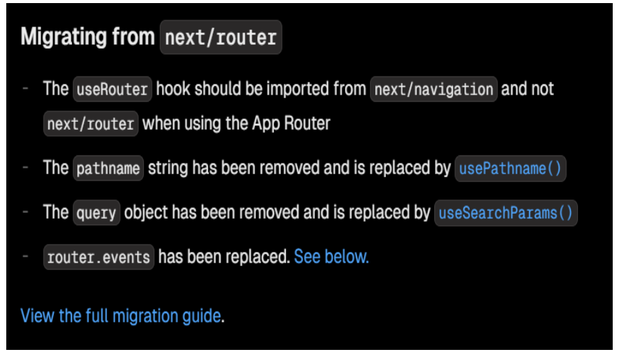
```tsx
emailjs
  .send(
    process.env.NEXT_PUBLIC_SERVICEID as string,
    process.env.NEXT_PUBLIC_TEMPLATEID as string,

    {
      from_name: formData.name,
      from_email: formData.email,
      message: formData.message,
    },
    process.env.NEXT_PUBLIC_PUBLICKET
  )

  .then(
    (result) => {
      console.log("Email sent successfully:", result.text);
      alert("메세지가 전송되었습니다.");

      setFormdata({
        name: "",
        email: "",
        message: "",
      });
    },
    (error) => {
      console.error("Error sending email:", error.text);
      alert("메세지 전송실패");
    }
  );
};
```

상세 기여 내용

- useState 훅을 사용하여 필드 값을 저장하는 상태객체를 생성
- 각 필드에 대한 유효성 검사를 위한 오류 처리 (errors)
- 기본 폼 제출 동작을 방지하고 이메일 전송 전 입력 필드를 유효성 검사하는 sendEmail 함수 구현
- Emailjs 라이브러리를 사용하여 사용자가 입력한 데이터를 기반으로 이메일 전송하는 기능 구현
- 이메일 전송 시 성공 실패 여부 알림 제공

```tsx
<form
  onSubmit={sendEmail}
  className="flex flex-col  w-full lg:w-2/3 ml-0 lg:ml-8"
>
  <label className="block mb-2  ">
    <div className="flex justify-between items-center">
      Name
      {errors.name && (
        <span className="text-xs text-red-500 text-end">
          {errors.name}
        </span>
      )}
    </div>
```

Portfolio site




```tsx
"use client";
import { IoLogoGithub } from "react-icons/io5";
import { FaRegFilePdf } from "react-icons/fa6";
import { usePathname } from "next/navigation";
import { projects } from "../data/projects";
import Image from "next/image";

const ProjectDetailPage = () => {
  const pathName = usePathname();
  const projectId = pathName.split("-").pop();
  const id = parseInt(projectId || "0");
  const project = projects[id - 1];

  return (
    <div className="bg-customBlue">
      <div className="max-w-5xl  mx-auto rounded-2xl flex flex-col min-h-screen px-8 lg:px-0  ">
        <div className=" ">
          <p className="text-2xl font-bold text-gray-300 mt-10 mb-2 ">
            {project.title}
          </p>
          <p className="text-base text-gray-300 font-semibold mb-3">
            {project.description}
          </p>
        </div>
        <div className="flex  py-6 border-b-2 border-gray-600 ">
          <div className="  w-1/2 hidden lg:block ">
            <Image
              src={project.image}
              alt={project.title}
              width={500}
              height={400}
              className="rounded-3xl border-2 border-black "
            />
          </div>
```

문제 해결 경험

- Next.js 프로젝트에서 경로명을 관리하기 위해 useRouter 훅을 사용하였으나, 오류로 인해 제대로 동작하지 않았습니다. 공식 문서를 확인해본 결과 Next.js 13 이상에서는 useRouter 대신 usePathname, useSearchParms 등의 훅을 사용하여 라우팅을 관리해야 한다는 것을 알게 되었습니다. 이를 통해 보다 효과적으로 경로 및 쿼리 파라미터를 처리할 수 있음을 이해하였습니다.

Portfolio site

개선 점

[코드 리팩토링]

handleChane 함수와 같은 이벤트 핸들러를 별도의 커스텀 훅으로 만들어 깔끔한 코드 작성

[상태 관리 개선]

현재 상태 관리를 위해 useState를 사용하고 있지만, 복잡한 상태를 관리해야 하는 경우 useReducer 를 고려해 볼 수 있습니다.

useReducer : 상태 업데이트 로직을 정의하는 순수함수이다. 여기서 순수 함수란 동일한 입력값이 주어졌을 때 항상 동일한 결과를 반환한다.

[테스트 코드 추가]

현재 작성한 코드에는 테스트 코드가 없기 때문에, 해당 기능들이 제대로 작동하는지 확인할 수 있는 메커니즘이 없습니다.

1. 기능검증
2. 버그 발견
3. 유지보수 용이성
4. 신뢰성 향상

이와 같은 이유로 테스트 코드 작성의 중요성을 알았으며 앞으로의 프로젝트에서는 테스트 코드를 포함하여 개발을 진행해야겠다고 다짐했습니다.

Portfolio site

개인 포트폴리오 사이트