Portfolio Conversion Documentation

Darshan M
darshanmrd17@gmail.com

Portfolio Conversion Documentation

Introduction: This documentation outlines the process of converting a static HTML portfolio template into a dynamic React.js application while integrating API data and enhancing its visual appeal with animations using the Framer Motion library. The focus was on fetching data from the API endpoint for specific sections of the portfolio template, including the Hero Section, Services, Projects, Skills, Testimonials, and Social Media Handles.

Table of Contents:

1. Project Overview:

Original HTML Template

The original HTML template consisted of various sections, including the Hero Section, Services, Projects, Skills, Testimonials, and Social Media Handles.

Objectives

Convert the static HTML template into a dynamic React.js application.

Fetch data from the API endpoint for the Hero Section, Services, Projects, Skills, Testimonials, and Social Media Handles.

Render dynamic content based on the fetched data within React components.

Enhance the visual appeal with animations using the Framer Motion library.

Tools and Technologies

- React.js
- JavaScript
- HTML
- CSS
- API (to fetch data)

- Framer Motion

2. React.js Conversion:

Componentization

Identified each section of the portfolio template and created separate React components for Hero Section, Services, Projects, Skills, Testimonials, and Social Media Handles.

Utilized props to pass dynamic data to the components.

3. Integration of API Data:

3.1 Fetching Data:

Utilized APIs to fetch data dynamically for the Hero Section, Services, Projects, Skills, Testimonials, and Social Media Handles.

3.2 Rendering Data:

Used JavaScript array methods like map() to iterate over API data and render dynamic content within React components.

Example Code Snippet:

```
export default function Experience() {
  const [experiences, setexperiences] = useState([]);
  const [selected, setselected] = useState(null); // selects the
current section and shows the description of the choosen one
  const [enable, setenable] = useState(false); //to see weather the
current section is selected or not

  const fetchdata = async () => {
    const data = await fetch(
      "https://portfolio-backend-
30mp.onrender.com/api/v1/get/user/65b3a22c01d900e96c4219ae"
    );
    const res = await data.json();

    setexperiences(res.user.timeline);
  };
  useEffect(() => {
    fetchdata();
  }, []);
  const handleenable = (id) => {
```

Portfolio Conversion Documentation

```
    setselected(id === selected ? null : id);

    setenable(!enable);
  };
  //framer motion library used to animate things a bit :)
  const variants = {
    initial: {
      opacity: 0,
      y: -220,
    },
    animation: {
      opacity: 1,
      y: 0,
      transition: {
        duration: 0.4,
        type: "spring",
        stiffness: 400,
        damping: 40,
      },
    },
  };

  return (
    <>
      {experiences.map((exp) => {
        return (
          <>
            <div>
              <div
                className="experience-cont"
                style={{ width: "100%", height: "100px", cursor:
"pointer" }}
                onClick={() => {
                  handleenable(exp._id);
                }}
              >
                <button className="experiencetoggle">+ </button>
                <h2 class="section-subheading">
                  <span>{exp.company_name}</span>
                </h2>

                <h1>{exp.jobTitle}</h1>
              </div>

              <div>
                {exp.bulletPoints.map((points) => {
                  return (
                    <>
```

Portfolio Conversion Documentation

```
                    {selected === exp._id && (
                      <motion.div
                        className="bullet_points"
                        variants={variants}
                        initial="initial"
                        animate="animation"
                      >
                        <p>{points}</p>
                      </motion.div>
                    )}
                  </>
                );
              })}
            </div>
          </div>
        </>
      );
    })}
  </>
);
}
```

4. **Styling:**

   Integrated CSS classes from the original CSS file to style React components.

   Made adjustments and additions to CSS as necessary to ensure proper styling and layout.

5. **Animation with Framer Motion:**

   Used Framer Motion to add animations to various components, such as fade-ins, transitions, and hover effects.

   Enhanced user experience and visual appeal by incorporating subtle animations throughout the portfolio.

6. **Conclusion:**

   The conversion of the static HTML portfolio template into a dynamic React.js application was successful. By integrating API data for specific sections such as the Hero Section, Services, Projects, Skills, Testimonials, and Social Media Handles, and incorporating animations using the Framer Motion library, the portfolio now showcases dynamic content with enhanced visual appeal. React components render data fetched from the API, providing a more interactive and engaging user experience.