

|  |
| --- |
| Website for Raj Dai |
|  |
| June 21  Ground Stack Services  Authored by: Sunil Dangal |



# Handyman & Landscaping website

|  |
| --- |
| Using Modern UI/UX in React JS with Tailwind At first, I am going to setup React Project using Vite. After that I will also setup Tailwind CSS. Tailwind CSS will make it easier to style our application’s.  I am going through the [Tailwind CSS](https://tailwindcss.com/) website and seeing the documentation for setting up Tailwind CSS in a Vite project. I am going to select tailwindcss v3.4.17 version to create using react.  I am going to create a folder name as rajdaiwebsite and open it in vs code. Inside that I am going to install Vite latest.  npm create vite@latest ./ --template react    After that, I am going to install all the dependencies  npm install  And to see in local host, I ran  npm run dev  And we can see live Vite + React in the <http://localhost:5173/> link. Now, again, I am going to install tailwindcss  npm install -D tailwindcss@3 postcss autoprefixer  npx tailwindcss init -p |

Now, next step is to configure the template paths as seen in tailwindcss website. So, I am going to copy this

content: [

"./index.html",

"./src/\*\*/\*.{js,ts,jsx,tsx}",

],

and paste it inside tailwind.config.js file and it would look like this.

/\*\* @type {import('tailwindcss').Config} \*/

export default {

content: [

"./index.html",

"./src/\*\*/\*.{js,ts,jsx,tsx}",

],

theme: {

extend: {},

},

plugins: [],

}

And then it will know, where to find the tailwindcss class names. Next, as from the [Tailwind CSS](https://v3.tailwindcss.com/docs/guides/vite) website documentation, we need to add the tailwindcss directives to my CSS. So, I am going copy all this:

@tailwind base;

@tailwind components;

@tailwind utilities;

I will paste it inside the index.css file. It is inside src/index.css directory. Make sure to remove the existing code inside before pasting and it would look like this:

@tailwind base;

@tailwind components;

@tailwind utilities;

After this, I am going to run this for build process as per the website instructions:

npm run dev

And after this, I will make some changes to App.jsx file by adding Hello raj dai header.

I have removed the existing codes whichever we didn’t needed for this project.

import './App.css'

function App() {

return (

<h1 className="text-3xl font-bold underline">

Hello raj dai !

</h1>

)

}

export default App

And we can see the output as: Hello raj Dai! (Underlines and bold)

A screenshot of a computer

Description automatically generated

Output is underlined and bold that’s shows tailwind is correctly reading utility class names. We successfully have setup Tailwind CSS with Vite project.

Now, I am going to format the files and folder just to make it bit cleaner and easy to navigate.

Current Project Structure Look like this:

rajdaiwebsite/

├── .vscode/ # VS Code settings

│ └── settings.json

│

├── node\_modules/ # Dependencies (auto-generated)

│

├── public/ # Public assets (served as-is)

│ └── favicon.ico

│ └── logo.png

│ └── robots.txt

│ └── ...

│

├── src/ # Your actual app source code

│ ├── assets/ # Images, icons, and static media

│ │ └── logo.svg

│ │ └── hero.jpg

│ │ └── handyman.png

│ │ └── landscaping.png

│ │

│ ├── components/ # Reusable UI components

│ │ ├── Header.jsx

│ │ ├── Hero.jsx

│ │ ├── Services.jsx

│ │ ├── About.jsx

│ │ └── Footer.jsx

│ │

│ ├── pages/ # Full-page components

│ │ ├── Home.jsx

│ │ ├── ServicesPage.jsx

│ │ ├── AboutPage.jsx

│ │ └── ContactPage.jsx

│ │

│ ├── styles/ # Optional custom styles or Tailwind extensions

│ │ └── typography.css

│ │

│ ├── App.jsx # Root component

│ ├── App.css # App-specific styles (minimal if using Tailwind)

│ ├── index.css # Tailwind base import

│ ├── main.jsx # App entry point

│

├── .gitignore # Files and folders to ignore in Git

├── rajdaiwebsite\_documentation.docx # Project documentation (can move to /docs)

├── sling.config.js # Custom config (if used)

├── index.html # HTML template (used by Vite)

├── package.json # Project metadata and dependencies

├── package-lock.json # Dependency lockfile

├── postcss.config.js # Tailwind/PostCSS config

├── tailwind.config.js # Tailwind custom configuration

├── vite.config.js # Vite project config

├── README.md # Project readme

I have created settings.json file inside vscode folder because, to specifically apply vs code settings just for this project. settings.json look like this:

{

"editor.defaultFormatter": "esbenp.prettier-vscode",

"editor.formatOnSave": true,

"editor.codeActionsOnSave": {

"source.fixAll.eslint": "explicit",

"source.addMissingImports": "explicit"

},

"prettier.tabWidth": 2,

"prettier.useTabs": false,

"prettier.semi": true,

"prettier.singleQuote": false,

"prettier.jsxSingleQuote": false,

"prettier.trailingComma": "es5",

"prettier.arrowParens": "always",

"[javascriptreact]": {

"editor.defaultFormatter": "esbenp.prettier-vscode"

},

"[css]": {

"editor.defaultFormatter": "vscode.css-language-features"

},

"[svg]": {

"editor.defaultFormatter": "jock.svg"

}

}

a

Now, I am going to create a cleaner project structure. I am going to delete App.css which is inside src folder and remove import App.css from App.js file. All the styles, I am going to user will be inside index.css file.

I am going to update tailwind.config.js file with the color, plugins, effects that I am going to user for this project. I am adding special class names for h1, h2, h3, font family, colors, etc. and so on. This will be much easier later because we don’t have to declare color individually. Currently it looks like this:

/\*\* @type {import('tailwindcss').Config} \*/

import { fontFamily } from "tailwindcss/defaultTheme";

import plugin from "tailwindcss/plugin";

export default {

content: [

"./index.html",

"./src/\*\*/\*.{js,ts,jsx,tsx}",

"./public/assets/\*\*/\*.{js,ts,jsx,tsx}",

],

theme: {

extend: {

colors: {

brand: {

primary: "#2F855A", // Forest Green - nature, trust

secondary: "#8B5E3C", // Earthy Brown - wood, craftsmanship

accent: "#A3D977", // Leafy Green - freshness

highlight: "#F6AD55", // Warm Yellow - attention

light: "#F5F3EB", // Light neutral background

dark: "#2D3748", // Dark text/background

},

neutral: {

100: "#ffffff",

200: "#f5f5f5",

300: "#e2e8f0",

400: "#cbd5e1",

500: "#94a3b8",

600: "#64748b",

700: "#475569",

800: "#334155",

900: "#1e293b",

},

},

fontFamily: {

sans: ["var(--font-sora)", ...fontFamily.sans],

code: "var(--font-code)",

grotesk: "var(--font-grotesk)",

},

letterSpacing: {

tagline: ".15em",

},

spacing: {

0.25: "0.0625rem",

7.5: "1.875rem",

15: "3.75rem",

},

opacity: {

15: ".15",

},

transitionDuration: {

DEFAULT: "200ms",

},

transitionTimingFunction: {

DEFAULT: "linear",

},

zIndex: {

1: "1",

2: "2",

3: "3",

4: "4",

5: "5",

},

borderWidth: {

DEFAULT: "0.0625rem",

},

backgroundImage: {

"radial-gradient": "radial-gradient(var(--tw-gradient-stops))",

"conic-gradient":

"conic-gradient(from 225deg, #F6AD55, #A3D977, #2F855A, #8B5E3C)",

"hero-pattern": "url('/src/assets/hero-bg.jpg')",

},

},

},

plugins: [

plugin(function ({ addBase, addComponents, addUtilities }) {

addBase({});

addComponents({

".container": {

"@apply max-w-[77.5rem] mx-auto px-5 md:px-10 lg:px-15 xl:max-w-[87.5rem]":

{},

},

".h1": {

"@apply font-semibold text-[2.5rem] leading-[3.25rem] md:text-[2.75rem] md:leading-[3.75rem] lg:text-[3.25rem] lg:leading-[4.0625rem] xl:text-[3.75rem] xl:leading-[4.5rem]":

{},

},

".h2": {

"@apply text-[1.75rem] leading-[2.5rem] md:text-[2rem] md:leading-[2.5rem] lg:text-[2.5rem] lg:leading-[3.5rem] xl:text-[3rem] xl:leading-tight":

{},

},

".h3": {

"@apply text-[2rem] leading-normal md:text-[2.5rem]": {},

},

".h4": {

"@apply text-[2rem] leading-normal": {},

},

".h5": {

"@apply text-2xl leading-normal": {},

},

".h6": {

"@apply font-semibold text-lg leading-8": {},

},

".body-1": {

"@apply text-[0.875rem] leading-[1.5rem] md:text-[1rem] md:leading-[1.75rem] lg:text-[1.25rem] lg:leading-8":

{},

},

".body-2": {

"@apply font-light text-[0.875rem] leading-6 md:text-base": {},

},

".caption": {

"@apply text-sm": {},

},

".tagline": {

"@apply font-grotesk font-light text-xs tracking-tagline uppercase":

{},

},

".quote": {

"@apply font-code text-lg leading-normal": {},

},

".button": {

"@apply font-code text-xs font-bold uppercase tracking-wider": {},

},

});

addUtilities({

".tap-highlight-color": {

"-webkit-tap-highlight-color": "rgba(0, 0, 0, 0)",

},

});

}),

],

};

So basically, I have used these colors:

## 🎨 Recommended Color Palette

| Purpose | Color Name | Hex Code | Notes |
| --- | --- | --- | --- |
| **Primary** | Forest Green | #2F855A | Represents nature, plants, trees – ideal for landscaping/gardening |
| **Secondary** | Earth Brown | #8B5E3C | Wood/decking/carpentry tone, grounding and professional |
| **Accent** | Leaf Green | #A3D977 | Freshness, growth – use for call-to-actions or highlights |
| **Neutral Light** | Warm Beige | #F5F3EB | Background color, easy on the eyes |
| **Neutral Dark** | Charcoal Gray | #2D3748 | For text, contrast, footer |
| **Highlight** | Golden Yellow | #F6AD55 | Use sparingly for buttons or hover states, adds warmth, and draws attention |

What I have done inside tailwind.config.js is that I have declared above mentioned colors as well as added plugins already. This allows me to define the components like .container, .h1 and so on later and use in the website.

Now, I am going to update index.css file with the following styles:

/\* Import Google Fonts \*/

@import url("https://fonts.googleapis.com/css2?family=Sora:wght@300;400;600&display=swap");

@import url("https://fonts.googleapis.com/css2?family=Source+Code+Pro:wght@400;600;700&display=swap");

@import url("https://fonts.googleapis.com/css2?family=Space+Grotesk:wght@300&display=swap");

/\* Tailwind Directives \*/

@tailwind base;

@tailwind components;

@tailwind utilities;

/\* CSS Variables for Fonts \*/

:root {

--font-sora: "Sora", sans-serif;

--font-code: "Source Code Pro", monospace;

--font-grotesk: "Space Grotesk", sans-serif;

}

/\* Global Styles \*/

\* {

scroll-behavior: smooth;

}

@layer base {

body {

@apply font-sans bg-brand-light text-brand-dark text-base;

}

}

/\* Rotation Utilities \*/

.rotate-45 {

@apply rotate-[45deg];

}

.rotate-90 {

@apply rotate-[90deg];

}

.rotate-135 {

@apply rotate-[135deg];

}

.rotate-180 {

@apply rotate-[180deg];

}

.rotate-225 {

@apply rotate-[225deg];

}

.rotate-270 {

@apply rotate-[270deg];

}

.rotate-315 {

@apply rotate-[315deg];

}

.rotate-360 {

@apply rotate-[360deg];

}

.-rotate-45 {

@apply -rotate-[45deg];

}

.-rotate-90 {

@apply -rotate-[90deg];

}

.-rotate-135 {

@apply -rotate-[135deg];

}

.-rotate-180 {

@apply -rotate-[180deg];

}

.-rotate-225 {

@apply -rotate-[225deg];

}

.-rotate-270 {

@apply -rotate-[270deg];

}

.-rotate-315 {

@apply -rotate-[315deg];

}

.-rotate-360 {

@apply -rotate-[360deg];

}

Inside the index.css file, I have applied class names for rotation’s animations. I have imported font’s from google and selected the fonts. I have also used light color scheme because this website for the business will be outdoor, trust-based, and natural. Now this is just a basic base class name, I am going to setup more class names & write more styles later in this project. Also, I am going to work on arranging project file and folder structure too.

Now, I am going to install react-router-dom by opening up in another terminal.

npm install react-router-dom

I am going to go to main.jsx inside src folder and I am going to wrap the entire application with a router. After that, I have also added favicon icon inside assets folder and set the targeted location to load up the icon inside index.html.

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="./src/assets/favicon.svg" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Choden Landscaping and Handyman Services</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.jsx"></script>

</body>

</html>

Now, I am going to work on making the layouts. In this project, App.jsx is going to be the homepage for this website. I am going to make some changes to App.jsx file.

I have declared App with ES6 const function and wrapped h1 with react fragment that allows later to add more elements inside the fragment. So I have downloaded and added ButtonGradient.svg file inside. /src/assets/svg folder. Below the h1 elements, I am going to add button gradient to render and import it as well. I am doing this to give gradient effect to the buttons.

const ButtonGradient = () => {

return (

<svg className="block" width={0} height={0}>

<defs>

<linearGradient id="btn-left" x1="50%" x2="50%" y1="0%" y2="100%">

<stop offset="0%" stopColor="#89F9E8" />

<stop offset="100%" stopColor="#FACB7B" />

</linearGradient>

<linearGradient id="btn-top" x1="100%" x2="0%" y1="50%" y2="50%">

<stop offset="0%" stopColor="#D87CEE" />

<stop offset="100%" stopColor="#FACB7B" />

</linearGradient>

<linearGradient id="btn-bottom" x1="100%" x2="0%" y1="50%" y2="50%">

<stop offset="0%" stopColor="#9099FC" />

<stop offset="100%" stopColor="#89F9E8" />

</linearGradient>

<linearGradient

id="btn-right"

x1="14.635%"

x2="14.635%"

y1="0%"

y2="100%"

>

<stop offset="0%" stopColor="#9099FC" />

<stop offset="100%" stopColor="#D87CEE" />

</linearGradient>

</defs>

</svg>

);

};

export default ButtonGradient;

There are lots of linearGradient inside ButtonGradient effect. They will be used within buttons. There are different IDs in those LinearGradients and I am going to use them later accordingly.

Now, I am going to create a folder inside src folder named as components and create component there as Button.jsx

Inside Button.jsx, I am going to write rafce and it will automatically create reactarrowFunctionExportComponent for faster process or using github copilot or by downloading extensions called as ES7 React/Redux/React-Native/JS snippets by rodrigovallades. Button.jsx looks like this:

const Button = () => {

return <div> Button </div>;

};

export default Button

Now, I am going to import this Button.jsx inside App.jsx for testing purpose. So inside App.jsx, I am going to add inside the <> <> (react fragment) within a div, and div will have class name with padding top, large devices padding top and overflow hidden. I am also going to give margin top class for button with href of login.

App.jsx

import ButtonGradient from "./assets/svg/ButtonGradient";

import Button from "./assets/components/Button";

const App = () => { {/\* Declaring App with es6 const function \*/}

return (

<> {/\* Wrapping inside react fragment to add more elements later \*/}

<h1 className="text-3xl font-bold underline"> Hello raj dai </h1>

<div className="pt-[4.75rem] lg:pt-[5.25rem] overflow-hidden">

<Button className="mt-10" href="#login">

Test

</Button>

</div>

<ButtonGradient />

</>

);

};

export default App

Now, I checked the output, and I can still see Button text only, I cannot see the Button effects. To fix this, I am going to transform the button into reusable component. I have started with Button only first to understand how to transform something to reuse later in this project to save time and learn more to make complex reusable components for future projects.

I am going to start now by making this button accept some of the props. Therefore I am going to destructure the props by adding, className, href, onclick, px, color values or etc. There are so much things I can add in there, but for now let’s do this inside Button.jsx

Sometimes, lets say if I want to make the button a link, or rectangular button or whatever. To make that happen, I am going to render two sub functions called as const renderButton with arrow function, instant return where it returns a button components. And this button component will render a span element rendering the children and it will also render the special button SVG component where we will pass white color.

I am going to import ButtonSVG from the file path ./assets/svg/ButtonSvg and in the end I will return the renderButton component. This is what Button.jsx look like:

import ButtonSvg from '../svg/ButtonSvg'

const Button = ({className, href, onclick, children, px, white}) => {

const renderButton = () => (

<button>

<span> {children} </span>

{ButtonSvg(white)}

</button>

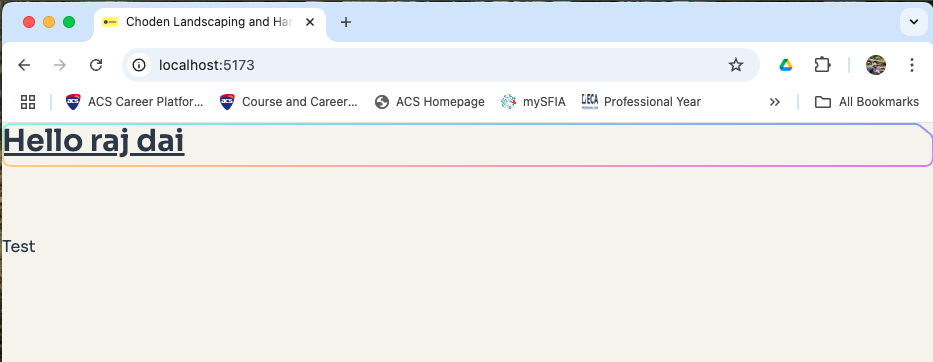
);

return renderButton();

};

export default Button;

Now the output is like this now:



Now, I am going to modify the styles more. I am going to add more additional classes in Button.jsx file. I have added this:

const classes = `button realtive inline-flex items-center justify-center h-11 transition-colors

hover:text-color-1 ${px || "px-7"} ${white ? "text-n-8" : "text-n-1"} ${className || ""}`;

And I can see, inside App.jsx, when I change padding top or margin top value, it just changes the position and rest of the effects remain same, which means I can reuse it in different places preserving its effects. That’s why it is better this way.

I am going to make it more reusable. I am going to define const span classes and I will give it relative z-10 and use that class in the button. Now let’s say if I wanted to render a link.

For that, I am going to create a new function named as renderLink and I will be having renderButton and renderLink and I can choose whenever I want to use them. If the href property exists then, it shows link else renderButton. App.jsx =

import ButtonGradient from "./assets/svg/ButtonGradient";

import Button from "./assets/components/Button";

const App = () => { {/\* Declaring App with es6 const function \*/}

return (

<> {/\* Wrapping inside react fragment to add more elements later \*/}

<h1 className="text-3xl font-bold underline"> Hello raj dai </h1>

<div className="pt-[4.75rem] lg:pt-[5.25rem] overflow-hidden">

<Button className="mt-10" href="#login">

Test

</Button>

</div>

<ButtonGradient />

</>

);

};

export default App

Now if we change it to href then is will show link and if we remove href inside the button then it will act as a normal button.

import ButtonSvg from '../svg/ButtonSvg'

const Button = ({className, href, onclick, children, px, white}) => {

{/\*Creating new property classes to define in below button\*/}

const classes = `button realtive infline-flex items-center justify-center h-11 transition-colors

hover:text-color-1 ${px || "px-7"} ${white ? "text-n-8" : "text-n-1"} ${className || ""}`;

{/\* creating span classes \*/}

const spanClasses = 'relativez-10'

// creating renderButton to render button with classes and children

const renderButton = () => (

<button className = {classes}> {/\* Using classes property that we made at the top from parent \*/}

<span className = {spanClasses}> {children} </span> {/\* Using parent spanClasses property \*/}

{ButtonSvg(white)}

</button>

);

// creating renderLink to render

const renderLink = () => (

<a href={href} className={classes}>

<span className={spanClasses}>{children}</span>

{ButtonSvg(white)}

</a>

)

return href

? renderLink()

: renderButton(); {/\* If href is present then renderLink else renderButton \*/}

};

export default Button;

By this way, I can also expand this component later in the future to make it more dynamic. Now I am going to go inside the components folder and create new file Header.jsx

Now inside Header.jsx, I wrote rafce and it automatically add reactArrowFunctionExportComponent. I am going to import react because I don’t need it. And, inside App.jsx, I am going to remove the button and add Header because I am going to render header to learn to make reusable header component. I have also removed the h1 element. I have left the padding top there because the header is fixed, and I don’t want to overlap with the content. And the top, I will show the nav bar. Now, I will head over to the heard and start creating. App.jsx =

import ButtonGradient from "./assets/svg/ButtonGradient";

import Button from "./assets/components/Button";

const App = () => { {/\* Declaring App with es6 const function \*/}

return (

<> {/\* Wrapping inside react fragment to add more elements later \*/}

<div className="pt-[4.75rem] lg:pt-[5.25rem] overflow-hidden">

<Header /> {/\* Importing Header component from components folder \*/}

</div>

<ButtonGradient />

</>

);

};

export default App;

Header.jsx =

const Header = () => {

return (

<div>

</div>

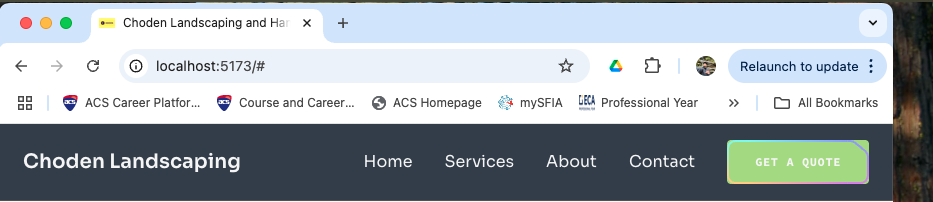
)

}

export default Header

I am going to start with the structure by returning a div element with a classname inside. I have added a padding to the top of the page with anchor tag which will render the image.

Now what I did is, I have used AI Assistance help for me to generate new tailwind.config.js, Header.jsx and index.css file using a prompt: [this is my current tailwind.config.js = [], Header.jsx and index.css = [], Now give me professional one for Handyman and Landscaping single page website]. Now after that, I was able to generate the header:



Now, I am going to replace Choden Landscaping with logo later.

I have also created a new folder inside src folder called constants folder. Now based on my simple, professional, single-page landscaping and handyman website), here's a cleaner and more maintainable structure for my project which is created using AI assistance. After creating revised project structure, the nav bar did not showed up in the browser. There was an issue inside Button.jsx Previously I was calling like this:

{ButtonSvg(white)} //

This is wrong, so I did this:

<ButtonSvg white={white} />

Now check below my project structure and revised Button.jsx

rajdaiwebsite/

├── .git/ # Git version control (hidden)

├── .vscode/ # VSCode settings (hidden)

├── node\_modules/ # Installed packages (auto-generated)

├── public/ # Static files (favicon, etc.)

│ └── favicon.svg

├── src/ # Source code

│ ├── assets/ # Images, fonts, SVGs

│ │ ├── images/ # JPG/PNG/WebP images

│ │ ├── svg/ # SVG assets

│ │ └── favicon.svg

│ ├── components/ # Reusable UI components

│ │ ├── Button.jsx

│ │ ├── Header.jsx

│ │ └── svg/ # SVG React components (if needed as JSX)

│ │ ├── ButtonGradient.jsx

│ │ └── ButtonSvg.jsx

│ ├── constants/ # Static data (e.g., links, content)

│ ├── styles/ # Custom CSS/Tailwind overrides

│ │ └── index.css

│ ├── App.jsx # Root component

│ └── main.jsx # Entry point for React + Vite

├── .gitignore

├── eslint.config.js

├── index.html # Entry HTML (Vite uses this)

├── package.json

├── package-lock.json

├── postcss.config.js

├── tailwind.config.js

├── vite.config.js

├── rajdaiwebsite\_documentation.docx # Project documentation

└── README.md # Project overview

import ButtonSvg from '../assets/svg/ButtonSvg.jsx';

const Button = ({ className, href, onClick, children, px, white }) => {

// Construct Tailwind classes

const classes = `button relative inline-flex items-center justify-center h-11 transition-colors hover:text-brand-highlight ${px || "px-7"} ${white ? "text-brand-dark" : "text-neutral-100"} ${className || ""}`;

const spanClasses = 'relative z-10';

// Render as <button>

const renderButton = () => (

<button className={classes} onClick={onClick}>

<span className={spanClasses}>{children}</span>

<ButtonSvg white={white} />

</button>

);

// Render as <a>

const renderLink = () => (

<a href={href} className={classes}>

<span className={spanClasses}>{children}</span>

<ButtonSvg white={white} />

</a>

);

return href ? renderLink() : renderButton();

};

export default Button;

It looks like this now:

A screenshot of a computer

Description automatically generated

Now Next step is to create a layout and structure inside App.js

This will create structure for scrolling and linking sections, etc., that I need.

I am going to add sections that will contain service cards, contact, etc., inside App.jsx

These service cards can be used to show, services, about us, contact form, etc. inside this single page website. So, I have added this structure inside App.jsx, below header and inside div. I have also added scroll-smooth inside div class to have smooth scrolling for good user experience.

<main>

<section id="home" className="min-h-screen flex items-center justify-center bg-hero-pattern bg-cover bg-center text-white">

<h1 className="text-4xl md:text-5xl font-bold">Welcome to Choden Landscaping</h1>

</section>

<section id="services" className="py-section bg-brand-light text-brand-dark">

{/\* I'll add service cards here later \*/}

</section>

<section id="about" className="py-section bg-white text-brand-dark">

{/\* Company history / experience / values \*/}

</section>

<section id="contact" className="py-section bg-brand-dark text-white">

{/\* Contact form or contact info \*/}

</section>

</main>

As you can see, I have selected background hero-pattern and if you can see inside tailwind configuration file, I have already declared hero-bg.jpg inside it and we can see the background image and h1 text. Now, I will add service card to show the services for this business.

Inside components folder, I am going to create ServiceCard.jsx file and add below codes to show the services.

const ServiceCard = ({ title, description, icon }) => {

return (

<div className="card text-center">

<div className="text-4xl mb-4">{icon}</div>

<h3 className="text-xl font-semibold mb-2">{title}</h3>

<p className="text-sm text-gray-600">{description}</p>

</div>

);

};

export default ServiceCard;

Now this is just a basic prop’s I have added inside Service Card. I can adjust to add more later. This is just for reusable component. For now, I am going to show this service card in the webpage.

Now all the icons I have used inside services section are from react-icons. I have installed react-icons by doing:

npm install react-icons

Now, icons are installed, I am going to import the ServiceCard , icons from Font Awesome (fa), icons from Material Design(md) and Game Icons(gi) which are needed for the service card, inside App.jsx file.

I have also added, all the service cards for all the services needed for Choden Business. Below is the code I have added inside App.jsx and then we can see output like this:

import ServiceCard from "./components/ServiceCard";

import { FaLeaf, FaTools, FaTree, FaHammer } from "react-icons/fa"; // install if needed

import { MdOutlineYard } from 'react-icons/md';

import { GiMirrorMirror } from 'react-icons/gi';

<section id="services" className="py-section bg-brand-light text-brand-dark">

<div className="container grid md:grid-cols-2 lg:grid-cols-4 gap-6">

<ServiceCard

title="Gardening"

description="Lawn care, pruning, and plant maintenance."

icon={<MdOutlineYard />}

/>

<ServiceCard

title="Landscaping"

description="Garden design, turf, and retaining walls."

icon={<FaTree />}

/>

<ServiceCard

title="Concreting"

description="Paths, driveways, and slabs for all spaces."

icon={<FaHammer />}

/>

<ServiceCard

title="Handyman Services"

description="General repairs and household fixes."

icon={<FaTools />}

/>

<ServiceCard

title="Glazing"

description="Glass repair and replacement for windows and doors."

icon={<FaWindowRestore />}

/>

<ServiceCard

title="Security Door Installation"

description="Secure, custom-fit doors for safety and peace of mind."

icon={<FaShieldAlt />}

/>

<ServiceCard

title="Sliding Door & Window Replacement"

description="Upgrade old sliders and windows with ease."

icon={<FaDoorOpen />}

/>

<ServiceCard

title="Shower Screen Installation"

description="Stylish and durable shower screens fitted to your bathroom."

icon={<FaShower />}

/>

<ServiceCard

title="Bathroom Mirror & Light Installation"

description="Brighten bathrooms with elegant mirrors and lighting."

icon={<GiMirrorMirror />}

/>

<ServiceCard

title="Fly Screen Installation"

description="Keep bugs out while letting fresh air in."

icon={<FaShieldAlt />}

/>

</div>

</section>

Output:

A screenshot of a computer

Description automatically generated

Now I am again going to add about section to show business history, experience, etc,. Here is the code below that I am going to add inside App.jsx file.

<div className="container max-w-4xl mx-auto px-4">

<h2 className="section-title text-brand-primary">About Us</h2>

<p className="mb-6 text-lg leading-relaxed">

At <strong>Choden Landscaping & Handyman Services</strong>, we bring over <strong>10 years of hands-on experience</strong> in transforming outdoor spaces into practical, beautiful environments. From turf laying to garden beds, concrete works to general repairs — we’re the team locals trust for quality workmanship and reliability.

</p>

<p className="mb-6 text-lg leading-relaxed">

Based in Hobart, our crew understands the Tasmanian climate and soil, which helps us deliver durable, low-maintenance solutions that stand the test of time. Whether it's a new garden feature, a fresh concrete path, or seasonal maintenance, we treat every job — big or small — with the same care and professionalism.

</p>

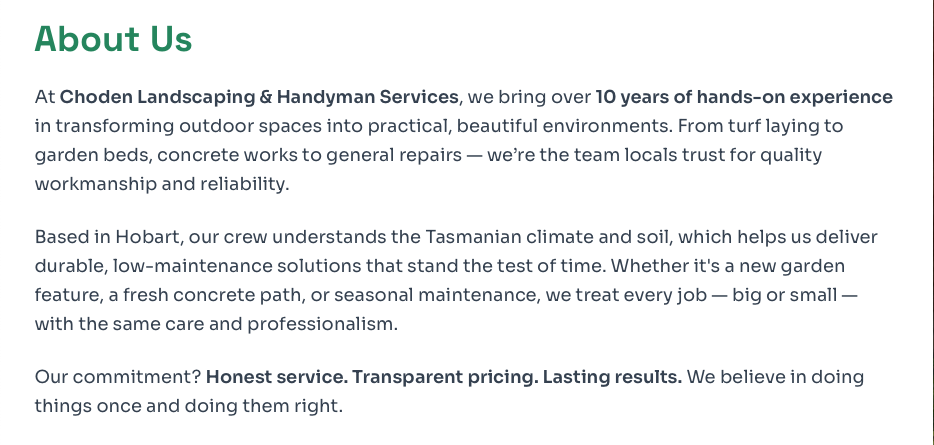
<p className="text-lg leading-relaxed">

Our commitment? <strong>Honest service. Transparent pricing. Lasting results.</strong> We believe in doing things once and doing them right.

</p>

</div>

Output:



Now another last thing I need to add is contact section. Just like above process, I am going to add this code for contact section inside App.jsx

<div className="container text-center">

<h2 className="section-title text-white">Get in Touch</h2>

<p className="mb-6">Call us at <strong>0450 123 456</strong> or email <strong>info@chodengardens.com</strong></p>

<a href="tel:0450123456" className="btn-primary inline-block">Call Now</a>

</div>

Now, I also need to add a footer. I am going to create Footer.jsx inside components folder and import it into App.jsx and Footer will be seen in the home page. This is code for Footer.jsx

const Footer = () => {

return (

<footer className="bg-brand-dark text-white py-6 border-t border-neutral-700">

<div className="container text-center text-sm">

<p>&copy; {new Date().getFullYear()} Choden Landscaping & Handyman Services. All rights reserved.</p>

<p className="mt-2">

Website created by <a href="" className="text-brand-accent hover:underline">Green Stack Services</a>

</p>

</div>

</footer>

);

};

export default Footer;

And then I have imported inside App.jsx like this:

import Footer from "./components/Footer";

Now we can see the footer contents inside the website. Now this is just a basic structure of what I can show inside the webpage.

Now what I did is, since I am not going to use complex ButtonSvg.jsx and ButtonGradient.jsx, I am planning to delete the svg folder completely and just make asimple Button.jsx to make it less complex and I have organized here and there and this is what current project structure looks like.

rajdaiwebsite/

├── node\_modules/ # Auto-generated packages (do not edit)

├── public/ # Static files served as-is

│ └── favicon.svg # Favicon icon

├── src/ # Source code (main app logic)

│ ├── assets/ # Media and static assets

│ │ ├── images/ # Project-related images

│ │ └── favicon.svg # Duplicate for ease of use in head

│ ├── components/ # Reusable UI components

│ │ ├── Button.jsx

│ │ ├── Footer.jsx

│ │ ├── Header.jsx

│ │ └── ServiceCard.jsx

│ ├── styles/ # Tailwind and global styles

│ │ └── index.css

│ ├── App.jsx # Main application layout

│ └── main.jsx # Entry point for React (Vite mount point)

├── .gitignore # Files and folders ignored by Git

├── eslint.config.js # ESLint configuration

├── index.html # Main HTML template for Vite

├── package.json # Project metadata and dependencies

├── package-lock.json # Exact dependency tree (auto-generated)

├── postcss.config.js # PostCSS configuration for Tailwind

├── rajdaiwebsite\_documentation.docx # Project notes & documentation

├── README.md # Project overview and instructions

├── tailwind.config.js # Tailwind CSS configuration

└── vite.config.js # Vite development server configuration

Now, just like that, with basic editing in the frontend, we have our website live preview here: <https://rajdaiwebsite.vercel.app/>