Table of Contents

[Pre-Requisites 2](#_Toc169300393)

[a. Familiarity with HTML and CSS 2](#_Toc169300394)

[b. Tailwind CSS Knowledge: 2](#_Toc169300395)

[c. Text Editor or IDE: 2](#_Toc169300396)

[Design Choices 2](#_Toc169300397)

[a. Semantic HTML 2](#_Toc169300398)

[b. Minimalistic Approach 2](#_Toc169300399)

[c. Animated Sections 2](#_Toc169300400)

[d. Responsive Design 2](#_Toc169300401)

[e. Color Scheme 3](#_Toc169300402)

[f. Accessibility 3](#_Toc169300403)

[g. File Structure and Creativity 3](#_Toc169300404)

[Challenged Faces 3](#_Toc169300405)

[Recreating Everything Using Tailwind 3](#_Toc169300406)

[Creating Buttons as Labels (Without JS) 3](#_Toc169300407)

[Hamburger Icon Nav Bar (No JS) 3](#_Toc169300408)

[Custom CSS Definitions in Tailwind 4](#_Toc169300409)

[Box Shadow Effect in Tailwind 4](#_Toc169300410)

[Animations and Transitions 4](#_Toc169300411)

[Mixing Tailwind and Manual CSS 4](#_Toc169300412)

[Targeting Multiple Elements with Tailwind 4](#_Toc169300413)

[Conclusion 4](#_Toc169300414)

[Appendix 5](#_Toc169300415)

# Pre-Requisites

Before embarking on the development of the portfolio website, the following pre-requisites in put in place –

## Familiarity with HTML and CSS

* A solid understanding of HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) is essential.

## Tailwind CSS Knowledge:

* Familiarization with utility-first approach.
* Explore the official Tailwind CSS documentation to understand available classes and features.

## Text Editor or IDE:

* Choosing Visual Studio Code as the text editor or integrated development environment (IDE)

## Design Choices

The following design choices were made for the portfolio website. The images in succeeding section can be used as reference.

### Semantic HTML

* Wisely opted for semantic HTML tags (e.g., <header>, <section>, <footer>).
* Semantic tags enhance accessibility and provide meaning to both humans and search engines.

### Minimalistic Approach

* Choice of a single-page design keeps things concise and user-friendly.
* By avoiding unnecessary complexity, I ensured a seamless browsing experience.

### Animated Sections

* Animations breathe life into the website and were wisely incorporated.
* Subtle animations for smooth transitions between sections.

### Responsive Design

* Embraced responsive design using media queries.
* Website adapts beautifully to various devices, including desktops, iPads, and Samsung Galaxy S8+.

### Color Scheme

* Consistent color scheme reflects your personality and branding.
* Ensures good color contrast for readability.

### Accessibility

* Accessibility by using semantic tags, alt text for images, and proper focus states.
* Testing website with screen readers to ensure it’s accessible to all users.

### File Structure and Creativity

* Files were organized logically. A well-structured directory makes maintenance easier.
* Grouping related files (HTML, CSS, images) into separate folders.
* The day/night toggle switch—adds a personal touch.

## Challenged Faces

Let me share my experiences with each of the challenges I encountered during the development of my portfolio website using HTML, CSS, and Tailwind CSS: –

### Recreating Everything Using Tailwind

Transitioning from traditional CSS to Tailwind was both exciting and challenging. I embraced the utility-first approach, which initially felt different from my usual workflow. Adapting my styling practices meant letting go of custom class names and relying on predefined utility classes. The benefit? Rapid development and consistent styling across components.

### Creating Buttons as Labels (Without JS)

I wanted buttons that behaved like labels, triggering specific displays without JavaScript. Tailwind’s hover and focus states allowed me to achieve this. By styling <label> elements as buttons, I could toggle visibility of related content. It felt empowering to create interactive elements without relying on event handlers.

### Hamburger Icon Nav Bar (No JS)

Ah, the classic hamburger icon! I tackled this challenge without a single line of JavaScript. I used an <input type="checkbox"> as my toggle switch. With Tailwind, I styled the checkbox and associated label to create the hamburger collapsible menu meant for smaller screen sizes. CSS sibling selectors did the magic—showing/hiding the menu based on checkbox state.

### Custom CSS Definitions in Tailwind

Tailwind is fantastic, but sometimes I needed custom styles. I extended the default configuration in tailwind.config.js. For flexible width values, I tweaked the breakpoints and spacing scales. It felt liberating to adapt Tailwind to my needs while maintaining its efficiency.

### Box Shadow Effect in Tailwind

Ah, the beloved box shadow! In traditional CSS, defining a box shadow was straightforward. But in Tailwind, it took me a moment to grasp the utility classes for shadows. I learned that Tailwind provides shadow classes like shadow-sm, shadow-md, and shadow-lg. Adjusting the shadow intensity was a matter of choosing the right class.

### Animations and Transitions

Animations and transitions add life to a website, right? Tailwind handles them differently. For animations, I used the animate utility class. It felt magical! Transitions were a bit tricky. I had to set up custom transition properties in my tailwind.config.js. Once I got the hang of it, my buttons and modals danced gracefully.

### Mixing Tailwind and Manual CSS

Mixing the two worlds—Tailwind and manual CSS—was like blending coffee flavors. I felt apprehensive about overriding rules. Tailwind’s utility classes are powerful but specific. When I needed to target an element manually, I used IDs (like #myElement) instead of classes. It was a dance of precision—knowing when to let Tailwind lead and when to take the lead.

### Targeting Multiple Elements with Tailwind

Manual CSS lets you target multiple elements of the same class in one go. But Tailwind? Each element got its own set of classes. Copy-paste became my mantra. I styled each component individually. The result? A tad messy, but hey, it worked!

# Conclusion

Building my portfolio website was both exhilarating and challenging. As I embarked on this journey armed with only HTML, CSS, and Tailwind, I encountered moments of frustration, countless trial-and-error sessions, and late nights tweaking every pixel. Stripping away JavaScript forced me to focus on the essentials. I learned that simplicity doesn’t mean sacrificing functionality—it means prioritizing what truly matters. Tailwind became my trusty companion. Its utility classes felt like spells in a developer’s grimoire. I tweaked margins, shadows, and widths until they danced harmoniously. Each breakpoint in media query was a step toward a seamless user experience. My buttons, modals, and profile image swayed gracefully with Tailwind’s animate class. Lastly, the clock ticked but I persisted. Eight days—sometimes feeling like an eternity, other times like a fleeting moment. Was it worth it? Absolutely. Because beyond the pixels and stylesheets, I discovered resilience, patience, and the joy of creation. My portfolio isn’t just a showcase; it’s a reflection of my journey—a pixelated memoir.

# Appendix





