For your final portfolio, provide in some accessible form (e.g., GitHub repo or shared Google drive) and for each of the above, provide:

a reasonably documented example / subset of work you have done that semester involving that topic/tool/framework

(within a single PDF or similar) a brief discussion of what role that topic/tool/framework plays in developing web site

HTML

HTML is the language that defines the structure and content of web pages. It is the backbone of the site and is what the browsers interpret to show content to the user. It uses tags for various elements to differentiate between things like headings, paragraphs, images, links. You can specify bolding, italics, or what level of heading. It interacts with css for styling. By differentiating components, it makes it easier to parse a website for information, allowing accessibility.

CSS

CSS is a stylesheet defining how HTML content is represented. HTML is the ingredients CSS is the recipe for the dish. It allows developers to specify fonts, colors, spacing, positioning. It allows HTML to be content and css to be design. CSS is essential for making pages appealing and friendly for the user.

JavaScript

While HTML and CSS provide content and static design, Javascript adds interactivity and dynamic behavior like animations. It allows a page to respond to user interactions (eg. clicks, keys, scrolling), update content, or animate. If information isn't hardcoded in the html, javascript can pull and update from other places. It can listen to users and respond based on actions. For example, if I, the user, scroll halfway down the page, the header might become sticky and animate.

ssh & scp (w/ ssh keys)

SSH allows us to secure login to remote servers. It is authenticated by keys. SCP uses SSH to securely transfer files between a local computer and a server. In class, I used it to upload my websites to catapult. I would need to securely copy my local datafiles and secure login with SSH to the catapult server then copy the files up. SSH keys work in pairs of a public and private key.

React (w/ components)

Often in websites, things are repeatedly used. In HTML, if you want to change them you have to change them all. React offers a solution with component based architecture. It allows development of reusable and nestable components. This means you can develop each component separately and then bundle them together with something like vite to make your HTML website. React encourages modularity, every part of the website has its own file which makes it a lot easier to maintain. It allows production of a dynamic website but in a much more efficient manner.

Bootstrap

Bootstrap takes the idea of react components but just predesigns them for you so you can use them without having to worry about creating or styling them yourself. It offers some CSS framework but also javascript components like navigation bars and buttons. By using these components, they will already be mobile friendly and accessible. It greatly speeds up development too.

Tailwind CSS

Tailwind is like a huge CSS stylesheet that you don't have to write and you can use it directly in your HTML. It gives you full control over styling without actually having to write a CSS file. It does mean more classes in your code but it is consistent and speedy. It does allow a lot of customization and you can even add your own custom utility classes. It is also really handy for mobile configuration as it has presets of things like sizes for different size screens.

ShadCN/UI

Shadcn is a component based toolkit much like react where you can install the components into your codebase. They are already accessible and they are fully customizable. It also uses tailwind so it integrates really well with tailwind styling. It makes it super easy to quickly add components where you only need to add what you use. Some of the components available are dropdowns, tables, calendars, etc.

Design for user experience (Krug)

Design with the user in mind means making a website intuitive, efficient, and pleasant to use. Krugs book, Don't make me think, gives foundational principles emphasizing that users shouldn't have to think when they use your site. Everything should be obvious and self-explanatory. As you develop a website these ideas should be in your mind; making pages self-evident, designing for scanning instead of reading, clear visual hierarchies (h1, h2, h3, etc) maintaining consistency, and not taxing the user.

Accessibility

Websites get accessed by everyone which means they need to be usable by everyone, including people with various disabilities. There are set standards (WCAG) that need to be met. These include add text alternatives for things that aren't text (like images or charts), all elements are navigable by keyboard as alternative to cursor, good color/texture contrast, website is usable by screen readers. This is ethical to be inclusive but also often a legality. It also means a website will get more users by being accessible.

Figma

Figma is a design tool that has really good transfer from an interactive design into a javascript developer write up to implement into the actual code. You can collaborate with your team to visualize the end product before you actually code it. This makes it a lot easier to refine the user experience, try different things, communicate ideas without the hassle and obstacles of trying to code it. You can prototype a lot of components that you can later get through things like react.

Cursor

Cursor is an AI powered editor that fully integrates with your workspace. When you ask it questions it looks at the whole context of your workspace as opposed to just one file and can make changes for you. It has multiple modes, the most important being ask and agent. Ask allows you to ask questions in the context of your workspace and whatever else you provide it. You can brainstorm ideas or give the model a good idea of what you want before it actually does it. Agent has the power to actually change code and even run terminal commands. Cursor really boosts productivity and takes away a lot of the busy work. It is also really good for solving problems. It does still have errors and can even get messed up on things like syntax. Overall, if you learn to use it right and be explicit about what you want it is a very efficient tool.

Google Firebase for backend

Firebase handles the backend of websites for developers. This includes storing data, and data inputted by users in a secure manner. Instead of having to write a whole server from scratch firebase does everything and is very robust. It is great for prototypes or small apps so that the focus can be on the features of the app not on whether my server is secure and whether I need to build one.

What was your experience in using React+Vite for building web apps compared to "rolling your own" using HTML, CSS, & JavaScript?

It was a lot easier with React by having the components and being able to reuse things instead of repeatedly typing. It just made the code overall a lot cleaner and easier to make changes. I could just change a component once instead of multiple times.

What are the primary takeaways you had from reading Krug's book and your corresponding analysis of sites?

Don’t make people think. That book gave me such a good perspective on design and I am so glad we read it. Now when I look at the website I am constantly thinking in the perspective of the user. Making navigation easy to get back and forth between pages. Making things obvious like you are walking down a path not getting lost in the woods.

What is the importance of accessibility (give a few explicit examples), and what steps can (and should) you take in assessing the accessibility of your site?

You need to make sure colors are differentiated for people with color blindness. This can be with the right color scheme or if that’s not possible adding textures to differentiate items. Websites also need to be tabbable so that people who can’t use a mouse can navigate through the page. There are many more aria features to add like captions for pictures for blind people. There are many analyzers for assessing accessibility to a site and can tell you what needs to be added. They can be added as extensions in vscode.

In what ways did the different design sprints, and use of Figma, help you in thinking about what an end product should look like and how it should function?

Design sprint was so essential to the process. My initial idea was not the end product. If there was no design sprint and collaboration of ideas I would have rabbit holed down one preliminary path and not have the best product. There were a lot of questions and context brought out of the collaborative design sprint. We thought about the user end, the developer side, how do we provide all the information of the dcs major and not make them think? How do we let students plan out their major with the least effort and time possible? It meant we could really look from the perspective of krug too with figma because it was purely focused on the design and not having to write the code just yet. Getting the gestalt view first made it way easier to make components and piece them together.

What takeaways do you have from working with AI/LLMs through Cursor (or similar) in building web applications?

You really have to tell it what you want. Sometimes it can get fixated on errors if you give it a niche one and it can create more errors by trying to fix one. I really love working with Cursor now that I have had a lot of time with it. I've gotten good at feeding it a list of tailwind v4 directives to use as a guide when doing all the installs particularly for postcss, and any of the other syntax it should use in several files. This made it a lot easier and anytime there was a mistake I would just say refer to this file. I also started using copilot pro on vs code Insiders instead since it is only 10 bucks a month and the insiders version has agent beta on it. I feel like I get everything vs code has to offer as well as cursor.

What was your favorite thing (or deemed most useful) that we covered this semester, and why?

There were so many things. I loved creating the aliases. I now have an alias to upload any built site to catapult so I can regularly update it quite easily. I want to get a lot more comfortable with them and create more useful ones. I am trying to be in the mindset of using a terminal to navigate my computer.
I really loved how we engaged with AI. The cursor is super helpful and I feel comfortable creating websites with it now.
The krug book was super essential, and changed my mindset on design. I use it in other classes too and in my powerpoint presentations.

What do you wish we had covered, or had covered in more detail, and why?

I wish we covered backend stuff more. I feel like I can create websites well and it's really easy uploading to catapult now. Although, I don't really know what to do if I had a client that wants to put up a website. I think I will watch some tutorials on firebase to get a good sense of it, I had many struggles with it. I don't think adding more backend to this class would do much though, I think it should be a second class to extend web development.