CSCB20 – Assignment 2 Report
Justin Zhang, Nisith DeSilva, Bill Jin

What issues and problem did you notice with some of the previous websites?

Examining the previous course websites, we noticed that the styling was quite lacking. That is to say, the websites lacked modern elements of design as well as animations. Most of the pages had bare bones section formatting dominated by font families such as Arial or Times New Roman. Text sections were either divided by lines or by nothing at all. In particular, one of the given CSC148 sites had all its information condensed into one scrollable page, rather than categorizing information into separate webpages. Lastly, the previous CSCB20 website had a lack of graphics/images which made searching for information slightly difficult.

Moreover, we found that information was not organized very well – the previous CSCB20 website lacked easily accessible information about TAs. Because we were unable to access the calendar provided, we are unsure whether or not the site included individual information about the office hours of each TA. Another glaring issue we identified was that the sites did not include a list of technologies used and required in the course along with proper corresponding instructions – for example, an introduction to programming course might required a specific Python IDE with a guide on how to install it; or, it might provide an installation guide for the GCC compiler. While this kind of information may be included in lecture notes, it absolutely helps to make it more accessible on the course website (which is the purpose of the site).

How did you address these issues in your redesigned course website?

We aimed to address these issues by identifying each specific problem and creating a corresponding user story, and we also addressed styling issues by creating accurate mockups to follow when creating the website.

We made sure to provide clear, distinguishable information on each page by organizing text into sections or tables. For example, the *Labs* page contains a table with each week listed next to its tutorial work and solutions, but the *Contact* page contains distinct boxes with information about the course instructor and each TA, as well as containers for images of each person. We organized the *Assignments* page differently from the *Labs* page upon mutual agreement that assignments should be organized in a way that encapsulates all information about an assignment into one section rather than in a table. Each assignment block contains a description of the assignment, the submission deadline, and the weighting of the assignment.

We included a section on the home page with the heading *Technologies* which is meant to list out the aforementioned required technologies for the course (i.e. Python IDE, compilers). We also included a section on the *Resources* page which lists optional/helpful resources such as online tutorials and software for testing like DB Browser for SQLite.

What are some challenges that you and your team member faced? How did you go about addressing these challenges?

One of the main challenges faced was unifying all the pages, as we each worked separately on certain pages, before compiling them all together. We referenced certain

stylesheets in multiple pages, which allowed us to apply styling to common elements, like information boxes and the navigation bar. Reusability of code is a good thing in general, as it allows for code to be accessed in different contexts with very little need to modify it, which applies to this situation as well.

However, there were still a number of unique elements that would needed their own specific styling changes, some of which were quite similar, yet had specific enough differences that they needed their own styling. For example, most of the pages had only one header on each information subsection. Thus it made sense for the box element for the header to be specifically styled so that the header would stay in one location, the top left of the information box. The labs page that we mocked up required 3 headings all one one line to simulate a table, so instead it made more sense to make a new selector to style the headers. Making small adaptations like this is one way in which we navigated this challenge.

Another challenge was combining different layouts, such as grid and flexbox and using each or even both at times when the situation called for it in the different pages. For example, for cases where there needed to be overlap between different elements, using flexbox made sense, as it would allow for the elements to remain on the same axis while letting there be overlap by setting the position to relative. In other cases it would make sense to use grid for more precise control of the position of each element in relation to other elements. And when we were unsure which would be more useful, we could combine both, for example nesting a flexbox containing some information elements inside a grid, or vice-versa.

A final challenge we experienced and overcame was adapting our website so that it would work on different platforms, mainly mobile and desktop. We achieved this by using media queries in most pages to shrink certain elements or change how an element could interact with other elements in the page. For example, we made it so that the navigation bar present in all pages would become a vertical drop-down menu when viewed on a screen with size 800 pixels or less. Another example of this is applying a media query to the calendar so that it switches to agenda mode, also a drop-down, from a full monthly calendar when viewed on a smaller screen. In general we also set width or height for many elements as a percentage of the size of the viewport, so that when the screen size was changed, the elements would be responsive.