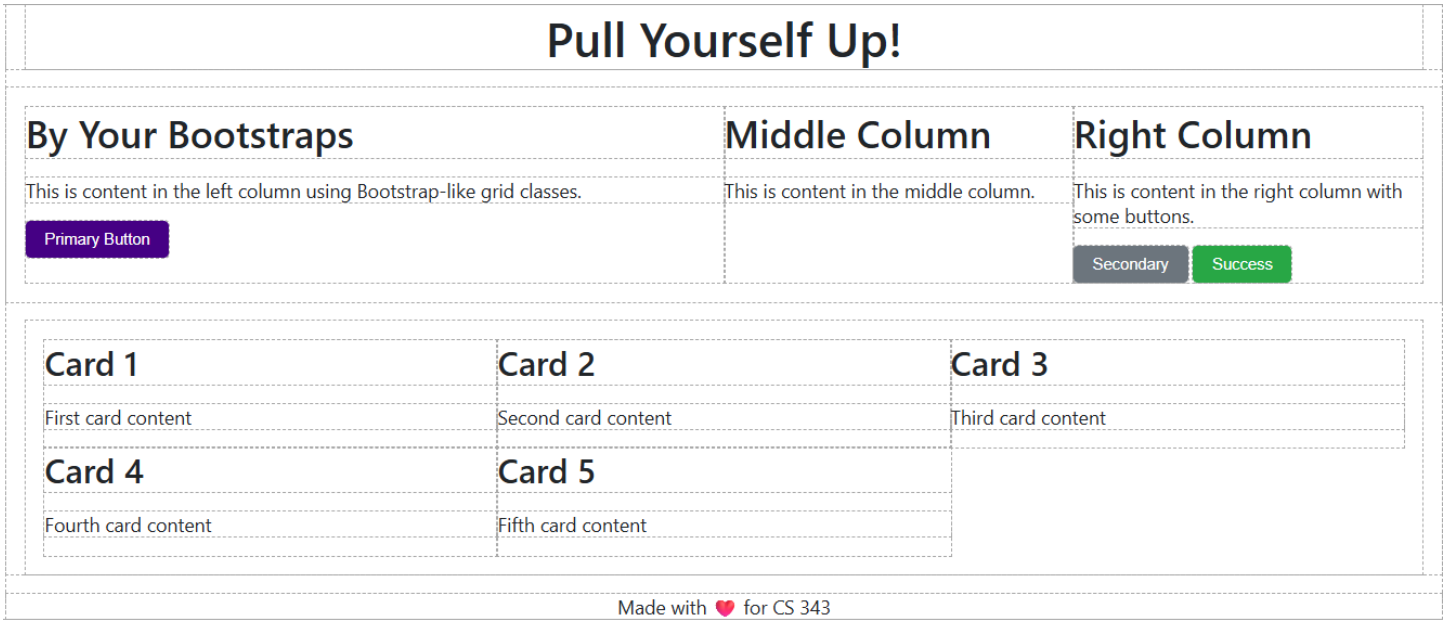
Activity: Building Bootstrap

Part 1: Bootstrap Classes

Take a look at the image below, which shows the desired website style and layout:



Answer the following questions using the image above and the provided HTML code:

1. In the HTML, identify all the elements with the "row" class. Then, in the image above, **draw a box** around each row and its contents.

2. Based on the image, what percentage of the row width does the "By Your Bootstraps" section take up?
   Circle one: 10% 20% 33% 25% 50% 75% 100%

3. Based on the image, what percentage of the row width do the middle and right columns take up?
   Circle one: 10% 20% 33% 25% 50% 75% 100%

4. Look at the elements for the left and right columns. What class do they have in common?

5. What class does the "By Your Bootstraps" section have? What part is different? What does that mean?

6. What class will make an item take up the full width of a row?

7. What percentage of the row width would the class `col-md-4` take up?
   Circle one: 10% 20% 33% 25% 50% 75% 100%

8. Are all columns inside a row displayed side-by-side? Justify your answer based on the image.

9. **Circle the type of display** do you think rows have: inline | block | flex | grid

Part 2: Building Bootstrap

Open `shoestring.css` (our knockoff version of bootstrap). First, look through the included styles to see how CSS variables are created and used. These variables define the page's color theme.

Your goal is to add declarations to the empty rules to style the page as the image shown on page 1.

Tips:

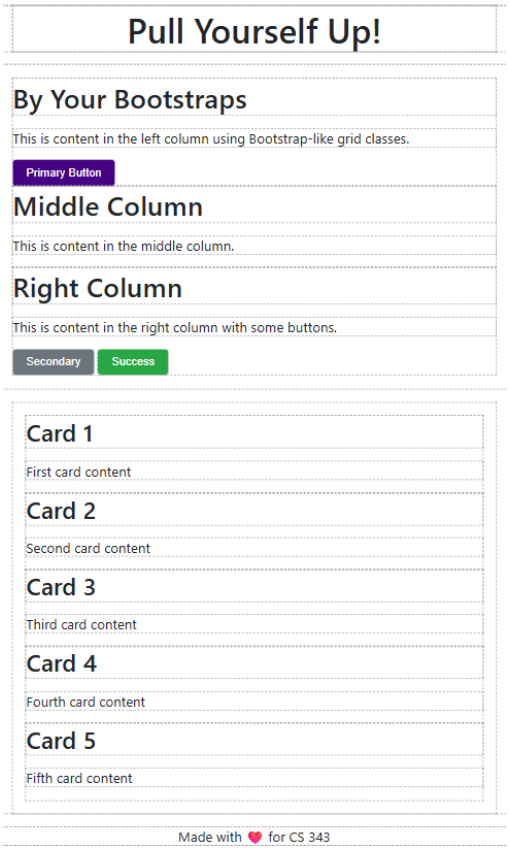
- **.container** should be centered on the page and have a dynamic width, but no more than 1200px.
  - To center a div, set its left and right `margin` to `auto`
  - The `max-width` property will allow an element to have a dynamic width up to that value
- The **m-0** and **p-0** classes should specify a margin or padding of `0`
- The **m-3** and **p-3** classes should specify a margin or padding of `1rem`
- **.row** should be a flexbox, position its contents horizontally, and allow contents to wrap if needed
- The **col-md-** classes should each set the element's width to a percentage of its parent
  - E.g., `col-md-1` should be 1/12th of its parent, or `8.33333%`
- Each of the **btn-** classes should set the button's "fill color" to the corresponding page theme color
  - E.g., `btn-primary` should make its overall color be the same as the page's primary color
    → Reference the CSS variable already created! Use `var(…)`

Part 3: Responsive Design

Finally, the "md" in `col-md-6` means a "medium breakpoint" – this means that when the width of the viewport is above 768px, the page will render normally – each column will only take up a fraction of the width.

However, once the viewport is smaller than 768px, instead, each column will shift to a "mobile friendly layout": each column takes up the entire width of the page, and are displayed vertically instead of side-by-side.

See the image on the right showing the desired mobile view:



At the bottom of `shoestring.css`, **write a media query** that applies if the width is below 768px, and inside, write rule(s) to make each column take up the full width and be displayed vertically like in the image above.

➔ What do you think is the advantage of writing multiple classes like this as opposed to styling elements using their ids?