Carson Perry

CST-247 Mr.Hughes  4/6/21

Activity 3

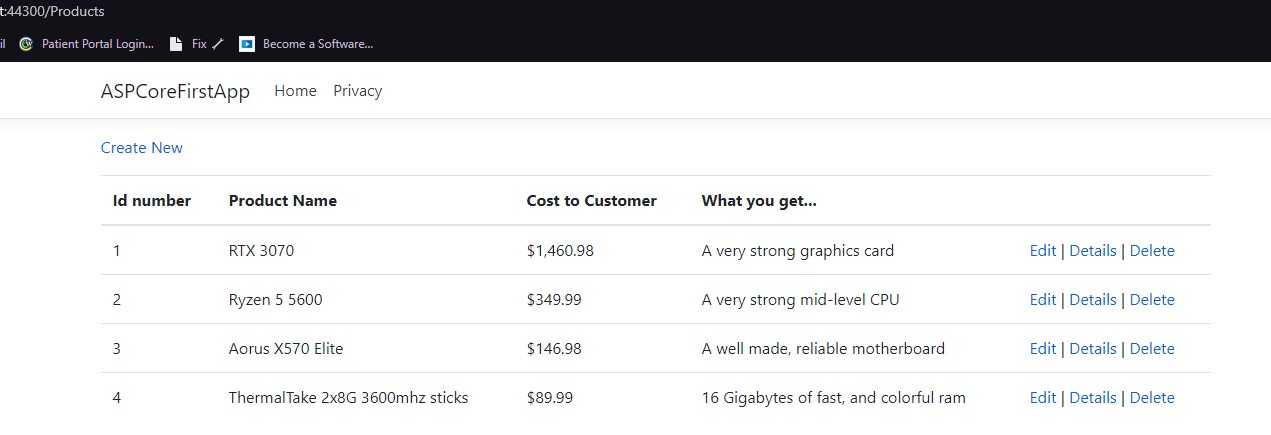
## Step 1.8

Graphical user interface, website

Description automatically generated

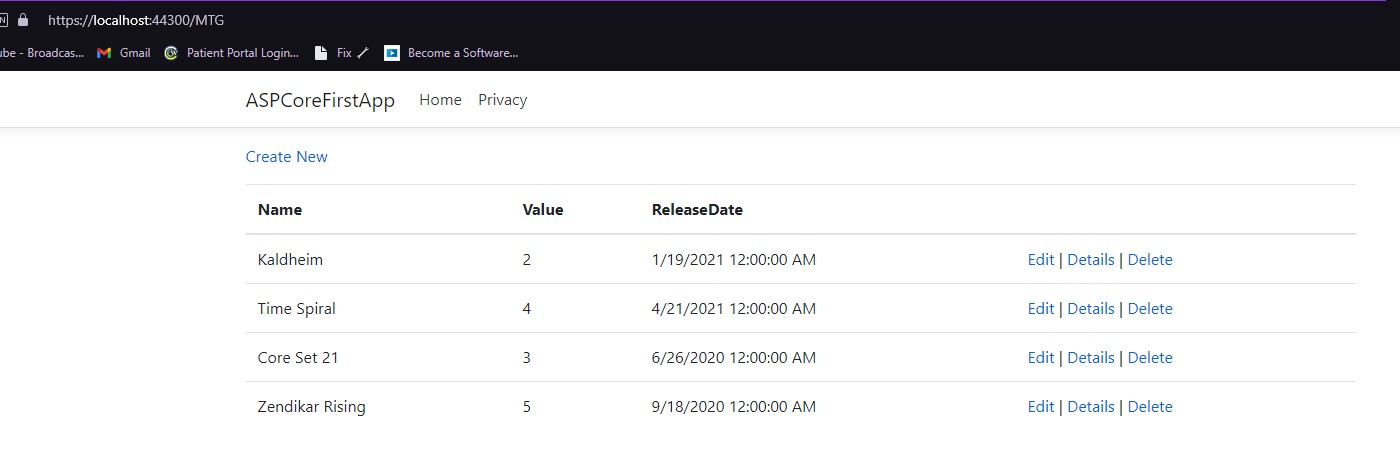
This is showcasing a list of products rendered into a page.

## Step 1.9



This is displaying how the DataType validation tag adds the dollar signs when set to currency.

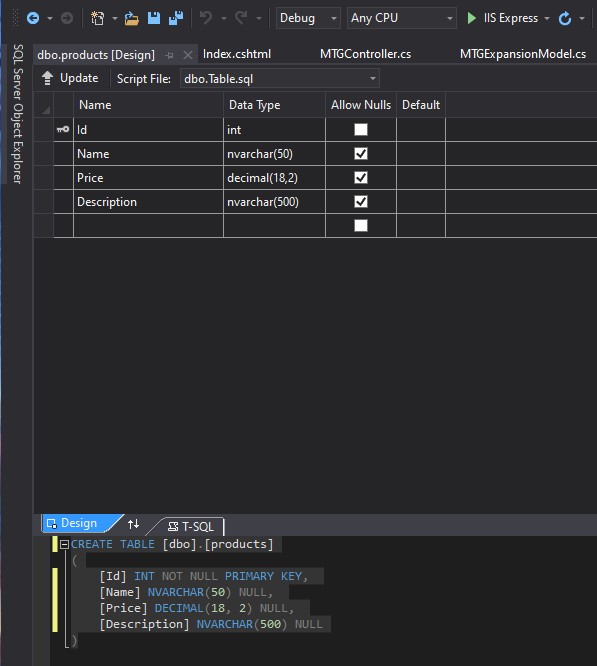
## Coding Challenge step 5



This demonstrates creating my own model (based on the card game Magic: The Gathering), and displaying that in a list, with the date DataType.

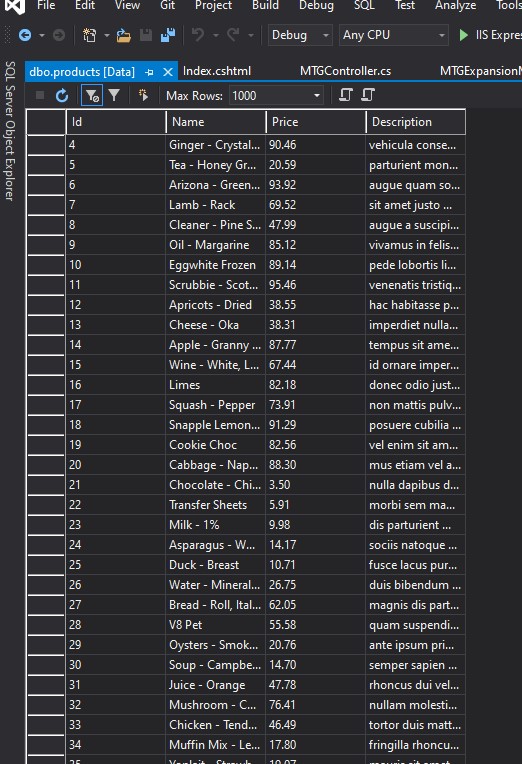
# Database Table for Products

## Step 2



This shows the SQL command of creating the products Table in SQL and the structure of it.

## Step 3



This shows the data placed into the database after running the script generated by mockaroo.

## Step 14

Table

Description automatically generated

This demonstrates using the database to load in data, instead of hard coded information.

## Step 18

Graphical user interface

Description automatically generated

This shows using a GET request for searching wine as the key string in the url bar.

## Step 25

Graphical user interface, text, application

Description automatically generated

This displays the search form created for the purpose of entering a key to search for among the products within the database.

## Step 27

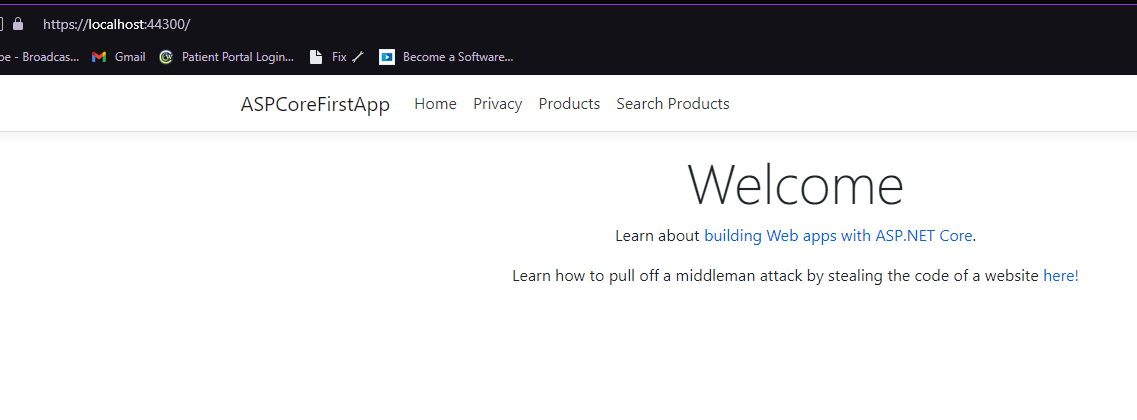
A picture containing text

Description automatically generated

This shows the result of the last form when searching for ‘car’.

# Add Links to the Navbar

## Step 3



This demonstrates adding more links to the navbar.

## Summary of concepts:

The main concepts that were taught in this lesson were implement for loops into a page to iterate through a list of models to display on a page, as well as loading those models from a database. A side concept we learned was using Bogus and Mockaroo for fake data for the sake of testing mass amounts of information. We also reviewed searching through a set of data and returning only what we want with a key, as well as adding more buttons to a navbar that lead to other pages.

# Visual Confirmation of Page Sections

## Step 4

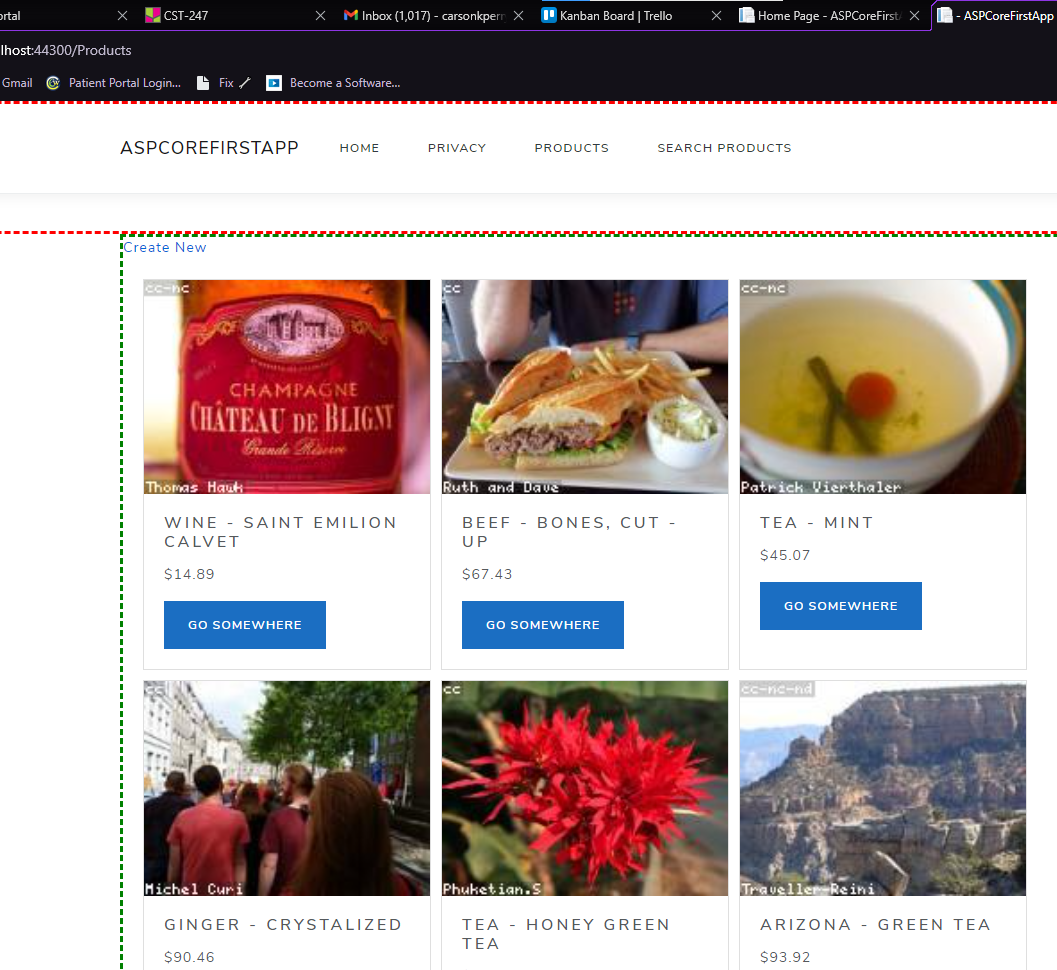
Graphical user interface

Description automatically generated

This demonstrates updating the css and adding a border to the different rendered sections.

# Flexbox Container

## Step 4



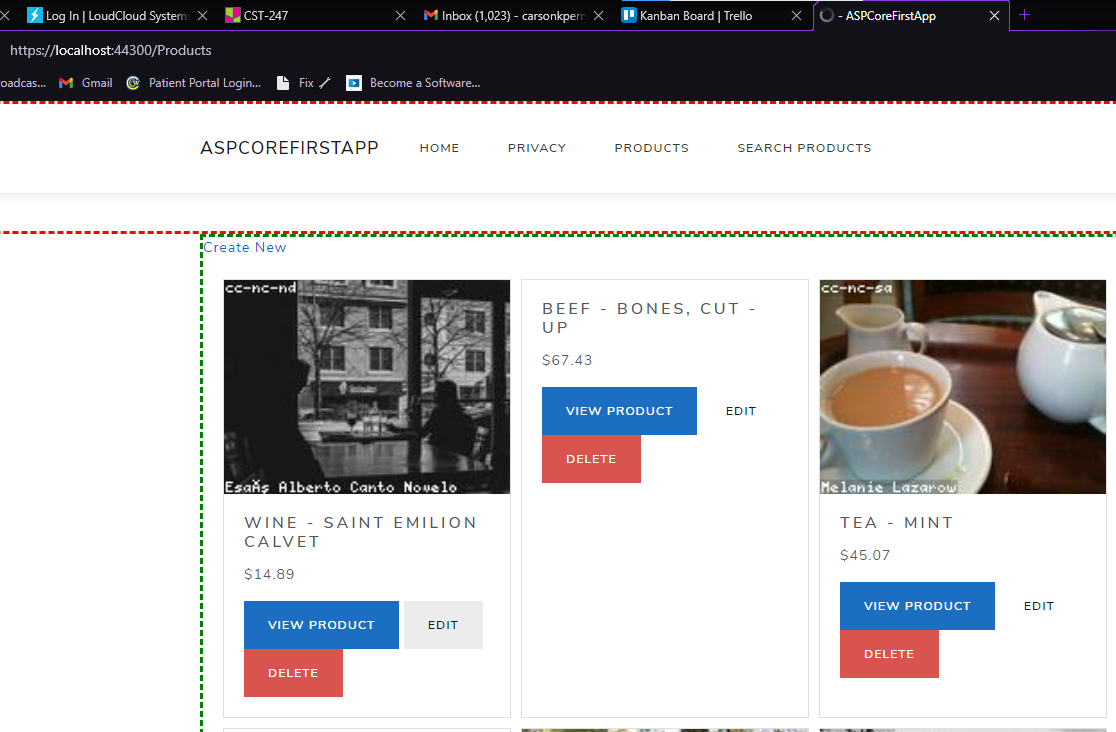
This demonstrates using another site to get autoloaded fake images using GET requests with loading model information into the url.

## Summary of Concepts:

In this section, I have demonstrated using layouts, rendering partial pages, changing the default bootstrap theme/changing css, and using flexboxes instead of a table. Flexboxes have a bit more freedom and look much nicer than a table in my opinion. They allow for easily formatted images, a nice looking button as well, and format neatly into the page.

# Part 3 CRUD SQL Operations:

## Coding Challenge:



I assume there would be at least one screenshot wanted for this section, but there are no steps for it, so here’s the finished product after the coding challenge. This is demonstrating adding a button to each product, that allows you to view the single product, edit the product, or delete the product, each having to communicate with the database.

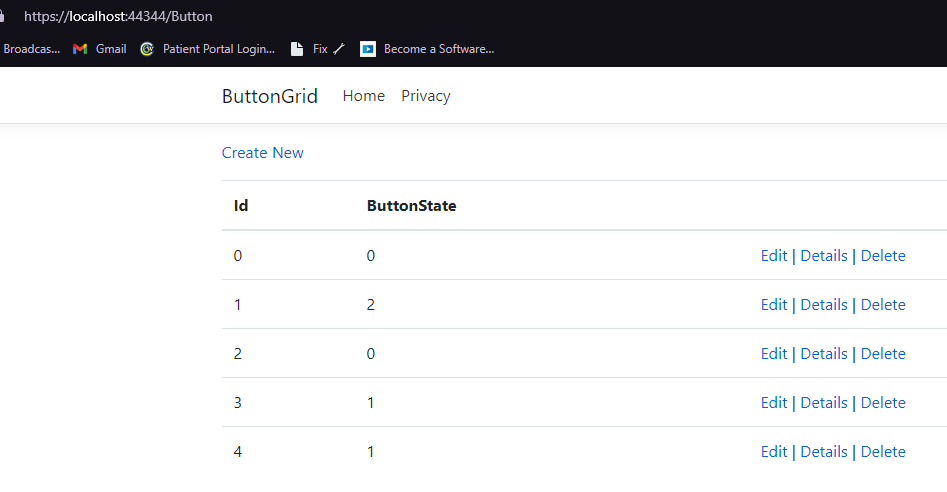
## Summary:

In this section, I demonstrated the ability to retrieve a product model from a database, updating (modifying) an existing row in the database using a product model, as well as deleting a row using an integer ID. All of this, as well as including a button on each product box for each of those actions, that are tied to the product model that box represents.

# Part 4:

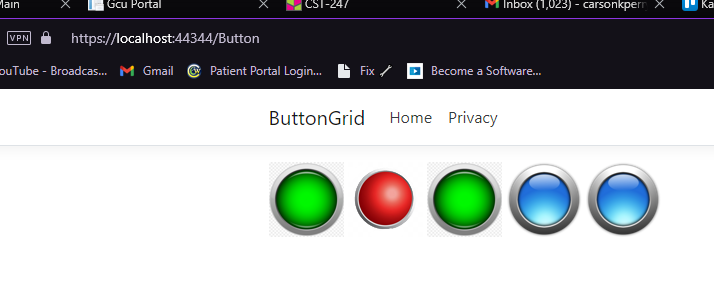
# How to Create a button list:

## Step 6:



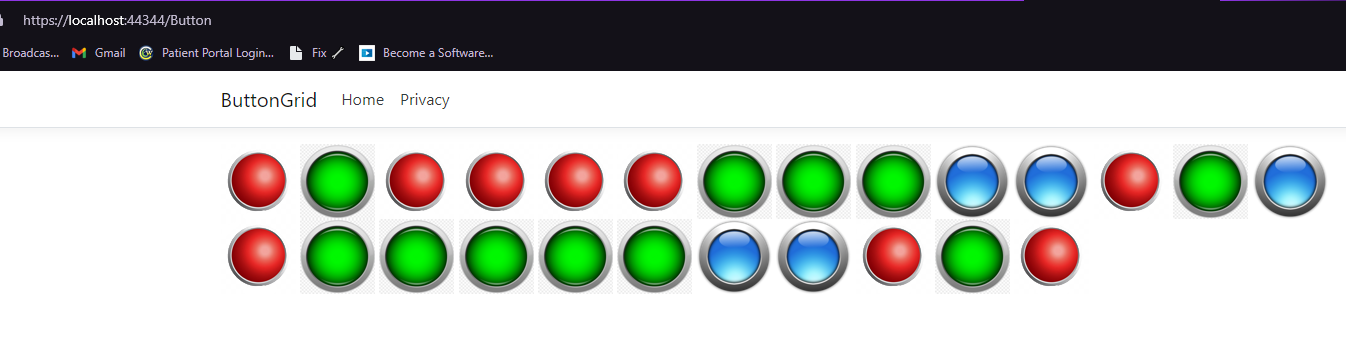
This is demonstrating creating a static list of objects in a controller, then displaying that list in a page.

## Step 8:



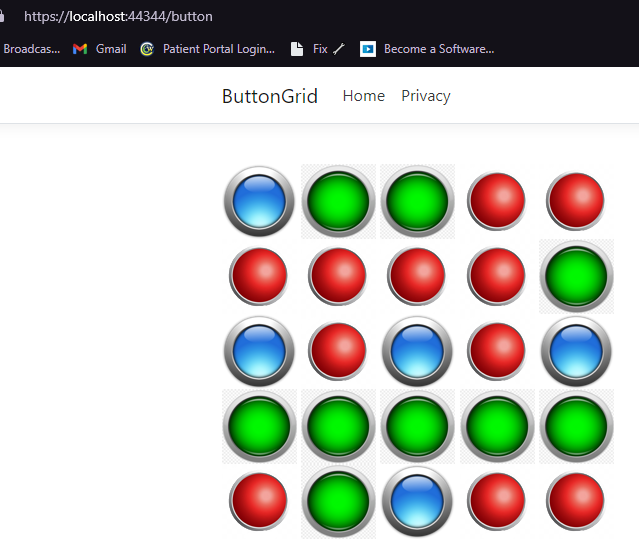
This demonstrates displaying an image from a local source, based on the information of the current model.

## Step 11:



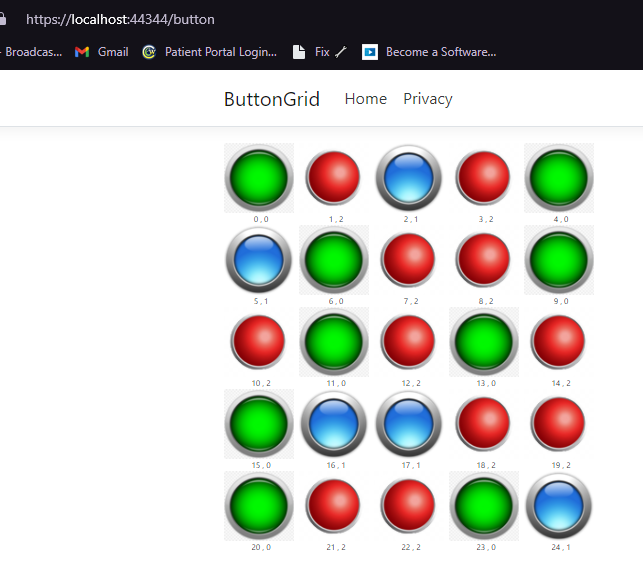
This demonstrates iteratively creating a button with the next ID, while randomly generating the button state.

## Step 14:



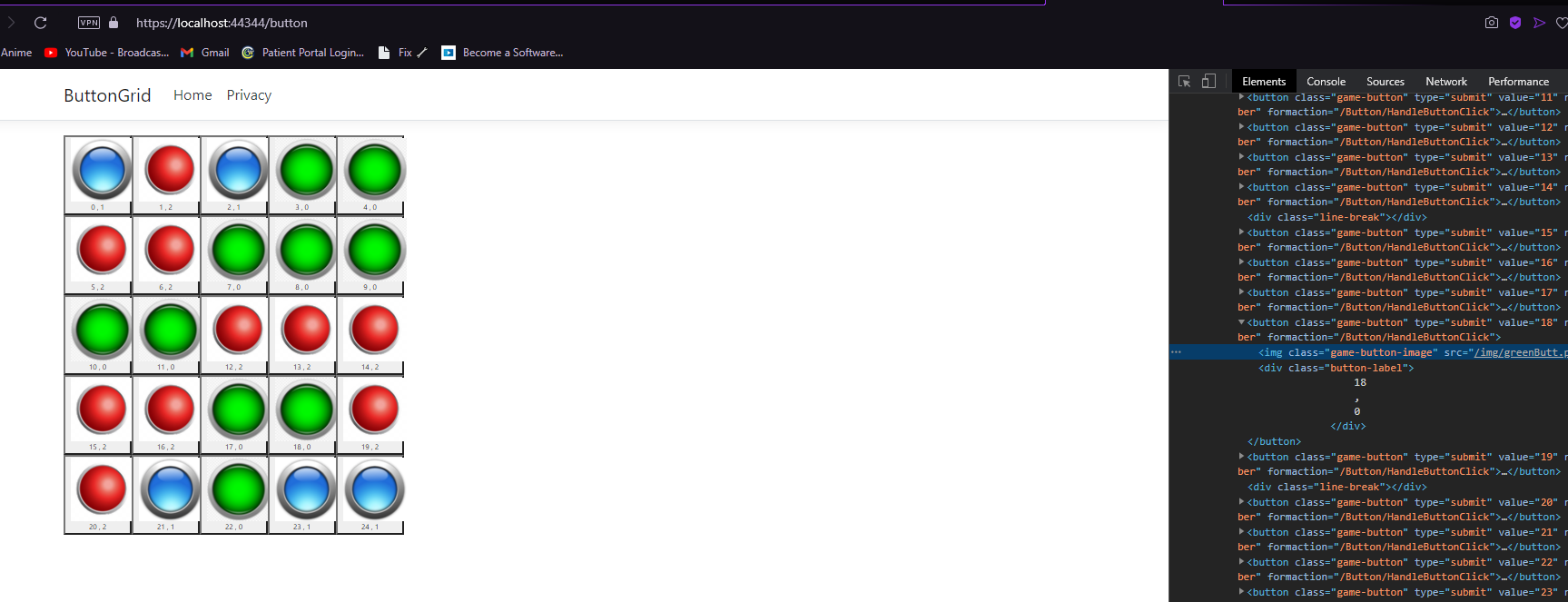
This demonstrates using a for loop to go through a list of objects, then rendering those objects in the form of images, but adding a <br /> tage every 5 buttons.

## Step 16:



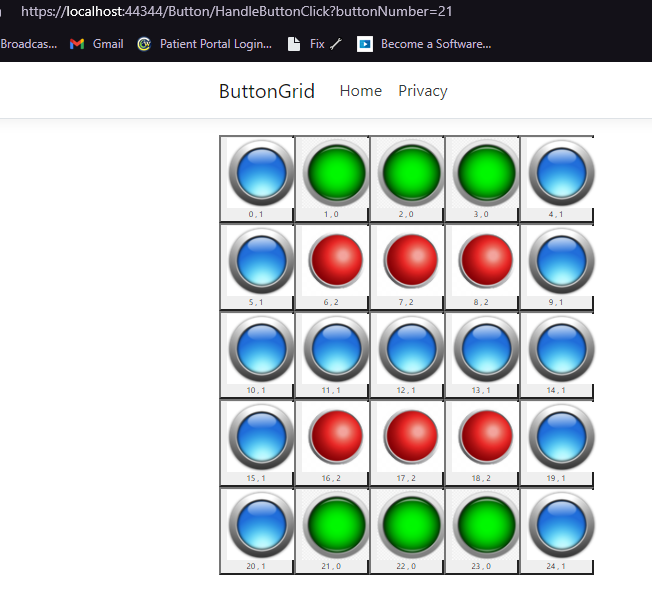
This is demonstrating the same button grid, but implementing flex wrapping and other flexbox css, to format the images better as well as have a label for each.

## Step 19:



This demonstrates implementing an actual button for each of our button images that then go to our button controller when clicked on

## Step 21:



This demonstrates the buttons actually working how they’re supposed to and changing the image when clicked on.

## Summary:

In step 4, I created a grid of buttons, that have information generated from a list of objects. Each of these buttons has their own id, that is then sent on button click, to the buttoncontroller that then changes that button’s buttonstate property, then sends the modified object list back to the index page, changing that one button. This demonstrates manipulating objects graphically on a web app.

## Git hub link just in case I can’t fit the zip files:

https://github.com/arsonull/arsonull.github.io/tree/master/Sem4/C%23/act3