**EAD – CA3 Report**

Anthony Kuznetsov – x00162229

Links: [Project](https://yellow-beach-0ba2fec03.2.azurestaticapps.net/) | [API](https://chandan-02.github.io/anime-facts-rest-api/) | [GitHub Repo](https://github.com/anthonykuz/ead-ca3) | [Azure API Response](https://anime-facts.azure-api.net/v1/get-facts)

The Blazor project I built is called **“The Anime Index”**. This is a single-page website that you can query an anime title and it returns a list of facts about that show along with a poster image and a fixed title.

**Functionality**

The HTML aspect of the page was simple enough to create. I have all the content within a global container for setting min and max width’s so that the containers don’t scale up or down past a certain threshold.

Then after that I had split each component (header, search, image, fact cards) into their own containers with unique classes for styling and positioning.

Text

Description automatically generated

Due to some unforseen circumstances, during the later stages of development the API I was utilizing for this project was shut down as Heroku was disbanding it’s free tier. What I chose to do was to create an API on Azure via APIM that sent a mock JSON response via a link that the API used to send. You can find the [JSON response here](https://anime-facts.azure-api.net/v1/get-facts).

Graphical user interface, application

Description automatically generated

I had to rework my classes as before I was querying one result at a time and deleting it afterwards for the next query – now I calling a function to send the GET request as the razor page loads and it deserializes all the results from the response neatly into a list of objects.

Text

Description automatically generated

Text

Description automatically generated

The blueprint for the AnimeTitle class was created with paste special from JSON to C# and it created all these fields for me automatically.

A screenshot of a computer

Description automatically generated with medium confidence

The list of facts is an object of “Fact” and was also generated using paste special:

Text

Description automatically generated

And the way the query works is that when you search and either press enter or click the search button it will run a function call that validates input and if it passes, it fixes the query string to include underscores and lowercase letters and checks it against the list of object titles. If it gets a match, it creates a copy via an implicit copy constructor and that object is the one that is displayed on the razor page.

Text

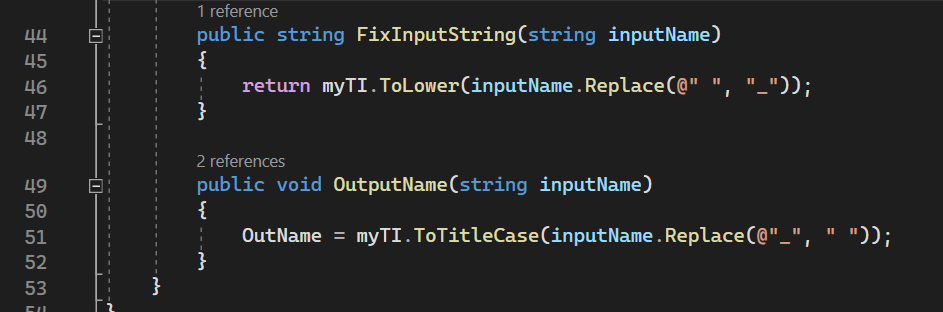
Description automatically generated

Class methods include “InsertHTML” that loops through all the facts and inserts them into a markup string which is then interpreted into markup via **MarkupString** in the body. There is an underline in the foreach loop as it suggests using a StringBuilder that does not seem to work when converted via MarkupString.

Text

Description automatically generated

And finally, the last couple methods use **TextInfo** to shift the letters of the query string to title case/ lowercase and remove/add the underscore.



The project is hosted via Azure Static Web App service and has a pipeline into the GitHub repository that triggers when a push is made. You can view the page [here](https://yellow-beach-0ba2fec03.2.azurestaticapps.net/).

**Design**

I **originally** decided to use the **“anime-facts-api”** found on the [GitHub open-source API list](https://github.com/public-apis/public-apis) because it fit all the requirements for the CA; that being it used CORS and did not require an API key. *That has since been changed to a custom Azure API GET request made during initialization.*

For design - I knew that because the functionality of the app was simple, the interface should be as well. And I began where *most* students would with design… being a basic mock-up in MS Paint:

A picture containing diagram

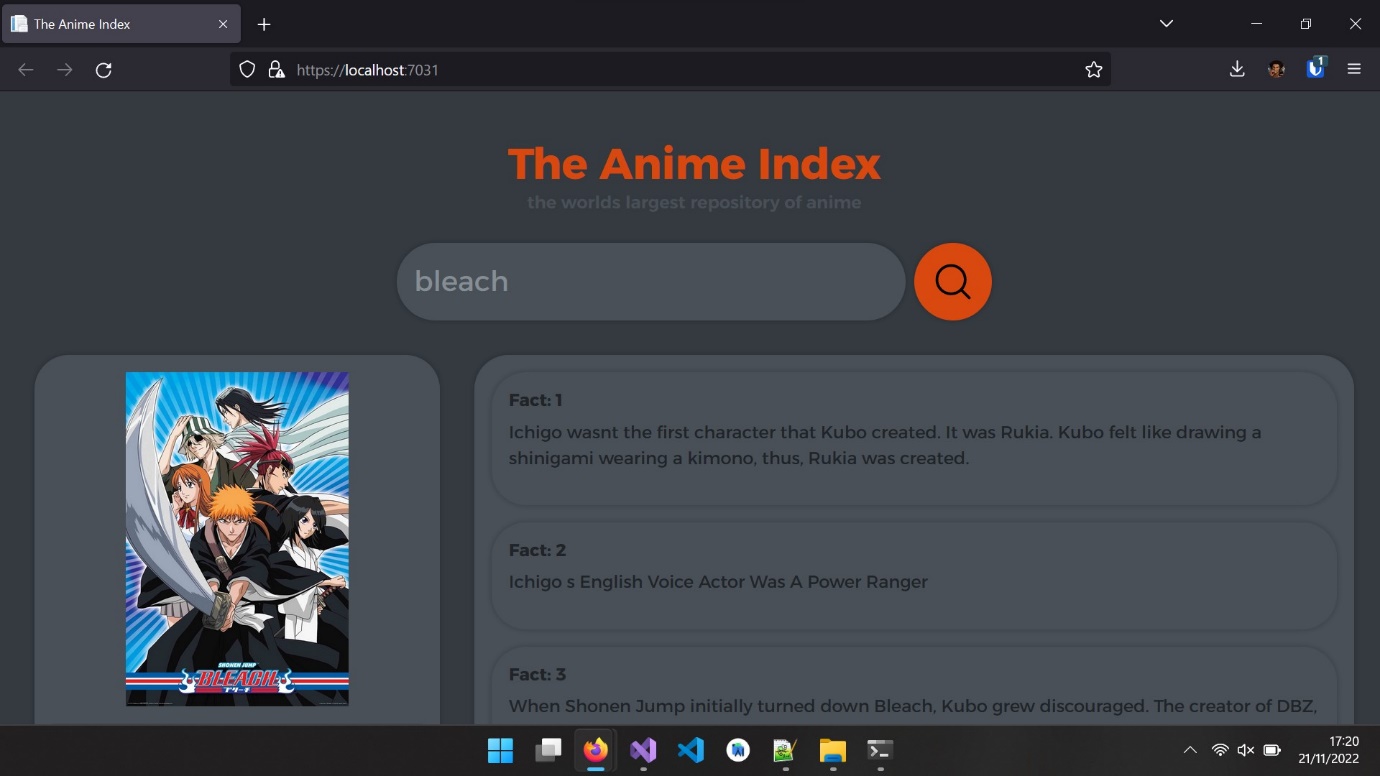
Description automatically generated

All it was comprised of was the heading and subheading at the top, with the search bar below that and then a grid of two columns:

1. (Left)  
   The anime title and cover artwork.
2. (Right)  
   The fact cards that would be gathered from the API call.

There is also a column to the left of that, which is the default Blazor bootstrap navigation menu that was removed shortly afterwards.

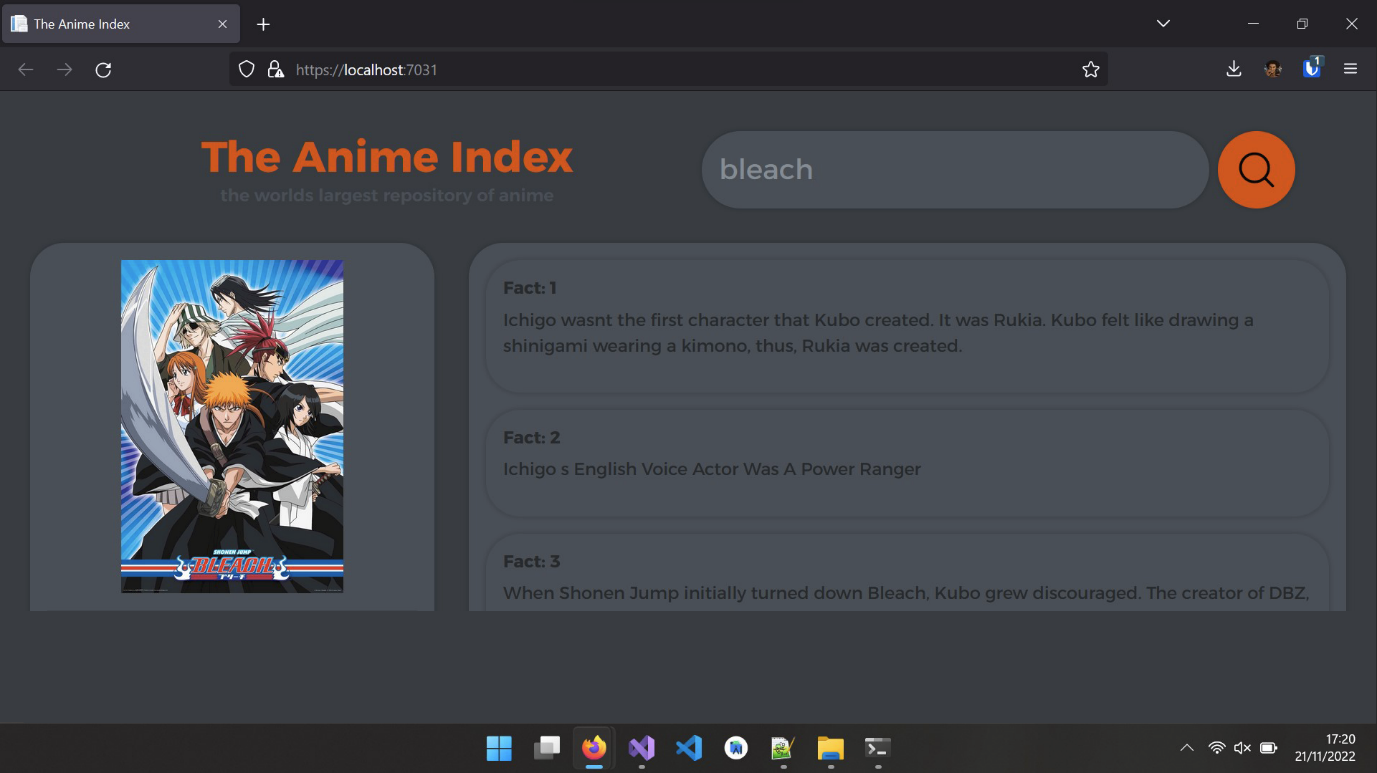
Mock-ups are an important, yet overlooked, step necessary to see the evolution of a design. From this artwork I was able to develop this:



This is the first iteration of the design which replicates all the aspects of the initial mock-up but in a much more presentable finish. I had also removed the default Blazor additions such as the “About” header and the navbar which were unnecessary components of this project.

I had asked for some additional input from none other than my own family members that suggested for me to move around the logo/header and subheading to be alongside the search bar instead of above it. And they also suggested for me to use an accordion instead of just a list to add an additional layer of interactivity and cleanliness.

And so, I came up with a quick mock-up in Photoshop that showed these changes:



As suggested, I had moved the logo to be alongside the search bar and moved up the core content. What I had done as well in practice, was have the content containers only stretch out to the max viewport so there was no unnecessary scrolling on the page, only within the fact cards container.

Along with that, I had also changed the colour scheme slightly to make text more readable, and to make it easier to see the interactable parts of the website which includes highlighting fact cards, search bar and search button on **mouse hover**.

These changes can be observed in the final design:

Graphical user interface, application

Description automatically generated

It is also worth noting that all the styling here is custom made and not utilizing any external Bootstrap libraries. I managed to achieve this look by using (maybe overusing) CSS grids for positioning the containers, flex boxes for aligning elements along rows/columns.

All of this can be found in the index.css file found in the static directory:

Text

Description automatically generated

I am also using a custom font provided by Google Fonts called [Alexandria](https://fonts.google.com/specimen/Alexandria) with font weights of 400, 600 and 700.

I created the accordion (collapsed fact cards, open on click) by swapping a CSS class via a JavaScript function.