CA3 ASP.Net Core Blazor Project

BLAZOR WEB PAGE USING WEBAPI

X00160711 Cianan Nicolai

2022

Contents

[Introduction 2](#_Toc121943599)

[Description of System 2](#_Toc121943600)

[Screenshots 2](#_Toc121943601)

[URI for Deployed App 5](#_Toc121943602)

[https://blazorchucknorris.azurewebsites.net/ 5](#_Toc121943603)

[Description of Testing 5](#_Toc121943604)

[Code Quality & Metrics 5](#_Toc121943605)

[Common Issues Encountered 6](#_Toc121943606)

[API URL 7](#_Toc121943607)

[GitHub Repo URL 7](#_Toc121943608)

# Introduction

An ASP.Net Core Client Side Blazor application is required which provides simple interactive functionality to display data from a public API and provide other simple features. A Single-Page Application is required implemented using Blazor (client side). 1/2 pages will suffice. The application should interact with a public API of your choice (GET only). The API should be RESTful (JSON), a public open API (with no authentication) is fine.

## Description of System

This project uses the free API chucknorris.io to generate facts also known as jokes about Chuck Norris. The features that were intended to deliver upon where random jokes which is done on the first page, random joke by category selection and a search functionality to look through the jokes that contain that word.

Bootstrap for blazor otherwise known as [blazorstrap.io](https://blazorstrap.io/) was used to help with some quick basic formatting.

.Net packages that were used were the following, [Microsoft.AspNetCore.Http](https://www.nuget.org/packages/Microsoft.AspNetCore.Http) and [newtonsoft](https://www.newtonsoft.com/json).

## Screenshots

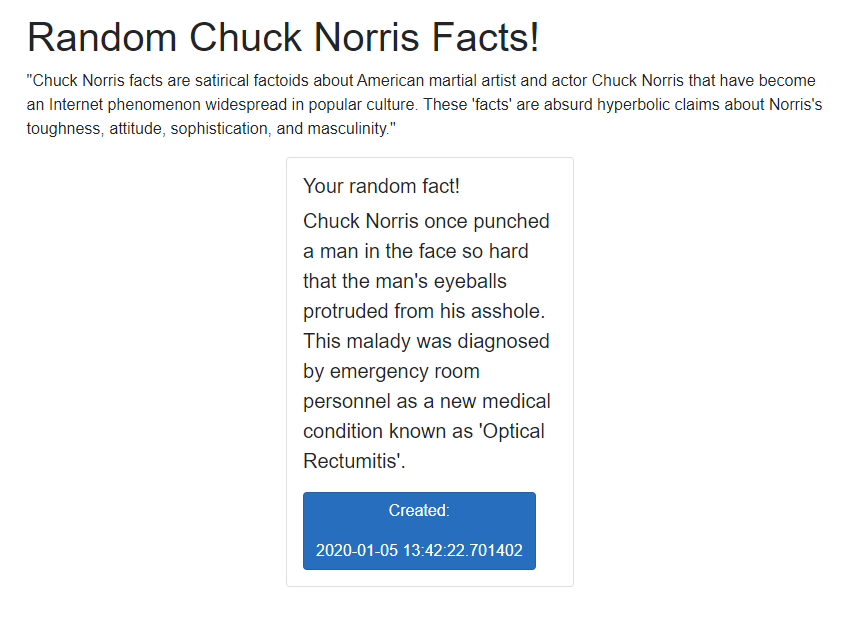
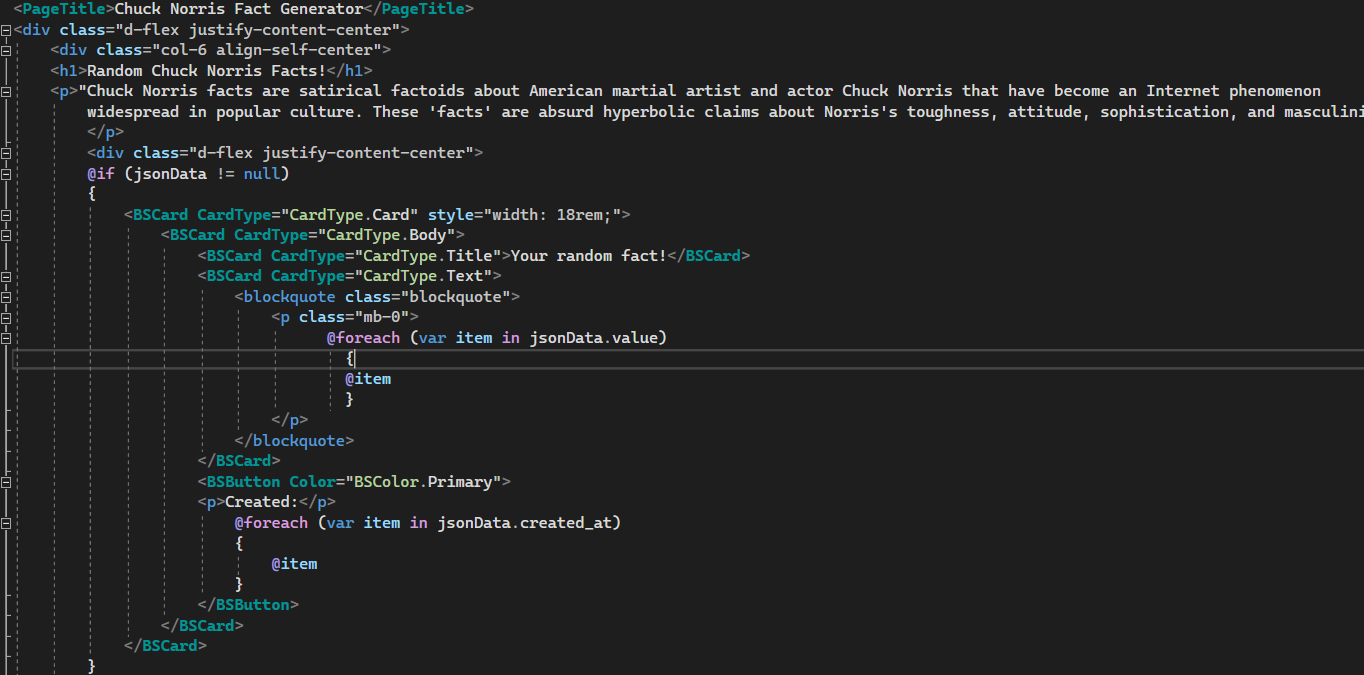


Figure Random Chuck Norris Facts



Here you can see me using the Blazor for Bootstrap to create the cards, mixed with a little bit of html to centre then display the jsonData.



OnInitialisedAysnc() initializes the list, then uses http to GET from the json asynchronously. The jsondata uses newtonsoft to deserialize the json it pulled then populates the Apiresponse model.

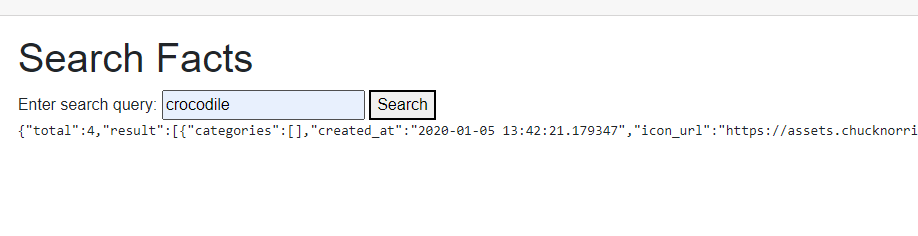
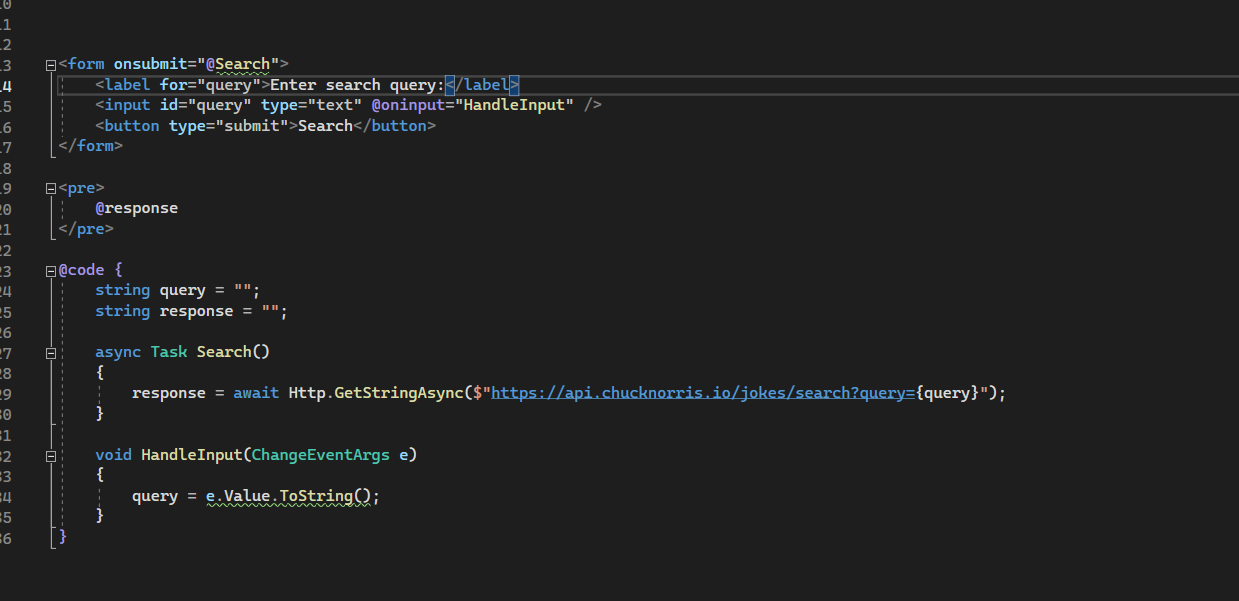


Figure 1 Search Chuck Norris Facts, as of writing this I struggled to display it correctly.



The handle input sets the query to string, then puts the string into the query. The input is taken from the Search.

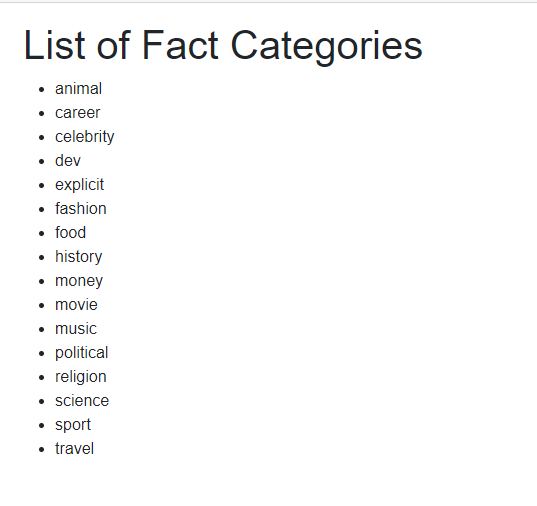
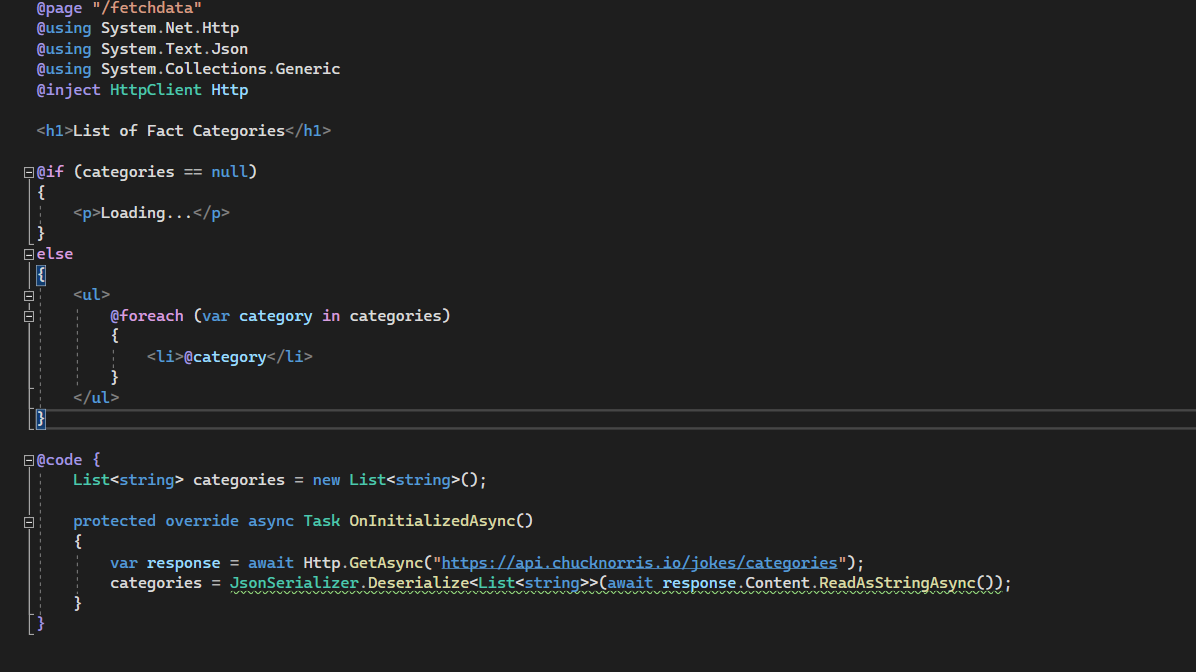


Figure 2 A list of all the categories that the website has,



This pulls from the categories api then displays them.

## URI for Deployed App

## <https://blazorchucknorris.azurewebsites.net/>

## Description of Testing

I set up unit testing however I failed to figure out how to appropriately do so, the theoretical tests I would attempt would be testing that the search function outputs search results, testing that there was a response from the API and testing that the categories correctly showed up as they should.

## Code Quality & Metrics

The visual studio code metric tells me these are of decent quality.





## Common Issues Encountered

The webapi out putted it’s results as json files however these were not in arrays, I didn’t realise this at the start and spent a significant amount of time struggling with formatting the json’s output correctly and still do.

I switched API’s twice during this project, the first one was switched out as RapidAPI limited the number of requests you could make from an API which proved problematic and the second one was switched out as I couldn’t consistently pull from the API.

My original code was I would said a lot more advanced and you can see it under the folder ‘Cocktails’ until ‘spring cleaning’ git push to main, I would have preferred to do my code this way but because of the many issues and the incredible amount of stress I basically stripped down my code to bare bones.

## API URL

https://api.chucknorris.io/

## GitHub Repo URL

https://github.com/CiananNicolai/ASP.Net-Core-Blazor-Project-API