Deliverables

1) I derived the app architecture from Microsoft’s .net Core libraries.

The main structure for the app comes from the default .net Core packages created upon running “*dotnet new webapp -o aspnetcoreapp*”. I deleted most of what I didn’t need from the default packages and implemented the rest by modifying “Index.cshtml” and “Index.cshtml.cs” under the “Pages” directory. I chose to do this for a couple of reasons, but it boiled down to it being the first thing I managed to get working. Due to inexperience with creating web apps from scratch, the overall structure of the app was subjected to a somewhat rashly concocted solution and I probably left several unnecessary files in the repo.

That being said, the work that “Index.cshtml” and “Index.cshtml.cs” do were not put together haphazardly; it is within these two files I established my design of the site. The design I chose here was to section a selected page’s content into categories such that the scraping process was supplemented by knowing exactly what I was looking for in its html. “OnGet()” (located in “Index.cshtml.cs”) is the ‘main method’ equivalent in this app; it selects a project, parses its html, and loads the content into fields that are used by “Index.cshtml” for display. I stuck with this as my end result because the logic ended up being both easy to read and concise.

In order to ensure no project was repeated within a round I stored a mapping from project URL to whether it’s been viewed already within a json file. I chose this because the web app is hosted on the local host there is no need to differentiate between users and thus all the data pertaining to whether a user has seen a file or not can be kept on the local host’s machine. If this mapping had to be sustained across multiple users then I would have used SQL to store and retrieve data.

3) How this design meets the 5 requirements:

1. Parsing the json file produces both a mapping from project URL to whether it’s been visited within the current round and a list of projects that haven’t been visited yet. To ensure a different project must be rendered every time the page is refreshed I select the next project to be displayed from the list.
2. The list is derived from all projects in a mapping marked as unvisited. Once a project is selected, the json file is updated. The only way a project can be rendered is if its marked as unvisited. The only way a project marked visited can be rendered again is if the derived list is empty, at which point all projects in the mapping are reverted to unvisited and a new project is selected.
3. The content displayed in the app’s html is directly derived from the selected project’s page from https://unity.com/madewith. “Index.cshtml” itself is somewhat bare; there are just five divs and they only display information stored in the fields of the c# script which are filled exclusively with information retrieved from the selected project’s page.
4. The c# script only loads information given from <https://unity.com/madewith> (the minimum). Parsing json is O(n), where n is the number of sites. Parsing HTML is O(m), where m is the number of html nodes in the body of the selected project’s page.
5. C# was used.

4) This design could be improved in a number of ways:

* The overall architecture can be optimized. Now that I’ve gained experience with how to build a web app from scratch, next time I could structure the project to be optimized for the specific task it’s designed for.
* SQL might be more performant than parsing json.
* Having an instance that I could host this app from would be much easier for the user. Currently the readme has them going through several different steps just to run the project, which is not friendly. Hosting from not localhost would abstract away the process the user has to go through to boot the app up.
* Videos from vimeo don’t have thumbnails
* The way I scraped for text was simple, and it resulted in the text being very spaced out across the page. Developing a better procedure for scraping would make this much more readable.
* There is a 1/484 chance that the same project be displayed twice if it is the last project of the previous round and the first project of the next round. Having a store that holds the last project used would avoid this all together, but I didn’t realize this until an hour before submission…
* Functionally, this project is correct. Visually, this project is bad. A custom CSS file that would make the app look a lot better.