Daniel Chavez
Professor King
CSCI 324
16 November 2025
Daniel Chavez

Annotated Bibliography

Websites and Articles

"Blazor Components: Event Handling." Microsoft Learn, Microsoft, 10 Nov. 2025, learn.microsoft.com/en-us/aspnet/core/blazor/components/event-handling?view=aspnetcore-10.0

This documentation was used to understand how to manage user interactions on Blazor web pages. I learned how to use things like @onclick and @onbing and used them in the sql search features, populating the sql search bar, and also sorting objects on the front-end side. It was directly used in the Spacecraft.razor page.

bootdey. 9 Oct. 2025, www.bootdey.com/snippets/view/Gradients-dashboard-cards.

This design snipped provided me with inspiration for the CSS design across all razor pages in the application. Specifically things like gradients and dashboard card layouts, and then pieces of the design were later given to Claude to general css code.

codepen. 9 Oct. 2025, codepen.io/alvarotrigo/pen/zYEGVdV.

Similar to the above templates, this also was used for inspiration for the CSS design, specifically things like layout of the client side interface. Like I mentioned earlier, pieces were later used and grabbed and given to Claude to create the css for the pages.

codepen. 9 Oct. 2025, codepen.io/alvarotrigo/pen/ExwYyNW.

Similar to the above templates, this was the third source for inspiration for the CSS design, specifically things like layout of the client side interface. Like I mentioned earlier, pieces were later used and grabbed and given to Claude to create the css for the pages.

dribbble. 9 Oct. 2025, dribbble.com/shots/13933793-School-App-Loggin.

Similar to the above templates, this was the fourth source for inspiration for the CSS design, specifically things like layout of the client side interface. Like I mentioned earlier, pieces were later used and grabbed and given to Claude to create the css for the pages. I specifically grabbed things like card layout and design.

"Elements/iframe." MDN Web Docs, MDN, 14 Nov. 2025, developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/iframe.

This documentation provided me with an understanding of how iframes work and how to implement them. This allowed me to integrate a YouTube video player onto the *ISS.razor* page, which displayed a live feed of the ISS.

"how to implement a function in blazor webpage." Google Search, 27 Oct. 2025.

The google AI overview provided me with an example of how to correctly call a C# function from the backend to the frontend and then how to use it in a Blazor component. This was later used in all razor pages, but a specific example is in ISS.razor page in the @code section.

"Order a list of objects by its DateTime property." Stack Overflow, 29 Oct. 2025, stackoverflow.com/questions/25053887/order-a-list-of-objects-by-its-datetime-property.

This forum gave me specific C# code and method to correctly sort custom objects (like space programs) based on fields like Datetime. This was used in Programs.razor.cs so that the space Programs can be sorted in chronological order on the Programs page.

Open Meteo API Documentation. 18 Oct. 2025, open-meteo.com.

I read the APU document that allowed me to create the data models, and the API calls for the weather service. I also learned the return types, data format, and parameters needed to call the API. This allowed me to build the openMeteoService.cs service.

"ServiceProvider.CreateScope in .NET." *The Medium*, Zaynt, 25 Oct. 2025, medium.com/@zaynt.dev/understanding-serviceprovider-createscope-in-net-a-deep-dive-a0d4549cc4b1.

This article was useful for debugging a bug I had in the CacheRefresherService.cs file. I learned the idea of service scopes in .NET, which helped clarify that a scope was necessary to execute different services in the background. The issues was I could not call any services or functions outside the program.

Documentation

"ASP.NET Core Razor Pages." Microsoft Learn, Microsoft, 27 Oct. 2025, learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-10.0.

This documentation served as the foundation for understanding how the structure of a Razor Page works in Blazor. It taught me how the front-end components (the .razor files) connect to and call the backend logic (the .cs files) and background functions.

"Asynchronous programming." Microsoft Learn, Microsoft, 27 Oct. 2025, learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming/.

This documentation helped me understand the basic idea of C# asynchronous programming, specifically, how to use things like await Task.WaitAll(). This was used in all the Balzor Pages when we needed to call the API from the services module. As well, this is used in 3D.razor to show the ISS moving. By using async it helps improve loading and also allows the user to see a “live” webpage without having to refresh for new data.

"Background tasks with hosted services." Microsoft Learn, Microsoft, 25 Oct. 2025, learn.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-9.0

This documentation helped me understand how to implement the caching mechanism found inside the CacheRefresherService.cs. I learned how to create a background service that would call an API endpoint every 30 seconds to make sure the data was new and also give the illusion of cached calls. This can be seen when we load 3d and 2d webpages and they no longer take 30 seconds to a minute to load. This is because the refresher already called it previously.

"C# Reference: async." Microsoft Learn, Microsoft, 18 Oct. 2025, learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async.

This documentation was used to help me understand how to async calls as well, but this one specifically focused and touched on the async and async functions. This was used to help create my first service (openMeteoService.cs). This is later used in all services.

"EF Core: Entity Properties." Microsoft Learn, Microsoft, 5 Nov. 2025, learn.microsoft.com/en-us/ef/core/modeling/entity-properties?tabs=data-annotations%2Cwith-nrt

When trying to figure out how to implement a database in our application, I referenced this doc, which provided me with how to create and implement database entities using the Entity Framework. It explained how to use EF Core to store custom C# objects within a SQL database, which was necessary because traditional SQL columns do not handle complex JSON data well.

"EF Core: Getting Started." Microsoft Learn, Microsoft, 5 Nov. 2025, learn.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=netcore-cli.

This was the primary start point for setting up the Data directories and using EF core in a Blazor webpage. This was specifically used in Data/SpaceTrackerDbContext.cs which taught me how to connect a SQL database, set up the context, and define the entities that would be used to manage the columns or schema.

"How to: Serialize and deserialize JSON." Microsoft Learn, Microsoft, 6 Nov. 2025, learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/how-to.

I used this to help debug various JSON serialization and deserialization issues in the API services. It showed me how to use and understand System.Text.Json library, which allowed me to properly parse raw API json to our custom models.

"Overview of ASP.NET MVC Models." Microsoft Learn, Microsoft, 18 Oct. 2025, learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-aspnet-mvc3/cs/adding-a-model.

This documentation taught me how Models work in C# web applications. I used this to structure the data return types from the various APIs, which led to the creation of all the model classes seen in the application. One example is the n2yoModel.cs.

"SQLite database provider." Microsoft Learn, Microsoft, 25 Oct. 2025, learn.microsoft.com/en-us/dotnet/standard/data/sqlite/?tabs=net-cli.

This documentation showed me how to integrate a SQLite db into the C# application. This was necessary for SpaceDevsService.cs where data from there is rarely updated, so by creating a local database instead of calling the api every 10 seconds, we can just query the database, making loading times much faster.

"Working with DbContext: AddAsync." Microsoft Learn, Microsoft, 14 Nov. 2025, learn.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext.addasync?view=efcore-9.0.

This documentation was used to help seed the databases. It showed how to properly call the API, and the asynchronous functions needed to add new entities to the database. An example of this can be seen in AstronautSeeder.cs.

API Documentation

N2YO API Documentation. 18 Oct. 2025, www.n2yo.com/api/.

This documentation showed me information about the API return objects for satellite data. I used it to create the n2yoModel.cs model and the n2yoService.cs service. As well, it helped me figure out the specifics of the API call structure and parameters.

RocketLaunch Live API Documentation. 22 Oct. 2025, www.rocketlaunch.live/api.

This documentation showed me information about the API return objects for the rocket launch API. I used it to create the rocketLiveModel.cs model and the rocketLiveModelService.cs service.

SpaceDevs API Documentation. 22 Oct. 2025, ll.thespacedevs.com/docs/.

This documentation showed me information about the API return objects for the Space Devs API. I learned the API's return types, call, and parameters. It was also used to help debug issues with serialization in spaceDevsModel.cs and creating the proper models for each function.

AI and Utility Tools

ChatGPT. 16 Oct., 20, 25, 28-29 Oct.; 5, 10, 14, 16 Nov. 2025.

I used ChatGPT all over the place for code cleanup, commenting, and organization for all C# Blazor files (e.g., Astronauts.razor, Data/Entities files, and all Service.cs files). It also helped with nested objects/models in the n2yoModel.cs. It was also used in rocketLaunchLiveSerice.cs when I needed to nest an object inside a list as the API was returning a list, not just one json object. Finally, it assisted in hardcoding image overrides in *SpaceProgramSeeder.cs* to fix malformed links found from the spaceDevs api. This taught me how to debug models in C#, how to edit objects from a database without actually entering the database, and how json objects work when returned from an api.

Claude.Ai. 27, 28 Oct.; 9 Nov. 2025.

CSS Generation: Claude.Ai served as the primary tool for writing the actual CSS code for all css files (ISS.razor.css, Programs.razor.css, Spacecraft.razor.css). I provided it with specific design ideas and chunks of CSS from the various online templates (bootdey, codepen, etc.), and it put them into css files. This taught me a lot about creating a webpage that looks appealing and nice.