Daniel Chavez, Anthony Petrosino, and Jackson Wang

Professor King

CSCI 324

16 November 2025

C# Report

Our SpaceTracker application is designed to provide real-time information about rocket launches, upcoming rocket launches, and overall provide information about all things space. This includes things like space programs, famous astronauts, International Space Station data like its position, live feed, overhead passes, etc. The app was created in C# using a Blazor application for the front end and then C# for the backend.

One of the benefits of working with C# is its support for asynchronous programming. The async and await features of C# were heavily used to handle getting data from the frontend to the backend. For example, in 3D.razor the ISS position updates every 5 seconds without freezing the UI. These async patterns work seamlessly with C#'s HttpClient class, which we used extensively to make non-blocking HTTP requests to various space APIs. Since all our API calls were async, the application could fetch data from multiple endpoints simultaneously without freezing the UI, which meant users could navigate between pages while data was still loading in the background.

Another benefit of C# is its great support for database usage and support for LINQ (Language Integrated Query). LINQ allows for the front end to directly interact and query a database in our application. For example, programs can be sorted by start dates using: programs = programs.OrderBy(p => p.StartDate ?? DateTime.MaxValue).ToList();. This ease of access allows us to sort or manipulate data without having to add loops or implement sorting functions.

Another benefit of C# is our application benefited heavily from the vast .NET ecosystem. The .NET ecosystem allows for cross platform support, so MacOS, Linux, and Windows users were all able to run, edit, and implement features without any issues. C# comes with vast libraries like Blazor which is great for Web applications. The .NET ecosystem allows for the creation of "Models" which are conceptually objects, and this is how the core of the data moved around the application. In our services, we would call the api, receive a json response, and create a "Model" for that json response which we would then use in our application. The benefit of this over plain objects is that a model can be thrown into a sqlite db using .NET EF core which allows us to store nested Models like Program_Model.Image_Model.Image_url.