Using the documentation for ASP.NET and the YouTube series on building a simple to do api, I have built a simple to do list api, having CRUD operations for the to do tasks I learned about Middlewares that are used for handling pipelines, for example redirecting certain URLs to others, logging services or memory cache services.

I learned about CORS, and how it is used to filter out unauthorized requests from other origins(other ports / subdomains / domains). In ASP.NET we need to add a policy using builder.Services.AddCors and utilize it with app.UseCors.

I looked into SQLite, a simple but effective database engine, perfect for prototyping and projects that are smaller, but might prove inadequate for the use of the app that we will build, probably something among the lines of PostgreSQL or MongoDB.

MongoDB will be chosen as a modern and robust DB to be used in our project, as it will be perfect for handling our messages, feed, 'To Do's, and other data. Also having some experience with Firestore databases is a plus for me.

I looked into One to One(eg. data about user)/ One to Many(eg. A user has added many tasks to be done)/ Many to Many(no example found right now) relationships, and how they could be used to model out data in our future database.

I looked into Authentication(who the user is) and Authorization(what the user can do). For authentication we will probably go at first with JWT, being a simple way to manage user connection to our services. In ASP.NET we can use the JwtBearer for this.

I finalized by doing a small sketch of what data we should know about our user and task