Project 2– Carter Manley IT DEV 162-800

In this project, I decided to do a deep dive into Angular and the API technology. My project was originally an attempt to make a full stack application whose frontend fully communicates with the API backend. While I was unsuccessful in my original goal, the result was a deeper dive into the related technologies than I had planned, attempting bug fixes, refactoring code, etc.   
  
The result is the product of dozens of hours of hard work, and while in the end, I wasn’t able to make it run as intended, it reflects my best attempt to dive into a complex program and essentially rebuild it from scratch.

The main problem was getting the site to route properly to the urls in my Routes class. Also, in building the project in Angular 17, much was different than what we learned in class. I learned that in 17, the app.routing, app.modules, app.config are all done in completely different ways than in older versions of Angular. I was successful in getting some of the application data to communicate properly, and was finally able to get the browser to load in my HTML. However, the browser navigation to my components, view-members, and create-members, sadly does not function.

Requirements:  
  
Angular 17  
Node JS v.18.13.0

Visual Studio Code

Visual Studio 2022

\*I ran this app by running the *ng serve* command inside of:

*FullStack UI/FullStackUI/*

in Visual Studio Code, and running the API inside of:

*FullStack API*

inside of Visual Studio 2022.

This application is a registration and viewing database for members (for a club, retailer, etc.)

* Using Visual Studio 2022, I created a .Net Core Web API template. Then, I modified the template for my custom model, “member”. Members include the fields:
  + Name
  + Email
  + Phone
  + Points
  + Department
* The API was set up to connect to Swagger UI when it runs.
* created Get and Post endpoints in the API.
* Using Visual Studio, installed a new Angular frontend using:

*npm install -g @angular/cli*

* Created a folder to represent the model, “Member” in my Angular app.
* Created components to display data— View-Member, and Create-Member.
* Installed angular material for dynamically displaying component elements.
* Connected my app to my components, and connected each component to their own .ts file.

\*\*\*\*\* This is where I ran into some issues\*\*\*\*\*

At this point, I had trouble getting all components communicating properly. What I learned is that in Angular 17, the components are standalone by default, meaning that in their .ts files, they each import their own modules, like BrowserModule, RouterModule, etc. This became a problem when trying to set things up with an app.module.ts file to control all imports from a higher level. When I fixed an error in one file, it would prompt a new error in another file, and so it became a never-ending chain of error messages.

\*\*\*A note on success and failure\*\*\*

I attempted to build this application initially by following a tutorial video on YouTube. I was interested in trying a new approach from what we had learned in class. When I cook a new dish, I often like to try multiple recipes, and comparing what I learn from both helps me understand the process better. While I learned a lot from this process, I think I would have been better served trying something simpler to produce a completed project. However, the resulting hours I have sunk into this project have made me much more knowledgeable than I think a successfully completed project would have.