LEARNING SIGNALR

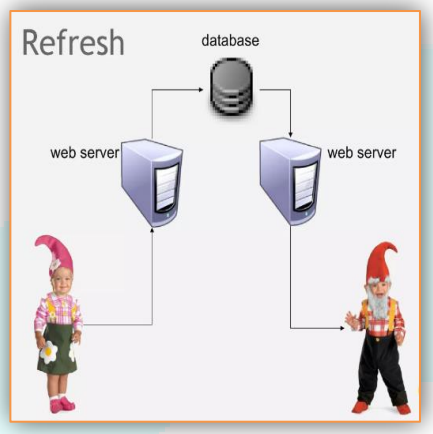
Comprehensive Guide To Learn SignalR.

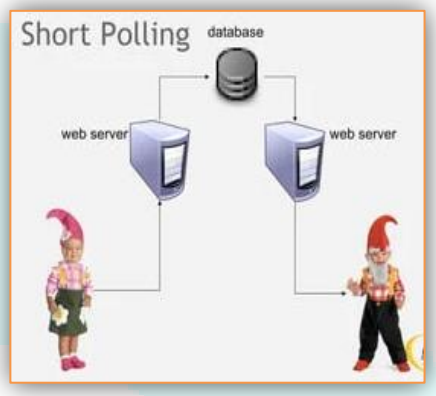
Table of Contents:

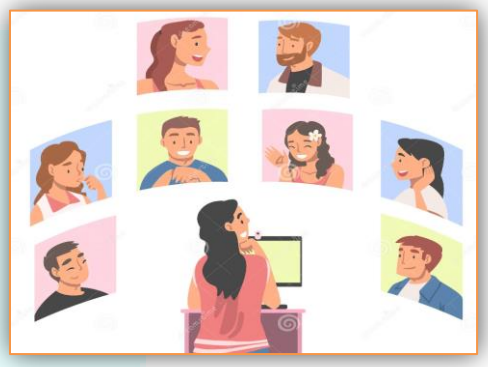
- ✓ **Created Under guidance of Nilesh Prajapati.**
- ✓ **Level of Document: Basic.**
- ✓ **Outcome: Basic understanding of SignalR and able to create a basic project.**

1. Introduction:

1.1 Overview of Real-Time Web Communication:







Importance of Real-Time Communication

- ✓ Enhanced User Engagement
- ✓ Efficient Collaboration
- ✓ Live Updates and Notifications
- ✓ Interactive User Interfaces

Impact on User Experience:

- ✓ Reduced Latency:
- ✓ Increased Interactivity:
- ✓ Timely Feedback
- ✓ Live Collaboration

In conclusion, embracing real-time communication on the web is pivotal for creating user-centric applications that cater to the evolving expectations of today's audience. Whether in social media, business tools, or interactive websites, the impact of real-time communication resonates across various digital experiences, shaping a more responsive and engaging online world.

1.2 Importance of SignalR in Web Development

One powerful tool that facilitates this level of interactivity is SignalR, a library for building real-time applications with ASP.NET and .NET Core. SignalR enriches web applications by enabling seamless, bidirectional communication between the server and clients, fostering a more engaging and responsive environment.

Exploring the Role of SignalR:

- ✓ Real-Time Bidirectional Communication
- ✓ Enhanced User Engagement
- ✓ Live Collaboration
- ✓ Push Notifications
- ✓ Dynamic Content Synchronization

Benefits for Interactive Web Applications:

- ✓ Real-Time Dashboards
- ✓ Live Chat and Messaging
- ✓ Online Gaming
- ✓ Stock Ticker and Financial Updates.
- ✓ Live Polling and Voting

In essence, SignalR stands as a pivotal tool for developers seeking to elevate the interactivity and responsiveness of their web applications. By incorporating SignalR, developers can craft dynamic, real-time experiences that captivate users and align with the evolving expectations of modern web interactions..

1.3 Introduction to SignalR

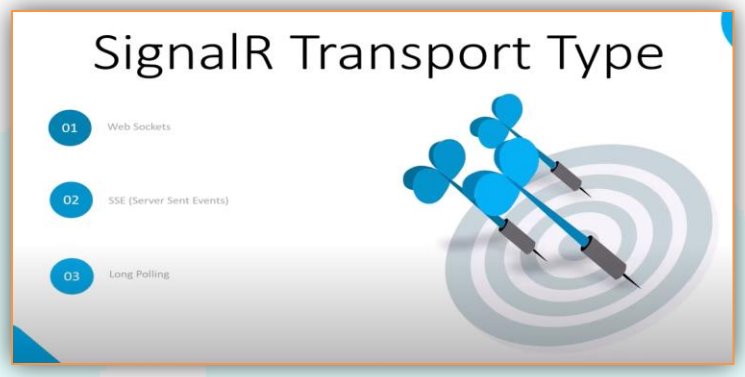
- ✓ SignalR stands out as a powerful and versatile library designed for building real-time applications within the ASP.NET and .NET Core ecosystems. Introduced by Microsoft, SignalR simplifies the complexities of real-time communication by providing developers with a seamless framework to implement interactive features in web applications.
- ✓ Library for implementation of realtime communication.
- ✓ It wrapper over 3 transport protocol:
  - Web Socket
  - Long Polling
  - Server Sent Event (SSE)

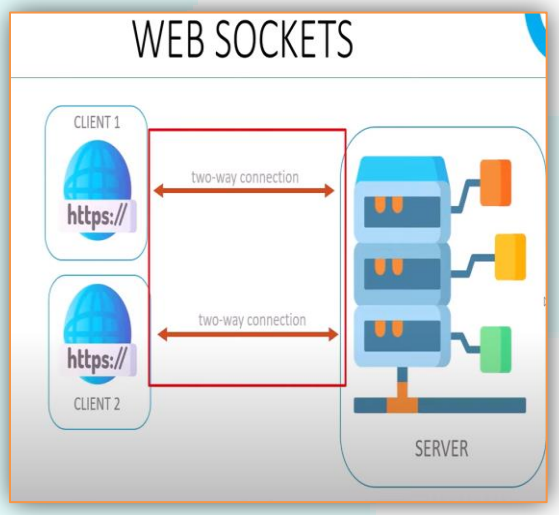
2. Prerequisites:

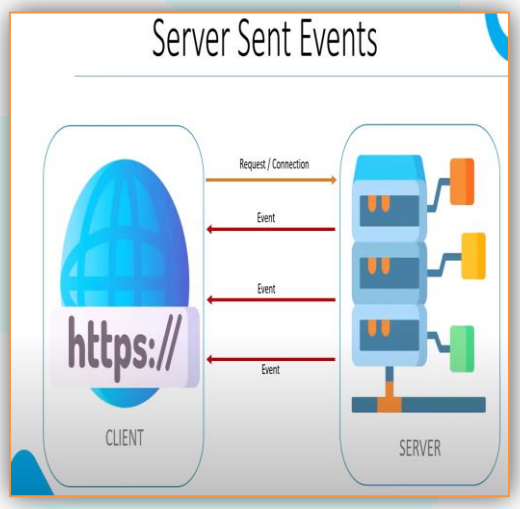
- ✓ Basic Understanding of Web Development.
- ✓ Set Up a Development Environment.
- ✓ Protocol Http request and response.
- ✓ Client server Architecture.

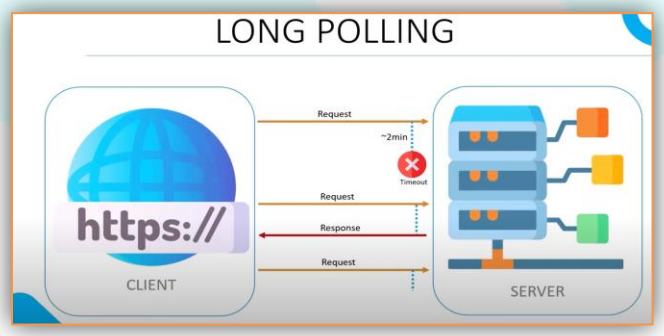
3. SignalR Overview:

3.1 Transport Type.



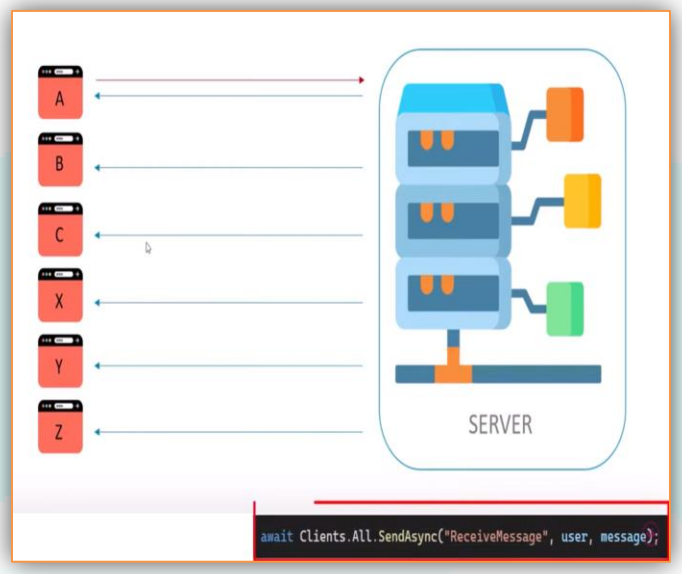




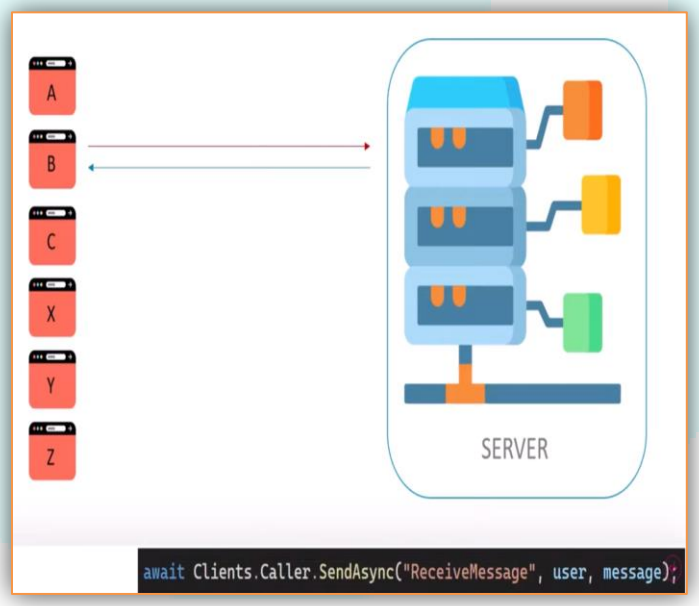


3.2 Properties and Method.

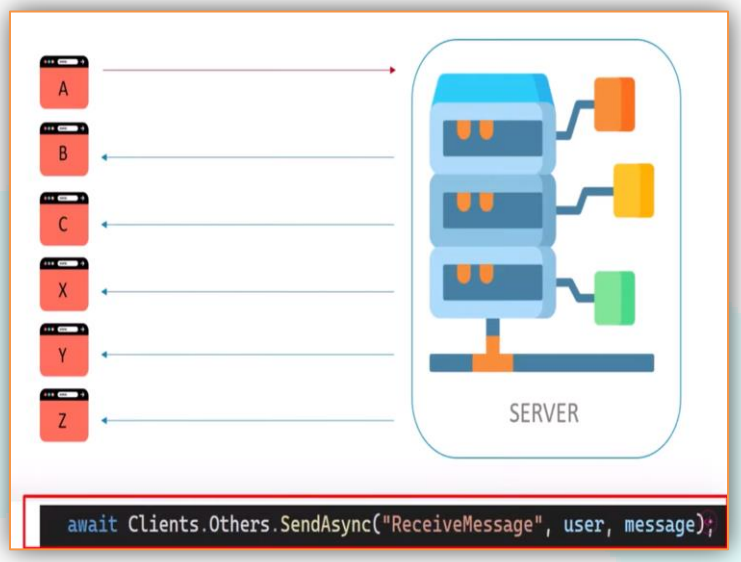
- ✓ Send data to all client



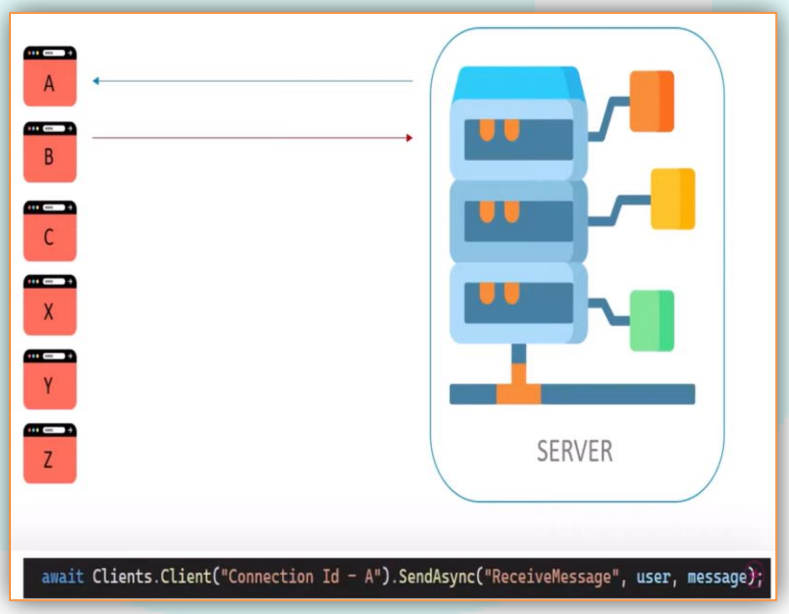
- ✓ Send data to caller client.



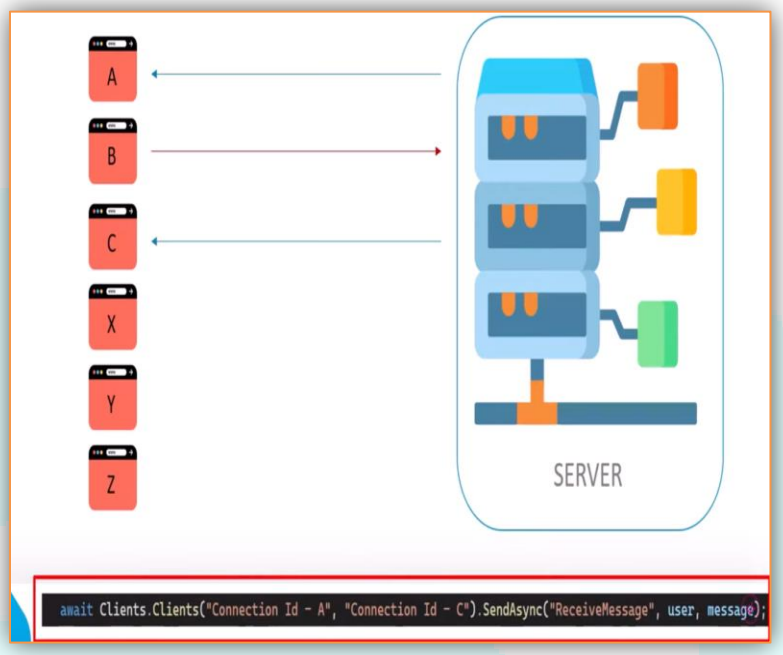
- ✓ Send data to all client except caller client.



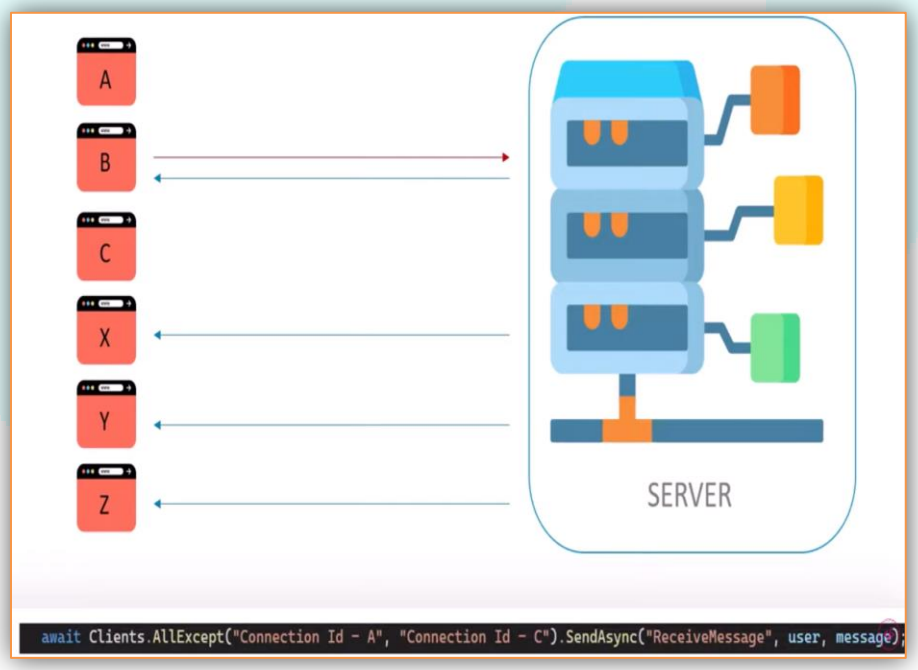
- ✓ Send data to specific client (Tip : Context.ConnectionId give the user connection id of current caller client)



- ✓ Send data to specific clients.



- ✓ Send data to all client but except some client.



✓ Send and Invoke method in signalR.

Snippet code :hubconnection.send(),hubconnection.invoke()

- Send returns a promise that is resolved when the client has sent the invocation to the server, or an error occurred. The server may still be handling the invocation when the promise resolves.

- Invoke returns a promise that is resolved when the server has finished invoking the method (or an error occurred). In addition, the Invoke promise can receive a result from the server method, if the server returns a result.

SignalR provides several default functions and methods that can be used on both the server and client sides. Here are some of the key default functions of SignalR:

✓ Server-Side Default Functions:

✓ OnConnectedAsync()` and `OnDisconnectedAsync(Exception exception)`

- **OnConnectedAsync**: This method is called when a new client connects to the hub.
- **OnDisconnectedAsyn**: Called when a client disconnects, either voluntarily or due to an exception.

```
public override async Task OnConnectedAsync()
{
    // Code to execute when a client connects.
}

public override async Task OnDisconnectedAsync(Exception exception)
{
    // Code to execute when a client disconnects.
}
```

✓ OnReconnected():

- Called when a client that was previously disconnected successfully reconnects.

```
public override async Task OnReconnected()
{
    // Code to execute when a client successfully reconnects.
}
```

Client-Side Default Functions:

- ✓ `start()` and `stop()`:

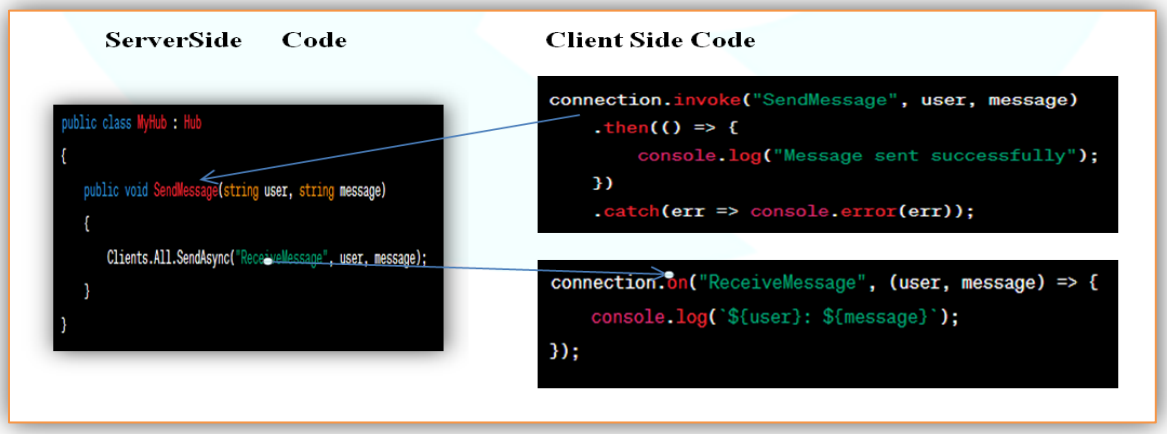
- ✓ start(): Initiates the connection to the hub.
- ✓ stop(): Closes the connection to the hub.

```
const connection = new signalR.HubConnectionBuilder()
    .withUrl("/hub")
    .build();

connection.start().then(() => {
    console.log("Connection started");
}).catch(err => console.error(err));
```

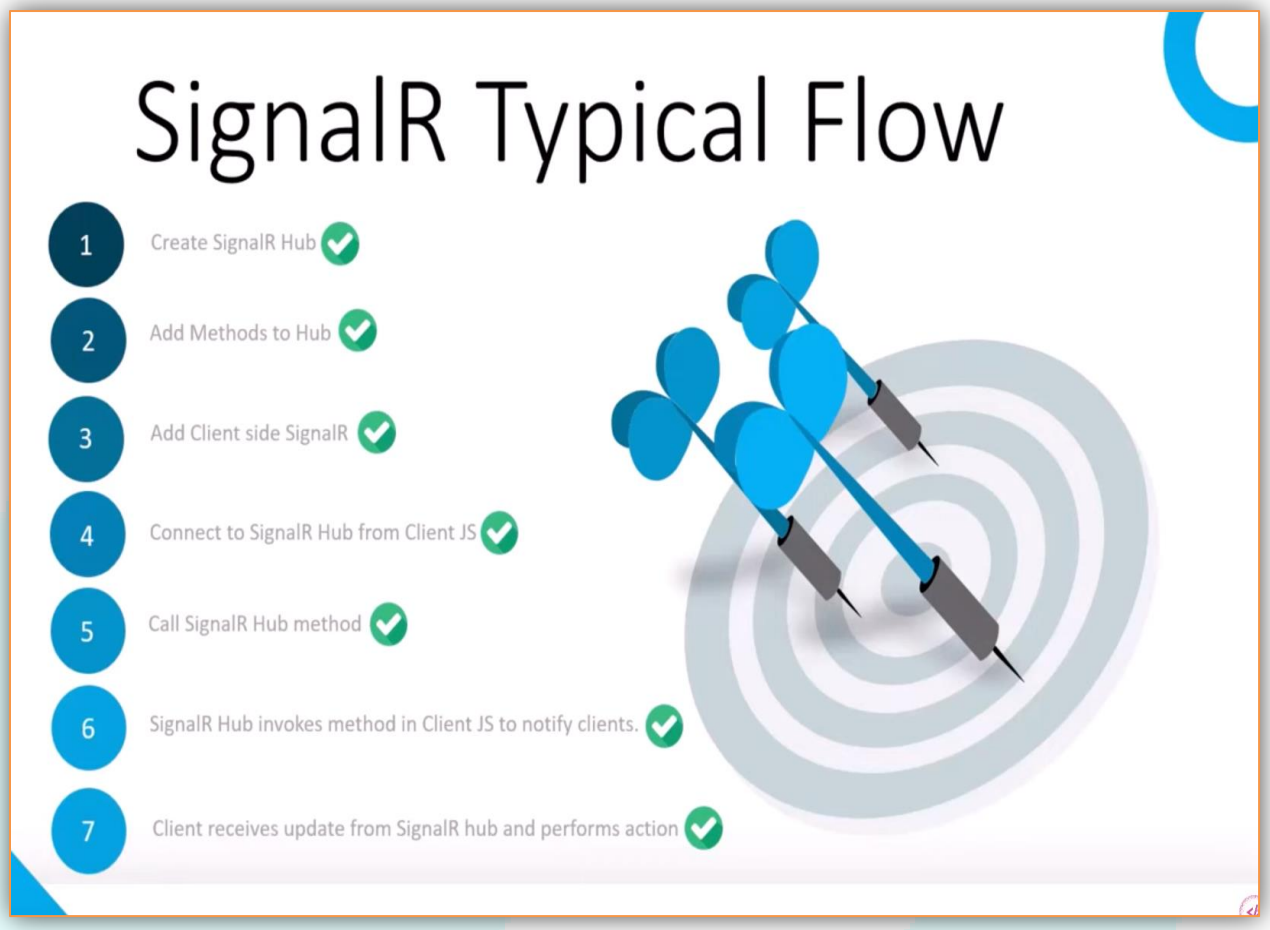
- ✓ state:
  - Represents the state of the connection, providing information such as whether the connection is connected or disconnected.

```
console.log(connection.state);
```

- ✓ Possible values: "Connecting", "Connected", "Reconnecting", "Disconnected"



4. Setting up SignalR:



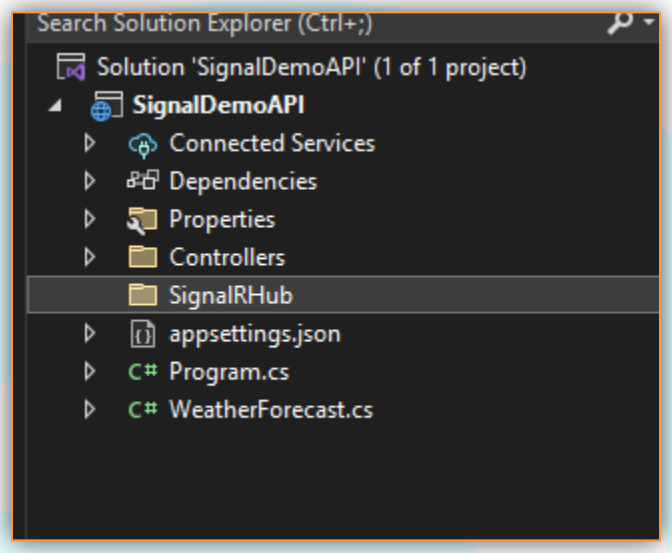
4.1 Installing SignalR

- ✓ Considering you have idea of creating .net core web Api project.(Click me to learn about how to create me.)
- ✓ For angular side and .net core api has inbuilt library for signalr so no need to install from nuget package.

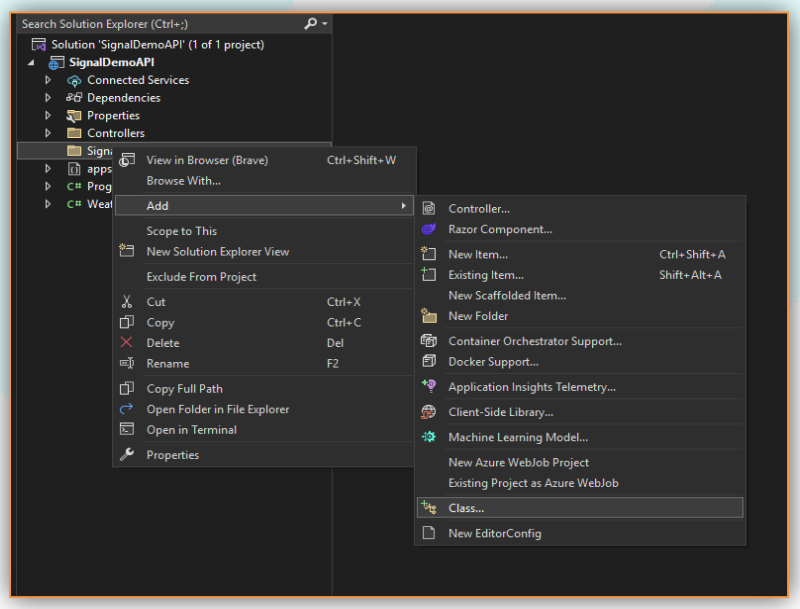
```
npm install @microsoft/signalr
```

4.2 Creating a SignalR Hub

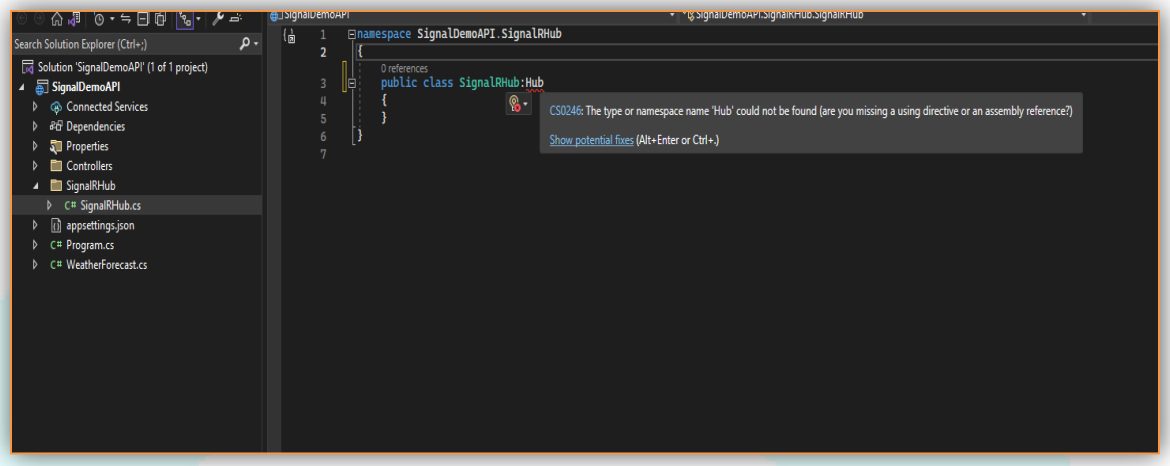
✓ Create a folder in your webapi .net core project with name SignalRHub.

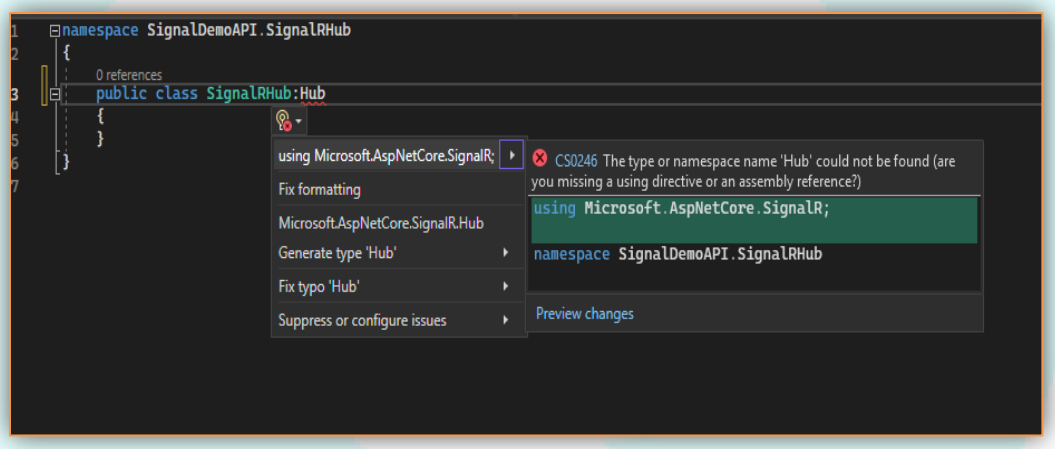


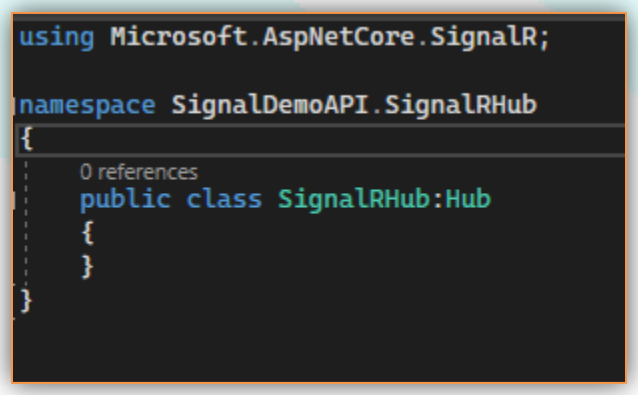
✓ Create a class in that folder with name SignalRHub.



- ✓ Inherit the SignalRHub class with Hub class from namespace call Microsoft.AspNetCore.SignalR.







```
using Microsoft.AspNetCore.SignalR;

namespace SignalDemoAPI.SignalRHub
{
    0 references
    public class SignalRHub:Hub
    {
        //first method to call when Client connects.
        0 references
        public override  Task OnConnectedAsync()
        {
            Console.WriteLine(Context.ConnectionId);    // Id  of Connected Client
            return base.OnConnectedAsync();
        }

        0 references
        public override  Task OnDisconnectedAsync(Exception? exception)
        {
            return base.OnDisconnectedAsync(exception);
        }

        0 references
        public async Task NewMessage(string msg)
        {
            await Clients.All.SendAsync("MessageReceived", msg);
        }

    }
}
```

- ✓ Created NewMessage Method method in Hub .
- ✓ Now its time to register SignalR services in Program.cs file.

Program.cs* | SignalRHub.cs

SignalDemoAPI

```
1   var builder = WebApplication.CreateBuilder(args);
2
3   // Add services to the container.
4
5   builder.Services.AddControllers();
6   // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
7   builder.Services.AddEndpointsApiExplorer();
8   builder.Services.AddSignalR(); // Add SignalR service
9   builder.Services.AddSwaggerGen();
10
11  var app = builder.Build();
12
13  // Configure the HTTP request pipeline.
14  if (app.Environment.IsDevelopment())
15  {
16      app.UseSwagger();
17      app.UseSwaggerUI();
18  }
19
20  app.UseHttpsRedirection();
21
22  app.UseAuthorization();
23
24  app.MapControllers();
25
26  app.Run();
27
```

- ✓ Add cors policy so that our client will able to connect the hub and server.

```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSignalR(); // Add SignalR service
builder.Services.AddCors(options =>
{
    options.AddPolicy("CorsPolicy",
        builder =>
        {
            builder.WithOrigins("http://localhost:4200")
                   .AllowAnyMethod()
                   .AllowCredentials()
                   .AllowAnyHeader();
        });
});   // Add Cors

// Add Swagger for API documentation
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "SignalDemo API", Version = "v1" });
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseCors("CorsPolicy");   // Add here
app.UseAuthorization();


app.MapControllers();

app.Run();
```

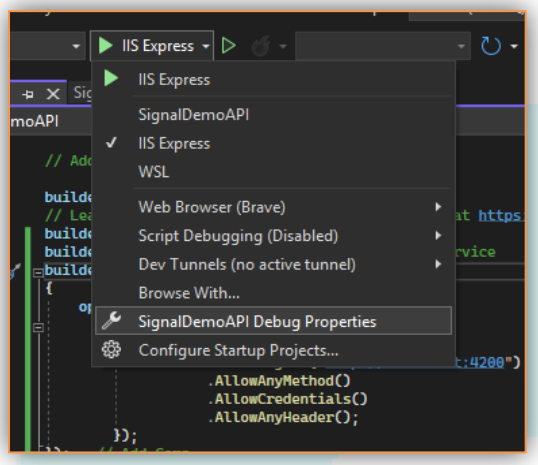
- ✓ Now let add end point of our hub.

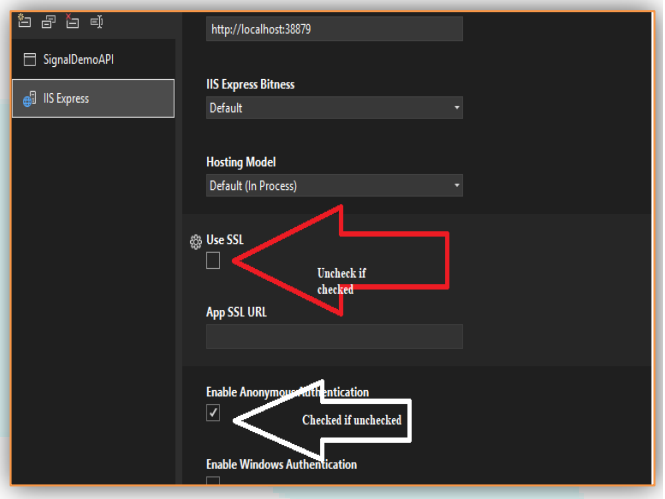
```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSignalR(); // Add SignalR service
builder.Services.AddCors(options =>
{
    options.AddPolicy("CorsPolicy",
        builder =>
        {
            builder.WithOrigins("http://localhost:4200")
                   .AllowAnyMethod()
                   .AllowCredentials()
                   .AllowAnyHeader();
        });
});   // Add Cors

// Add Swagger for API documentation
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "SignalDemo API", Version = "v1" });
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseCors("CorsPolicy");
app.UseRouting();// Add here
app.UseAuthorization();
app.UseEndpoints(endpoints =>
{
    endpoints.MapHub<SignalRHub>("/SignalRHub");
});

app.MapControllers();
```

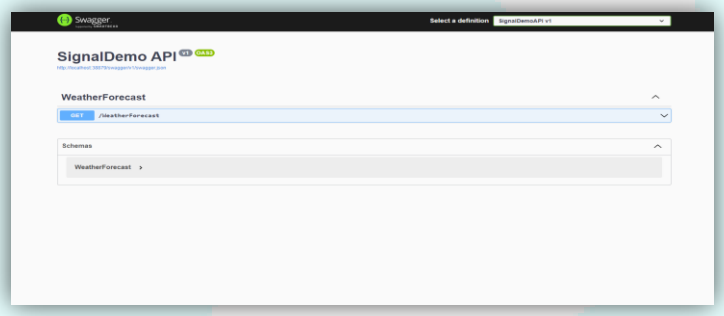
Hub Class Name

Name should in client call I will mention while client side writing

- Before we go client side code we have to check two thinks in VS code .







4.3 Integrating SignalR with Frontend Technologies: (Angular)

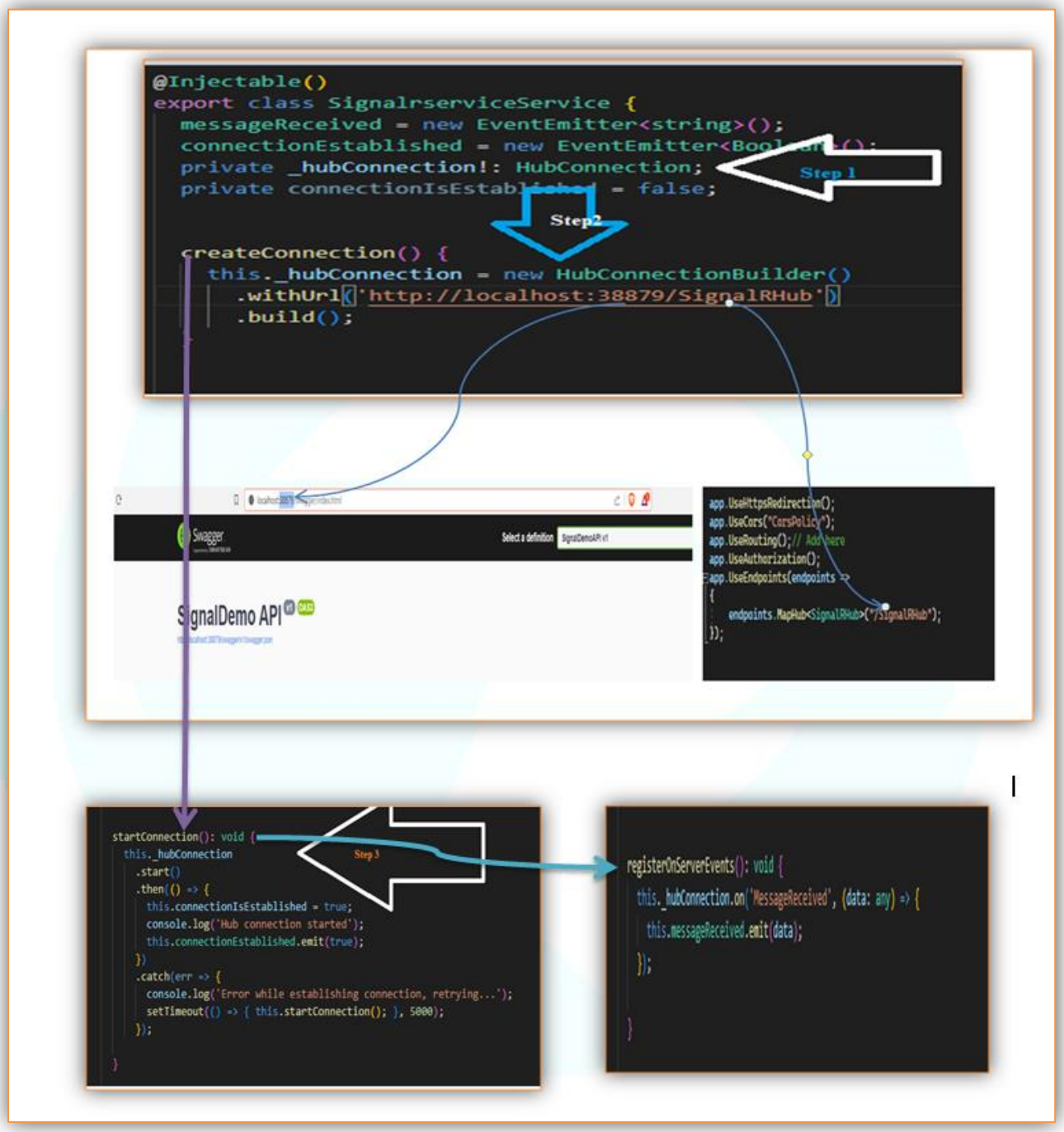
- ✓ Considering you have idea of how to create angular project.(Click me to know how to create angular project.)
- ✓ Now let create a service in angular project with name signalrservice.

```
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide\src\app> ng g s signalrservice
CREATE src/app/signalrservice.service.spec.ts (413 bytes)
CREATE src/app/signalrservice.service.ts (152 bytes)
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide\src\app> ng serve

Initial Chunk Files | Names         |  Raw Size
polyfills.js        | polyfills     |  83.46 kB |
main.js             | main          |  23.18 kB |
styles.css          | styles        |  96 bytes |

                    | Initial Total |  106.74 kB

Application bundle generation complete. [2.422 seconds]
Watch mode enabled. Watching for file changes...
  ➜  Local:   http://localhost:4200/
  ➜  press h + enter to show help
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide\src\app>
```

- ✓ Installing SignalR in angular project.

```
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide> npm install @microsoft/signalr

added 16 packages, and audited 928 packages in 19s

119 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\sit318.SIT\Desktop\GA-Superarilife\SoignalRDemoClientSide>
```

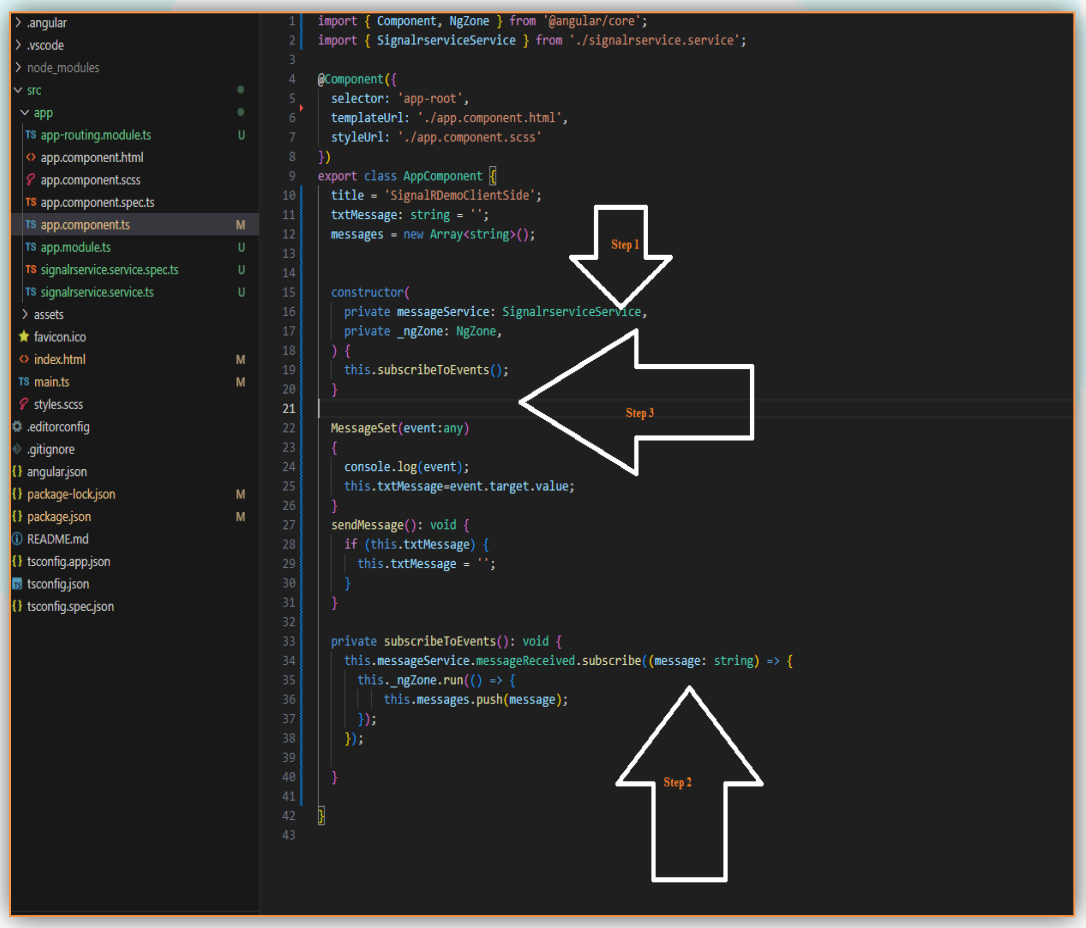
- ✓ Now its time to work on signalrservice and write client side code to connect with hub.(SignalRHub in our case).
- ✓ Steps to be in mind:
- CreateConnection→StartConnection→RegisterEvent.
- Creating object of HubConnectionBuilder.
- Basic recap of on,start,invoke…


@Injectable()
export class SignalrserviceService {
messageReceived = new EventEmitter<string>();
connectionEstablished = new EventEmitter<Bool
private _hubConnection!: HubConnection;
private connectionIsEstabli = false;
Step 1
Step2
createConnection() {
this._hubConnection = new HubConnectionBuilder()
.withUrl('http://localhost:38879/SignalRHub')
.build();
}
Swagger
Select a definition SignalDemoAPI v1
SignalDemo API
app.UseHttpsRedirection();
app.UseCors("CorsPolicy");
app.UseRouting();// Add here
app.UseAuthorization();
app.UseEndpoints(endpoints =>
{
endpoints.MapHub<SignalRHub>("/signalRHub");
});
startConnection(): void {
this._hubConnection
.start()
.then(() => {
this.connectionIsEstablished = true;
console.log('Hub connection started');
this.connectionEstablished.emit(true);
})
.catch(err => {
console.log('Error while establishing connection, retrying...');
setTimeout(() => { this.startConnection(); }, 5000);
});
}
Step 3
registerOnServerEvents(): void {
this._hubConnection.on('MessageReceived', (data: any) => {
this.messageReceived.emit(data);
});
}

```
import { HubConnection, HubConnectionBuilder } from '@microsoft/signalr';


@Injectable()
export class SignalrserviceService {

  messageReceived = new EventEmitter<string>();
  connectionEstablished = new EventEmitter<Boolean>();
  private _hubConnection!: HubConnection;
  private connectionIsEstablished = false;
  constructor()
  {
    this.createConnection();
    this.startConnection();
    this.registerOnServerEvents();
  }
```

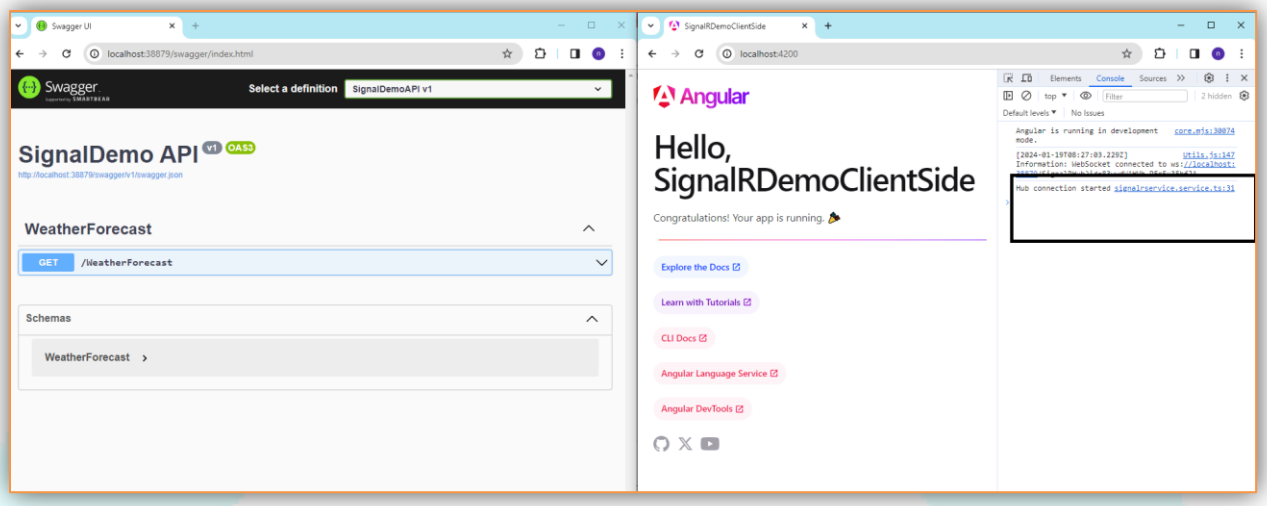
Add in constructor of Signalrservice
Step 4

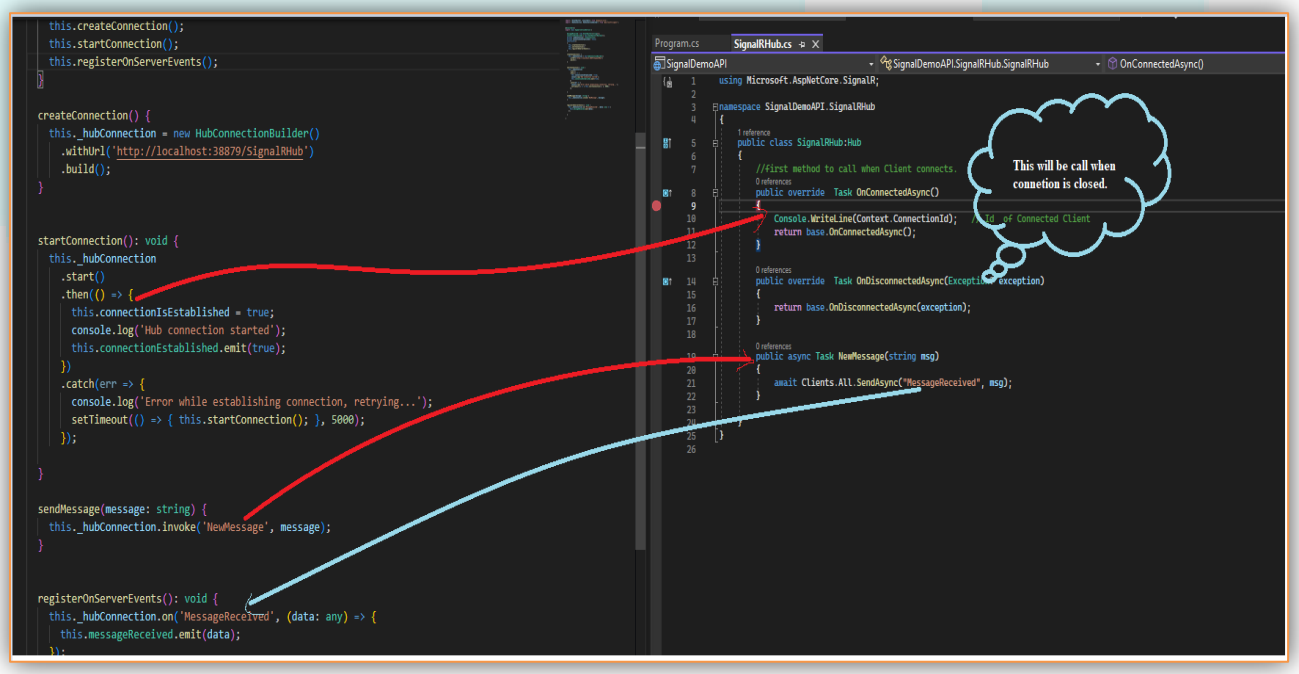
- ✓ Next step is to set up our landing page and call this signalrservice method and connect to our hub.
- ✓ Considering basic level documentation I had created all things in app component only you can also do it by creating new page and use routing.



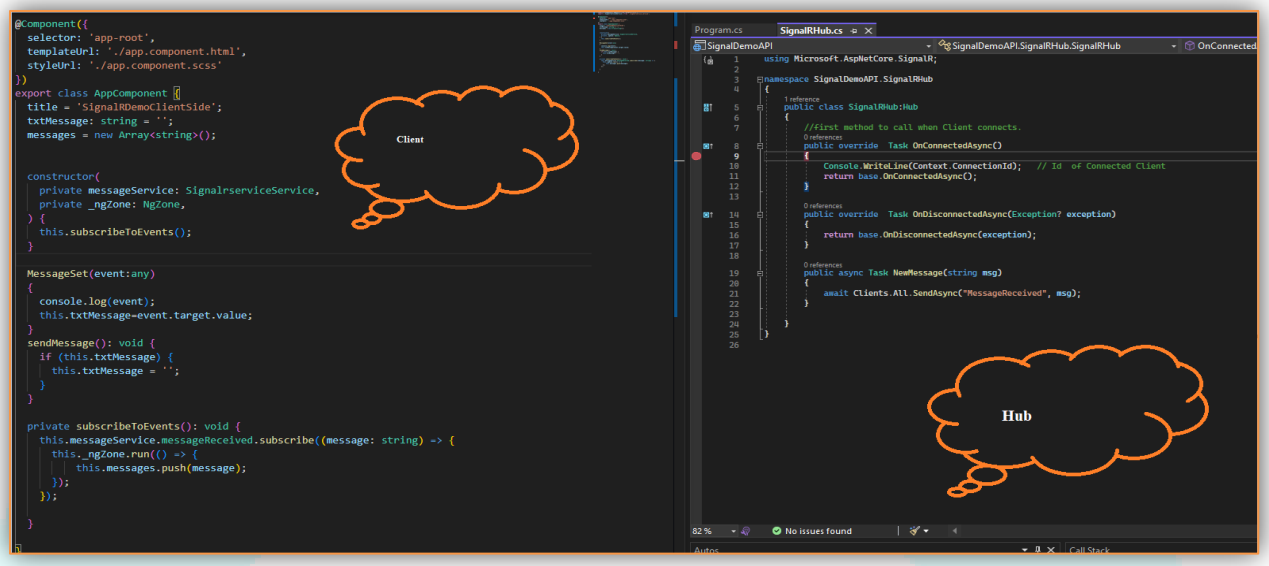
5. Understanding SignalR Architecture:

5.1 Connection Overview





5.2 Hubs and Clients

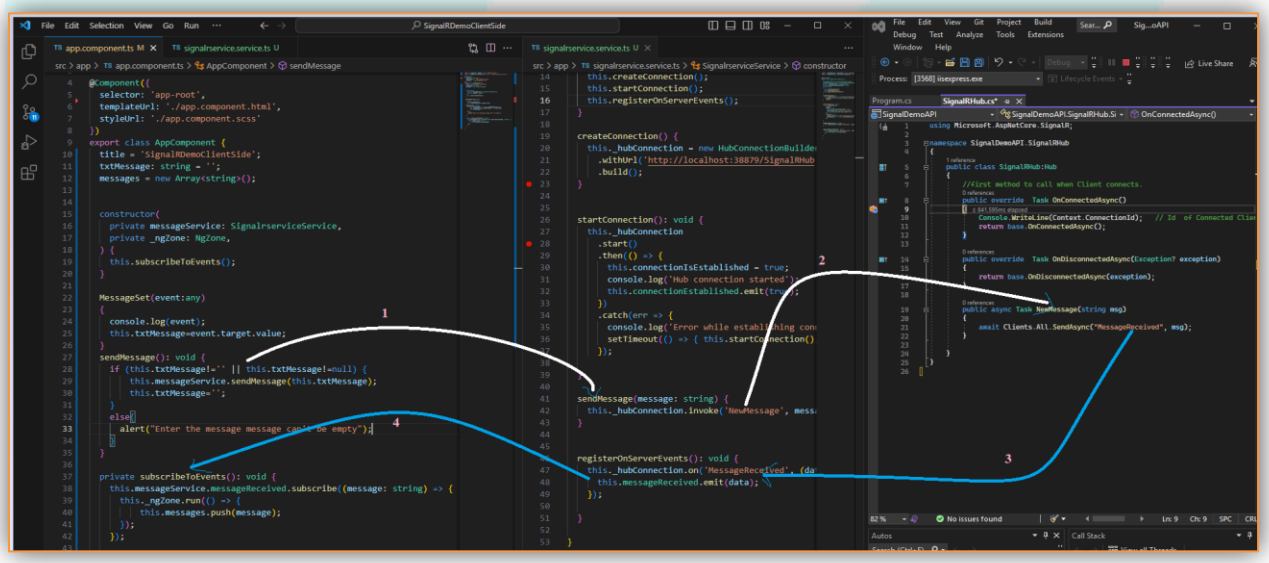


6. Customization:

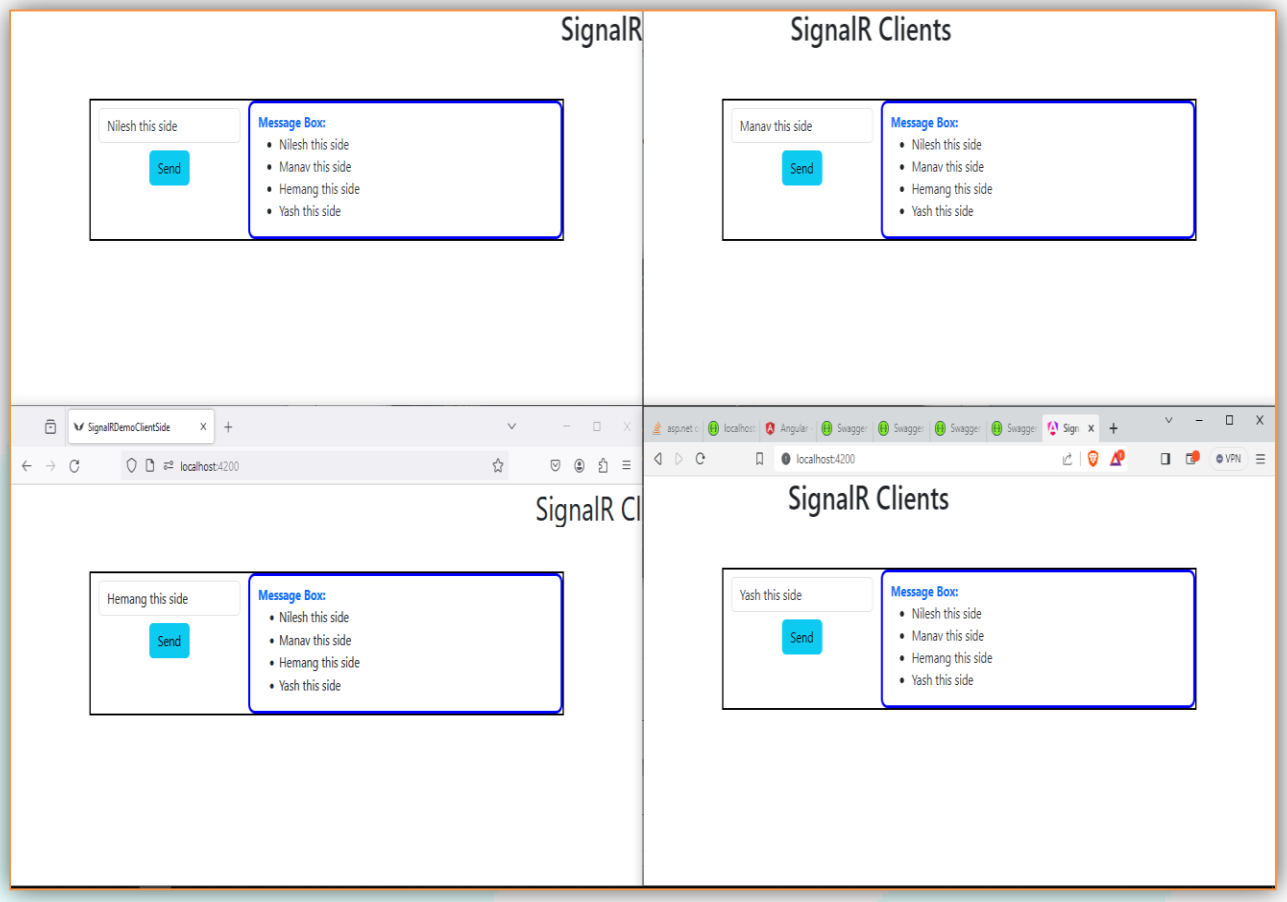
6.1 Creating Custom Hubs

- ✓ Override the hub function and can also create new hub in the same server as shown above and have to add endpoint similar to individual.

6.2 Setting Up SignalR Events



- ✓ Now you are done with basic project.
- ✓ Start your Server(.net core api) and your client (angular) in multiple browser and boom!!!!!



7. Best Practices:

- **Performance Optimization**
  - If possible use primary database like redis.
- **Code Organization and Structure**
- **Handling Connection Interruptions**

8. Resources:

8.1 Official SignalR Documentation

✓ Official Docs of microsoft of SignalR.

8.2 Online Tutorials and Community Forums

✓ Click to lear me through video.

9. Demo Project:

✓ **Click for Demo project**

Note: If any suggestion or any changes in demo please go ahead new suggestion are always welcome..


THANK
YOU