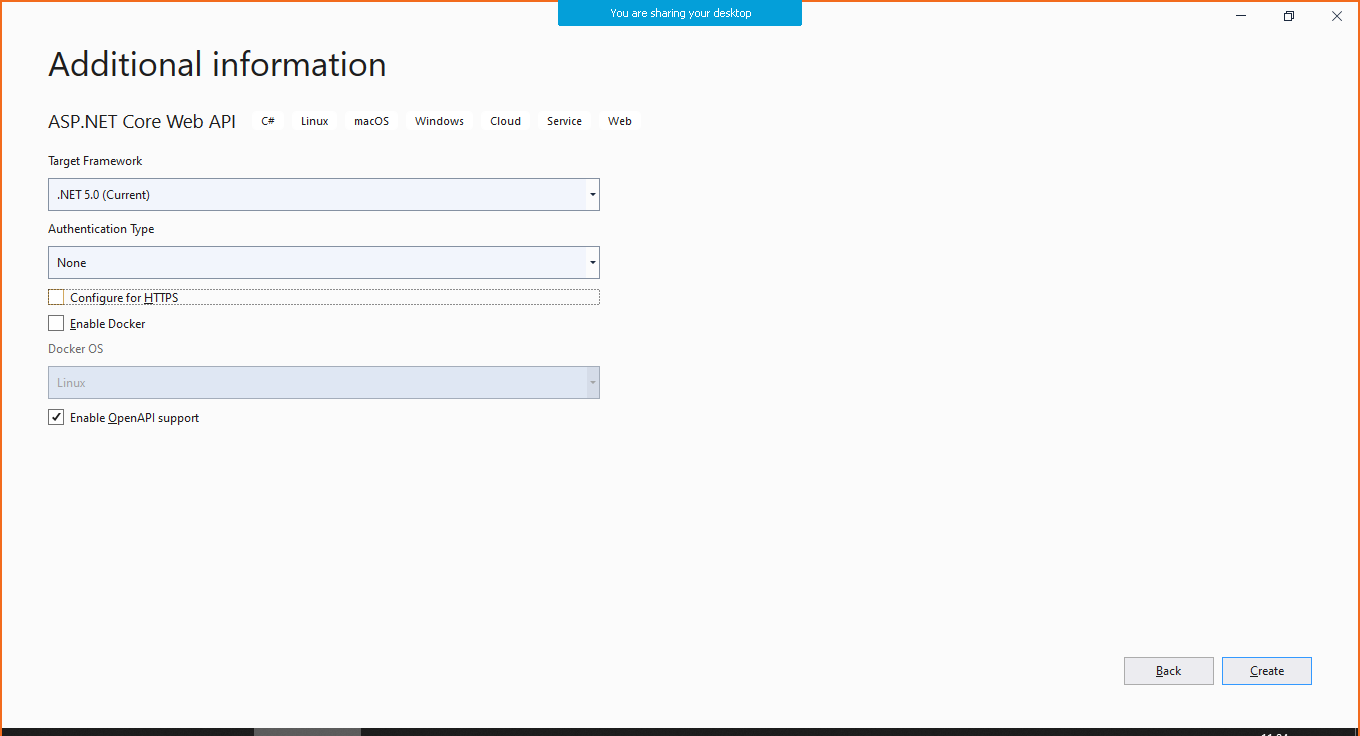
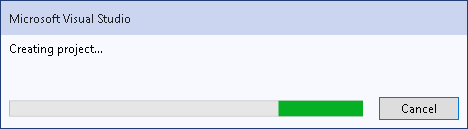
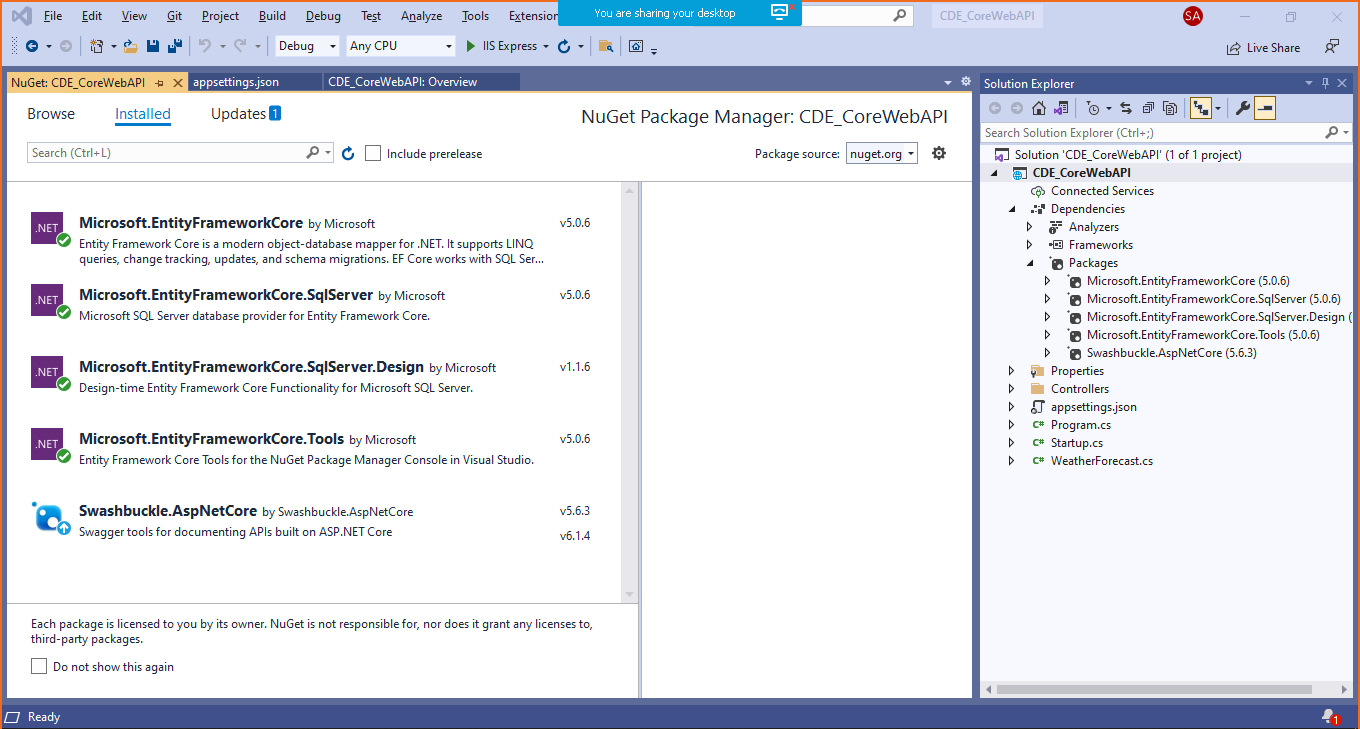
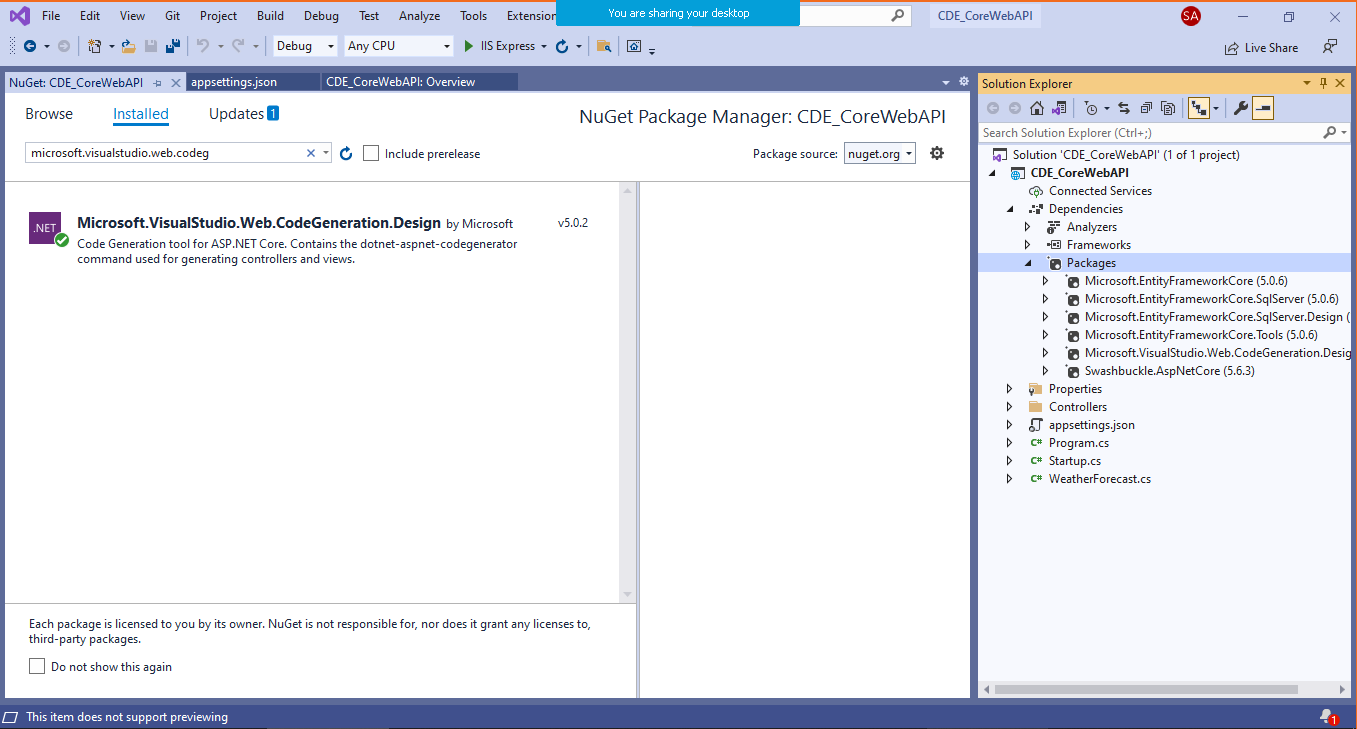
Pilot Project Sample – Angular consuming Web API

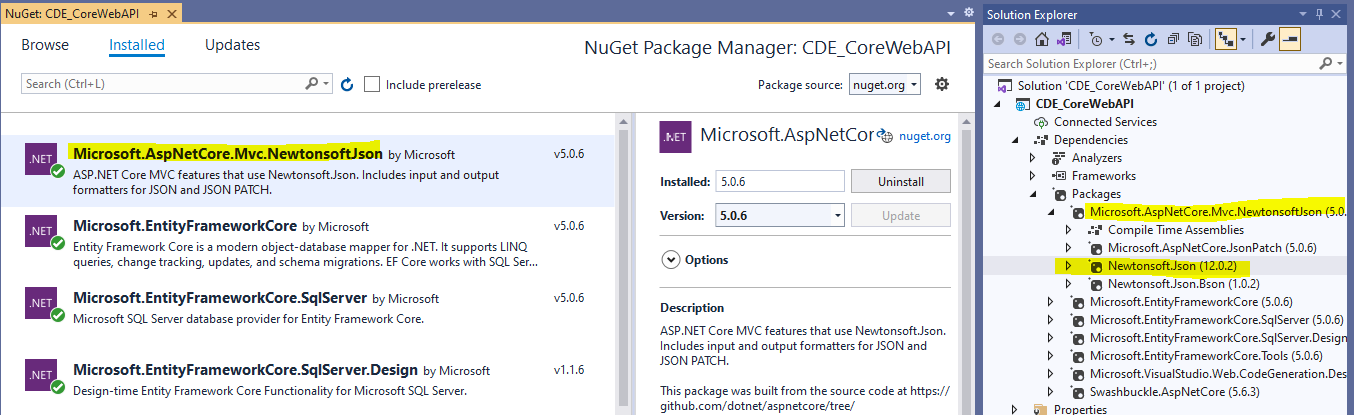


Step 2: Include needed packages for EF usage in Core Web API



Also include the following





Step 3: Add connection string in appsettings.json

"ConnectionStrings": {

"**MyConStr**": "server=ServerName;database=WebAPI\_DB;Trusted\_Connection=True;"

}

Step 4: Create **Employee** Model classes and **AppDBContext** class

using System.ComponentModel.DataAnnotations;

namespace CDE\_CoreWebAPI.Models

{

public **class Employee**

{

[Key]

public int EmpID { get; set; }

public string EmployeeName { get; set; }

public string Department { get;set; }

public string Designation { get; set; }

public int Salary { get; set; }

}

}

//Added...

using **Microsoft.EntityFrameworkCore;**

namespace CDE\_CoreWebAPI.Models

{

public class **AppDBContext:DbContext**

{

public **AppDBContext**(DbContextOptions<AppDBContext> **options**) : base(**options**)

{

//Empty..

}

public **DbSet**<Employee> **Employees** { get; set; }

}

}

Step 5: Add following **config** lines in **startup.cs (If you are using 2019 Visual Studio)**

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.OpenApi.Models;

**//Added...**

**using CDE\_CoreWebAPI.Models;**

**using Microsoft.EntityFrameworkCore;**

**using Newtonsoft.Json.Serialization;**

namespace CDE\_CoreWebAPI

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void **ConfigureServices**(IServiceCollection services)

{

**services.AddDbContext<AppDBContext>(item => item.UseSqlServer(Configuration.GetConnectionString("MyConStr")));**

**//Enable CORS – Cross Origin Resource Sharing**

**services.AddCors(c =>**

**{**

**c.AddPolicy("AllowOrigin", options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());**

**});**

**//JSON Serializer**

**services.AddControllersWithViews()**

**.AddNewtonsoftJson(options =>**

**options.SerializerSettings.ReferenceLoopHandling = Newtonsoft**

**.Json.ReferenceLoopHandling.Ignore)**

**.AddNewtonsoftJson(options => options.SerializerSettings.ContractResolver**

**= new DefaultContractResolver());**

**services.AddControllers();**

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "CDE\_CoreWebAPI", Version = "v1" });

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void **Configure**(IApplicationBuilder app, IWebHostEnvironment env)

{

**app.UseCors(options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());**

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "CDE\_CoreWebAPI v1"));

}

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

**If you are using Visual Studio 2022, Update your program.cs like below**

using MDUWebAPIDemo.Models;

using Microsoft.EntityFrameworkCore;

using Newtonsoft.Json.Serialization;

var builder = WebApplication.CreateBuilder(args);

//Add services to the container.

//Addition1

string conStr = builder.Configuration.GetConnectionString("MyConStr");

builder.Services.AddDbContext<AppDBContext>(con => con.UseSqlServer(conStr));

//Addition2 - Enable CORS – Cross Origin Resource Sharing

builder.Services.AddCors(c =>

{

c.AddPolicy("AllowOrigin", options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

});

////Addition3 - JSON Serializer

builder.Services.AddControllersWithViews()

.AddNewtonsoftJson(options =>

options.SerializerSettings.ReferenceLoopHandling = Newtonsoft

.Json.ReferenceLoopHandling.Ignore)

.AddNewtonsoftJson(options => options.SerializerSettings.ContractResolver

= new DefaultContractResolver());

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

//Addition1

app.UseCors(options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

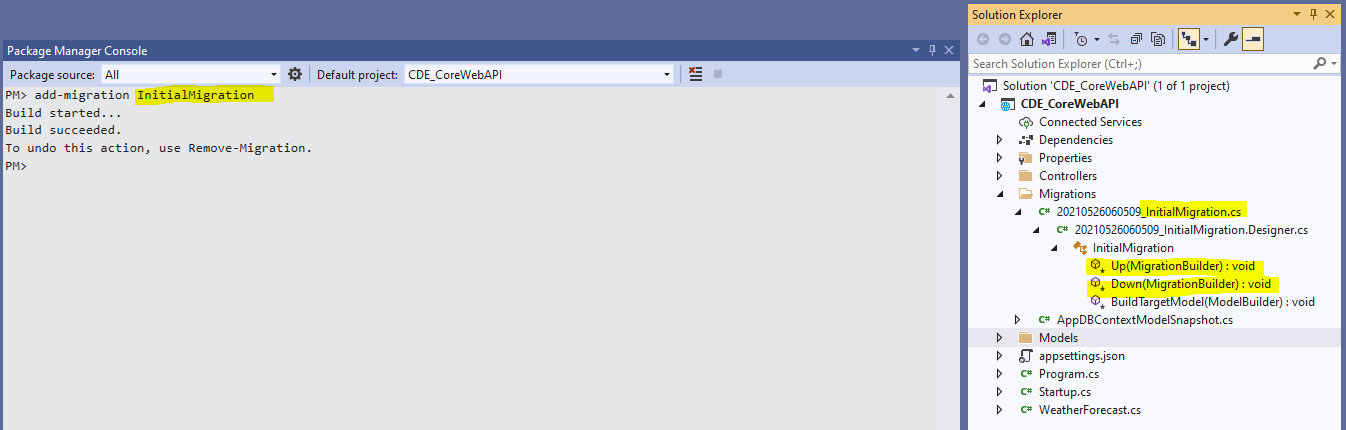
}

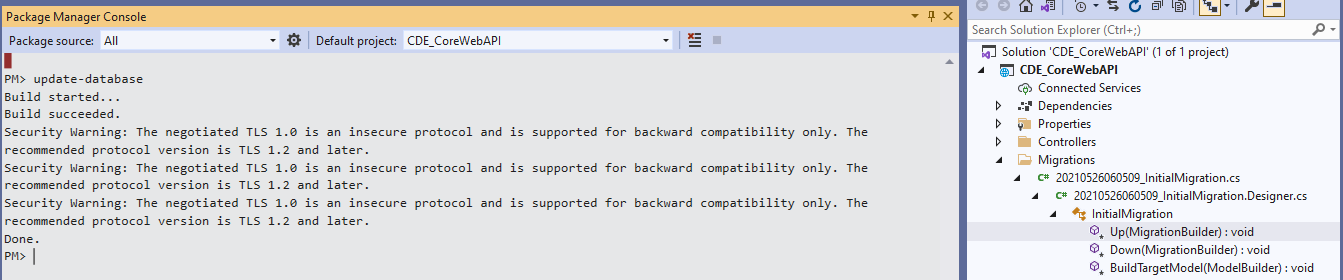
app.UseAuthorization();

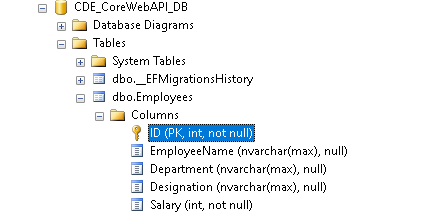
app.MapControllers();

app.Run();

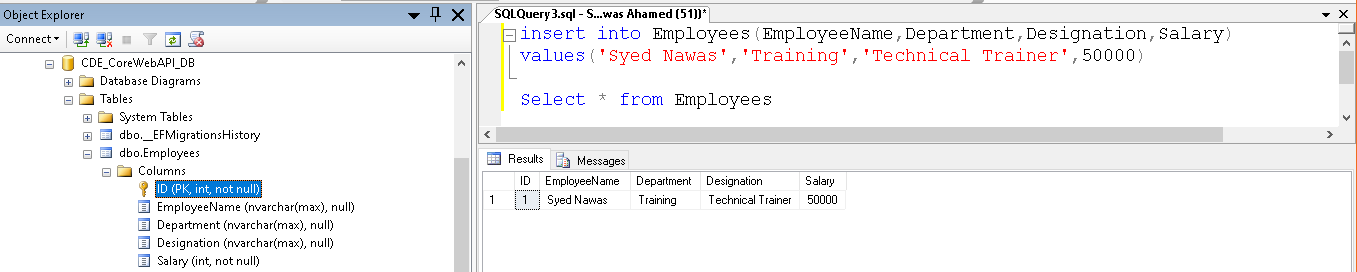
Step 6: Run DB Migration commands in Package Manager Console



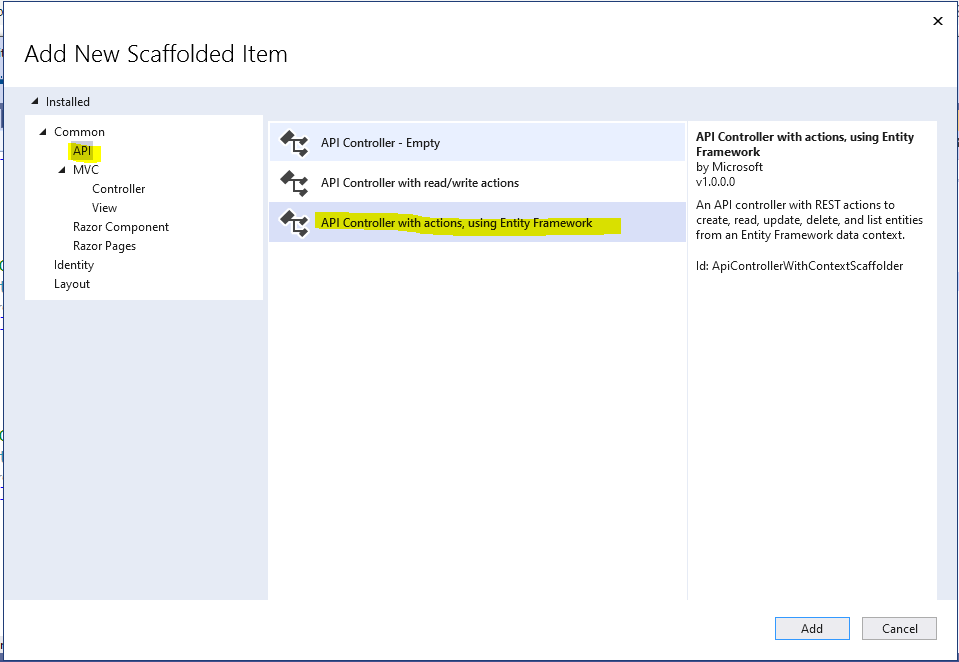


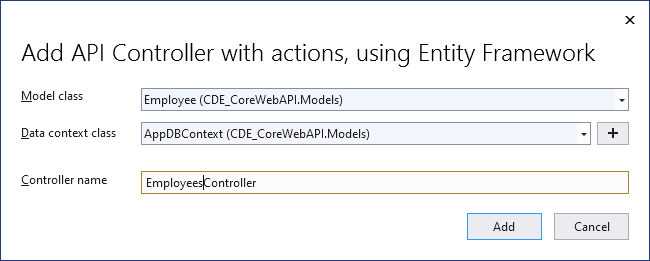
 DB created at the back-end

Populate with some records



Step 7: Add Controller - EmployeesController





**So, controller will be added with all EF auto generated code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using CDE\_CoreWebAPI.Models;

namespace CDE\_CoreWebAPI.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class EmployeesController : ControllerBase

{

private readonly AppDBContext \_context;

public EmployeesController(AppDBContext context)

{

\_context = context;

}

// GET: api/Employees

[HttpGet]

public async Task<ActionResult<IEnumerable<Employee>>> GetEmployees()

{

return await \_context.Employees.ToListAsync();

}

// GET: api/Employees/5

[HttpGet("{id}")]

public async Task<ActionResult<Employee>> GetEmployee(int id)

{

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

return employee;

}

// PUT: api/Employees/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutEmployee(int id, Employee employee)

{

if (id != employee.ID)

{

return BadRequest();

}

\_context.Entry(employee).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Employees

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Employee>> PostEmployee(Employee employee)

{

\_context.Employees.Add(employee);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetEmployee", new { id = employee.ID }, employee);

}

// DELETE: api/Employees/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteEmployee(int id)

{

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

\_context.Employees.Remove(employee);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool EmployeeExists(int id)

{

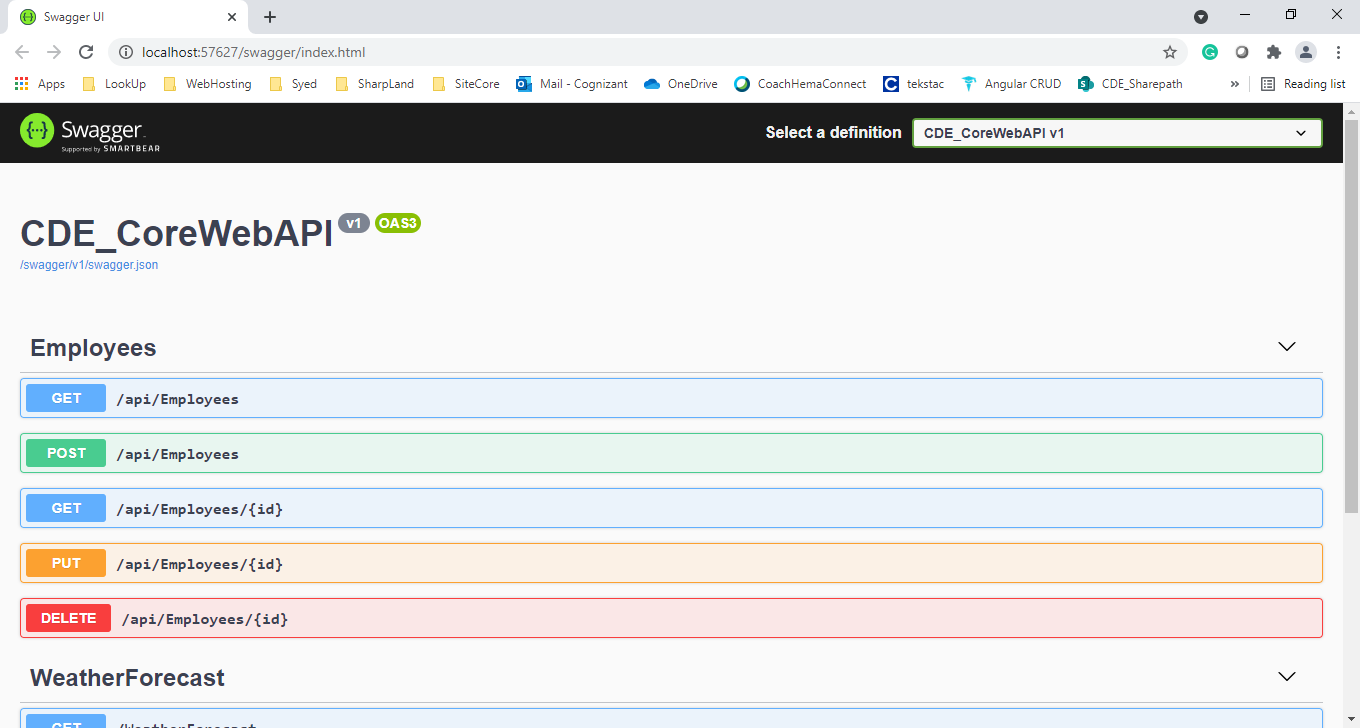
return \_context.Employees.Any(e => e.ID == id);

}

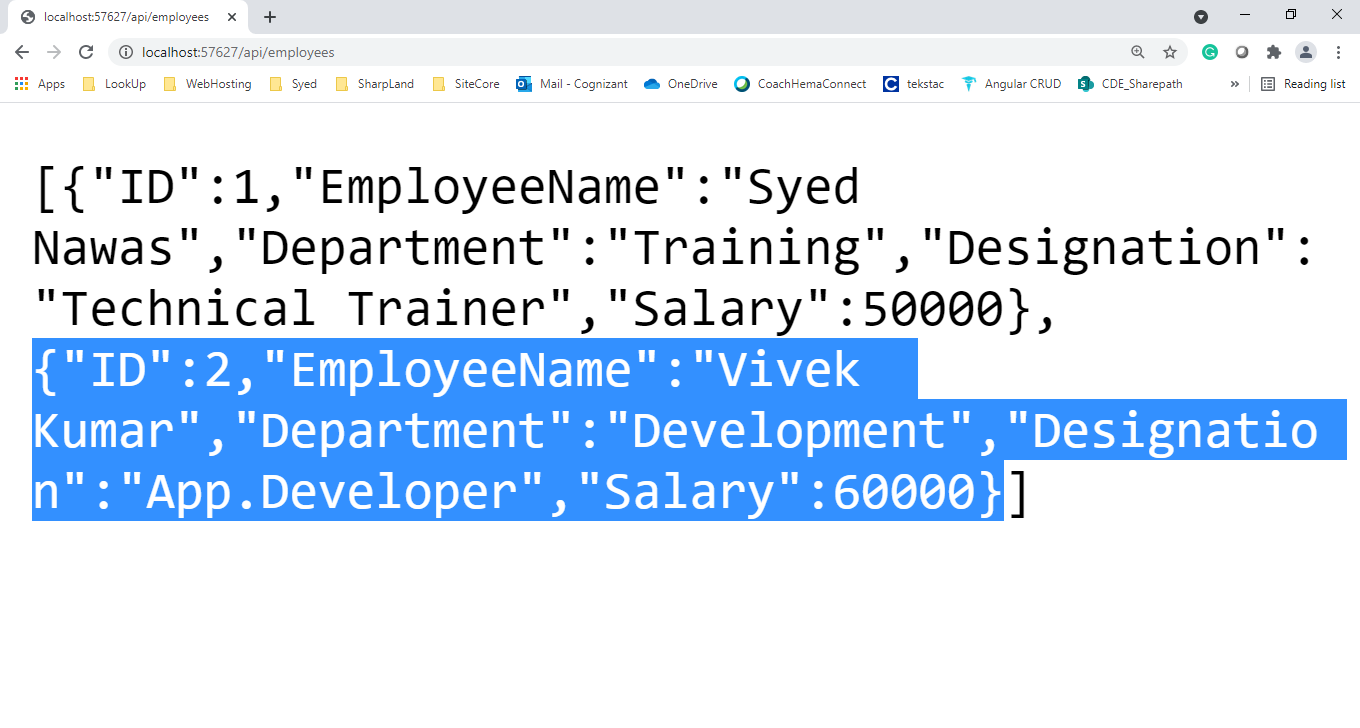
}

}

Step 8 – Run your web api project and test all GET with browser POST, PUT, and DELETE action methods with postman

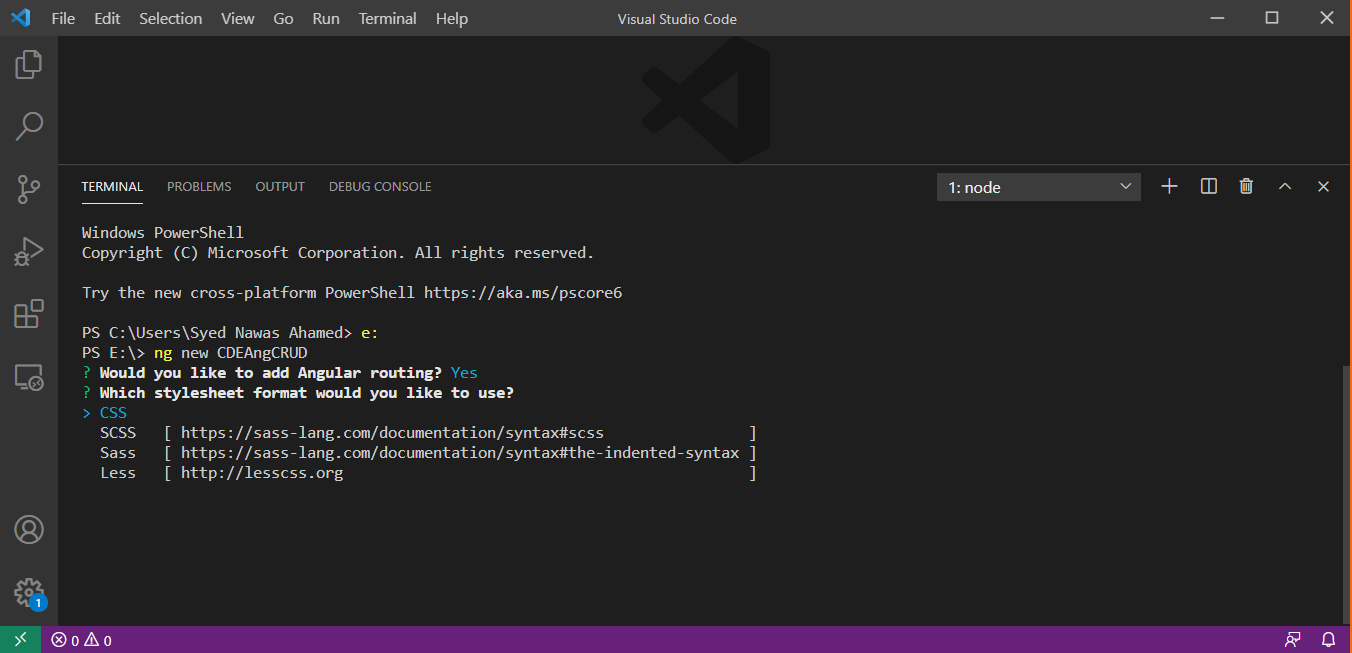


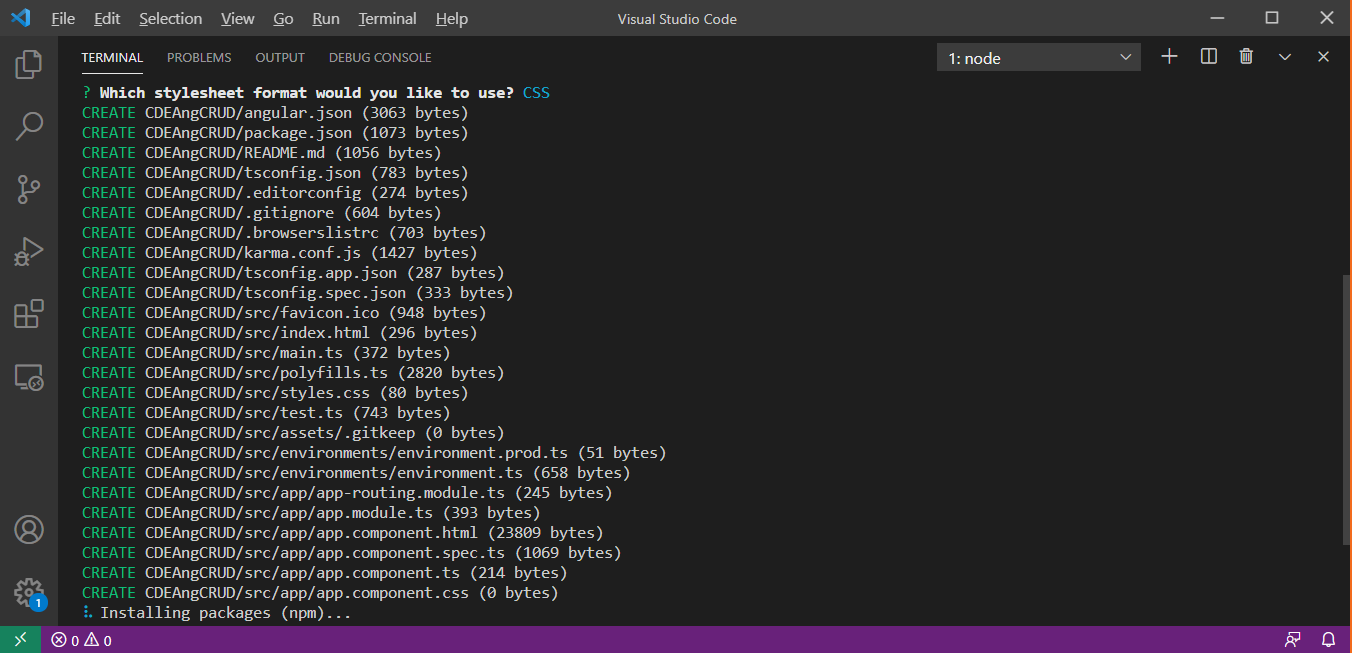
Change the url to [**http://localhost:57627/api/employees**](http://localhost:57627/api/employees) to test the get Method, others can be tested with **Postman**



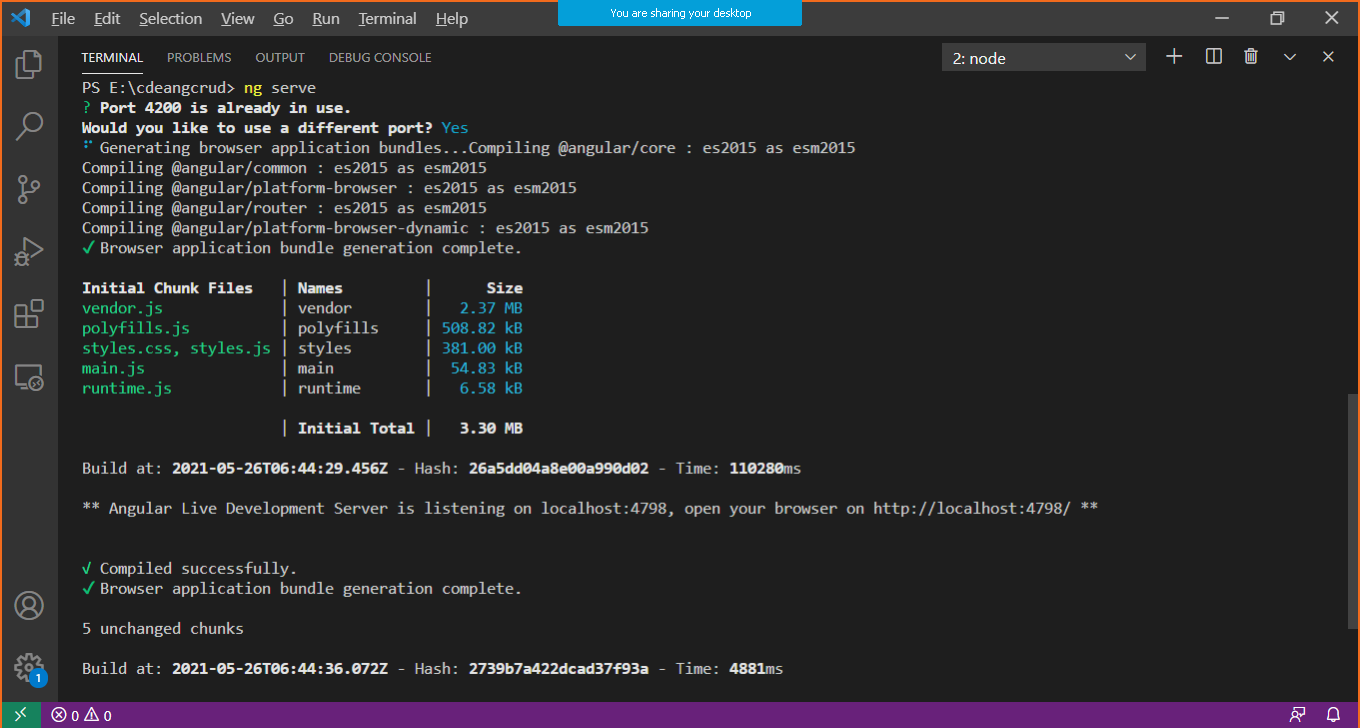
Angular Application Creation

Step1: Create ng App

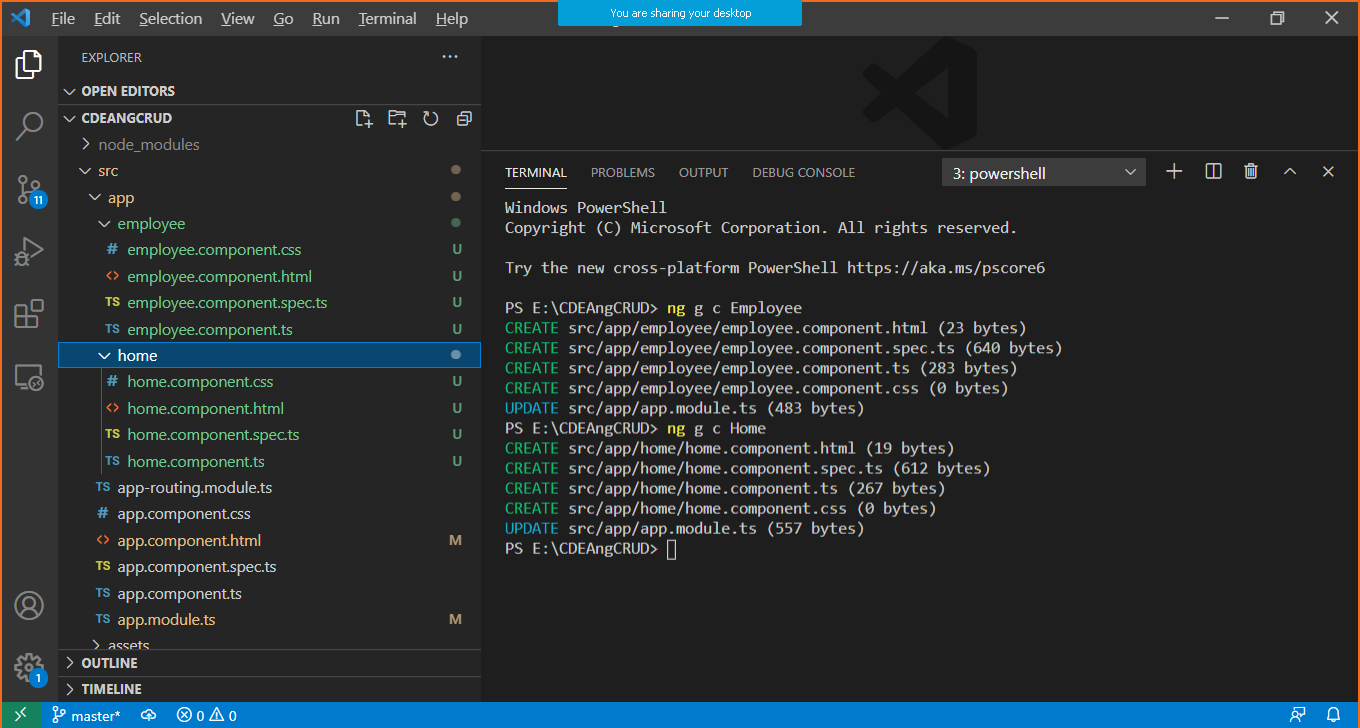




Test the application with >ng serve command



Step 2: Create a components called **Home** and **Employee** but don’t add any code just leave the components.



Include **bootstrap** in index.html, So **Bootstrap styles** will be applied to all components we are adding in application

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>CDEAngCRUD</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"

  integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">

</head>

<body>

  <app-root></app-root>

  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"

  integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"

  crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"

  integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"

  crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"

  integrity="sha384-w1Q4orYjBQndcko6MimVbzY0tgp4pWB4lZ7lr30WKz0vr/aWKhXdBNmNb5D92v7s"

  crossorigin="anonymous"></script>

</body>

</html>

Step 3: Create routing in **app-routing.module.ts**

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

/\*\*

 \* Added

 \*/

import {EmployeeComponent} from './employee/employee.component';

import {HomeComponent} from './home/home.component';

/\*\*

 \* Routes array should be initialized by JSON Path objects.

 \* Those objects will contain 2 fields 1. path string, 2. component name

 \*/

const routes: Routes = [

  {

    path:"home",

    component:HomeComponent

  },

  {

    path:"employees",

    component:EmployeeComponent

  }

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

Step 4: Create Navigation menu **app.component.html**

<h2 class="d-flex justify-content-center">Integration of .NET Core Web API with Angular</h2>

<hr/>

<nav class="navbar navbar-expand-sm bg-light navbar-dark">

  <ul class="navbar-nav">

      <li class="nav-item">

        <button routerLink="home" class="m-1 btn btn-light btn-outline-primary" Button>Home</button>

      </li>

      <li class="nav-item">

        <button routerLink="employees" class="m-1 btn btn-light btn-outline-primary" Button>Employees</button>

      </li>

    </ul>

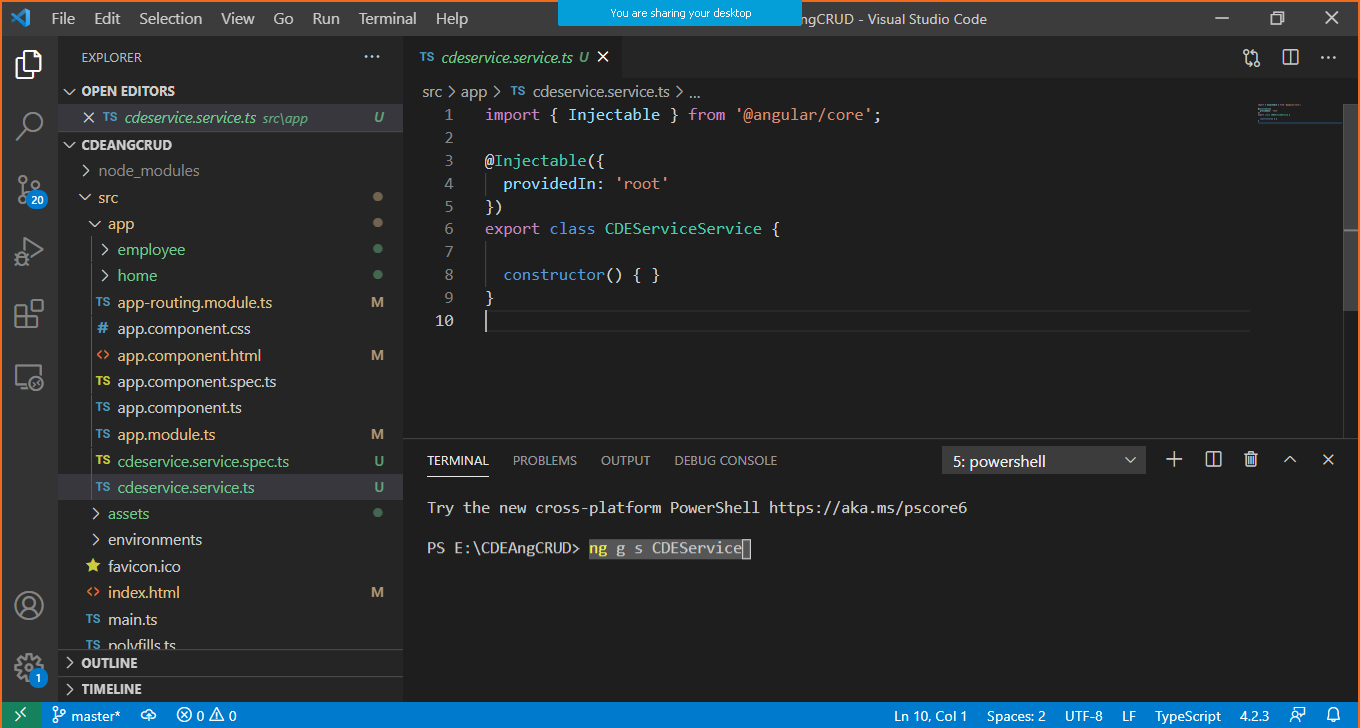
  </nav>

<router-outlet></router-outlet>

**Please confirm**

**Then path string you used in app-routing.module.ts should match with the value of routerLink attribute in bootstrap button**

Step 5: Generate **angular service** to fetch .NET Core Web API Data (JSON Objects)



Modify the above auto-geneated service as follows,

import { Injectable } from '@angular/core';

/\*\*

 \*You have to add the following because

Reason 1 - This service is going to act as a client of our core web api.

Reason 2 - This service is going to observe the data coming from core web api.

\*/

import {HttpClient} from '@angular/common/http';

import {Observable} from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class CDEServiceService {

  readonly APIUrl="http://localhost:57627/api/employees";

  constructor(private http:HttpClient) { }

  ServiceMethodGetEmployeeList():Observable<any[]>{

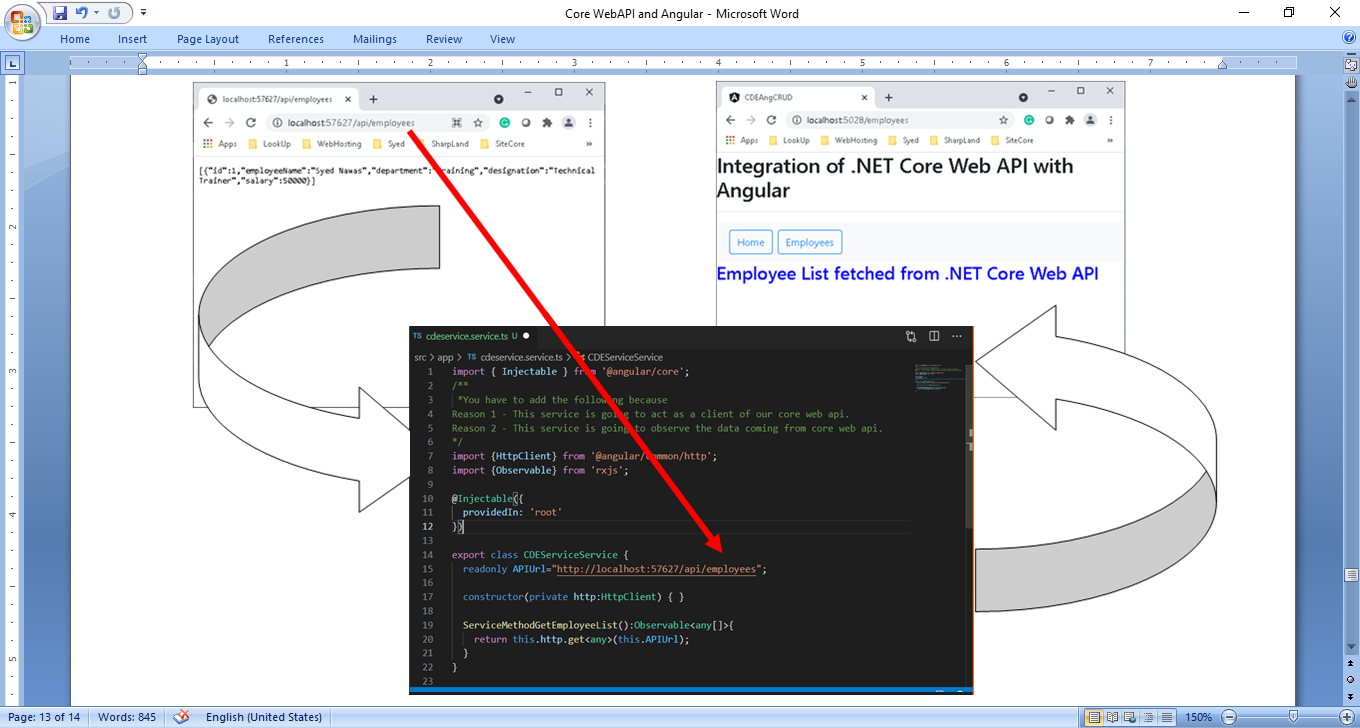
    return this.http.get<any>(this.APIUrl);

  }

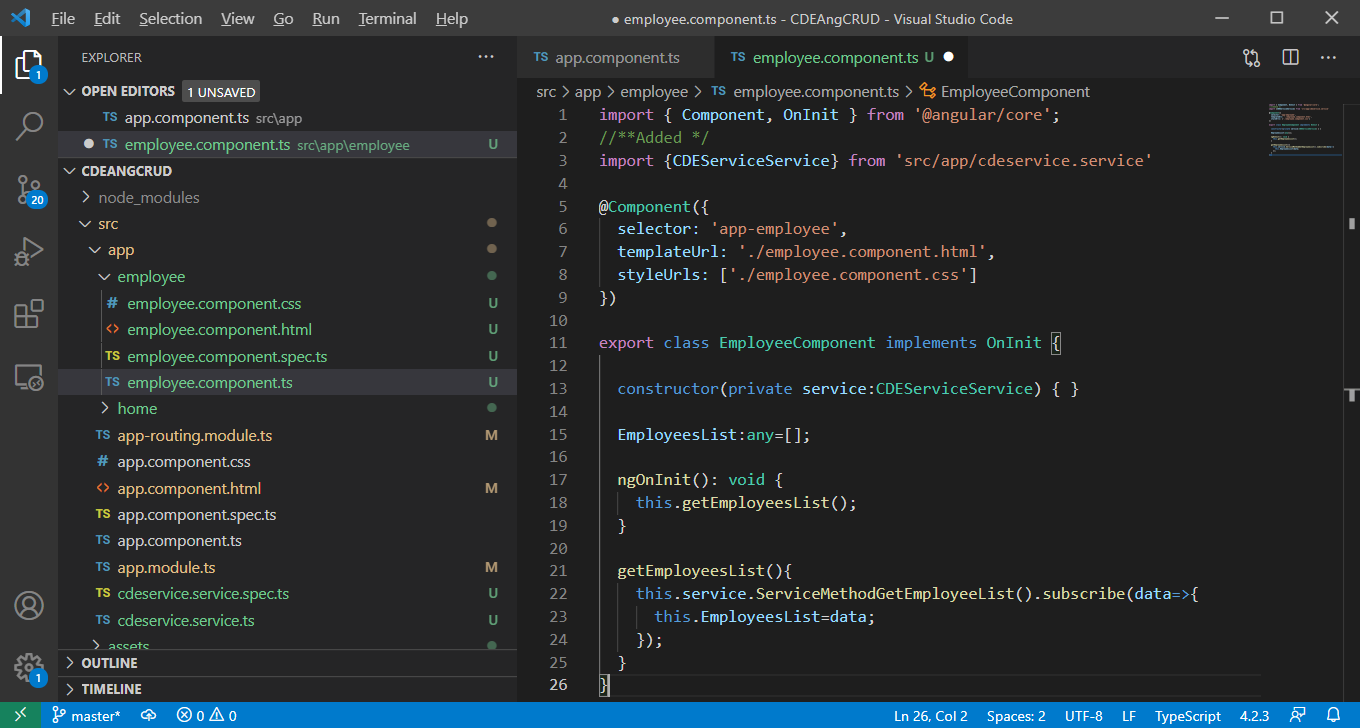
}

Step 6: Consume the above **Service (CDEServiceService)** and

call the **Service Method (ServiceMethodGetEmployeesList)** in Employee Component



Alter / add the following code in **employee.component.ts**



Add this Following in **employee.component.html**

<h3>Employee List fetched from .NET Core Web API</h3>

<table class="table table-striped">

    <thead>

        <tr>

            <th>ID</th>

            <th>Employee Name</th>

            <th>Department</th>

            <th>Designation</th>

            <th>Salary</th>

        </tr>

    </thead>

    <tbody>

        <tr \*ngFor="let dataItem of EmployeesList">

            <td>{{dataItem.ID}}</td>

            <td>{{dataItem.EmployeeName}}</td>

            <td>{{dataItem.Department}}</td>

            <td>{{dataItem.Designation}}</td>

            <td>{{dataItem.Salary}}</td>

        </tr>

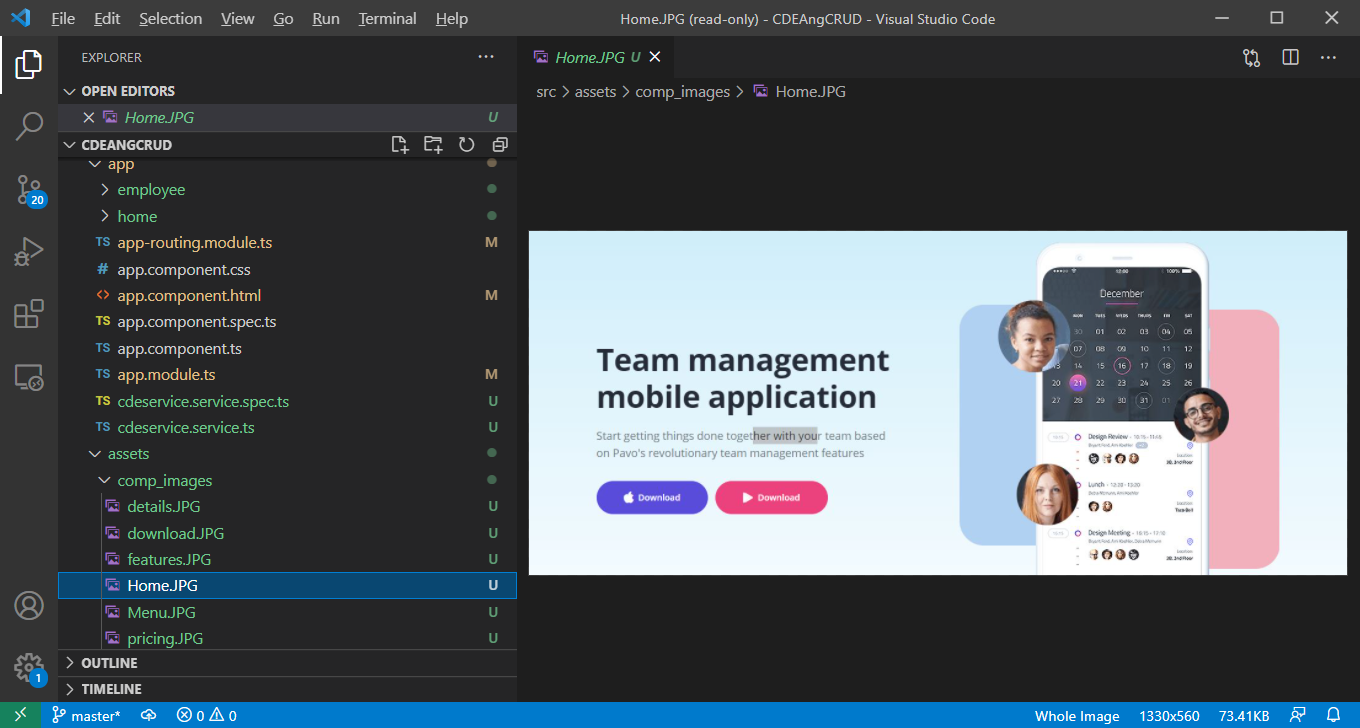
    </tbody>

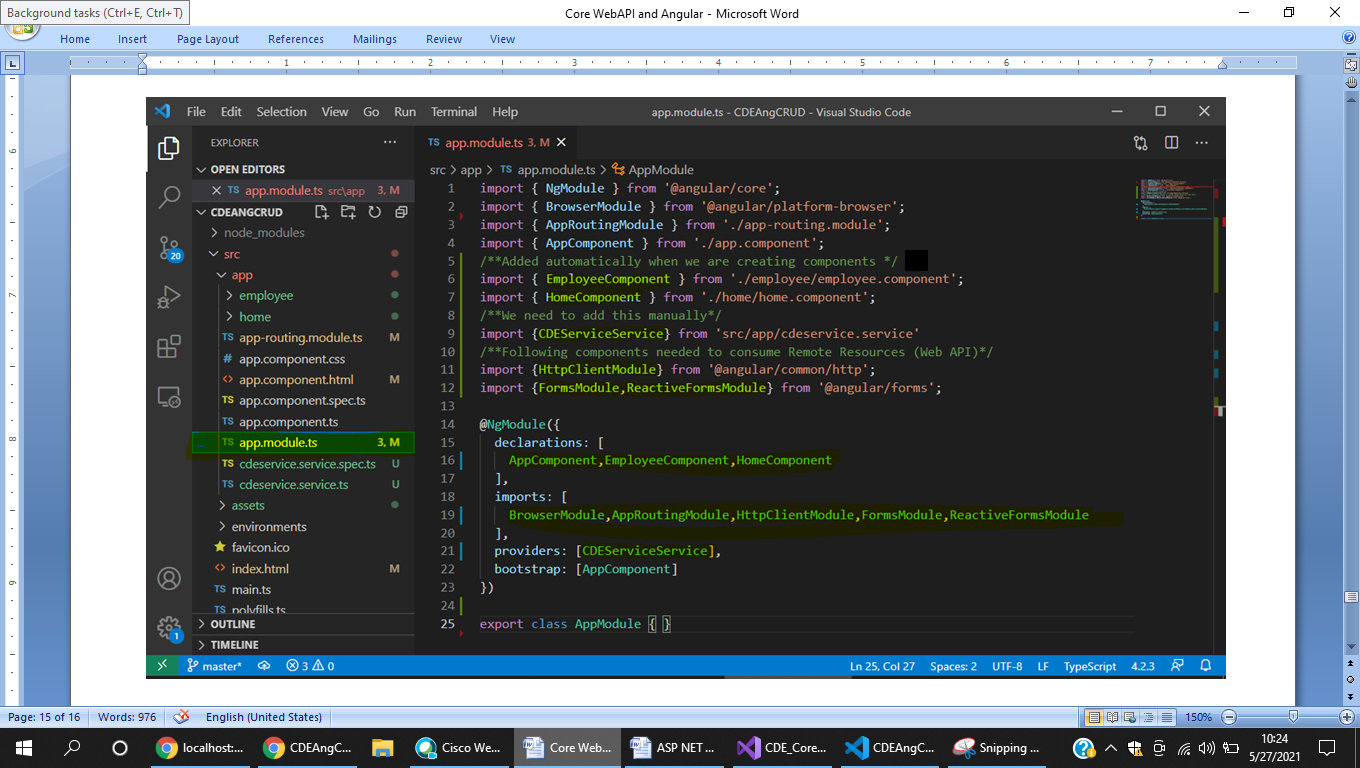
</table>

Step 7: Add this single line of code in **home.component.html**

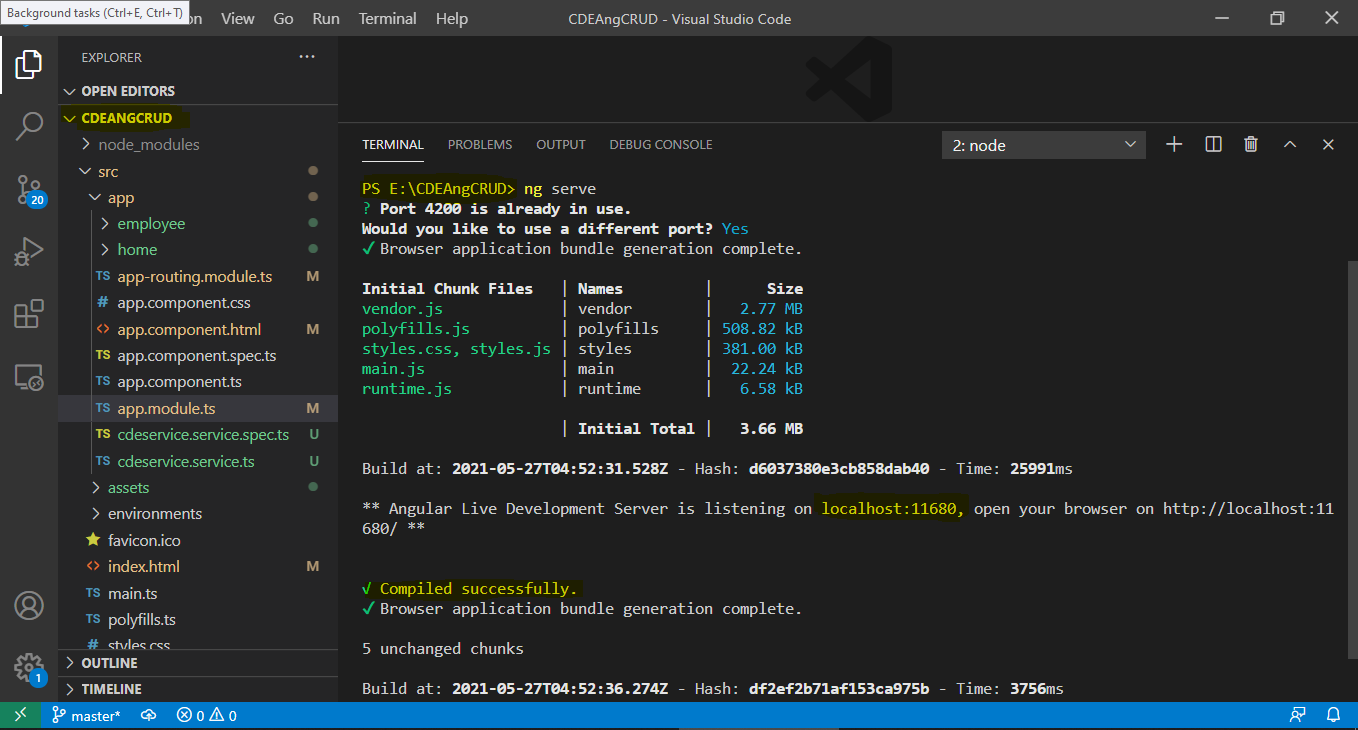
<img src='.\assets\comp\_images\Home.JPG' alt='Home Content'/>

This component is using an image called **“Home.JPG”** so make that available in **assests\comp\_images** folder

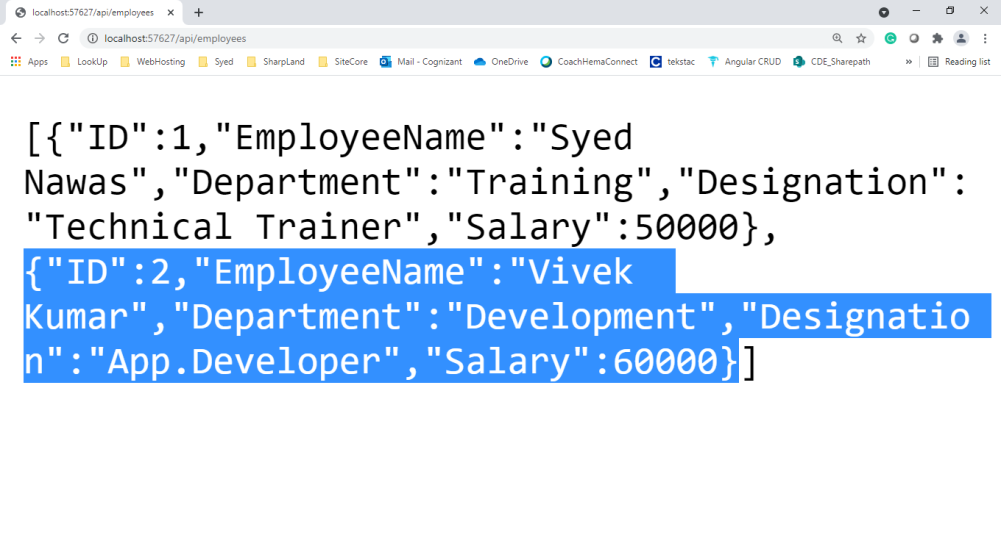


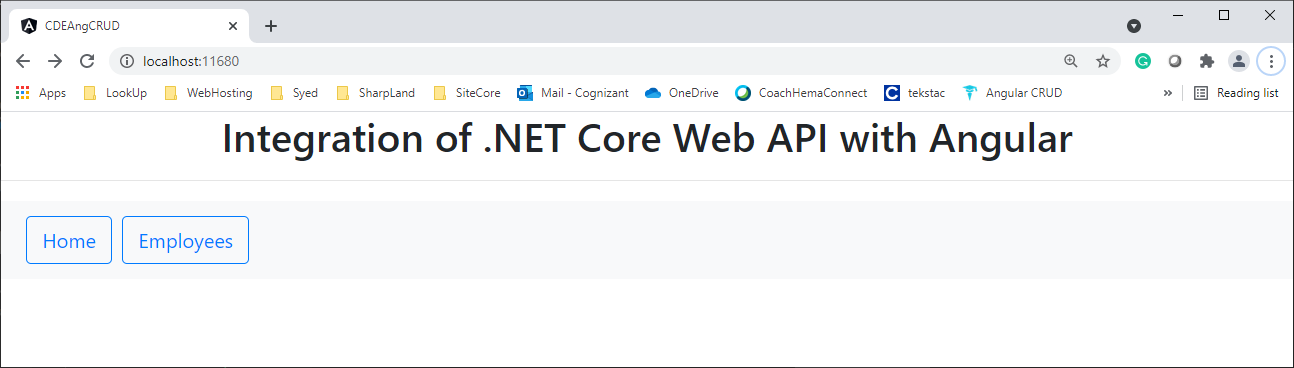
****Step 8: Do this important additions in **app.module.ts**

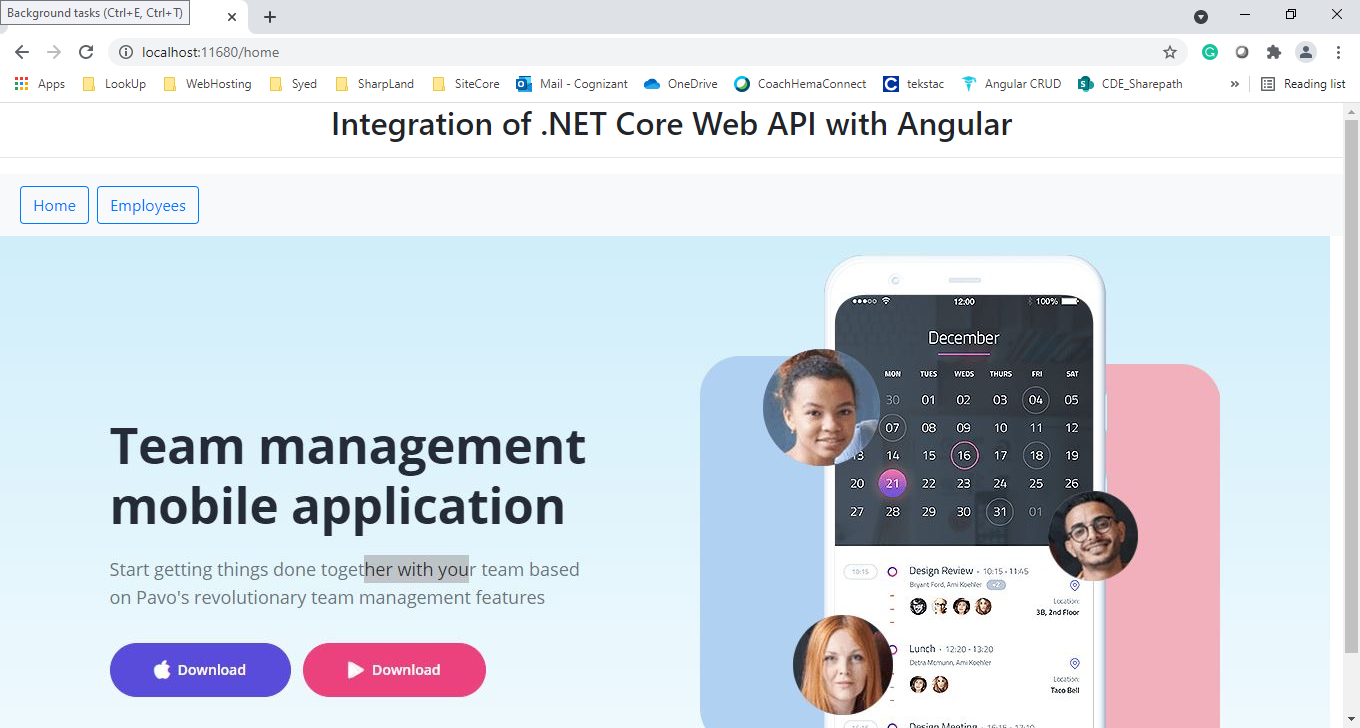
**Step 9**: Run your **Angular App** with **ng Serve and** notice the port number it is running.

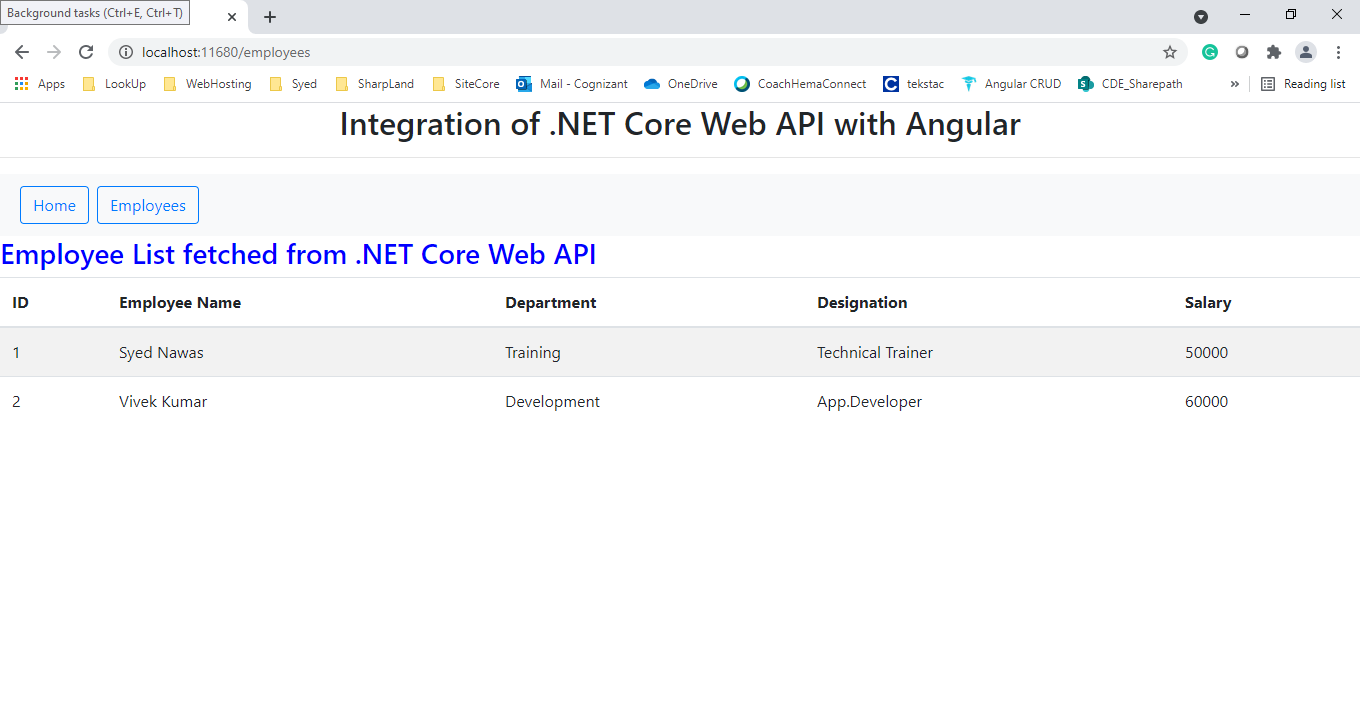


Confirm that **ASP.NET Core WebAPI** Project is already running.

****

****

****

****

**Angular - SPA**

Home

About

Courses

Contact

Results

Emp.Recs

Core Web API

GET - **R**ead

POST - **C**reate

Delete - **D**elete

PUT - **U**pdate

**Other PUT, DELETE, POST will be included later.**

**All the best**

employee.component.ts

EmployeeList:any=[]

Home

Ser.Meth.GetEmployeeList()

**http://localhost:57627/api/Employees**

WebAPI

EF

**http://localhost:57627/api/Employees**

employee.component.html

Create the Employee Data

Update the Employee Data

Delete the Employee Data