Requirements:

Postman  
Visual Studio 2019 Or Higher  
.Net 5.0.13 Or Higher  
VSC  
Node.js  
Sql Server  
Sql Server Management Studio  
Install These Library In Visual Studio Installer:  
ASP.Net And Web Development  
.Net Desktop Development  
.Net Core Cross-Platform Development

Backend:

1-Create ASP .Net Core Web API Project

2-Install Microsoft.AspNetCore.Mvc.NewtonsoftJson With Nuget

3-Add These Codes to These Functions In Startup.Cs:

Function ConfigureServices:

//Enable CORS

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 Newtonsoft.Json.Serialization.DefaultContractResolver());

Function Configure:

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

4-Create Models Folder In The Root Of Project And Entities Folder In This Folder To Add Your Entities To It

5-Add Your Entities With Their Properties As a C# Class To This Folder(Entities)

6-~~Create a Database~~

7-Write This Piece of Code in the “appsettings.json” File (we use this code For Establishing A Connection between database and backend)  
 "ConnectionStrings": {

"EmployeeAppCon": "Data Source=DESKTOP-5LCE2RF; Initial Catalog=EmployeeDB; Integrated Security=True; TrustServerCertificate=True;"

},

P.S: Personalize This Code According To The Information Of Your Server

8-Create a folder in Models folder called Contexts

9-Create a C# class in this folder called DatabaseContext

10-Add these packages to the project:  
Microsoft.EntityFrameworkCore  
Microsoft.EntityFrameworkCore.Tools  
Microsoft.EntityFrameworkCore.SqlServer

11-place this piece of code in databasecontext.cs:

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using OU\_API.Models.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace OU\_API.Models.Contexts

{

public class DatabaseContext : DbContext

{

private readonly IConfiguration \_configuration;

public DatabaseContext(IConfiguration configuration)

{

this.\_configuration = configuration;

}

public DbSet<User> Users { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

string SqlDataSource = \_configuration.GetConnectionString("OUAppCon");

optionsBuilder.UseSqlServer(SqlDataSource);

}

}

}

P.S: Personalize This Code According To The Information Of Your Project

12-Open Package Manager powershell and type Add-Migration Init and then type Update-Database  
(these commands create a database with its tables with respect to our C# classes)

13-Add folder called Services to the Models folder

14-You can add a folder for each service that you want to perform and then add an interface and two classes(Dto and Service) to this folder for your services

15-If you follow example codes correctly you can create your API and finally you should just remember your API’s name and domain address and use these API’s in frontend.

FrontEnd:

1-create react app(npx create-react-app “project name”)

2-run these commands in terminal :

npm install react-bootstrap bootstrap

npm install react-router-dom

3-place this code in index.html head:

<link

rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"

integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"

crossorigin="anonymous"

/>

4-Create some component for testing API

5-Create file “.env” and write your API in it(see the examples)

6-Add .env to gitignore file

7- npm install dotenv -save(run this command)

8-you can access to your API address with this code “process.env.REACT\_APP\_API”

9-see the example codes and copy them for testing API and then make your change on this files to create a webpage