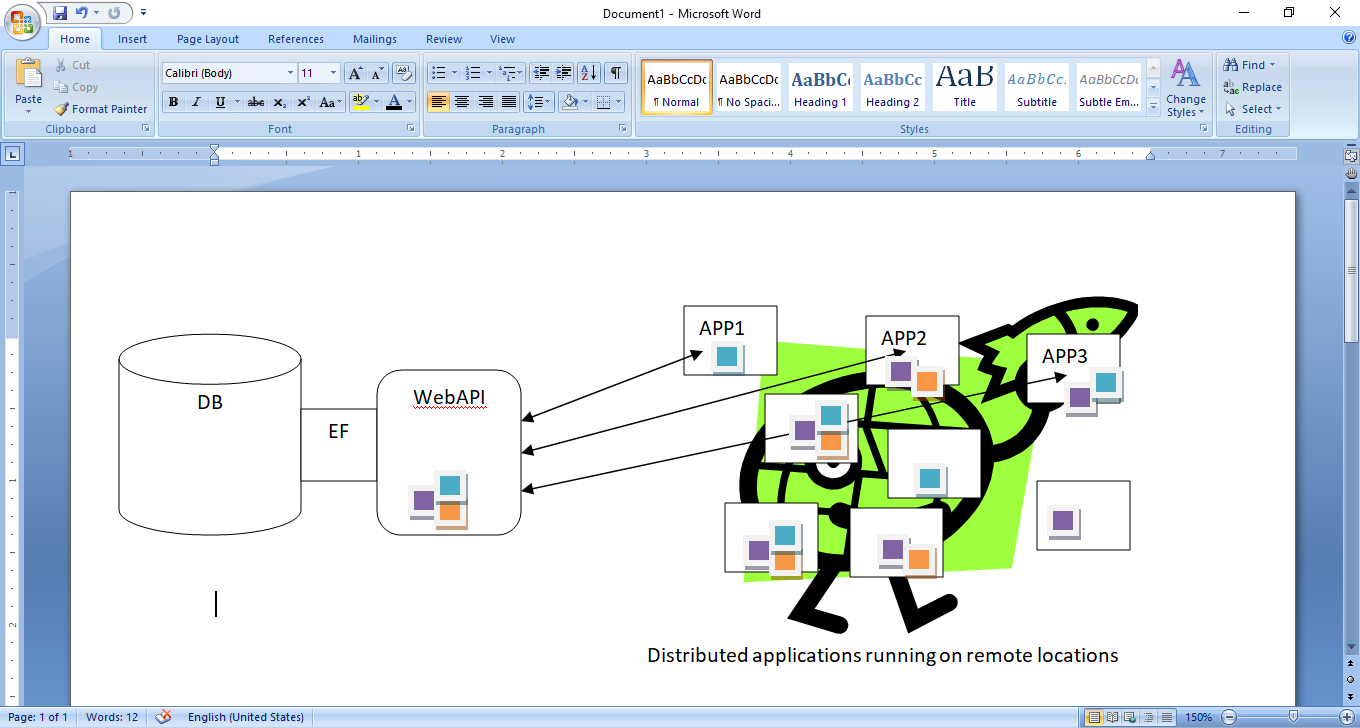
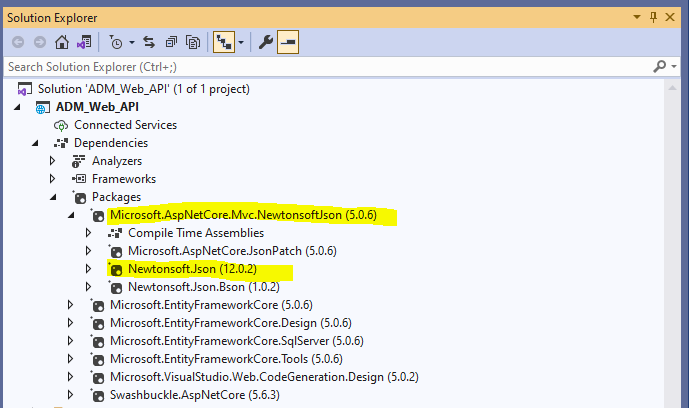
Creating .Net Core Web API



**Step1:** Create **ASP.NET Core Web API** Project 🡪 Give Project Name 🡪 Uncheck **“Configure for HTTPS”**

**Step2:** Install needed packages for EF integration in ASP.NET Core Web API Project by **NuGet Package Manager**



**Step 3**: Add Connection String in **appsettings.json** file

{

"Logging": {

"LogLevel": { "Default": "Information", "Microsoft": "Warning", "Microsoft.Hosting.Lifetime": "Information" }

},

"AllowedHosts": "\*",

**"ConnectionStrings":** {

**"MyConStr": "server=SYED83\\SQLEXPRESS86;database=ADM\_CoreWebAPI\_DB;Trusted\_Connection=True;"**

}

}

Step4: Add the folder called **Models** and create **Account, ApplicationDBContext** Classes in **Models** Folder

**Account Model class**

namespace ADM\_Web\_API.Models

{

public class Account

{

public int AccountID { get; set; }

public string AccountName { get; set; }

}

}

**AppDBContext class**

//Added...

using Microsoft.EntityFrameworkCore;

namespace ADM\_Web\_API.Models

{

public class **AppDBContext**:DbContext

{

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

{

//Empty constructor….

}

public DbSet<Account> **Accounts** { get; set; }

}

}

**Step 5**: add some configurations in startup.cs file as follows in both ConfigureServices(), Configure() methods

//Added...

using ADM\_Web\_API.Models;

using Microsoft.EntityFrameworkCore;

using Newtonsoft.Json.Serialization;

namespace ADM\_Web\_API

{

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)

{

//Enable CORS – Cross Origin Resourcing– HTTP Header based mechanism

**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());**

**//DBContext by consuming connection string**

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

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "ADM\_Web\_API", 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", "ADM\_Web\_API v1"));

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

**Step 6:** Run following commands in Package Manager Console (Visual Studio Tools Menu 🡪 NuGet Package Manger)

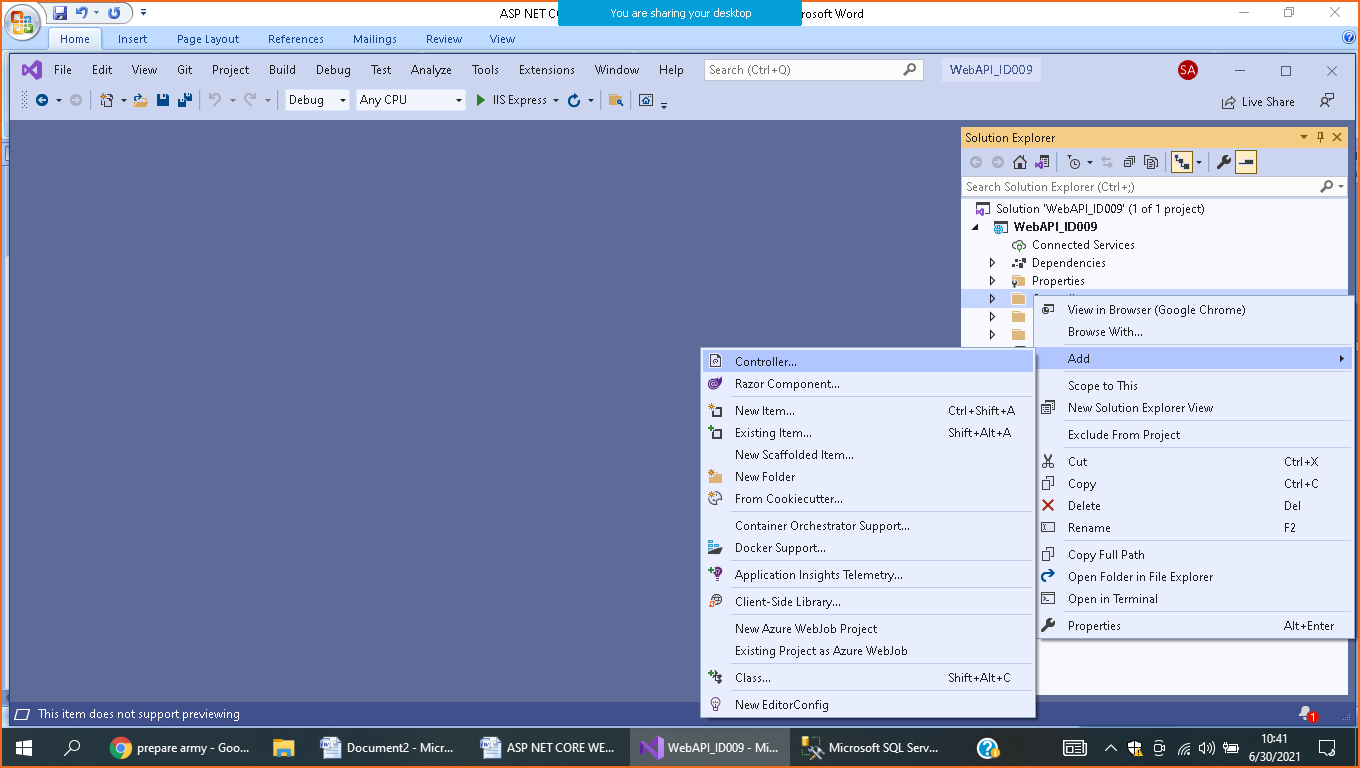
* add-migration InitialMigration

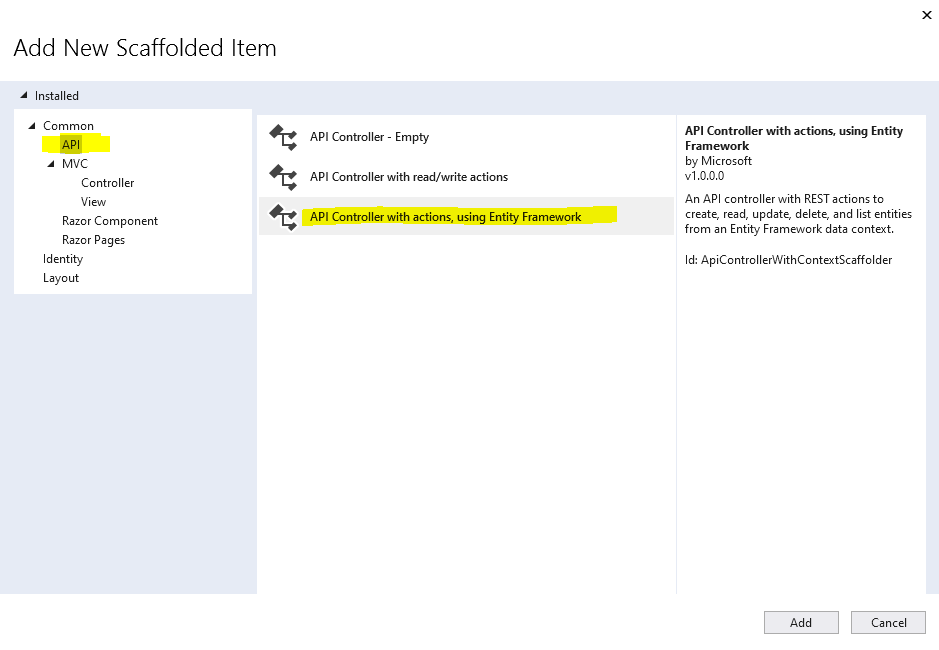
This command will add migration class with up(), down() methods and adds Migrations folder

* update-database

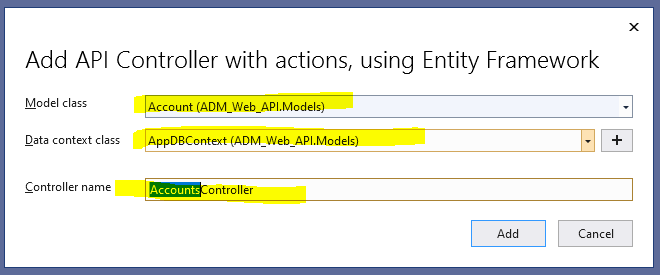
This command will create database and tables based on the migration class generated in previous step

**Step 7**: Add API Controller by **Add Controller option** of Context of Menu from Solution Explorer





Specify the Model Class, Data Context Class 🡪Click Add Button



**Controller will be added as follows with auto – generated code,**

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ADM\_Web\_API.Models;

namespace ADM\_Web\_API.Controllers

{

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

[ApiController]

public class AccountsController : ControllerBase

{

private readonly AppDBContext \_context;

public AccountsController(AppDBContext context)

{

\_context = context;

}

**// GET: api/Accounts**

[HttpGet]

public async Task<ActionResult<IEnumerable<Account>>> GetAccounts()

{

return await \_context.Accounts.ToListAsync();

}

**// GET: api/Accounts/5**

[HttpGet("{id}")]

public async Task<ActionResult<Account>> GetAccount(int id)

{

var account = await \_context.Accounts.FindAsync(id);

if (account == null)

{

return NotFound();

}

return account;

}

**// PUT: api/Accounts/5**

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

[HttpPut("{id}")]

public async Task<IActionResult> PutAccount(int id, Account account)

{

if (id != account.AccountID)

{

return BadRequest();

}

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

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!AccountExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

**// POST: api/Accounts**

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

[HttpPost]

public async Task<ActionResult<Account>> PostAccount(Account account)

{

\_context.Accounts.Add(account);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetAccount", new { id = account.AccountID }, account);

}

**// DELETE: api/Accounts/5**

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteAccount(int id)

{

var account = await \_context.Accounts.FindAsync(id);

if (account == null)

{

return NotFound();

}

\_context.Accounts.Remove(account);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool AccountExists(int id)

{

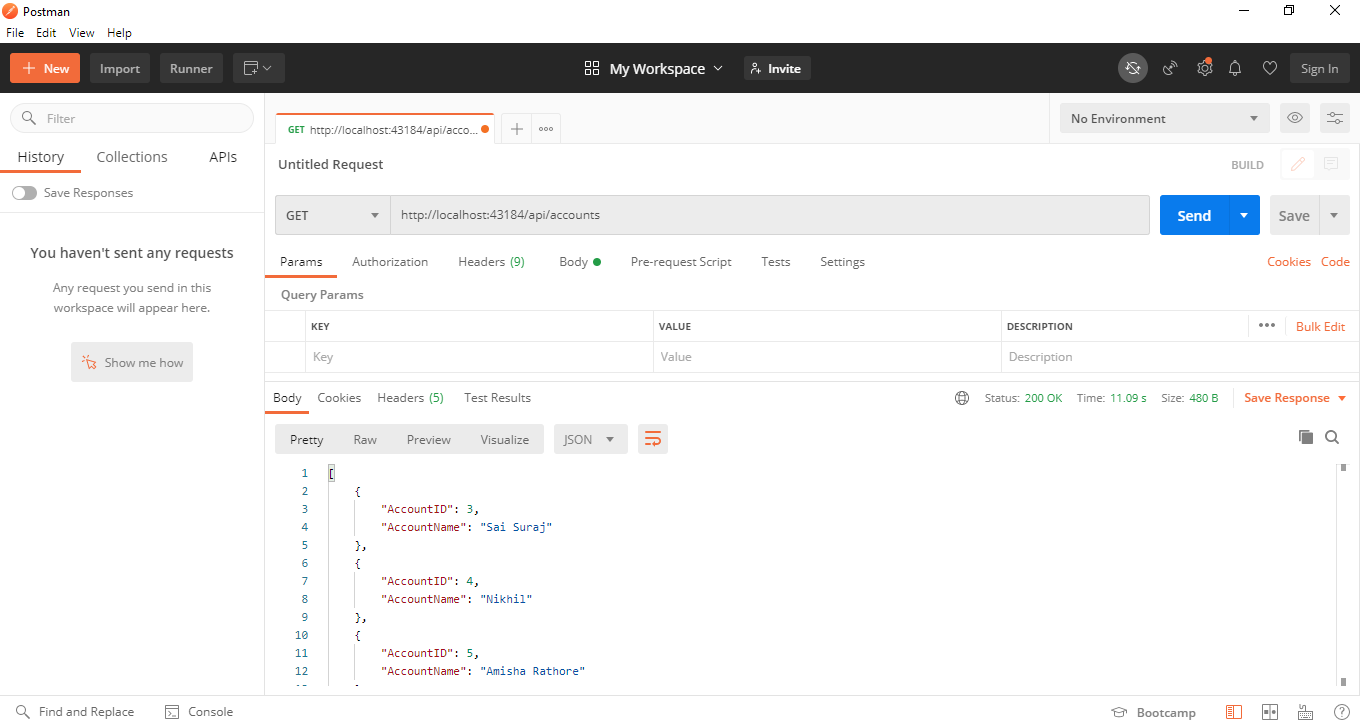
return \_context.Accounts.Any(e => e.AccountID == id);

}

}

}

**Step 8**: Run your Web API Project and test GET, POST, PUT, and DELETE actions using **Postman** tool.

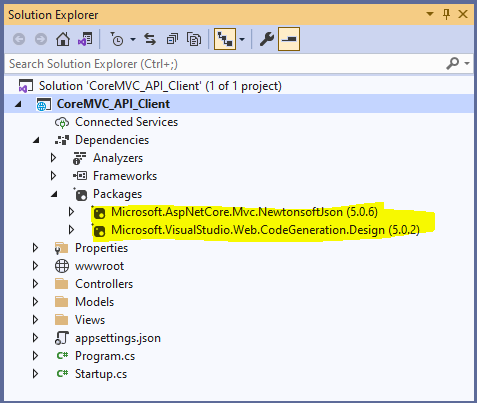


**Note down** the URL [**http://localhost:43184/api**/accounts](http://localhost:43184/api/accounts). It is required to consume the Web API from outside.

Consuming ASP.NET CORE Web API from ASP.NET Core MVC Application

**Step 1:** Create ASP.NET Core Web Application (Model-View-Controller) Category. Name it like **CoreMVC\_API\_Client**

**Step 2:** Include needed packages in to project



**Step 3:** Add "**WebAPIBaseUrl**" with URL copied from previous **WebAPI** project

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"**WebAPIBaseUrl**": "**http://localhost:43184/api**"

}

**Step 4**: Add Model **Account** in Models folder

namespace CoreMVC\_API\_Client.Models

{

public class Account

{

public int AccountID { get; set; }

public string AccountName { get; set; }

}

}

**Step 5**: Add new **MVC Controller** to access the WebAPI controller’s action methods **or** use default **Home Controller**.

using System.Collections.Generic;

using System.Threading.Tasks;

using CoreMVC\_API\_Client.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System.Diagnostics;

**//Added...**

using Microsoft.Extensions.Configuration;

using System.Net.Http;

**//Added by NuGet Package Manager**

using Newtonsoft.Json;

namespace CoreMVC\_API\_Client.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

**//Add these 2 Lines of code**

**private readonly IConfiguration \_config;**

**private readonly string apiBaseUrl;**

**//2nd Param in following constructor, we need to add it manually**

public HomeController(ILogger<HomeController> logger, **IConfiguration configuration**)

{

\_logger = logger;

**\_config = configuration;**

**apiBaseUrl** = \_config.GetValue<string>("WebAPIBaseUrl");

}

**//Change method header of "public IActionResult Index()" as below**

public **async Task<IActionResult>** Index()

{

IEnumerable<**Account**> **accounts** = new List<Account>();

using (var httpClient = new HttpClient())

{

using (var response = **await** httpClient.GetAsync(**apiBaseUrl** + "/accounts"))

{

string apiResponse = await response.Content.ReadAsStringAsync();

accounts = JsonConvert.DeserializeObject<List<**Account**>>(apiResponse);

}

}

return View(**accounts**);

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

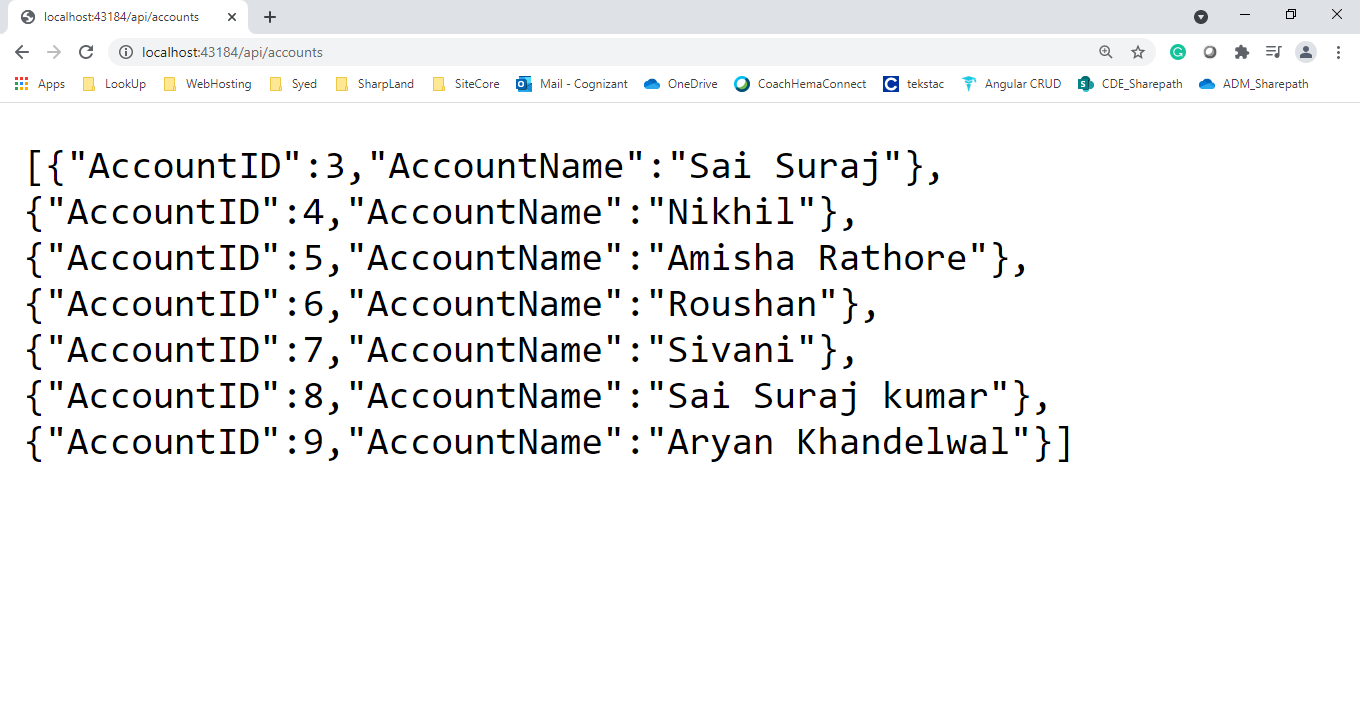
return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

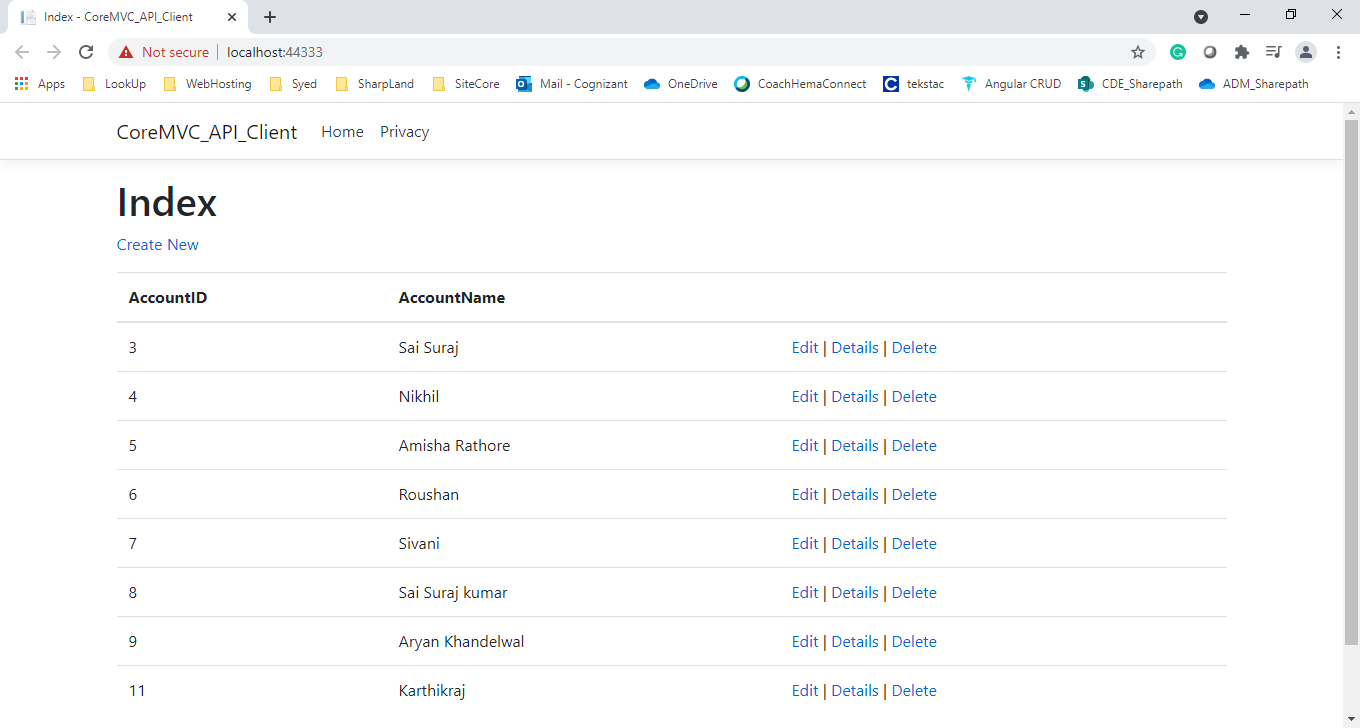
}

}

}

**Step 6**: Run your **ASP.NET MVC App** and Confirm that **ASP.NET Core WebAPI** Project is already running.

****

****

**All the best**