Session Storage

Program.cs:

// CONFIGURE THE SESSION STORAGE SERVICES

/ session storage:

// services.AddMvc().AddSessionStateTempDataProvider();

// Builder.Services.AddDistributedMemoryCache();

Builder.Services. .AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromSeconds(10);

options.Cookie.HttpOnly = true;

options.Cookie.IsEssential = true;

});

services.AddDistributedMemoryCache();

services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromSeconds(10);

options.Cookie.HttpOnly = true;

options.Cookie.IsEssential = true;

});

// Use in application

app.UseSession();

Controller.cs:

[HttpGet]

[Route("GetSessionData")]

public IActionResult GetSessionData()

{

// get session value

var data = HttpContext.Session.GetString("Student");

if(string.IsNullOrEmpty(data))

{

// return BadRequest("Expired session");

var studentlist = \_studentService.GetAllStudent();

return Ok(studentlist);

}

return Ok(data);

}

[HttpGet]

[Route("SetSession")]

public IActionResult SetSessionn()

{

var studentData = \_studentService.GetAllStudent();

// set session value

HttpContext.Session.SetString("Student", JsonConvert.SerializeObject(studentData));

return Ok(studentData);

}

Imemory Cache

Configure AddMemoryCache()

Program.cs:

// add service add memory cache

Builder.Service.AddMemoryCache();

services.AddMemoryCache();

Controller.cs: add ImemoryCache interface inside a constructor class of controller

private readonly IMemoryCache \_memoryCache;

public StudentController(

IMemoryCache memoryCache )

{

\_memoryCache = memoryCache;

}

// set Memory Cache value:

[HttpGet]

[Route("SetSession")]

public IActionResult SetSessionn()

{

// individual:

var studentData = \_studentService.GetAllStudent();

// set memory cache

\_memoryCache.Set("Student", JsonConvert.SerializeObject(studentData), TimeSpan.FromSeconds(80));

// get

var data = \_memoryCache.Get("Student");

if (data != null)

{

return Ok(data);

}

else

{

return Ok(studentData);

}

}

[HttpGet]

[Route("GetSession")]

public IActionResult GetSession()

{

// get memory cache valuea

var getStudents = \_memoryCache.Get("Student");

if (getStudents == null)

{

return BadRequest("No data found");

}

return Ok(getStudents);

}

[Get object from AWS S3 as a File stream](https://stackoverflow.com/questions/42145696/get-object-from-aws-s3-as-a-stream)

public FileStream GetFile(string keyName)

{

using (client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2))

{

GetObjectRequest request = new GetObjectRequest

{

BucketName = bucketName,

Key = keyName

};

using (GetObjectResponse response = client.GetObject(request))

using (Stream responseStream = response.ResponseStream)

using (StreamReader reader = new StreamReader(responseStream))

{

// The following outputs the content of my text file:

Console.WriteLine(reader.ReadToEnd());

// Do some magic to return content as a stream

}

}

IDistributedCache

private readonly IDistributedCache \_distributedCache;

public FilesController(

IAmazonS3 amazonS3,

IWebHostEnvironment webHostEnvironment,

IMemoryCache memoryCache,

IDistributedCache distributedCache

)

{

\_amazonS3 = amazonS3;

\_webHostEnvironment = webHostEnvironment;

\_memoryCache = memoryCache;

\_distributedCache = distributedCache;

}

[HttpGet]

[Route("GetFileByKeyAsync")]

public async Task<IActionResult> GetFileByKeyAsync(string bucketName, string key)

{

// var cachemem = \_distributedCache.GetString("s3BucketData");

var bucketExists = await Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(\_amazonS3, bucketName);

if (!bucketExists)

{

return NotFound($"Bucket {bucketName} does not existed.");

}

var s3object = await \_amazonS3.GetObjectAsync(bucketName, key);

//var data = File(s3object.ResponseStream, s3object.Headers.ContentType);

GetObjectRequest request = new GetObjectRequest()

{

BucketName = bucketName,

Key = key

};

using(GetObjectResponse response = await \_amazonS3.GetObjectAsync(request))

using (var reader = new StreamReader(response.ResponseStream))

{

var conttent = await reader.ReadToEndAsync();

// Distributed cache timeout option

var options = new DistributedCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(80),

};

// distribution cache setstring value

\_distributedCache.SetString("JsonFile", conttent, options);

return Ok(conttent);

}

}

[HttpGet]

[Route("GetJsonFileData")]

public IActionResult GetJsonFileData()

{

var data = \_distributedCache.GetString("JsonFile");

if (data == null)

{

return NotFound("No data found.");

}

return Ok(data);

}

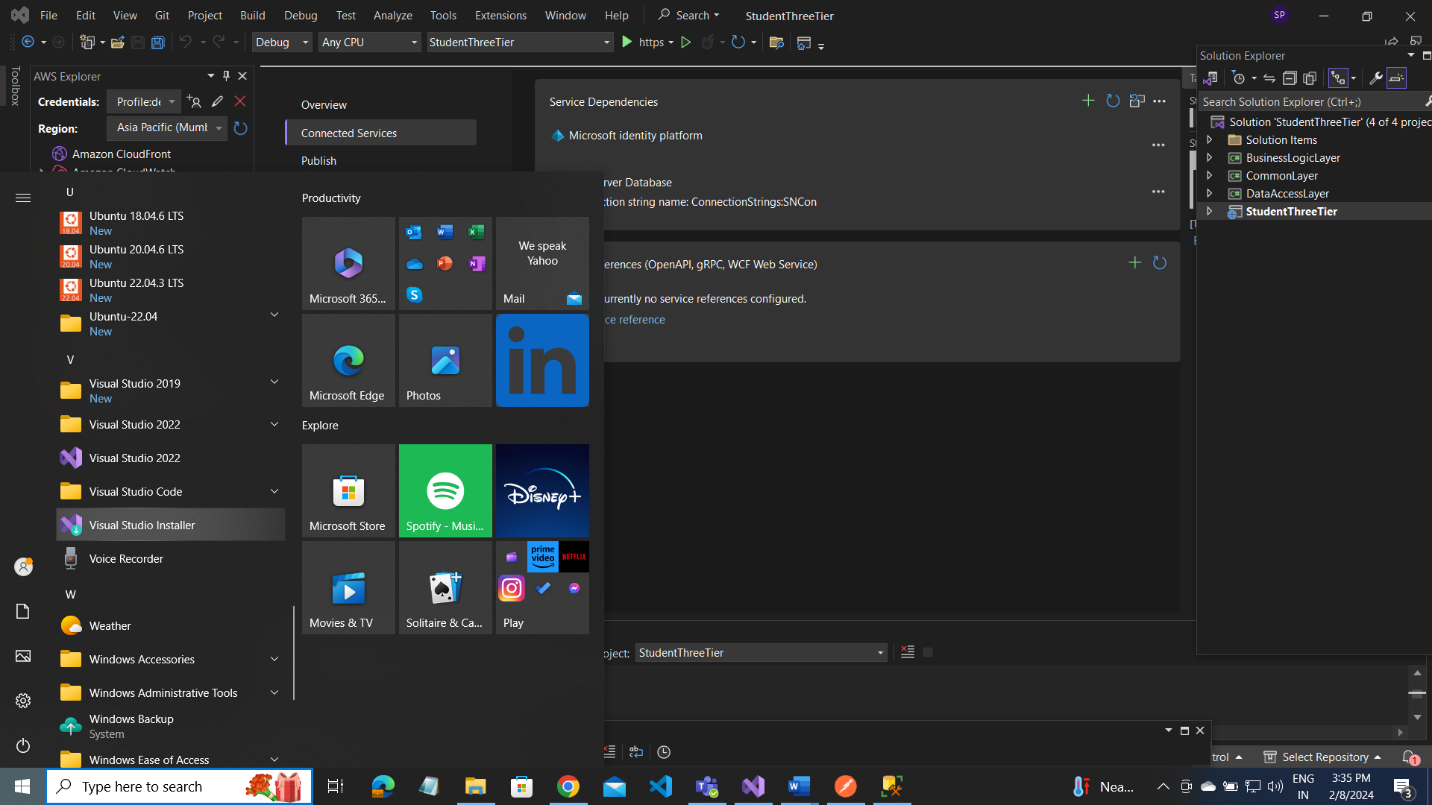
WCF( WINDOWS COMMUNICATION FOUNDER)

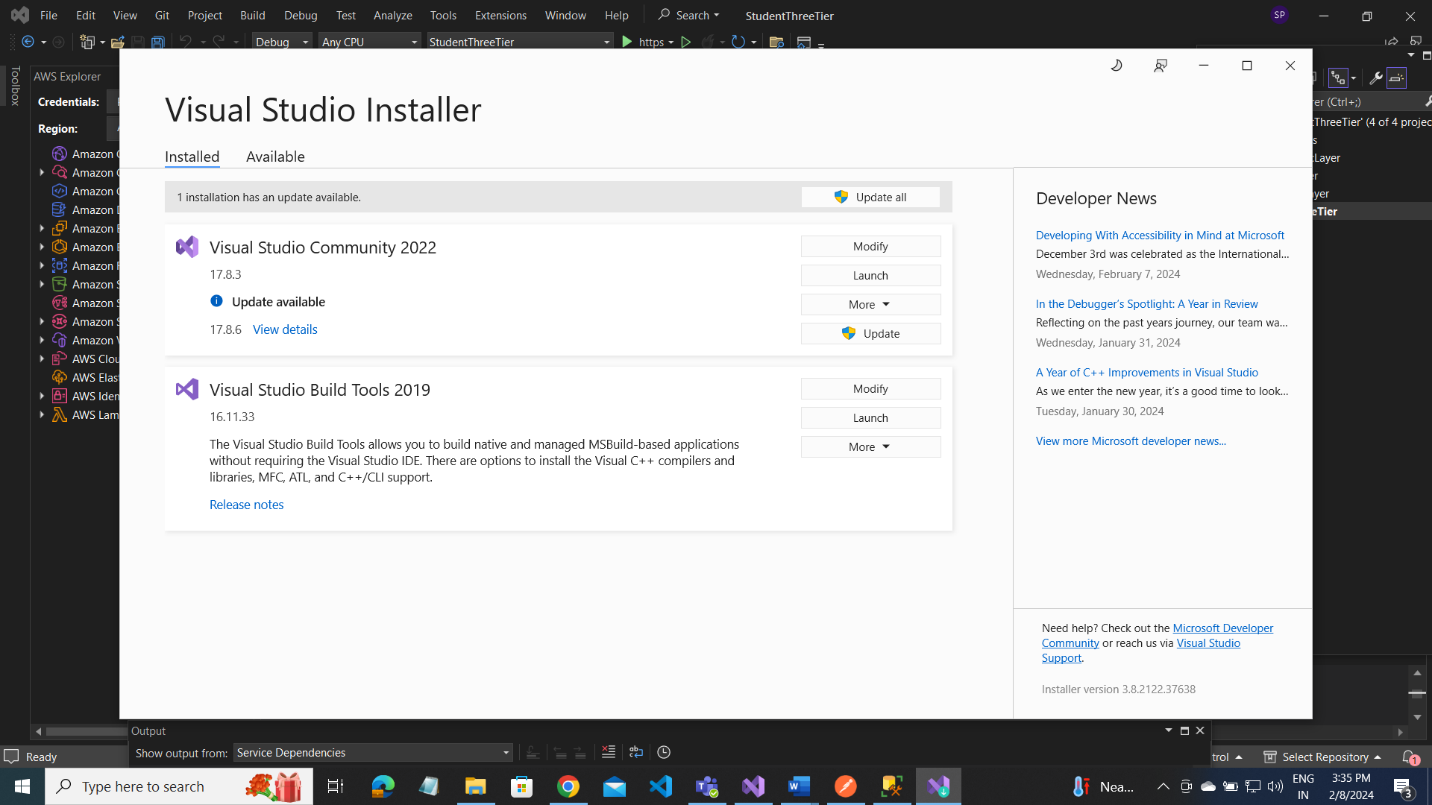
Windows Communication Foundation (WCF) is an RPC mechanism that enables rich client/server communication between processes. The advantages WCF has over other RPC mechanisms such as WebAPI, REST, or gRPC are that the contracts and data types are defined using .

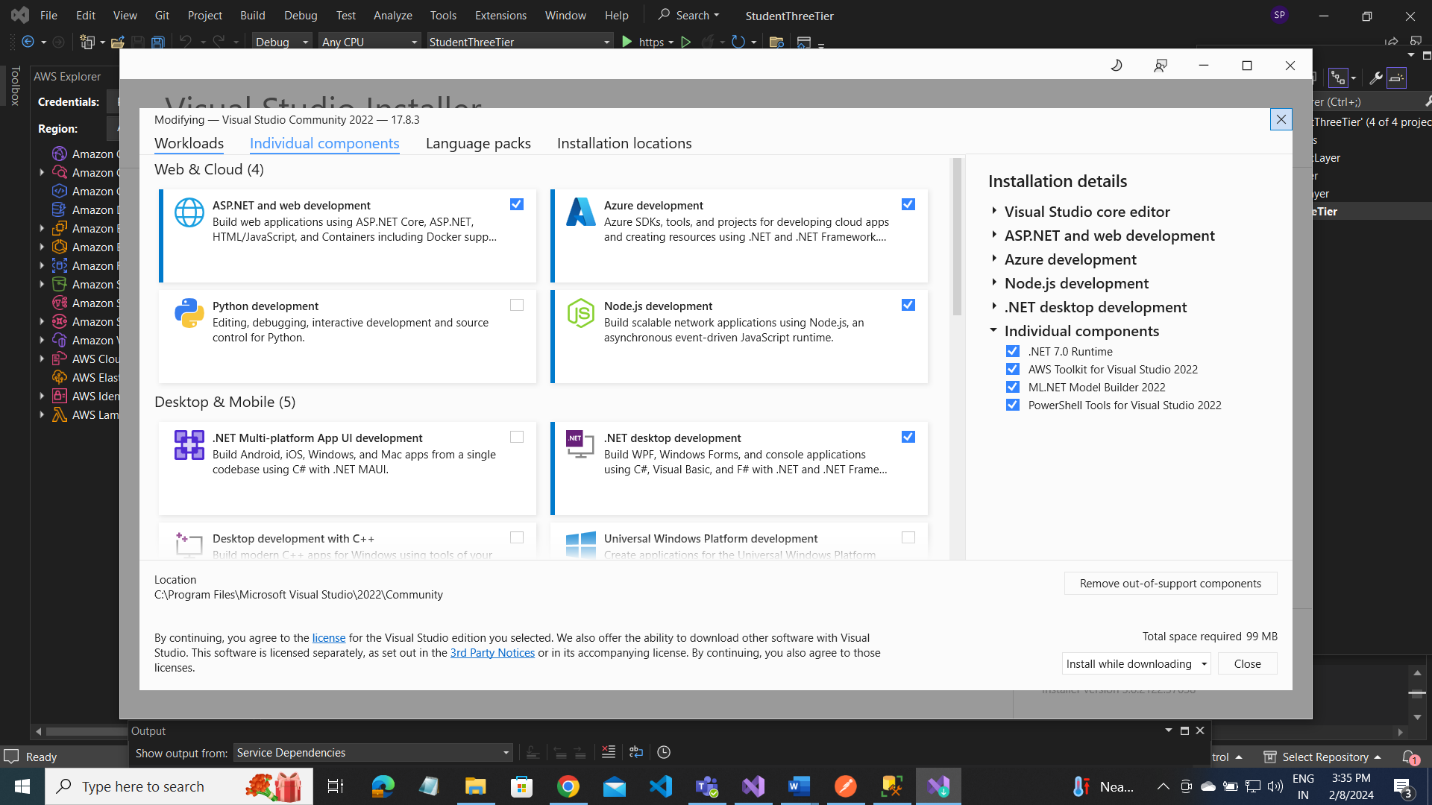
What is WCF in asp net core?

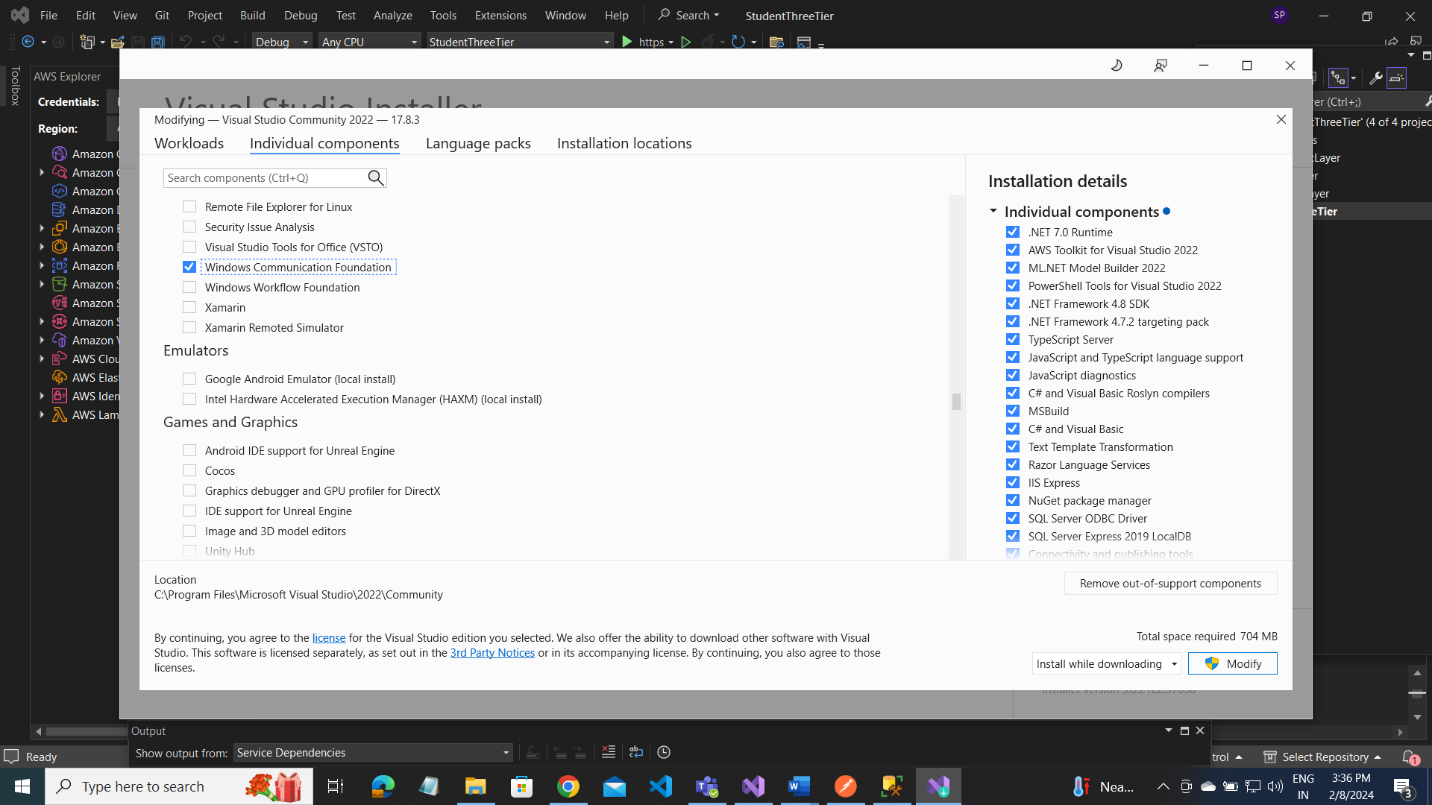
Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application.

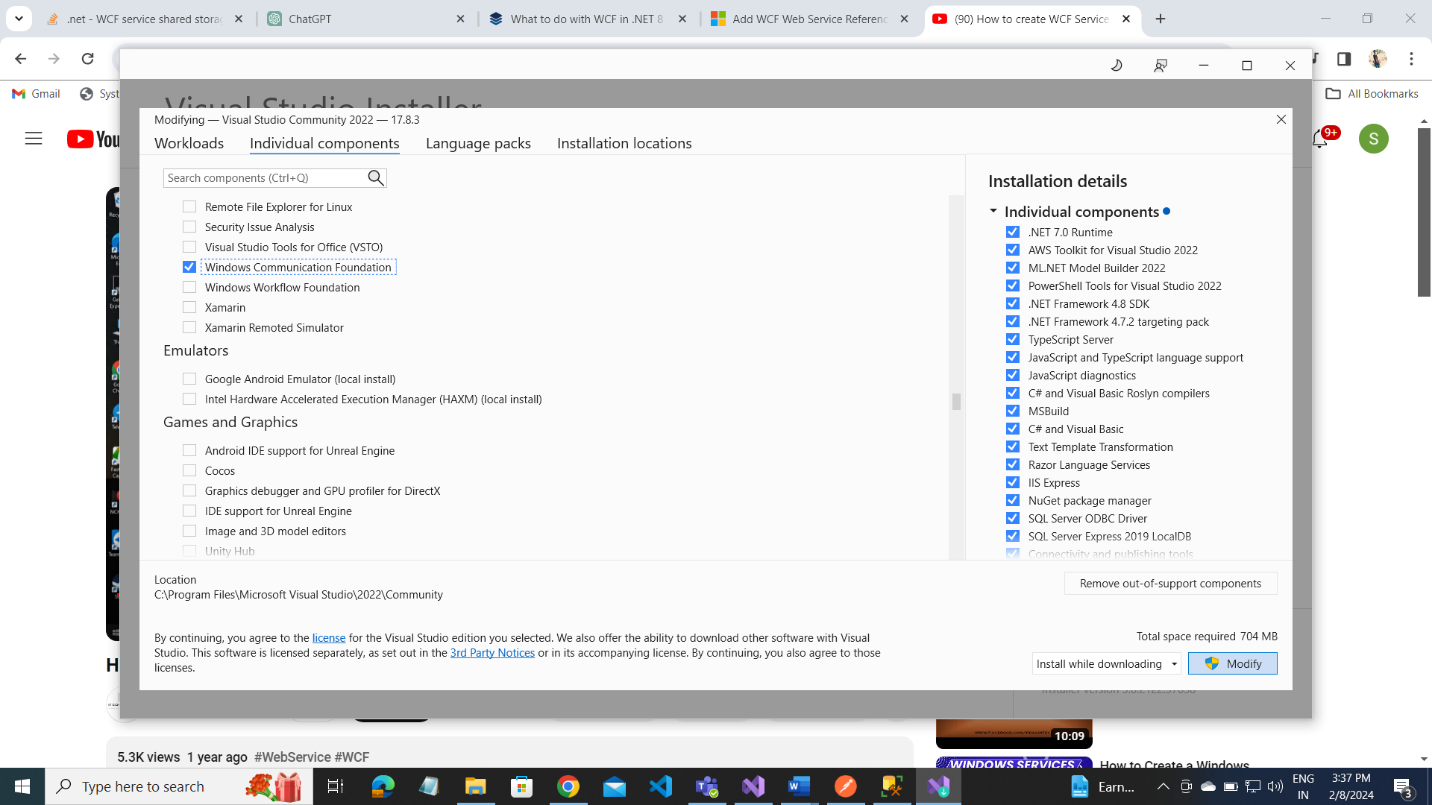
\*Before create project: install the windows communicatin founder in visual studio 2022.







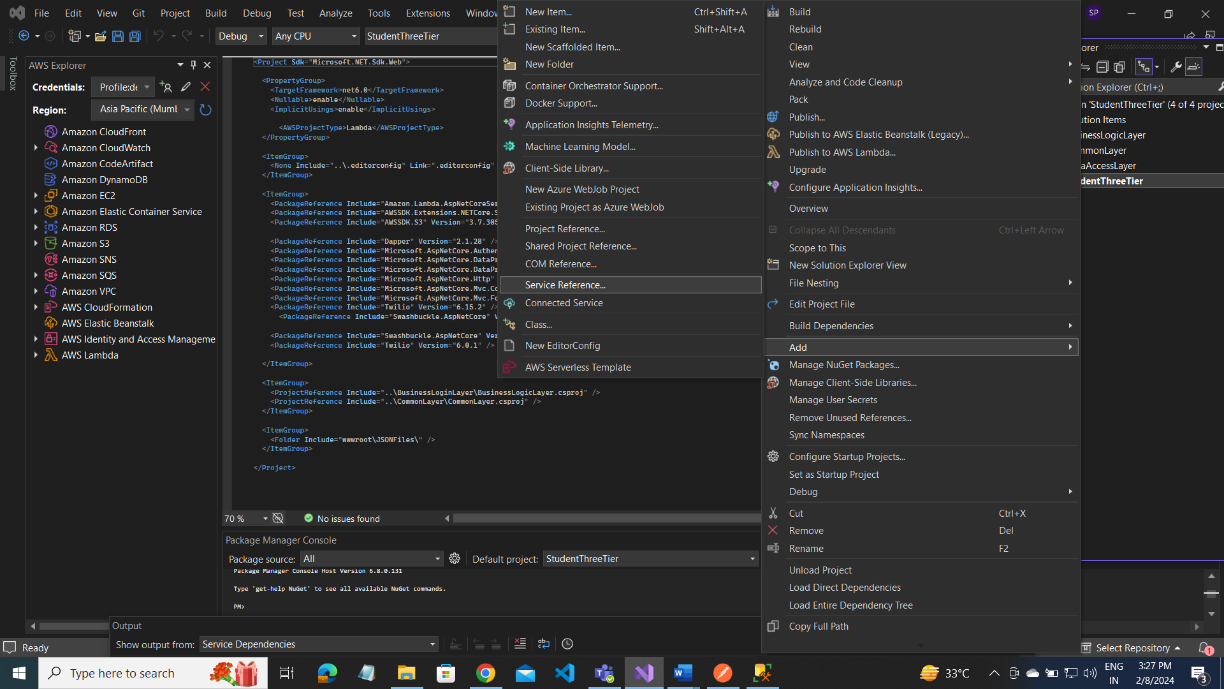


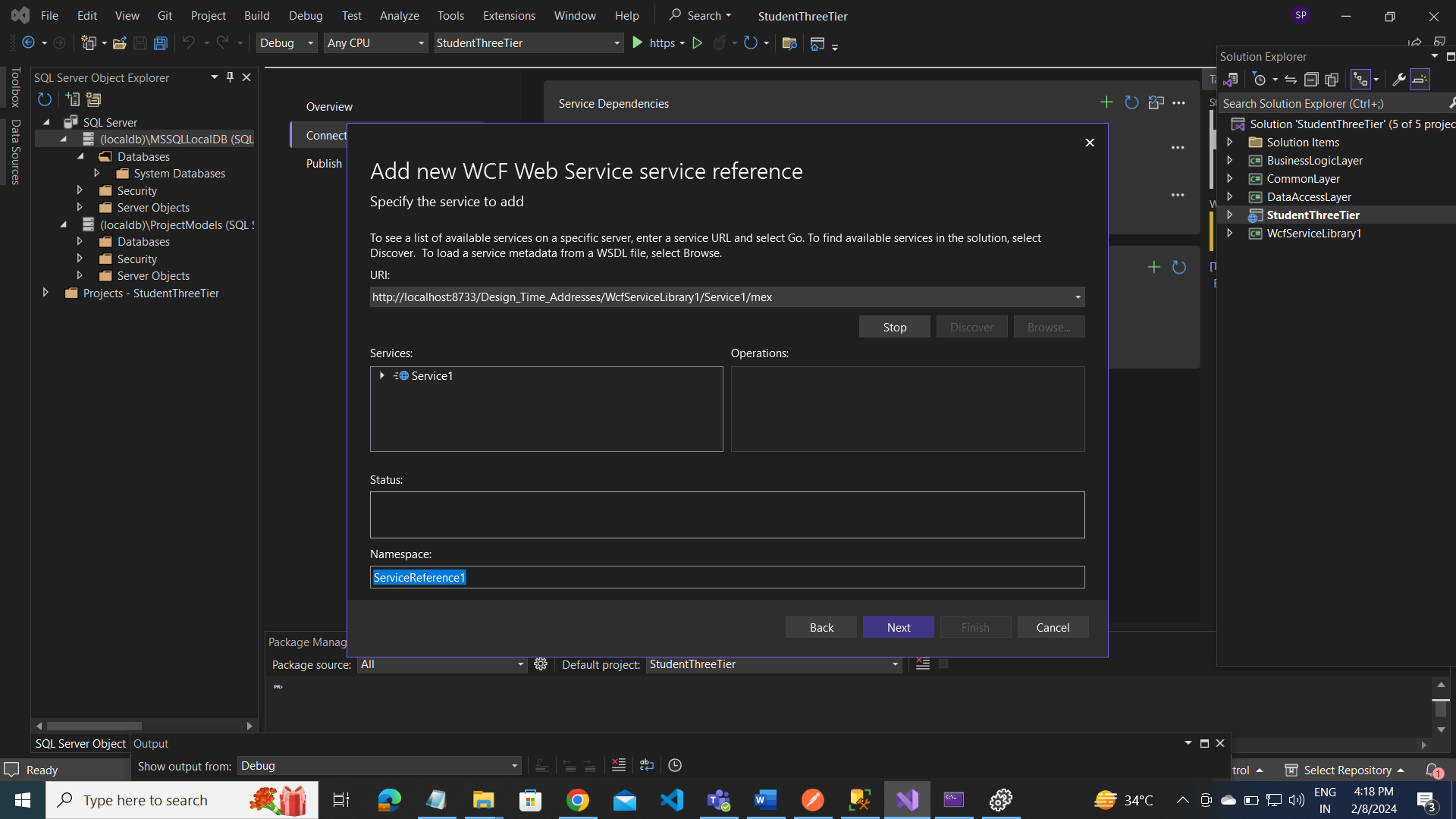


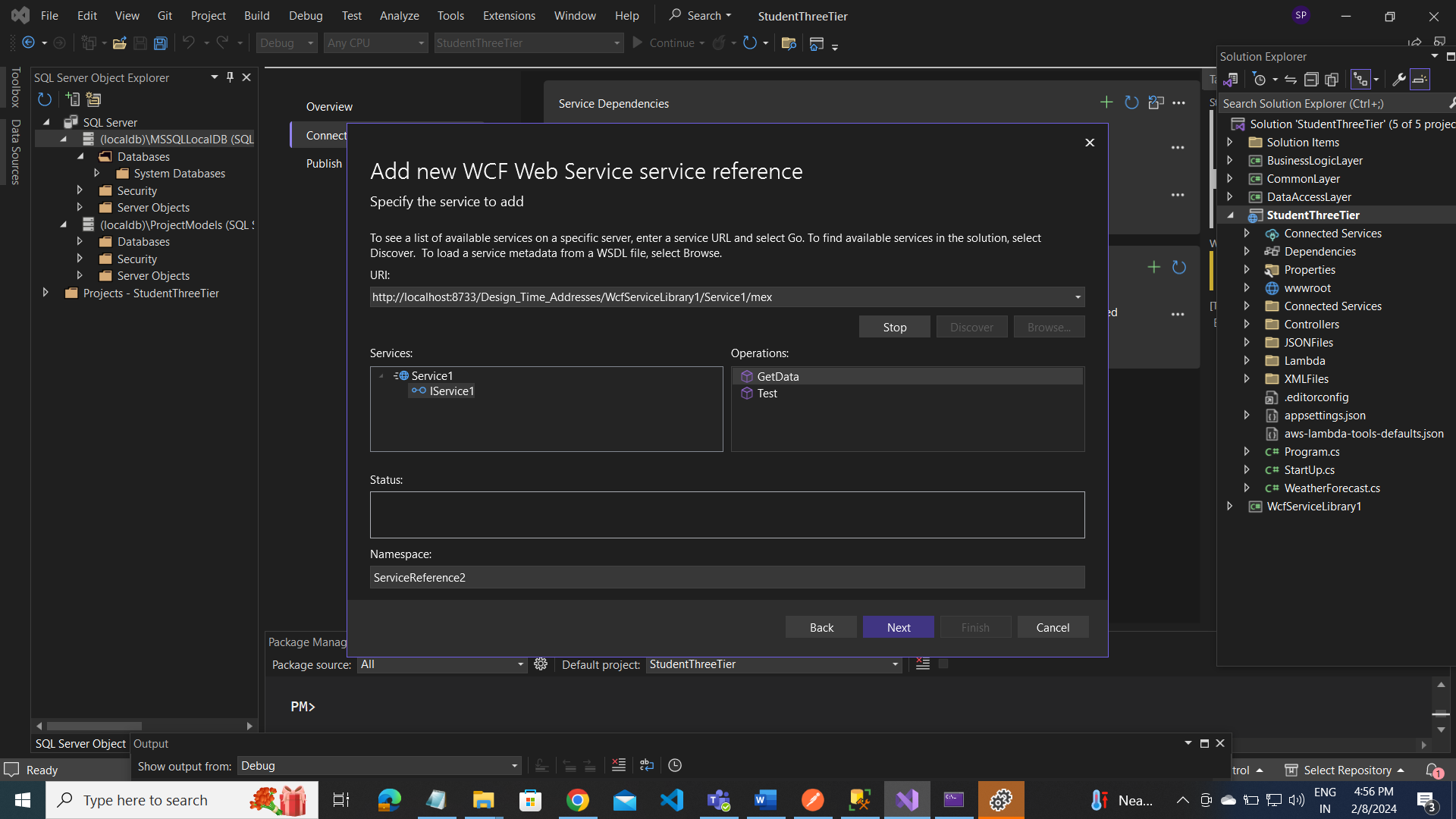
NEXT

1.create asp.net core6 web api project

2. Add service reference adding by clicking right of Your project name(StudentThreeTier)







Create WCF service:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService" in both code and config file together.

[ServiceContract]

public interface IService

{

[OperationContract]

string AddData(string data);

[OperationContract]

string GetData();

[OperationContract]

void Test();

// TODO: Add your service operations here

}

SERVICE:CS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

using System.Web.UI;

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Service" in code, svc and config file together.

public class Service : IService

{

private static string storedData;

public string AddData(string data)

{

return storedData = data;

}

public string GetData()

{

return storedData;

}

public void Test()

{

}

}

Usage in controller:

using TestService;

[HttpPost]

[Route("CreateEmployee")]

public async Task<Employee> CreateEmployee(Employee employee)

{

// var result = \_repo.AddEmployee(employee);

var result = await \_service.AddEmployee(employee);

var strEmpData = JsonConvert.SerializeObject(result);

ServiceClient client = new ServiceClient();

var data = client.AddData(strEmpData);

client.Close();

return result;

}

//[Authorize(Roles ="User")]

[HttpGet]

[Route("GetEmployeesList")]

public async Task<List<Employee>> GetEmployeesList()

{

ServiceClient client = new ServiceClient();

var getData = client.GetData();

client.Close();

// var result = \_repo.GetALLEmployees();

// var result = await \_service.GetALLEmployees();

List<Employee> employees;

if(!\_cache.TryGetValue("Employee", out employees))

{

employees = await \_service.GetALLEmployees();

\_cache.Set("Employee", JsonConvert.SerializeObject(employees), TimeSpan.FromMinutes(2));

}

return employees;

}

With Memory catch:

Contoller.cs:

[HttpGet]

[Route("GeWCFData")]

public IActionResult GetData()

{

//if(\_cache.TryGetValue("MyCatchData",out var result)) {

// return Ok(result);

//}

//else

//{

// ServiceClient client = new ServiceClient();

// string[] data = new string[] { "sai", "vijay", "vinay" };

// Console.WriteLine(data);

// var res = JsonConvert.SerializeObject(data);

// client.AddData(res);

// // var response = client.GetData();

// client.Close();

// return Ok(res);

//}

ServiceClient client = new ServiceClient();

var data = client.GetWCFData();

if(data == null || data == "No data found")

{

return NotFound("No data found.");

}

client.Close();

return Ok(data);

}

//[Authorize(Roles ="User")]

[HttpGet]

[Route("GetEmployeesList")]

public async Task<List<Employee>> GetEmployeesList()

{

// var result = \_repo.GetALLEmployees();

// var result = await \_service.GetALLEmployees();

List<Employee> employees;

if(!\_cache.TryGetValue("Employee", out employees))

{

employees = await \_service.GetALLEmployees();

\_cache.Set("Employee", JsonConvert.SerializeObject(employees), TimeSpan.FromMinutes(2));

}

return employees;

}

WcfService1:

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService" in both code and config file together.

[ServiceContract]

public interface IService

{

[OperationContract]

string AddData(string data);

[OperationContract]

string GetData();

[OperationContract]

void AddWCFData(string data);

[OperationContract]

string GetWCFData();

//[OperationContract]

//void Test();

// TODO: Add your service operations here

}

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

public class Service : IService

{

private static string storedData;

private ObjectCache \_cache = System.Runtime.Caching.MemoryCache.Default;

}

public void AddWCFData(string data)

{

try

{

if (!string.IsNullOrEmpty(data))

{

CacheItemPolicy policy = new CacheItemPolicy

{

AbsoluteExpiration = DateTimeOffset.Now.AddMinutes(2)

};

\_cache.Set("MyCatchData", data, policy);

}

}

catch (Exception ex)

{

throw ex;

}

}

public string GetWCFData()

{

try

{

string data = (string)\_cache.Get("MyCatchData");

//if (!string.IsNullOrEmpty(data.ToString()))

if(data != null)

{

return data.ToString();

}

else

{

return "No data found";

}

}

catch (Exception ex)

{

throw ex;

}

}