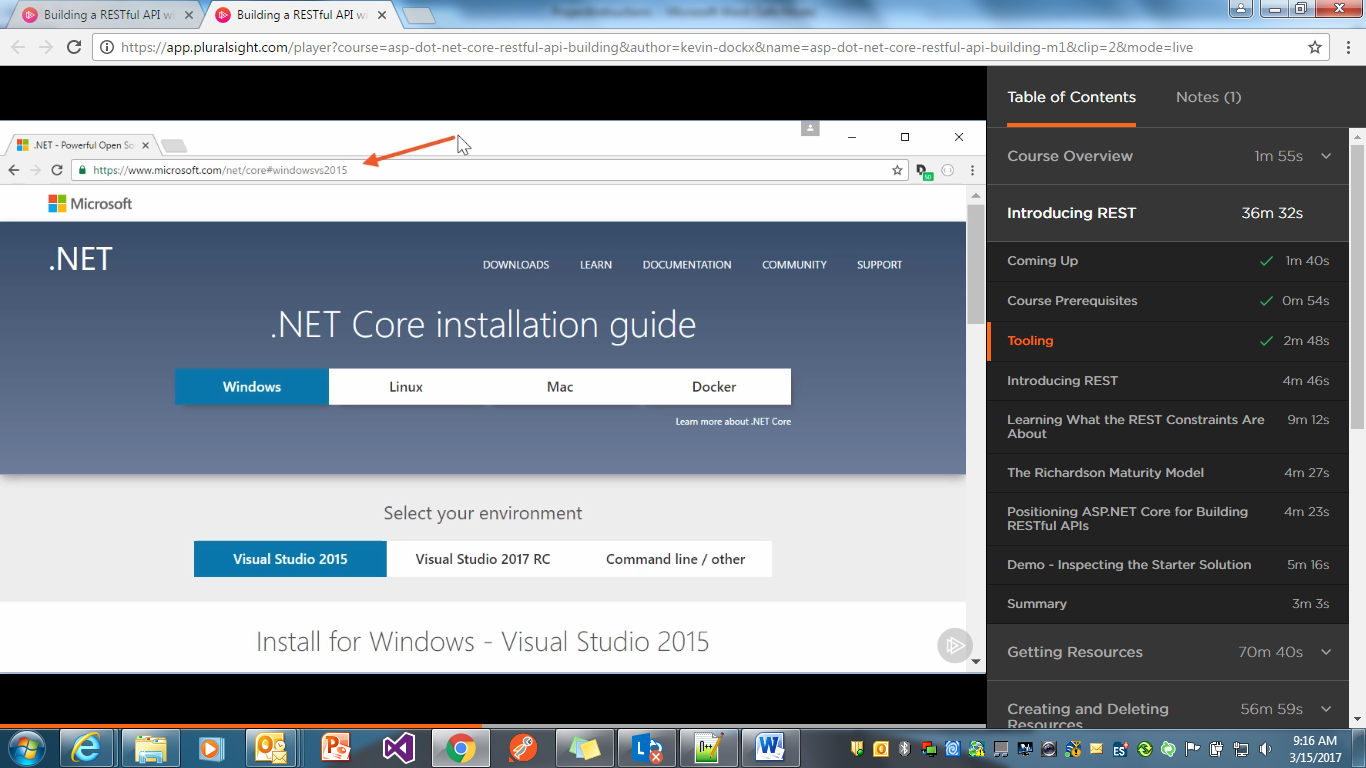
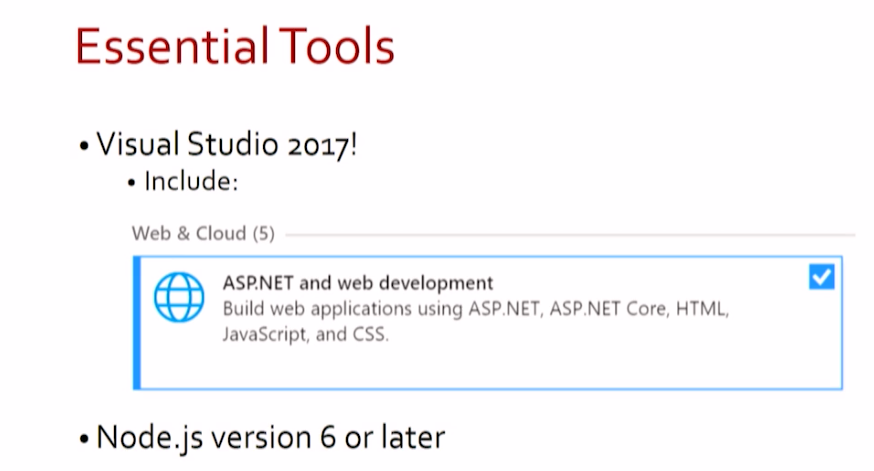
**.Net Core and Angular X project implementation**

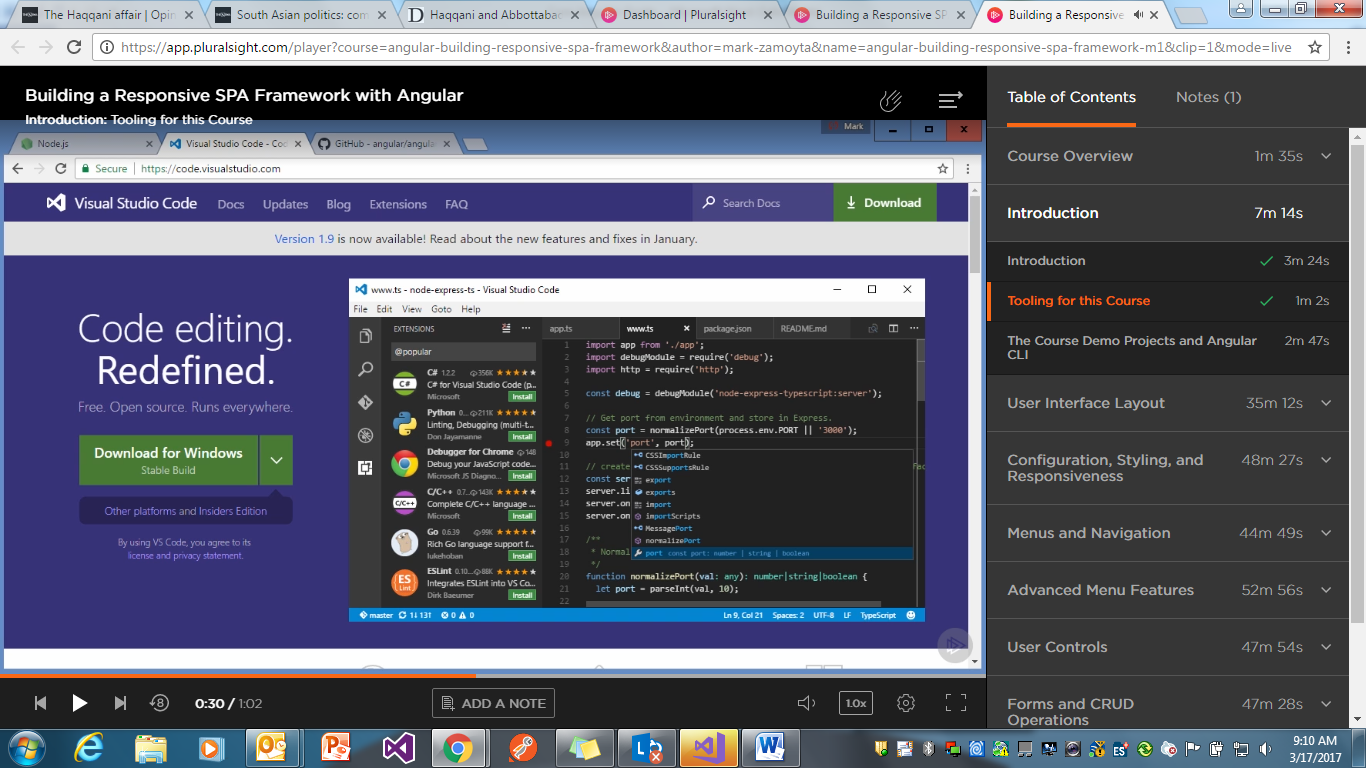
Pre – Requisites (TODO – Add other Pre-requisites as needed)

1. .Net Core 1.1 SDK installable – Note the Url might be different



1. Editor – Visual Studio 2017 or Visual Studio Code



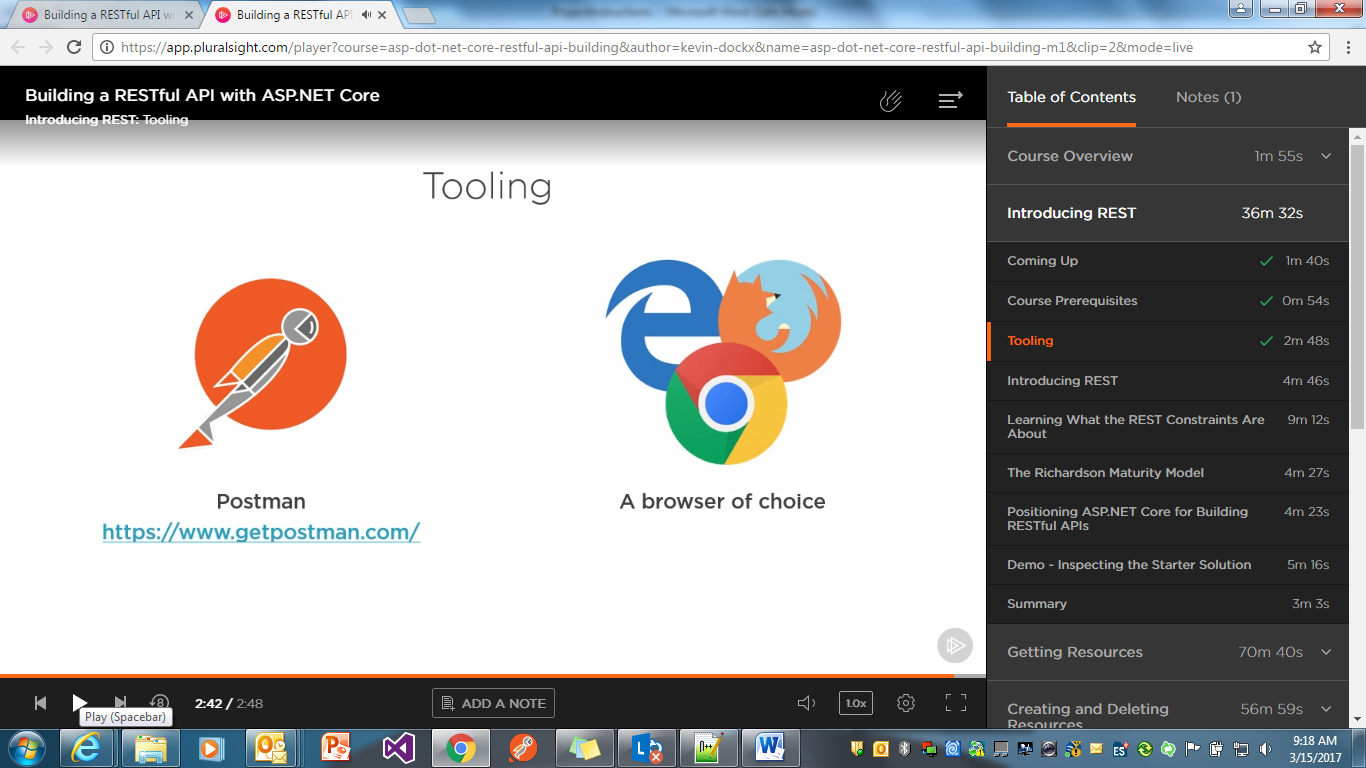


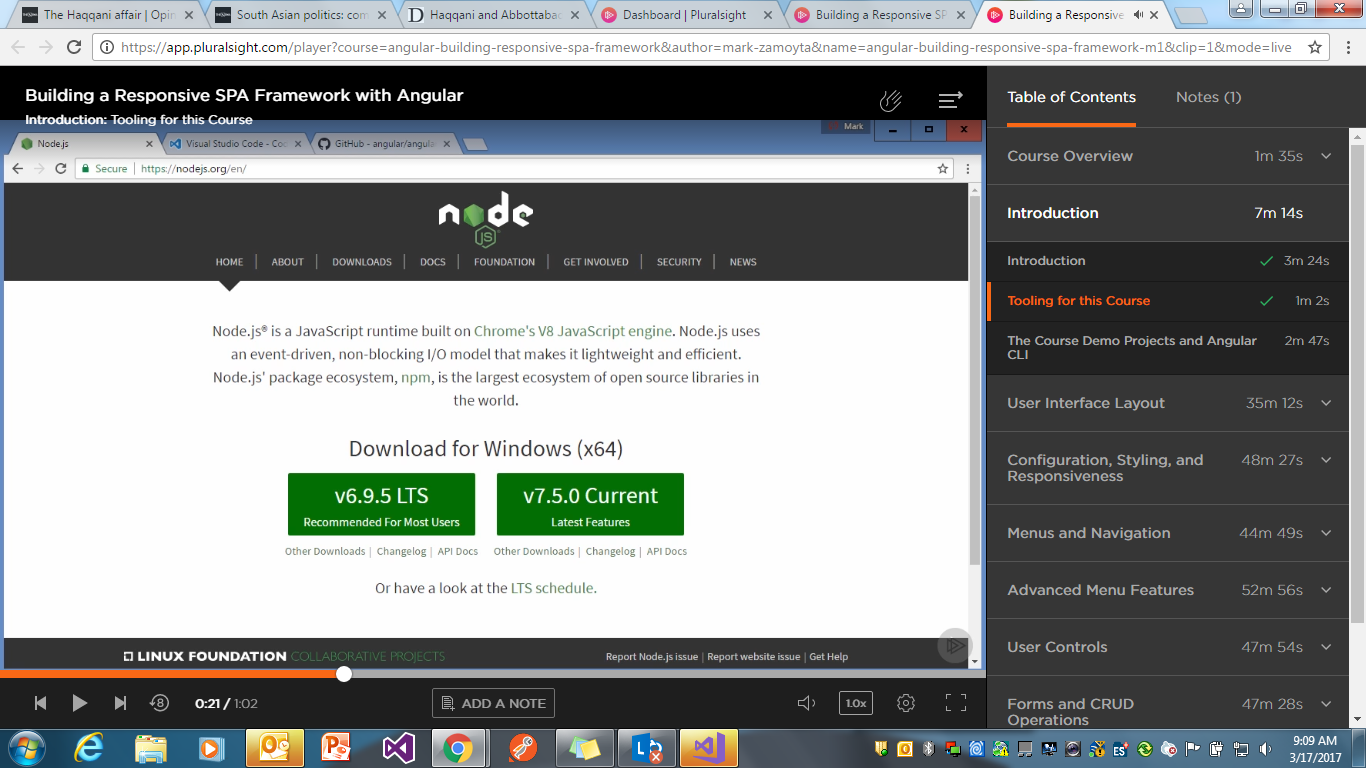
If Visual Studio Code , add the below c# extension for intellisense

Install C# extension in Visual Studio Code by following the below

Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

ext install csharp

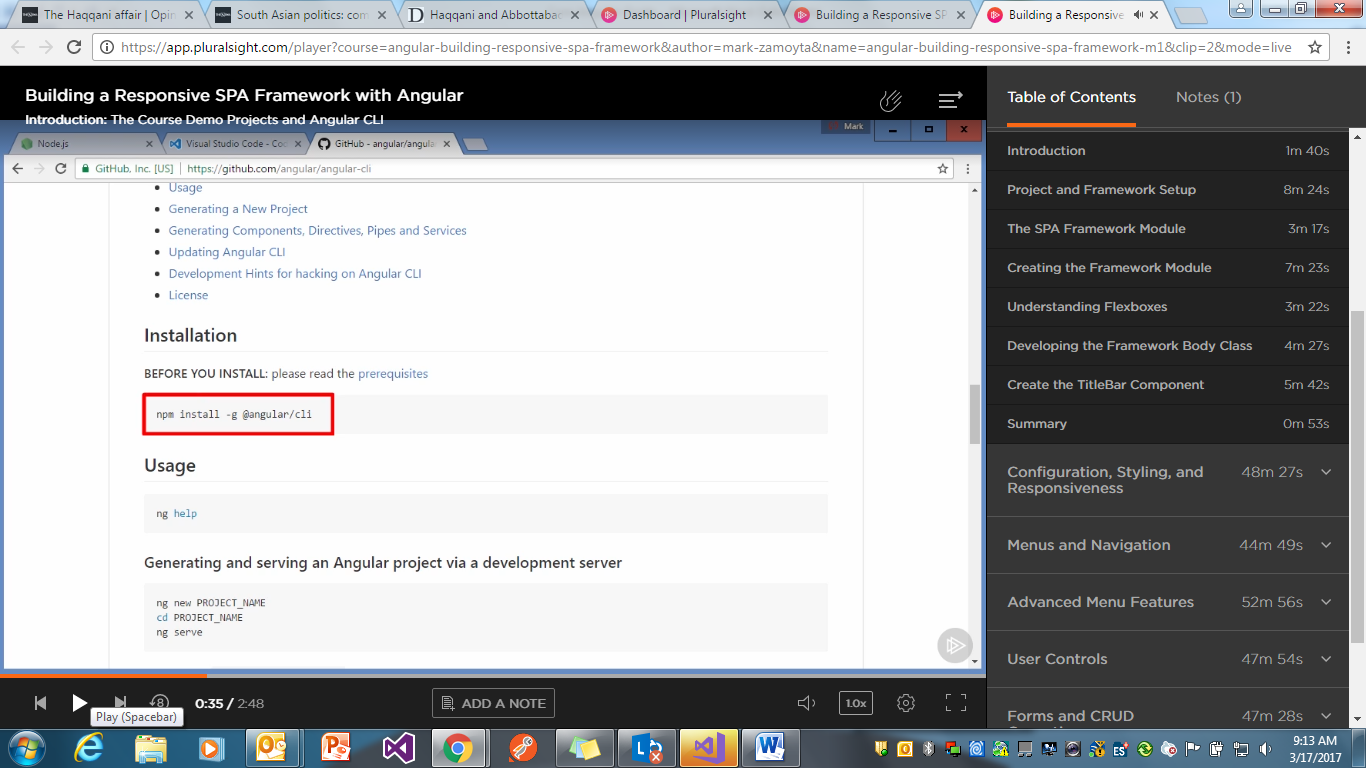




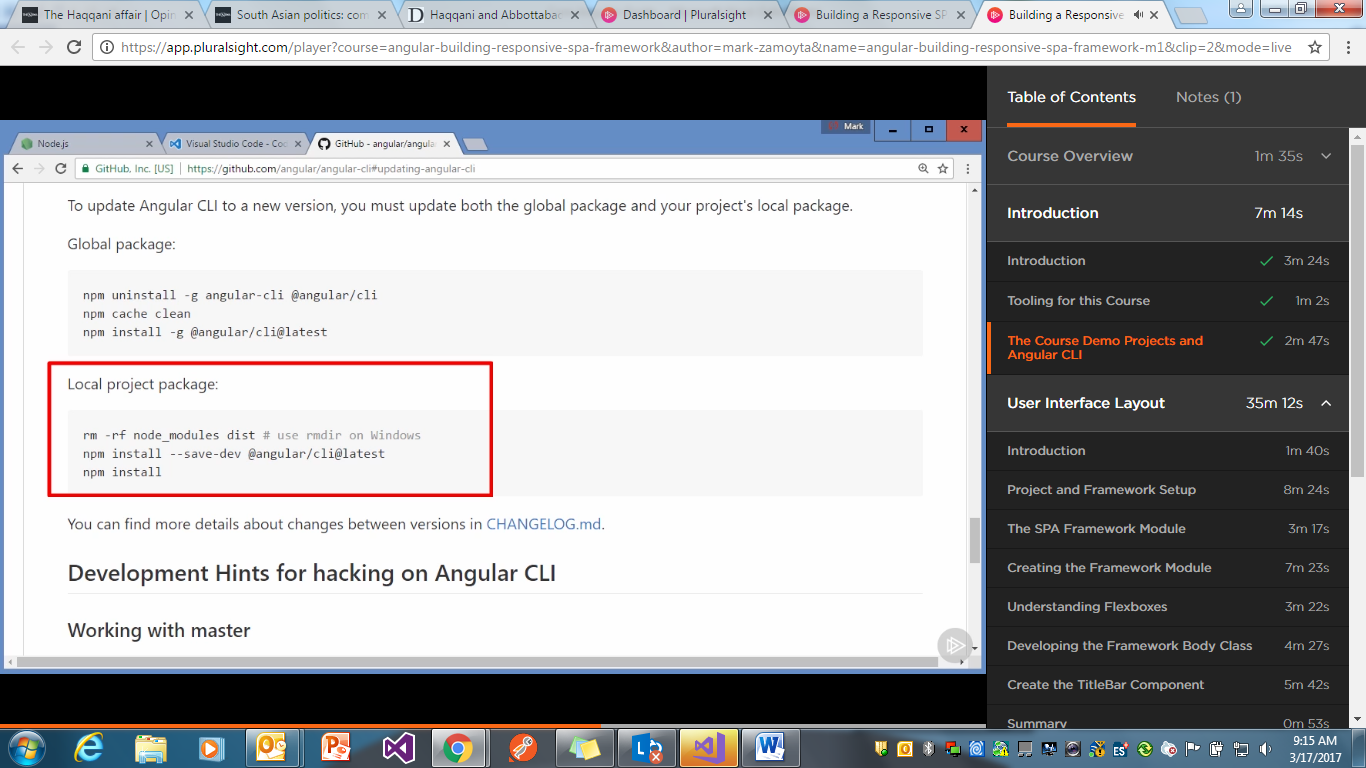
**MS Sql server 2014 or latest management studio.**

**Swagger**

If using Angular Cli





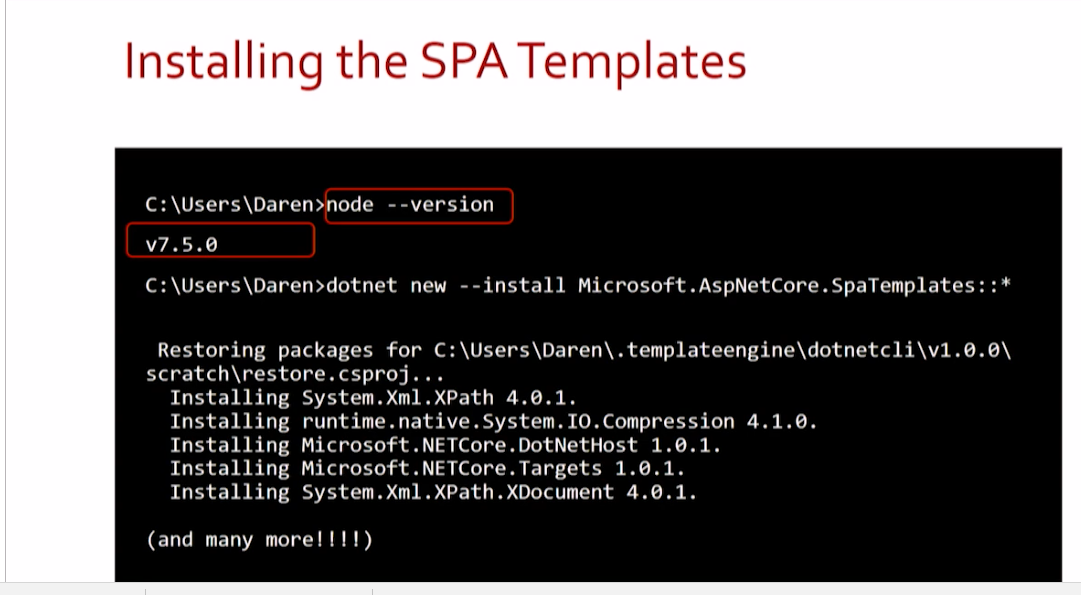


For building SPA Application make sure you have installed the SPA template

<https://channel9.msdn.com/Events/Visual-Studio/Visual-Studio-2017-Launch/WEB-103>

Minutes -->4:13

As in the below – The NODE version should be greater > **6**



After the templates are installed , verify the list of templates installed with the below command

**dotnet new –l**

**(TRY may be with an empty solution and then in that folder do the below –So in that case we wouldn’t need to create 2 folders ??)**

Now create your application folder Ex. D:\Dev\HelpDesk.

And create another folder HelpDesk.Web

And go to HelpDesk.Web folder from cmd and execute the below command

D:\MyProjects\ HelpDesk.Web >**dotnet new angular**

After then restore packages by

D:\MyProjects\ HelpDesk.Web >**dotnet restore**

And then client side packages by

D:\MyProjects\ HelpDesk.Web >**npm install**

**After you make add any client Side packages to the application make sure you run the command**

**webpack --config webpack.config.vendor.js**

**After you make any changes to the Client Side of the application make sure you run the command**

**webpack**

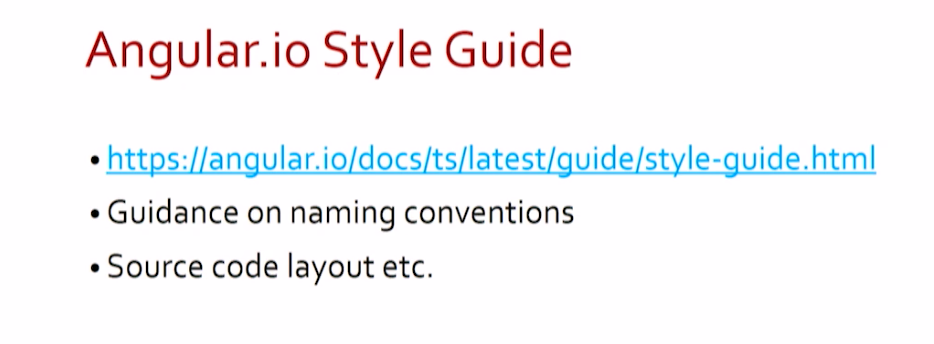
Use full Extensions for Visual Studio

<https://marketplace.visualstudio.com/>

<https://marketplace.visualstudio.com/items?itemName=MadsKristensen.Angular2SnippetPack>



Angular Style Guide



Generated Code Clean Up

References 🡪

<https://channel9.msdn.com/Events/Visual-Studio/Visual-Studio-2017-Launch/WEB-103>

Removed the

asp-prerender-module="ClientApp/dist/main-server"

from the

<app asp-prerender-module="ClientApp/dist/main-server">Loading...</app>

D:\MyProjects\HelpDesk\Views\Home\Index.cshtml

Removed the serverBundleConfig from webpack.config.js

// Configuration for server-side (prerendering) bundle suitable for running in Node

//const serverBundleConfig = merge(sharedConfig, {

// resolve: { mainFields: ['main'] },

// entry: { 'main-server': './ClientApp/boot-server.ts' },

// plugins: [

// new webpack.DllReferencePlugin({

// context: \_\_dirname,

// manifest: require('./ClientApp/dist/vendor-manifest.json'),

// sourceType: 'commonjs2',

// name: './vendor'

// })

// ],

// output: {

// libraryTarget: 'commonjs',

// path: path.join(\_\_dirname, './ClientApp/dist')

// },

// target: 'node',

// devtool: 'inline-source-map'

//});

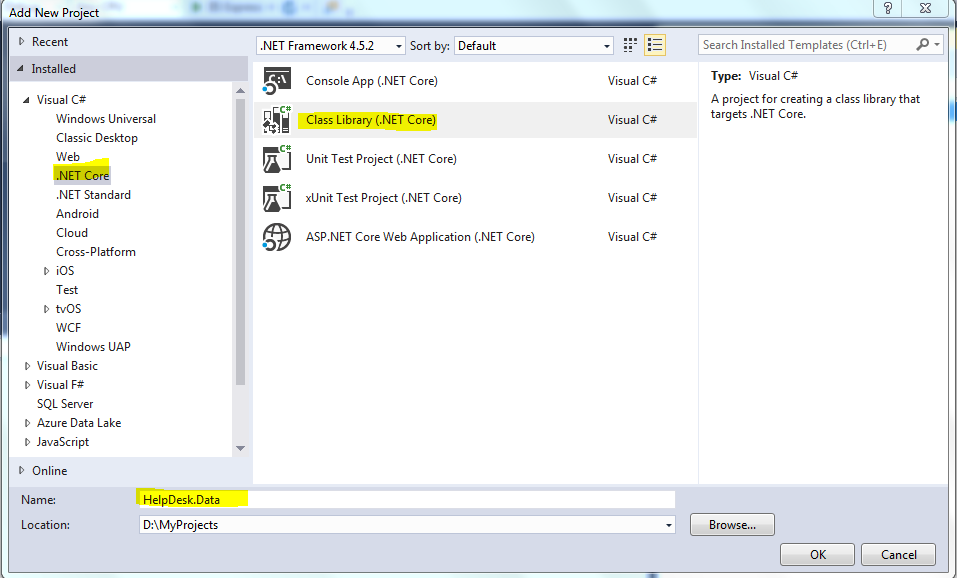
//return [clientBundleConfig, serverBundleConfig];

return [clientBundleConfig];

Deleted the boot-server.ts

Class Library

Add class library from the Visual Studio



Authentication & Authorization

Reference:

**Understanding ASP.NET Core Security**

(Has details regarding Aspnet Identity and Identity Server & Claims based Authorization).

**Building a RESTful API with ASP.NET Core**

# Entity Framework Core: Getting Started

(How to add migrations )

Add the class Library Ex. HelpDesk.Data, HelpDesk.Models

In the HelpDesk.Models project.

Add nuget reference to Microsoft.AspNetCore.Identity.EntityFrameworkCore

Add a new HelpDeskUser Class which inherits from IdentityUser. It has additional generic parameter if you want to change the Primary Key instead from default varchar to Guid .You can add additional user related properties in this class which will eventually be part of the AspNetUsers table

So ex.

public class HelpDeskUser:IdentityUser<Guid>

{

public bool IsActive { get; set; }

}

In the project HelpDesk.Data add the following Nuget References

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.AspNetCore.Identity.EntityFrameworkCore

References (TODO – Add other References as needed)

Add reference of HelpDesk.Models into HelpDesk.Data

Create a new class HelpDeskContext

public class HelpDeskContext : IdentityDbContext<HelpDeskUser, IdentityRole<Guid>, Guid>

Web Project related to Identity

Add project reference to the Class Library projects Ex. HelpDesk.model and HelpDesk.Data in the HelpDesk Web project

Add nugget reference to

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

In the configure Services add the

services.AddSingleton(Configuration);

services.AddIdentity<HelpDeskUser, IdentityRole<Guid>>()

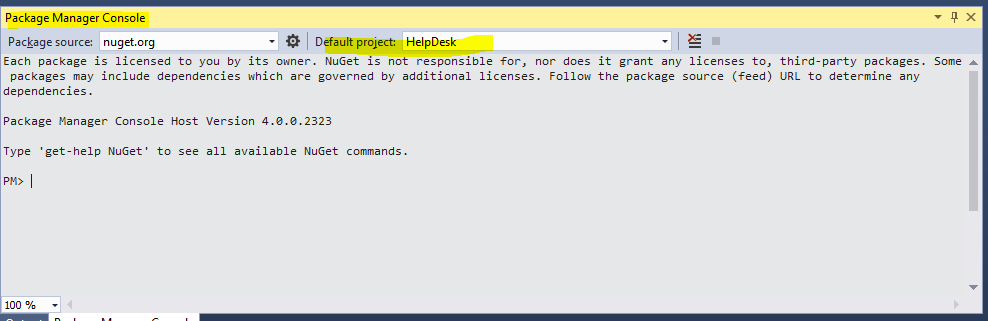
.AddEntityFrameworkStores<HelpDeskContext, Guid>();

services.AddDbContext<HelpDeskContext>();

In the Configure add the below

app.UseIdentity();

Go to Packaga Manager Console .Select the Web project ie HelpDesk



Execute the below command to see all the available commands

get-help entityframeworkcore

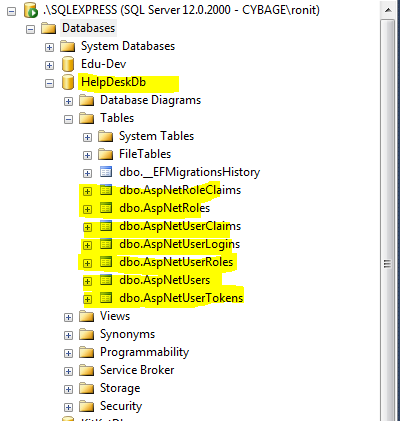
get-help add-migration

Add the migration

add-migration Database-Identity

update-database -verbose

After running the update the database should be created as shown



Controllers- Authentication & Authorization

Add new folder **Api** inside the controllers folder.

Define the attribute based routing as in the below link

<https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing>

Ex . On the controller level the routes is defined as.Note the below uses attribute based routing .

# Reference : Asp.Net Core Fundamentals

# Refer to attributes section for details

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

And at the action level , it is defined as

[HttpPost("[action]")]

Create ViewModels folder and define the AccountRegisterViewModel.

Demo of how to use Postman

Reference: Implementing and Securing an API with ASP.NET Core

Refer to Demo:Using Postman

Refer to [Demo: Implementing POST](https://app.pluralsight.com/player?course=aspdotnetcore-implementing-securing-api&author=shawn-wildermuth&name=aspdotnetcore-implementing-securing-api-m3&clip=3&mode=live)

Here is complete sample code for Register

[HttpPost("[action]")]

public async Task<IActionResult> Register([FromBody]AccountRegisterViewModel vm)

{

var user = new HelpDeskUser() { UserName = vm.LoginEmail, Email = vm.LoginEmail };

var result = await \_userManager.CreateAsync(user, vm.Password);

// Add roles if not present

if (!await \_roleManager.RoleExistsAsync("Admin"))

await \_roleManager.CreateAsync(new IdentityRole<Guid>() { Name = "Admin" });

if (!await \_roleManager.RoleExistsAsync("User"))

await \_roleManager.CreateAsync(new IdentityRole<Guid> { Name = "User" });

// Add user to a role

await \_userManager.AddToRoleAsync(user, vm.Role);

//Add claims to user

await \_userManager.AddClaimAsync(user, new Claim("Profession", vm.Profession));

if (result.Succeeded)

{

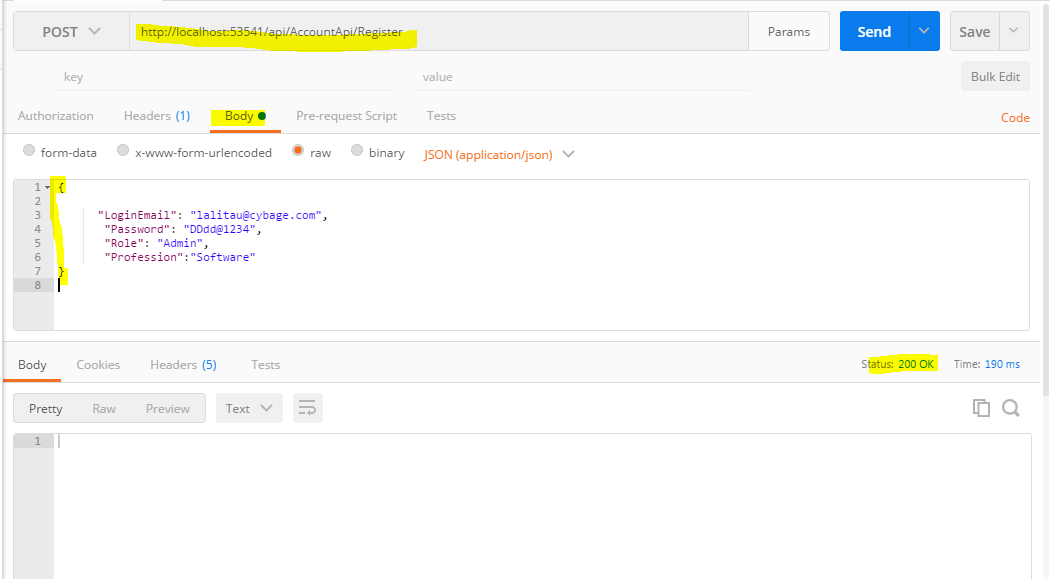
//DO you want to Sign IN ? after registration -- Ref

return Ok();

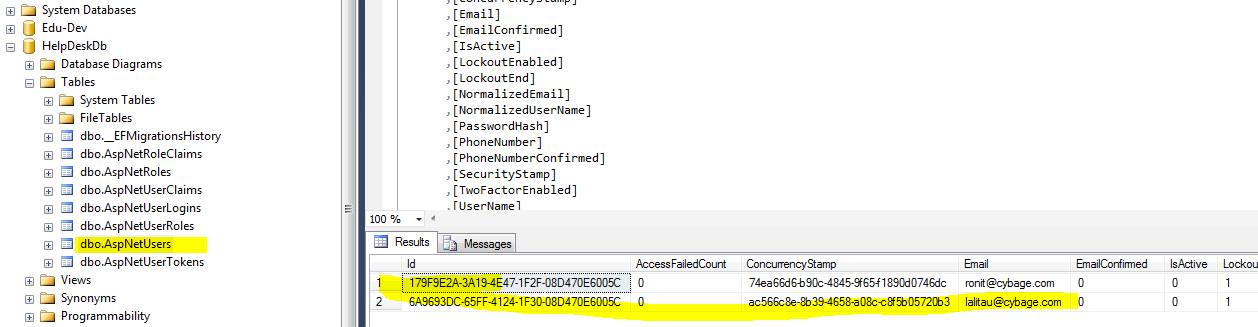
}

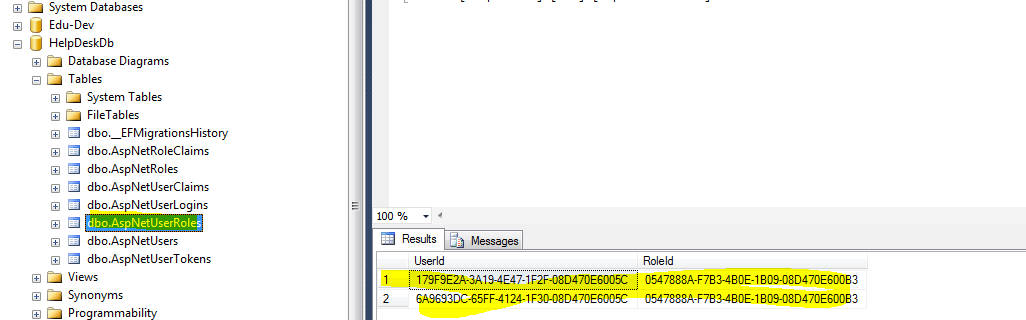
return BadRequest("Failed to register");

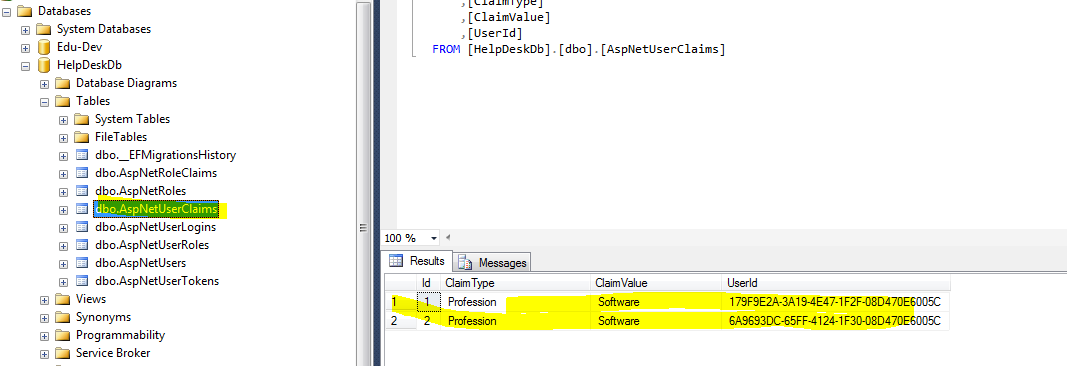
}

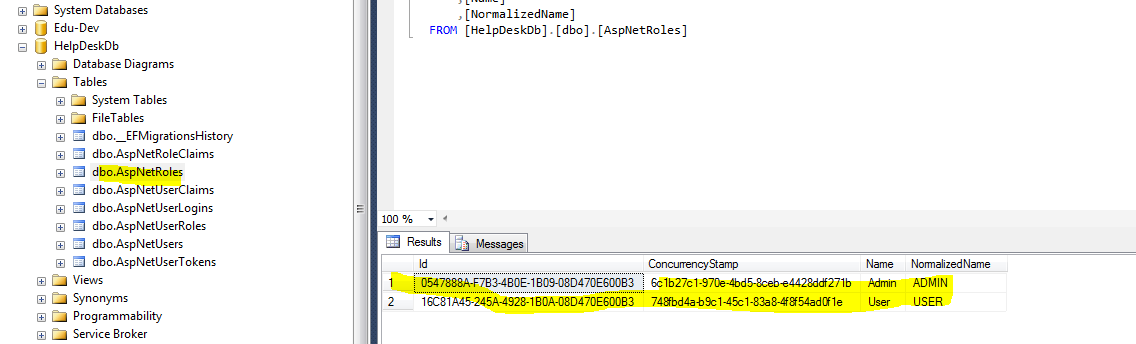


And the database entries









For external authentication like Google

Install the nugget package

Microsoft.AspNetCore.Authentication.Google

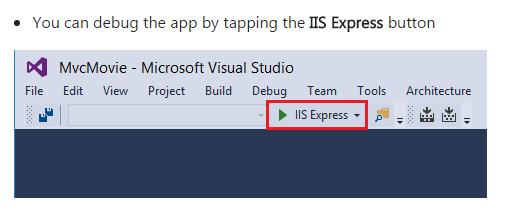
Add the middleware service

You need the client Id and secret and to get that you need to register at

http://tinyurl.com/extprovs

Debugging the server side application :

Click the IIS Express button as shown in the below link



<https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc>

Adding Documentation to API

References : Play by Play: Understanding API Functionality Through Swagger

Add the nugget package Swashbuckle to the Web project

Ex.In the current project- HelpDesk ,have installed the pre-release version , may be later want the release version.

Add the following swagger services in the ConfigureServices

services.AddSwaggerGen();

In the configure method

app.UseSwagger(); // This needs to be before the usemvc

app.UseSwaggerUi();//This needs to be after the usemvc

The default page to get the api details in swagger is

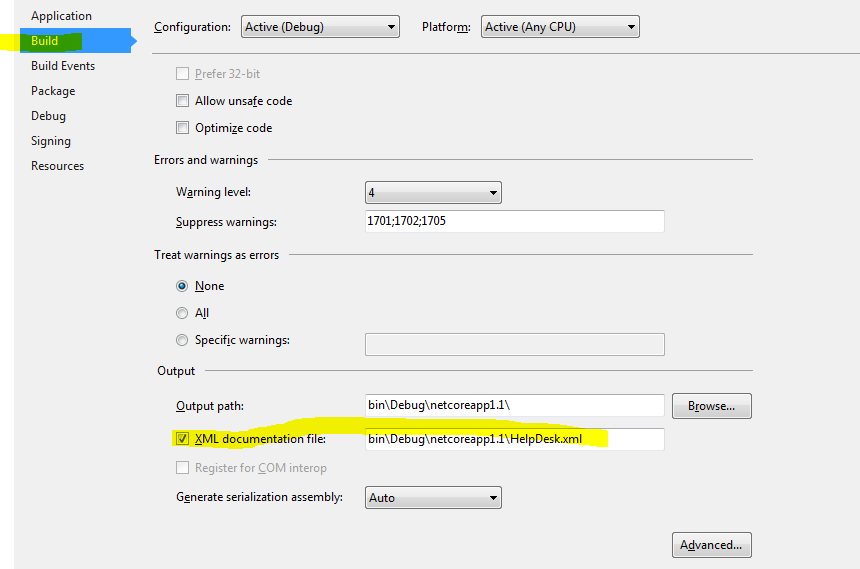
<http://localhost:53541/swagger/ui/index.html> .. Note the **port number** will defer based on your port of running application.

The above reference github repository for the PS course is at

<https://github.com/spboyer/superhero-api>

In order to have the xml comments reflected in the swagger documentation do the following setting .For reference

<http://www.talkingdotnet.com/add-swagger-to-asp-net-core-web-api/>



And add the below in COnfigureServices

services.ConfigureSwaggerGen(options =>

{

options.SingleApiVersion(new Info

{

Version = "v1",

Title = "Help Desk Api",

Description = "Api for Help Desk ",

TermsOfService = "None",

});

//Determine the base path of the application

var basePath = PlatformServices.Default.Application.ApplicationBasePath;

//Set the comments path for the swagger json and ui. The below is typically name of project which is case sensitive.

options.IncludeXmlComments(Path.Combine(basePath + "/HelpDesk.xml"));

});

Getting Started with ASP.NET Core in Visual Studio 2017(

<https://channel9.msdn.com/Events/Visual-Studio/Visual-Studio-2017-Launch/WEB-101>)

(Latest implementation of .net core 1.1 –Replacing project.json with .csproj and how to add core projects from cli.)

**Asp.Net Core Fundamentals**

(Basics of Asp.net Core )

# Building a Web App with ASP.NET Core, MVC 6, EF Core, and Angular

Building a RESTful API with ASP.NET Core

# Implementing and Securing an API with ASP.NET Core

# Understanding ASP.NET Core Security

(Has details regarding Aspnet Identity and Identity Server & Claims based Authorization).

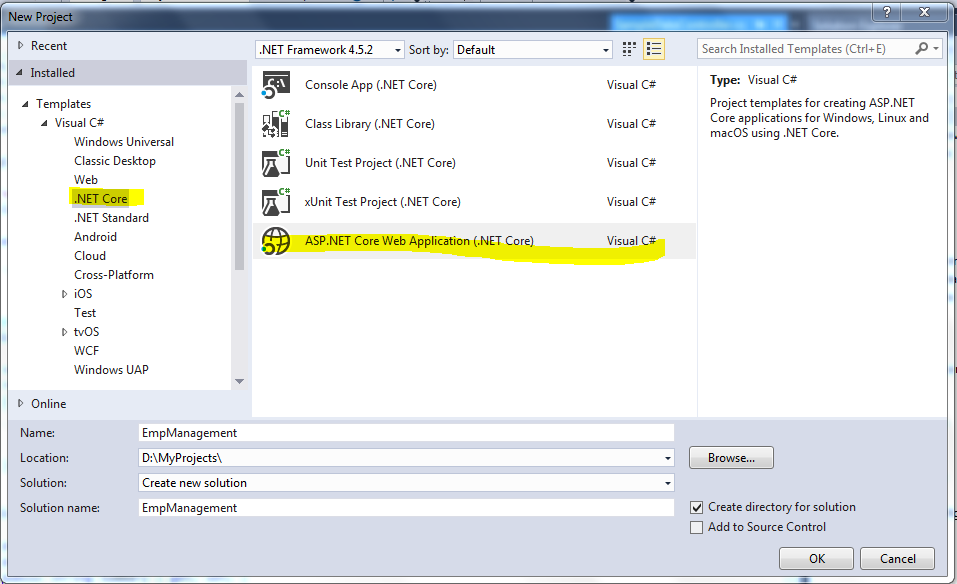
# Entity Framework Core: Getting Started

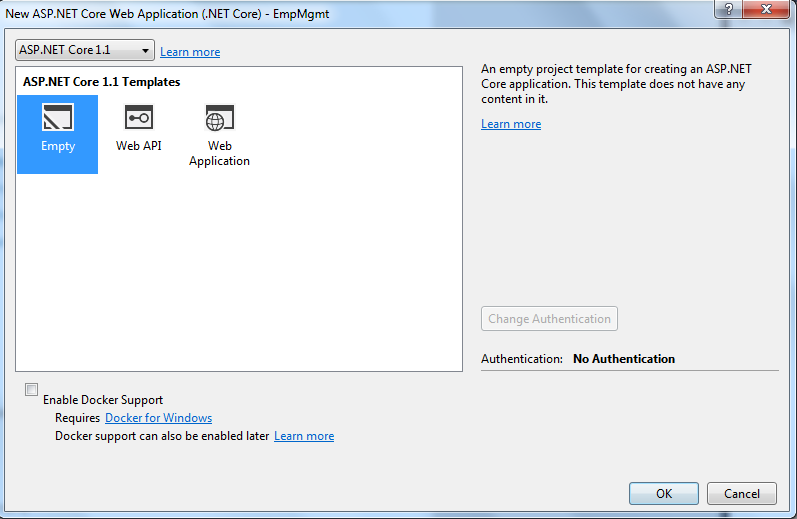
(How to add migrations )

# Building End-to-End Multi-Client Service Oriented Applications - Angular Edition

(Details regarding the project structure)

Project Implementation





Add the Controllers and the View Models folders

**Add a new class library from the cli**

Go to the folder which has the **.sln** file

Type **dotnet new** -- to see all the types of projects templates that can be created

Ex.

**D:\DEV\TicketClick>dotnet new classlib -n TicketClick.Data -o TicketClick.data**

Where –n is name of the project and

-o is the folder Name where it will be created in .

To add the newly created project to solution

**D:\DEV\TicketClick>dotnet sln add .\TicketClick.data\TicketClick.Data.csproj**

You can see the projects that are part of the solution by

**D:\DEV\TicketClick>dotnet sln list**

**Install the dependencies needed in the TicketClick.data and TickteClick projects**

Microsoft.EntityFrameworkCore

**Microsoft.EntityFrameworkCore.SqlServer**

Microsoft.EntityFrameworkCore.Tools

Microsoft.AspNetCore.Identity.EntityFrameworkCore

Add project Reference of TicketClick.data to the TicketClick project

Create the TicketClickContext

Register

services.AddDbContext<TicketClickContext>();

in the configure Services

Override the OnConfiguring in the TicketContext Class

Add the Db connection string in the config.json

"ConnectionStrings": {

"TicketClickConnection": "data source=.\\sqlexpress;initial catalog=TicketClickDb;Trusted\_Connection=true;MultipleActiveResultSets=true;"

}

Go to the Path of the Web Project in cmd and add

**In PM console for the Web Project when see the availale commands by**

**Get-help entityframeworkcore**

**To get help on particular migration command .Ex add-migration use the following**

**Get-help add-migration**

**Then run**

PM> add-migration InitialDatabase

**Error**

**The Entity Framework Core Package Manager Console Tools don't support PowerShell version 2.0. Upgrade to PowerShell version 3.0 or higher, restart Visual Studio, and try again.**

**Soln** – Download the latest Powershell command from the and select the installable applicable

Error

Your target project 'TicketClick' doesn't match your migrations assembly 'TicketClick.Data'. Either change your target project or change your migrations assembly.

Soln

Change your migrations assembly by using DbContextOptionsBuilder. E.g. options.UseSqlServer(\_config["ConnectionStrings:TicketClickConnection"], b => b.MigrationsAssembly("TicketClick"))

After this rerun

add-migration InitialDatabase

and then to create the database

**update-database -verbose**

Refer KitKat project on how to change the Guid as primary key instead of varchar for the Asp.net Identity Tables.

Add the TicketClickUser class which implements IdentityUser<Guid> in the new class library TickteClick.Models

Needs referende to Microsoft.AspNetCore.Identity.EntityFrameworkCore

By adding **GUID** in the above , we make sure the generated class has primary key as Guid.

Add the identity related services in the Startup configure

services.AddIdentity<TicketClickUser, IdentityRole<Guid>>()

.AddEntityFrameworkStores<TicketClickContext,Guid>();

services.AddDbContext<TicketClickContext>();

Add the Identity Middleware

app.UseIdentity();

Claims Overview from Roland Guit PS course

