dotnet cli cheat sheet ❗🚀 https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet

To display information about installed SDKs and Runtimes.

```
$ dotnet --info
```

To create an empty solution. If you don't provide the name of the solution it will use directory's name instead and create a solution file.

```
$ dotnet new sln
```

Create a new WebApi template project and output in to api folder, -o parameter is used to specify the output directory name.

```
$ dotnet new webapi -o api
```

Create add reference.

```
$ dotnet add reference ..\eExpIdentity\
```

Create a new Class library template project and output in to api folder, -o parameter is used to specify the output directory name.

```
$ dotnet new classlib -o eExpIdentity
```

Add the project to the solution. If you just specify the name of the folder it will go and find all the projects inside that folder and add them to the solution.

```
$  dotnet sln add .\eExpIdentity\
```

It will list all the projects inside the solution. You must be in the root folder where .sln file exists.

```
$ dotnet sln list
```

Create Console APP

```
dotnet new globaljson --sdk-version 5.0.403

md ConsoleApp

dotnet new sln -n ConsoleApp -o ConsoleApp

cd ConsoleApp

md src

dotnet new console -n ConsoleApp -o src/ConsoleApp

dotnet sln add src/consoleapp/consoleapp.csproj

dotnet build

dotnet run --project src/consoleapp
```

Create MVC APP

```
md MvcApp

dotnet new sln -n MvcApp -o MvcApp

cd MvcApp

md src

dotnet new mvc -uld --auth individual -n MvcApp -o src/MvcApp

dotnet sln add src/MvcApp/MvcApp.csproj

dotnet build

dotnet run --project src/MvcApp/MvcApp.csproj
```

Create API APP

```
md WebApi

dotnet new sln -n WebApi -o WebApi

cd WebApi

md src

dotnet new webapi -n WebApi -o src/WebApi --use-program-main true

dotnet sln add src/WebApi/WebApi.csproj

dotnet build

dotnet run --project src/WebApi/WebApi.csproj
```

To open Visual Studio Code in the current directory.

```
$ code .
```

Go to your Api folder and run the api project

```
$ cd api
$ dotnet run
```

It will keep an eye on the files changed and automatically recompile the app.

```
$ dotnet watch run
```

It will keep an eye on the files changed and automatically recompile the app.

```
$ dotnet restore
```

Will generate a cert for you or if it is already present it will confirm it’s presence but that does not mean that your OS trusts the cert.

```
$ dotnet dev-certs https
```

On windows it will show a warning message and if you select Yes it will entrust the self signed cert.

```
$ dotnet dev-certs https -t
```

Delete the local cert for development – you will be asked for confirmation.

```
$ dotnet dev-certs https --clean
```

Create and trust new cert.

```
$ dotnet dev-certs https -t
```

🛢 Generate & Run Migrations

```
$ dotnet --info
$ dotnet tool install --global dotnet-ef --version 3.1.9

$ dotnet ef -h

$ dotnet ef migrations add InitialCreate -o Data/Migrations

$ dotnet ef database update

$ dotnet ef migrations remove -p Infrastructure -s api


$ dotnet ef migrations add InitialiCreate -p Infrastructure -s api -o Data/Migrations


$ Add-Migration InitialMigration -Context IdentityContext -o Migrations

$  Update-Database -Context IdentityContext
$  Remove-Migration -Context IdentityContext
by visual studio
```

🛢 Test, Build and Release

```
$ dotnet build --configuration Debug

$ dotnet publish --no-build --configuration Debug --output /folder/of/your/choice
$ dotnet test --configuration Release --no-build
```

```
$ dotnet test Tailspin.SpaceGame.Web.Tests --configuration Release --no-build --
logger trx
```

🛢 publish application

```
First install dotnet-hosting-5.0.8-win.exe

$ dotnet publish

$ dotnet publish healthAPI.sln.sln

$ dotnet publish --no-build --configuration Debug --output /folder/of/your/choice
```

🛢 publish Angular Application in IIS

```
$ ng build --prod --base-href /expAppTest/

$ ng build --prod --base-href /healthApp/ --aot=false --build-optimizer=false
```

🛢 use Database first

use SQL Server.

```
$ dotnet ef dbcontext scaffold
"server=10.211.55.14\\MyInstance,1433;Database=BioMedEquipmentDB;User
ID=sa;Password=20100;" Microsoft.EntityFrameworkCore.SqlServer -o Models
```

use Orcale Database 12.

```
$ dotnet ef dbcontext Scaffold "Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.8.146)(PORT=1521))(CONNECT_D
ATA=(SERVER=DEDICATED)(SERVICE_NAME=ExpSysv1)));User
ID=ExpAppUserss;Password=Expssaaew;Persist Security Info=True"
Oracle.EntityFrameworkCore  -o Models -f
```

install angular 11.

```
$ npm i -g  @angular/cli@11.2.13
```

run angular 11 in https.

```
$ ng s -o ---ssl true
```