CMD Command and dependecies Steps

Made with VisualParadigm
For non-commercial use

1)Commands

***Install Angular CLI**: npm install -g@angular/cli

***Create New project:** ng new projectName

***Run Project :** ng serve -o

***Install Angular Material:**ng add@angular/material

***Install flexbox to UI application** : npm i flexboxgrid --save

***Generate component:** ng g c top-nav

***Generate Service**: ng g s servicename

-2/Extensions

*Angular Essentials

FrontEnd

3/ Site Web:
*flexboxgrid.com
***Material design:**
https://material.angular.io/components/categories
4/ Term Definition:

```
<div class="mt-1 mat-elevation-z8
```

```
private readonly route:ActivatedRoute,
```

***HttpClient:**
Performs HTTP requests. This service is available as an injectable class,with methods to perform HTTP requests. Each request method has multiple signatures,and the return type varies based on the signature that is called (mainly the values of observe and responseType).

***@ViewChild:Property decorator that configures a view query. The change detector looks for the first element or the directive matching the selector in the view DOM. If the view DOM changes, and a new child matches the selector, the property is updated**
***paramMap:An Observable that contains a map of the query parameters available to all routes.**
The map supports retrieving single and multiple values from the query parameter.
***subscribe():subscribe` is not a regular operator, but a method that calls Observable's internal `subscribe` function. It might be for example a function that you passed to Observable's constructor, but most of the time it is a library implementation, which defines what will be emitted by an Observable, and when it be will emitted.**

***Css Flex-Box Properties:**

```
<div class="row end-xs" >
```

```
(keyup)="filterStudents()"
```

```
filterStudents(){
```

```
this.dataSource.filter=this.filterString.trim().toLowerCas
```

***NgModel(): Creates a FormControl instance from a domain modeland binds it to a form control element.**

```
{path:'students/:id',
```

```
component:ViewStudentComponent}
```

***Observable:**
A representation of any set of values over any amount of time. This is the most basic building block of RxJS.
***ngOnInit():**
A callback method that is invoked immediately after the default change detector has checked the directive's data-bound properties for the first time, and before any of the view or content children have been checked. It is invoked only once when the directive is instantiated.
***ActivatedRoute:**
Provides access to information about a route associated with a component that is loaded in an outlet. Use to traverse the RouterState tree and extract information from nodes.
***paramMap:**
An Observable that contains a map of the required and optional parameters specific to the route. The map supports retrieving single and multiple values from the same parameter.
***Template variables :**
help you use data from one part of a template in another part of the template.
Use template variables to perform tasks such as respond to user input or finely tune your application's forms.

***datepicker toggle button:**
gives the user an easy way to open the datepicker pop-up

***// check is null or empty**
if (genderList == null || !genderList.Any())

```
<mat-option *ngFor="let gender of genderList"
```

```
[value]="gender.id">
```

```
private snackbar:MatSnackBar,
```

```
setTimeout(()=>
```

```
{this.router.navigateByUrl('');
```

```
},2000
```

2 seondes

===filter on data table

```
<input type="text"
class="search-input"
placeholder="Search Students"
[(ngModel)]="filterString"
(keyup)="filterStudents()"
>
```

```
filterString='';
filterStudents(){
this.dataSource.filter=this.filterStr
ing.trim().toLowerCase();
}
```

```
<form #studentDetailsForm="ngForm" >
```

```
@ViewChild('studentDetailsForm') studentDetailsForm?:
```

```
if(this.studentDetailsForm?.form.valid )
```

***NgForm:**
Creates a top-level FormGroup instance and binds it to a form to track aggregate
form value and validation status.
As soon as you import the FormsModule, this directive becomes active by default on all
<form> tags. You don't need to add a special selector.

```
***const formData= new FormData();
```

Provides a way to easily construct a set of key/value pairs representing form fields and their values, which can then be easily sent using the XMLHttpRequest.send() method. It uses the same format a form would use if the encoding type were set to "multipart/form-data".

BackEnd

A/Dependecies installed :
1.Entity Framework:
*Microsoft.Entityframeworkcore.SqlServer
*Microsoft.Entityframeworkcore.Tools
2.**Automapper**
*AutoMapper
*AutoMapper.Extensions.Microsoft.Dependency.Injection
3. ServerSideValidation:
*FluentValidationAspNetCore

B/Migrations cmd :
Add-Migration"m1"
Update-Database

C/ Terms Definition
***AddScoped():**

Made with
Visual Paradigm
For non-commercial use

The AddScoped method registers the service with a scoped lifetime,
the lifetime of a single request. By using the DI pattern, the controller:Doesn't use the concrete type MyDependency , only the IMyDependency interface it implements

***AutoMapper is a simple library that helps us to transform one object type into another.
It is a convention-based object-to-object mapper that requires very little configuration.
The object-to-object mapping works by transforming an input object of one type into an output
object of a different type.

***Include(): Specifies related entities to include in the query results.
The navigation property to be included is specified starting
with the type of entity being queried (TEntity).

return Ok(mapper.Map<Student>(student));

***CreateMap():

Create mapping configuration form the TSource type to the TDestination type

***AfterMap:

Execute a custom mapping action after member mapping

***public async Task<IActionResult> AddStudentAsync([FromBody]AddStudentRequest request):

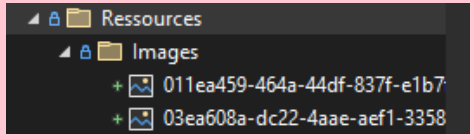
specifies that a parameter or property should be bound using the reequest body

***IFormFile profileImage:

represent a file sent with the HttpRequest

***path:

class System.IO.Path
Performs operations on string instances that contain file or directory path information. These operations are performed in a cross-platform manner.



***Directory:

Exposes static methods for creating, moving, and enumerating through directories and subdirectories. This class cannot be inherited.

```
public async Task<string> Upload(IFormFile file,
    string fileName)
{
    var filePath=Path.Combine(Directory.
        GetCurrentDirectory(),
        @"Ressources\Images",fileName);
    using Stream fileStream = new FileStream
        (filePath, FileMode.Create);
    await file.CopyToAsync(fileStream);
    return GetServerRelativePath(fileName);
}
private string GetServerRelativePath(string fileName)
{
    return Path.Combine(@"Ressources\Images", fileName);
}
```

***GetCurrentDirectory()

Gets the current working directory of the application.

Stream:

Provides a generic view of a sequence of bytes. This is an abstract class.

FileMode:

Create:

Specifies how the operating system should open a file.

Specifies that the operating system should create a new file. If the file already exists, it will be overwritten. This System.Security.Permissions.FileIOPermissionAccess.Write permission. FileMode.Create is equivalent to requ otherwise, use FileMode.Truncate. If the file already exists but is a hidden file, an UnauthorizedAccessExceptio-

CopyToAsync

Asynchronously copies the contents of the uploaded file to the target stream.

```
app.UseStaticFiles(new StaticFileOptions
{
    FileProvider = new PhysicalFileProvider
    (Path.Combine(app.Environment.ContentRootPath,
    "Ressources")),
    RequestPath = "/Ressources"
});
```

PhysicalFileProvider:

PhysicalFileProvider.PhysicalFileProvider(string root) (+ 1 overload)
Initializes a new instance of a PhysicalFileProvider at the given root

Made with Visual Paradigm
For non-commercial use

Made with
Visual Paradigm
For non-commercial use

directory.

ContentRootPath:

```
string IHostEnvironment.ContentRootPath { get; set; }
Gets or sets the absolute path to the directory that contains the application content files.

'ContentRootPath' is not null here.
```

var extension = Path.GetExtension(profileImage.FileName);

```
Returns:
The extension of the specified path (including the period "."), or null, or string.Empty. If path is null, Path.GetExtension(string) returns null. If path does not have extension information, Path.GetExtension(string) returns string.Empty.
```

Made with Visual Paradigm
For non-commercial use

Made with
Visual Paradigm
For non-commercial use