https://github.com/TryCatchLearn/DatingApp

DotNet> dotnet watch run

EntityFrameworkCore.Sqlite & Tools & EntityFrameworkCore.Design > nuget

Connectionstring> "ConnectionStrings" : { "DefaultConnection" : "Data source=datingapp.db"}

Code first Add db script> Search for dotnet-ef in nuget.org

Use matching entity framework

Migrations> dotnet ef migrations add InitialCreate -o Data/Migrations

Update> dotnet ef database update

Node> nvm install 14.17.5

Angular> npm install @angular/cli

Bootstrap> ng add ngx-bootstrap

Font-awsome> npm install font-awesome

For ssl (https certificate)

Generate key for a certificate

Add ssl folder inside client app

Add server.cert and server.key in the ssl folder

Go to angular.json and inside “serve” object, add

“options”:{

“sslKey”: “./ssl/server.key”,

“sslCert”: “./ssl/server.crt”,

“ssl”: true,

“browserTarget”: “test:build”

}

To debug using VS code:

Check for launch.json and tasks.json which are created when we generate assets for build and debug

dotnet ef database drop

dotnet ef database update

System.identymodel.tokens.jwt> nuget

HMac (Symmetric algorithm)> Hash-Based Message Authentication Codes like SHA256, SHA512 (Secure Hashing Algorithm)

Authentication> nuget> microsoft.aspnetcore.authentication.jwtbearer

Angular:

ng g -h> ng g c nav --skip-tests

getboostrap.com > examples> carousel

ng g s account –skip-tests

dropdowns> ngx-bootstrap

RxJS(Reactive extension for Java Script) > map, pipe, redue

Error handling toaster notification service> npm install ngx-toastr

"./node\_modules/font-awesome/css/font-awesome.css",

"./node\_modules/ngx-toastr/toastr.css",

Add the above inside angular.json “styles”: []

Route guards> ng g guard auth --skip-tests> canactivate

AuthGuard will subscribe to ReplySubject Observable

Theams> bootswatch.com

* Npm install bootswatch

New module>mkdir \_modules>ng g module shared –flat

Router>

<router-outlet></router-outlet>

<routerLink="/members" routerLinkActive="active">

const routes: Routes = [

{ path: '', component: HomeComponent },

{

path: '',

runGuardsAndResolvers: 'always',

canActivate: [AuthGuard],

children: [

{ path: 'members', component: MemberListComponent },

{ path: 'members/:id', component: MemberDetailComponent },

{ path: 'lists', component: ListsComponent },

{ path: 'messages', component: MessagesComponent },

{ path: '\*\*', component: HomeComponent, pathMatch: 'full' },

]}];

this.router.navigateByUrl('/members');

interseptors>api> mkdir \_interceptors> ng g interceptor --skip-tests

api> dotnet ef migrations add extendeduserentity

dotnet ef migrations remove

fully defined relationship

google>json generator

Repository and Unit of work patterns

DTO will avoid:

"message": "A possible object cycle was detected. This can either be due to a cycle or if the object depth is larger than the maximum allowed depth of 32. Consider using ReferenceHandler.Preserve on JsonSerializerOptions to support cycles.",

AutoMapper> nuget

TypeScript:

let data: number | string = 42;

data = “str”;

const multiplay = (x,y) => { x \* y; }

const multiplay = (x,y) => { return x \* y; }

const multiplay = (x,y) : int => { return x \* y; }

Json to ts

Google> angular bootstrap> valor-software.com

Photos google>Ngx-gallery> npm install @kolkov/ngx-gallery

Guard> ng g guard prevent-unsaved-changes --skip-tests

@ViewChild(‘editForm’) editForm: NgForm

This.editForm?.reset(this.member)

@HostListener(‘window:beforeunload’, [‘$event’]) unloadNotification($event: any){}

Loader> google> ngx-spinner> ng add ngx-spinner

Cloudinary>

Google> ng2-file-upload> npm install ng2-file-upload

Form Controls in another component> export class TextInputComponent implements ControlValueAccessor {

response.Headers.Add("Pagination", JsonSerializer.Serialize(paginationHeader));

response.Headers.Add("Access-Control-Expose-Headers", "Pagination");

time age> google> ngx-timeago

npm install ngx-timeago

directives> ng g d name --skip-tests

Section 16:

1. Using .Net Identity
2. Role management
3. Policy based authorization
4. UserManager<T>
5. SinInManager<T>
6. RloeManager<T>

SingleR

1. WebSockets
2. Server-Sent events
3. Long poling

Client> npm install @icrosoft/signalr