# فهرست مطالب

[فهرست مطالب 1](#_Toc135719117)

[مقدمه: 2](#_Toc135719118)

[تعریف مسئله 3](#_Toc135719119)

[راهکارهای حل مسئله:(نیاز سنجی) 3](#_Toc135719120)

[1-نحوه ی Authentication 3](#_Toc135719121)

[2- Entity های مورد نیاز براساس مسئله 3](#_Toc135719122)

[فیلد های مورد نیاز برای یوزر به شرح ذیل خواهد بود: 3](#_Toc135719123)

[ ID:long 3](#_Toc135719124)

[ Name:string(255) 3](#_Toc135719125)

[ LastName:string(255) 3](#_Toc135719126)

[ Email:string(500) 3](#_Toc135719127)

[2-UserRelations: 4](#_Toc135719128)

[درباره پروژه: 6](#_Toc135719129)

[فناوری های مورد استفاده در این پروژه: 6](#_Toc135719130)

[توضیحات مربوط به لایه ها و انواع ساخته شده: 7](#_Toc135719131)

[تغییرات مورد نیاز در آینده برای بهتر کردن پروژه: 13](#_Toc135719132)

[نحوه ی استفاده از برنامه: 14](#_Toc135719133)

[صفحه ی ورود و ثبت نام: 14](#_Toc135719134)

[ثبت نام: 14](#_Toc135719135)

[ورود به صفحه اصلی: 14](#_Toc135719136)

[نمای صفحه ی اصلی: 15](#_Toc135719137)

# مقدمه:

با سلام و احترام خدمت خوانندگان عزیز این داکیومنت ،لازم به ذکر است ریز توضیحات کدها در خود سورس کد وجود دارد و دراینجا به کلیت نحوه ی انجام کار متدهای مهم و نوع های مورد استفاده اشاره شده است. و در کد نویسی سعی بر این بوده که خود کد و توضیحات آن تمام نیازهای خواننده را رفع نمائید یا به عبارتی Self Description باشد.

# تعریف مسئله

ایجاد یک Social network API با ویژگی های زیر

* ایجاد یک API بر پایه Asp.net core
  + ایجاد امکان لاگین و ریجستر کردن برای یوزر ها
  + امکان مشاهده لیست همه ی یوزر ها و جست و جو براساس نام
  + امکان ایجاد موتال فرند و درخواست دوستی و ارسال پیام بعد از پذیرفتن یک نفر
  + ایجاد JWT و احراز هویت براساس آن
  + استفاده از swagger به جهت داکیومنت سازی و تست API ها
* موارد مهمی که در این تسک مورد نظر است:
  + دیزاین دیتابیس و جلوگیری از افزونگی در آن
  + طراحی کوئری ها با بالاترین راندمان
  + جزئیات طراحی آن به عهده برنامه نویس است

# راهکارهای حل مسئله:(نیاز سنجی)

## 1-نحوه ی Authentication

با استفاده ازJWT وایجاد توکن برای هر فرد نسبت به احراز هویت و کنترل دسترسی اقدام میشود.

### 2- Entity های مورد نیاز براساس مسئله

#### 1-User:

### فیلد های مورد نیاز برای یوزر به شرح ذیل خواهد بود:

### ID:long

### Name:string(255)

### LastName:string(255)

### Email:string(500)

* Birthday:DateTime
* Password:String
* About:String
* profilePicture:String
* CreationDate:DateTimeOffset

نکته:Emial به عنوان نام کاربری در نظر گرفته خواهد شد.

#### رفتارهای مورد نیاز برای یوزر:

* ایجاد یوزر

### 2-UserRelations:

این یک Value object برای user میباشد

#### فیلد های مورد نیاز به شرح ذیل است:

* UserA:User
* UserB:User
* Approve:bool
* RelationRequestMessage

نکته: با استفاده از یک رابطه چند به چند به یک انتیتی (Self-relation many-to-many)از مدل یوزر ها ایجاد میشود.

#### رفتارهای مورد نیاز برای دوستی ها(UserRelations)

* ارسال درخواست دوستی
* قبول درخواست دوستی
* امکان ارسال پیام به همراه درخواست

3-Message

این مورد نیز یک value object برای user میباشد

#### فیلد های مورد نیاز به شرح ذیل است:

* FromUser:User
* ToUser:User
* Message:string
* Like:bool
* IsRead:bool

نکته :این مدل از رابطه چند به چند با یک مدل (مدل User ) به وجود می آید

#### رفتارهای مورد نیاز برای پیام(Message)

* ارسال پیام
* لایک پیام
* آن لایک پیام
* تغییر به عنوان خوانده شده
* قابلیت ویرایش پیام تا 3 دقیقه

# درباره پروژه:

## فناوری های مورد استفاده در این پروژه:

|  |  |
| --- | --- |
| **توضیحات** | **فناوری یا کتابخانه** |
| **زبان برنامه نویسی** | **C#** |
| تکنولوژی (فریمورک) اصلی مورد استفاده | .Net Core 6 |
| نوع معماری مورد استفاده | Onion Architecture |
| لایه های مورد استفاده | Domain,Aplication,Infrastructure,presentation |
| لایه های کمکی | 00\_Framework |
| بانک اطلاعاتی مورد استفاده | SQL Server 2019 |
| فناوری دسترسی به داده | EFCore Code-First v 6.0.19 |
| تزریق وابستگی‌ها | Built-In IOC Container |
| سایر تکنولوژی های مورد استفاده | Swagger,JWT,JWE |
| زبان مورد استفاده سمت کلاینت | .net core 6 web Api,Swagger generator |
| لایه تست | TDD,Xunit,Moq,FluentAssertation |

## توضیحات مربوط به لایه ها و انواع ساخته شده:

|  |  |
| --- | --- |
| **نام نوع** | **توضیحات کوتاه** |
| لایبرری 00\_Framework | محلی برای ایجاد ابزار هایی که در سراسر برنامه استفاده میشود-به نوعی ایجاد ابزار های عمومی(علت جدا کردن این بخش امکان انتقال آن به سایر پروژه ها و استفاده آسان) |
| کلاس OperationResult | زبان مشترک برای برگرداندن نتیجه ی یک Operation |
| کلاس IAuthHelper | انتراع مربوط به کلاس AuthHelperکمک برای Authencitate یک یوزر و احراز هویت |
| کلاس AuthHelper | کمک برای Authencitate یک یوزر |
| کلاس AuthViewModel | مدل کمکی برای عملیات Authenticate |
| کلاس HashingOptions | کلاس کمکی برای عملیات رمز نگاری پسورد |
| کلاس IPasswordHasher | اینترفیس انتزاع رمزنگار پسورد |
| کلاس PasswordHasher | پیاده سازی رمزنگار پسورد |
| Enum RequestStatus | جهت کمک به مشخص نمودن وضعیت ارتباط بین دو یوزر |
| کلاس ValidatingMessage | پیام های اماده جهت نمایش در بخش اعتبار سنجی سمت کلاینت |
| کلاسApplicationMessage | پیام های آماده برای استفاده در بخش Application |
| کلاس EntityBase | جلوگیری از تکرار پراپرتی های مورد نیاز همه ی Entity ها |
| لایبرری SocialNetwork.Domain | لایه دامین برای نگهداری Entity ها و منطق مورد نیاز برای آنها |
| کلاس User | مربوط به Entity کاربر و منطق آن |
| کلاس Message | مربوط به Entity پیام و منطق آن |
| کلاس UserRelations | مربوط به Entity ارتباط های دوستانه و منطق آن |
| لایه Application | به دو لایبرری تقسیم میشود که یکی از آنها انتزاعات را در بر دارد و رابطه مستقیم با لایه دامین دارد و دیگری شامل پیاده سازی های این لایه میباشد که بر روی لایه ی قبلی قرار میگیرد و وظیفه رهبری عملیات های برنامه است و هیچگونه منطقی در آن جریان ندارد |
| لایبرری SocialNetwork.Application.Contracts | این لایه مربوط به انتزاعات برنامه در بخش اپلیکیشن است و وظیفه آن راهبری و استفاده از سایر بخش ها جهت انجام امور محوله است.تمامی تعاریف مدل ها و انتزاعات مربوط به لایه اپلیکیشن در این قسمت است مانند مدل های ساخت یوزر و تغییر اطلاعات یوزر و ... |
| لایبرری SocialNetwork.Application | همان طور که در بالا اشاره شد این بخش مسئول پیاده سازی های مربوط به بخش Application.Contract است. |
| لایه Infrastructure | این لایه تمام ارتباطات زیرساختی مانند دیتابیس و تکنولوژی های آن به همراه تزریق وابستگی های آن سیستم را برعهده دارد-که از دو لایبرری تشکیل شده است. |
| لایبرری CurrencyConvertor.Infrastructure.EfCore | لایه مربوط به دیتابس و ارتباط با آن و مدل کردن و انجام تنظیمات ان |
| بخش Mapping | پیکربندی جداول و نوع های آن را برعهده دارد-به عبارتی مسئولیت مپ کردن هر Entity به جدول به دوش این بخش است |
| کلاس SocialNetworkContext | این کلاس مسئول ایجاد و مدل کردن دیتابیس است و به نوعی دیتابیس ما حساب میشود |
| لایبرری SocialNetworkApi.Infrastructure.Bootstrapper | ایجاد تنظیمات و تزریق وابستگی های این سیستم(WireUp)-ارتباط با سایر سیستم ها(دراینجا سیستم دیگری نداریم) |
| Presentation لایه | این لایه بیرونی ترین لایه برنامه است که مسئول انجام امور مربوط به کلاینت (ساخت UI) و هاست کردن سرویس های مورد نیاز و کنترل درخواست ها ی مختلف است. |
| لایبرری SocialNetworkApi.Presentation.WebApi | این لایه نمای بیرونی برنامه ما است که مدخل برنامه نیز میباشد و با استفاده از swagger میتوان API ها را تست نمود هم چنین کنترلر های ما نیز در اینجا پیاده سازی شده است |
| پوشه Tools | یکسری پیاده سازی هایی که فقط مختص به این برنامه و فقط با hosting environment کار میکند. |
| لایبرری 01\_Test.SocialNetworkApi | تست های ما در این لایه میباشد و بر پایه TDD و با استفاده از XUNIT میباشد |

# تغییرات مورد نیاز در آینده برای بهتر کردن پروژه:

1-استفاده از معماری CleanArchitecture

2-استفاده از پکیج AutoMapper جهت مپ کردن های مختلف

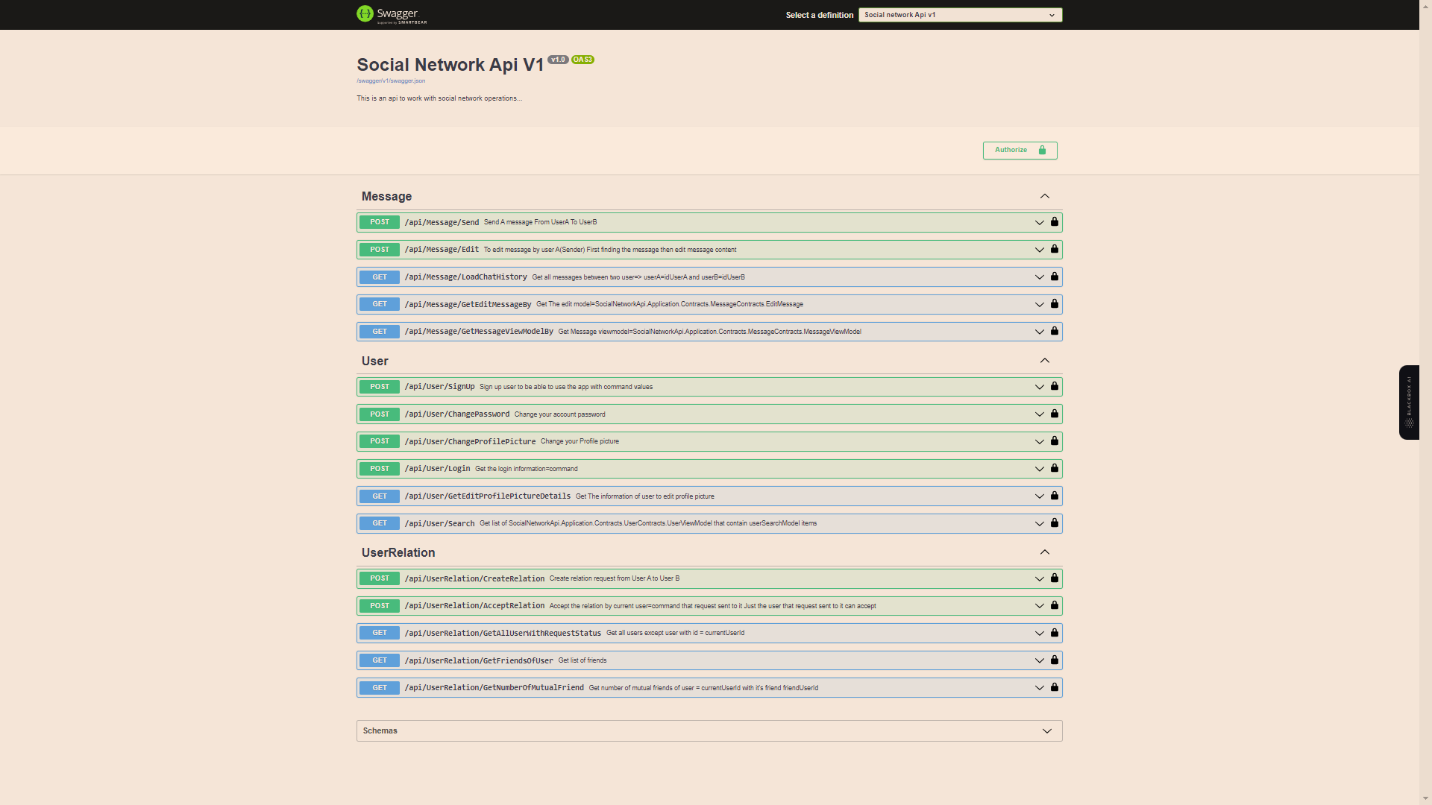
3-استفاده از پکیج MediatR جهت پیاده سازی CQRS و هم چنین استفاده از دیزاین پترن Mediator برای حل این مسئله

4-اضافه نمودن قابلیت لایک توسط گیرنده پیام

5-اضافه نمودن قابلیت خوانده شده(توسط گیرنده) به پیام ها

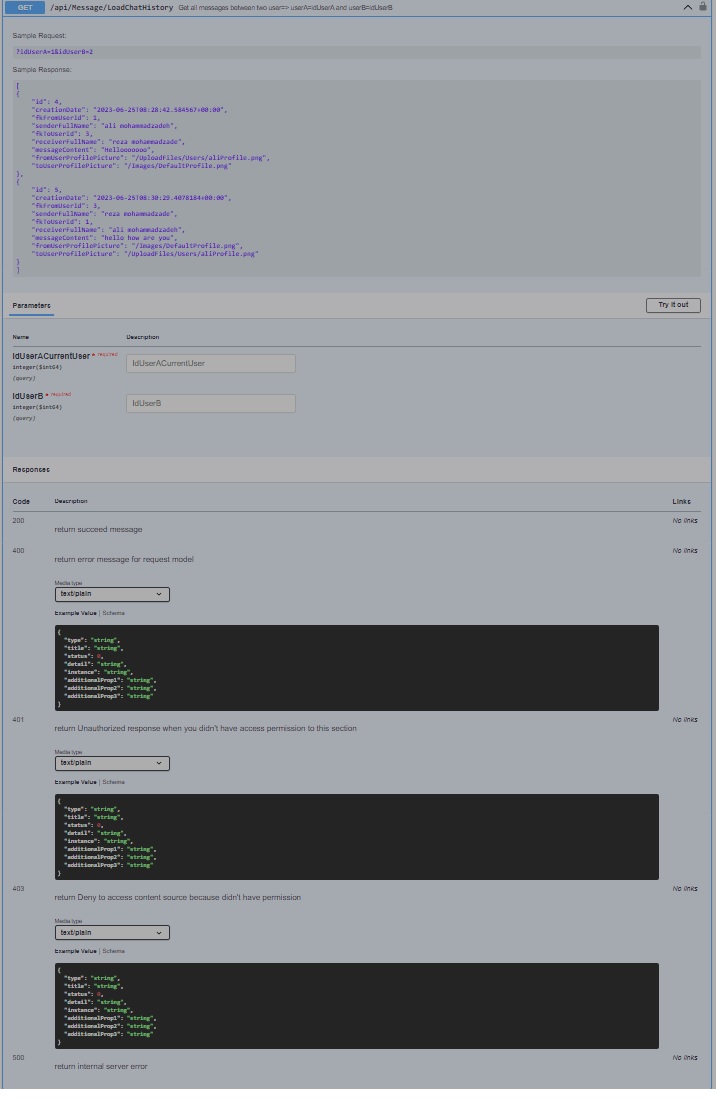
# نحوه ی استفاده از برنامه:

## نمای برنامه:



طبق تصویر صفحه ی لندینگ API ما این مورد میباشد که توضیحات هر API در جلوی آن به همراه آدرس API نوشته شده است.

در درون هر API در صورت امکان یک مثال نوع درخواست و یک نمونه response ارسال شده از سوی سرور قرارداده شده است.هم چنین در انتهای هر API کدهایی که توسط API بازگردانده میشوند (status code) آورده شده است. که در تصویر زیر آن را مشاهده مینمائید.



### ثبت نام:

برای ثبت نام وارد Api مربوط به sign up میشویم و اطلاعات را به صورت یک مدل JSON ارسال مینمائیم.و اگر مشکلی پیش نیاید یک کد از طرف سرور به همراه پیام "عملیات با موفقیت انجام گردید" ارسال خواهد گردید.

نکته:تمامی API های این برنامه به جز Sign up و Login نیاز به Authurize دارد.

### نحوه ی ورود:

پس از ثبت نام موفقیت امیز شما میتوانید با استفاده از نام کاربری (همان ایمیل وارد شده ) و همچنین پسورد خود از طریق Login وارد شوید، اگر نام کاربری و پسورد خود را به درستی وارد نمائید یک JWE token برای شما ساخته و ارسال میگردد.لازم به ذکر است تاریخ انقضای این توکن 180 دقیقه میباشد.که با استفاده از دکمه Authurize که در بالای صفحه تعبیه شده است به همراه کلمه "Bearer" قبل از توکن دریافتی عملیات احراز هویت را انجام میدهید.

لازم به ذکر است این توکن باید در هدر یک درخواست http و با نام هدر Authurization قرار بگیرد تا بتوان احراز هویت را انجام داد.

در زیر یک نمونه هدر مربوط به JWT اورده شده است:

'Authorization: Bearer eyJhbGciOiJBMjU2S1ciLCJlbmMiOiJBMjU2Q0JDLUhTNTEyIiwidHlwIjoiSldUIn0.SKeh2\_rGAhiOvgTmpFbxV0uCVlPnFUgo78E9X1k8fMBOc\_xxRN96LExG-GBLmDX50aQEy41-jkruluadTQtqvt2Bm5ovnMkP.oblYWEvMRswB3dLldI1r7g.EklNvbOt0Uys1peduDQMVtfjeyrJdck4QgcQGz3KdsI448LjTpuEjJgKitLcKe53SQ69noL0U\_PENNychHhWVSTKux5hmdlTph\_yKgZDgF8A9D1ehFGJtH5HgDfAZR8S21aGNMCrQzwFE7e4w334p\_GKn5AMKJ1cw67YlVRPIWGXknDgkJIiqF1pOs7VkfKfV82holRhelhAD2XNPHkPMQWC4NRyAHYs2-W7ZWhMuR12PBdfsKN13PiQAhIXkCCZSzlxnpPBGnuuGa\_HICkjK0VnzMKKIlnjO487GiqANEf5voEdncLjYqmQClo8oQzNQpQKQ3pC5RIJqNX4Yq-4YZcmT8rSB\_u6w7ODLXY4AMOwOSZAHAj8uPQfuM6y3N3i.DuZr3MNo9LTxZKxwDbWJsGKrQ2qhhIvXZh34WqE\_1I0'

**نکته**: در صورتی که نام کاربری و رمز عبور درست نباشد یک رشته string خالی("") از سمت سرور ارسال میگردد.