# 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
* 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 از سولوشن های ساده ی پیاده سازی Authenticate و ایجاد یک Custom Authenticate

### 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

نکته: با استفاده از یک رابطه چند به چند 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 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 مربوط به لایه ها و انواع ساخته شده:

|  |  |
| --- | --- |
| **نام نوع** | **توضیحات کوتاه** |
| لایبرری 0\_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 |
| 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 |
| اینترفیس IBaseRepository | انتزاعی برای جلوگیری از تکرار کد ها جهت ایجاد عملیات مربوط به دیتابیس مثل CRUD |
| کلاس BaseRepository | یک پیاده سازی آماده جهت اینترفیس بالا برای اعمال مشترک دیتابیس |
| لایبرری SocialNetwork.Domain | لایه دامین برای نگهداری Entity ها و منطق مورد نیاز برای آنها |
| کلاس User | مربوط به Entity کاربر و منطق آن |
| اینترفیس IUserRepository | انتزاعی جهت ارتباط و مپ کردن مدل User و استفاده از آن در برنامه |
| کلاس Message | مربوط به Entity پیام و منطق آن |
| IMessageRepository اینترفیس | انتزاعی جهت ارتباط و مپ کردن مدل Message و استفاده از آن در برنامه |
| کلاس UserRelations | مربوط به Entity ارتباط های دوستانه و منطق آن |
| اینترفیس IUserRelationRepository | انتزاعی جهت ارتباط و مپ کردن مدل Message و استفاده از آن در برنامه |
| 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 |
| بخش Repositories | این بخش مسئولیت پیاده سازی های انتزاعات بخش دامین که I<Entity>Repository است را برعهده دارد |
| کلاس SocialNetworkContext | این کلاس مسئول ایجاد و مدل کردن دیتابیس است و به نوعی دیتابیس ما حساب میشود |
| لایبرری SocialNetwork.Infrastructure.Configuration | ایجاد تنظیمات و تزریق وابستگی های این سیستم(WireUp)-ارتباط با سایر سیستم ها(دراینجا سیستم دیگری نداریم) |
| ServiceHosts 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.فایل SignalrAgent.js برای موارد مورد استفاده سایت از جاوااسکریپت در نظر گرفته شده است. |
| کلاس ChatHub | این کلاس برای استفاده از SignalR ایجاد گردیده است |
| پوشه Pages | این کلاس صفحات Razor page را شامل میشود که UI حساب میشود |
| کلاس UserIdProvider | این کلاس برای تغییر User identity از name به user id میباشد |
| 01\_SocialNetworkQuery لایبرری | این لایبرری را میتوان به عنوان واسط بین Application و ServiceHost به حساب اورد که عمل کوئری گرفتن و انتقال آن به Service host را انجام میدهد |

# 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

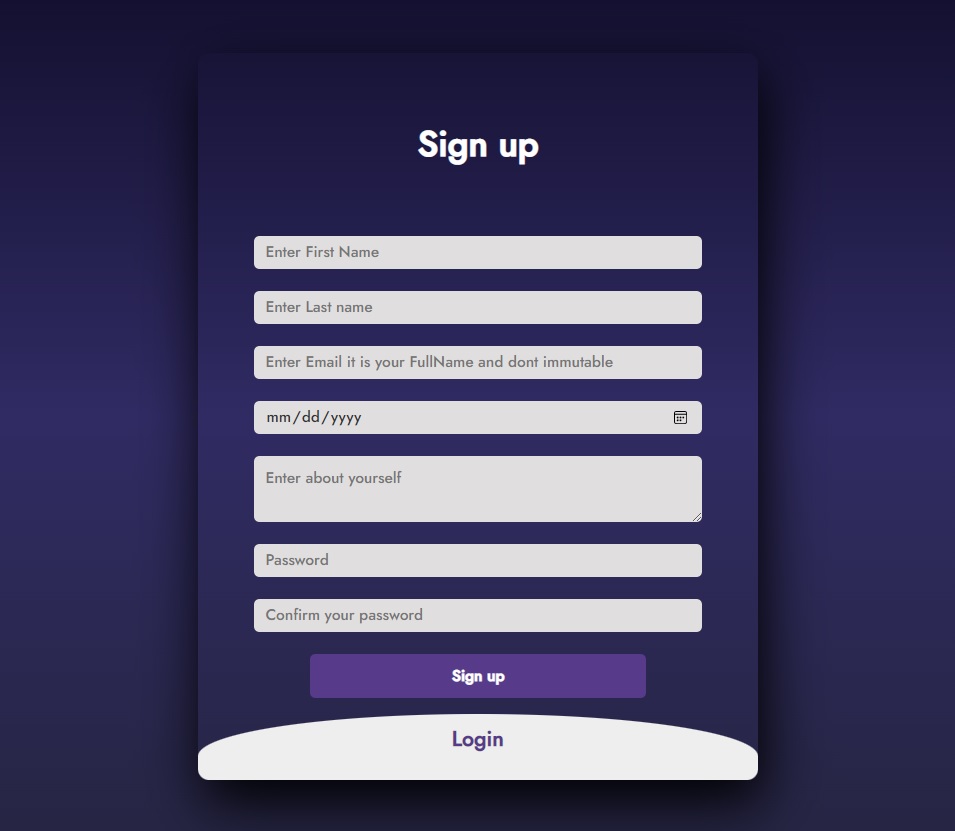
5-اضافه نمودن قابلیت خوانده شده(توسط گیرنده) به پیام ها

6-اضافه نمودن قابلیت ویرایش پروفایل

7-اضافه نمودن قابلیت انتخاب عکس پروفایل(در حال حاضر عکس پروفایل برای همه یک مورد یکسان میباشد.)

# نحوه ی استفاده از برنامه:

## صفحه ی ورود و ثبت نام:



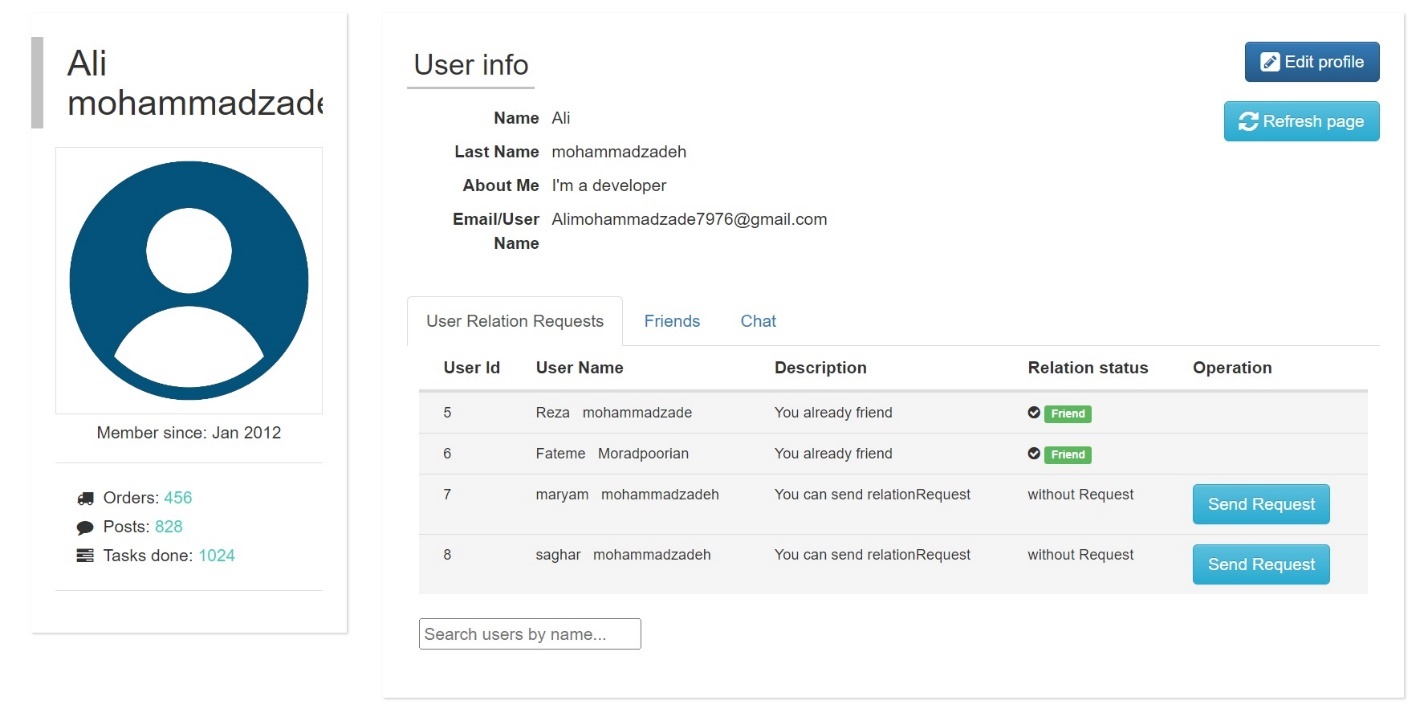
### ثبت نام:

برای ثبت نام تمامی موارد مورد نیاز را پرکرده و دکمه ی Sign up را میزنیم.

### ورود به صفحه اصلی:

دکمه لاگین پایین صفحه را میزنیم و نام کاربری(یا همان ایمیل)را وارد نموده و سپس دکمه لاگین را کلیک مینمایئم.و وراد صفحه ی اصلی میشویم.

### نمای صفحه ی اصلی:



#### قسمت User info :

همان طور که در صفحه ی بالا مشاهده میکنید میتوانید با زدن دکمه Send Request درخواست دوستی خود را به طرف مقابل ارسال نمائید.

افرادی که درخواست دوستی با آنها برقرارشده است ستون Relation status آنها friend را نشان میدهد.

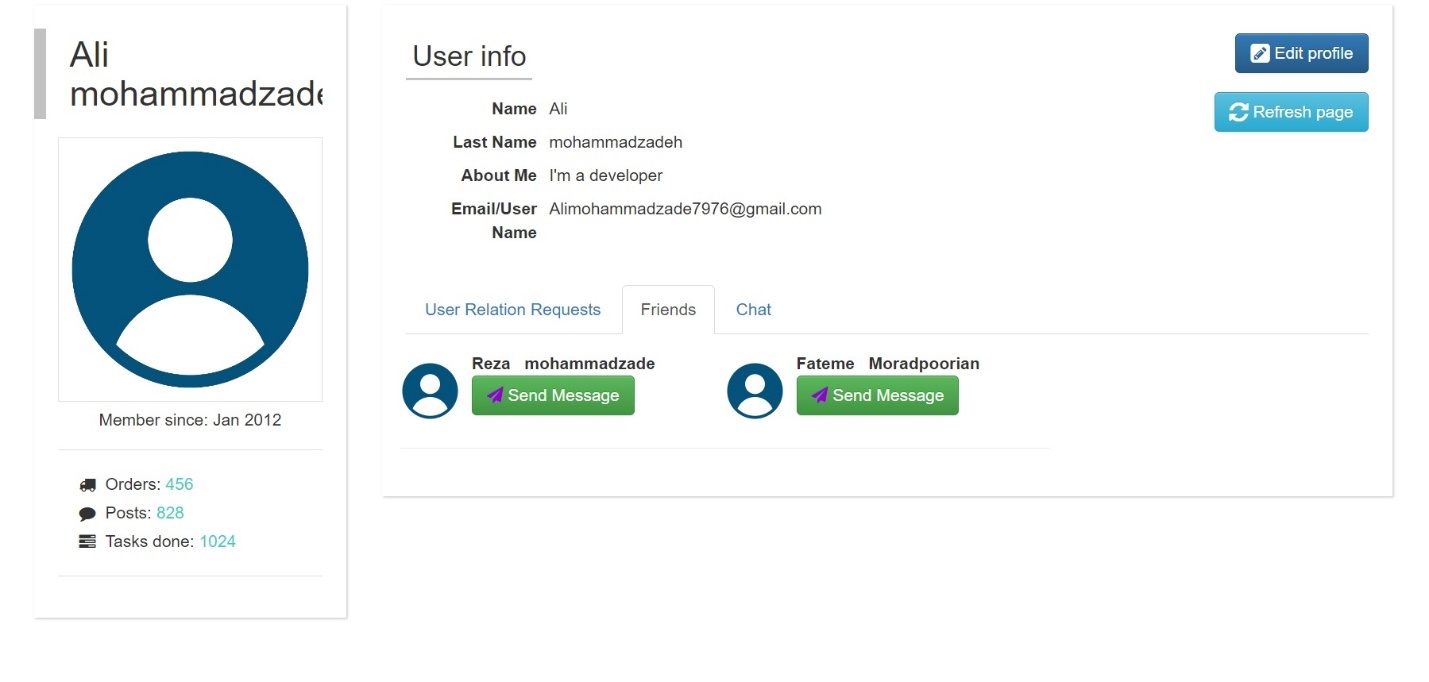
در کارد پایین در تب User Relation requests میتوانید همه ی افراد را براساس نام کاربری آنها جست و جو کنید.

دکمه ی Refresh Page در بالا میتوان کل صفحه را رفرش کرد.

هنگامی که شما به یک نفر درخواست داده باشید Relation status مساوی با pending خواهد بود.

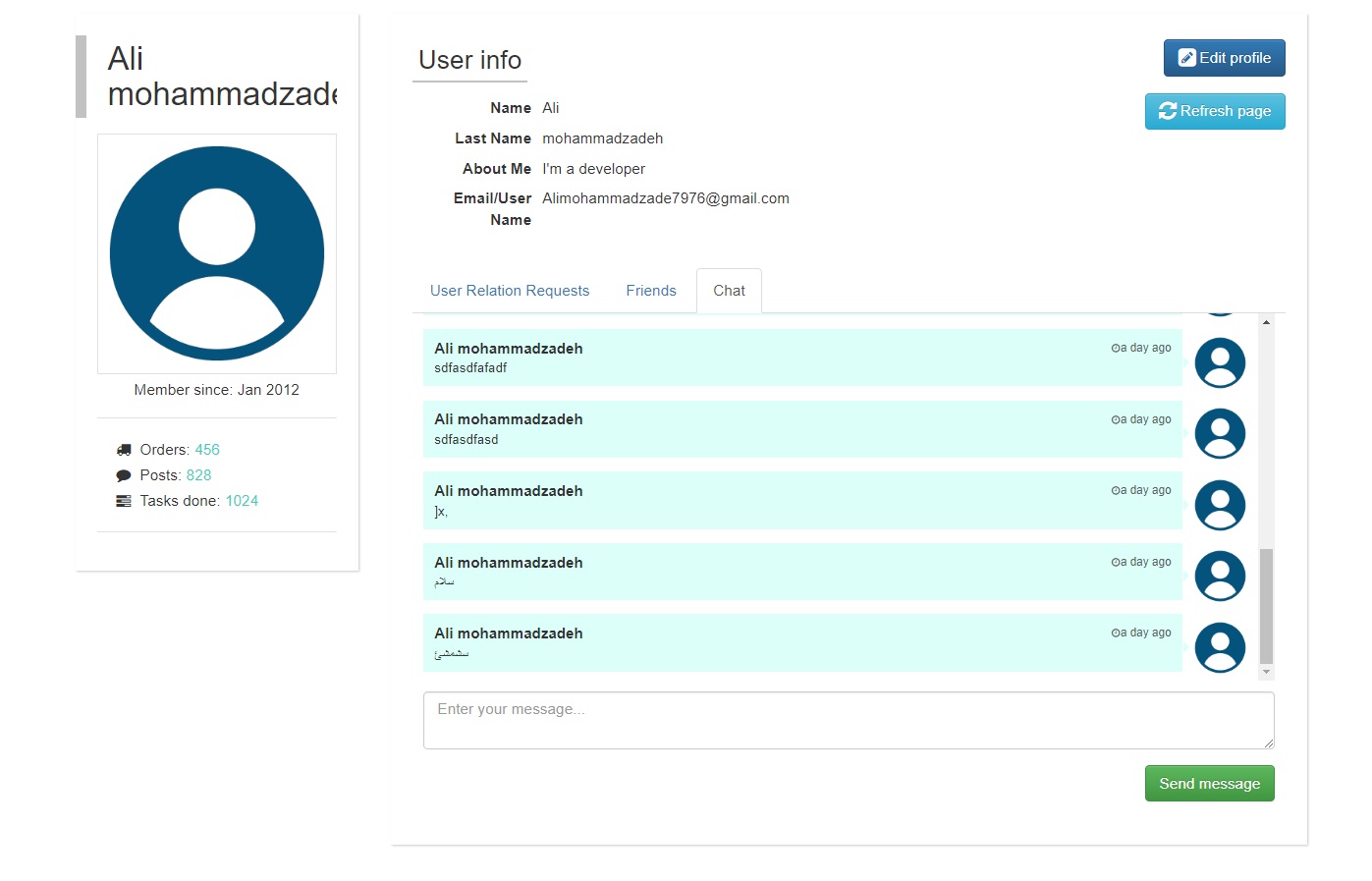
و اگر کسی برای شما درخواستی ارسال کرده باشد دکمه Accept در ستون Operation وجود خواهد داشت که میتوانید با زدن آن درخواست را بپذیرید.

#### تب Friends :



در این تب همه ی افرادی که درخواست شما را پذیرفته اند یا شما درخواست آن را پذیرفته اید وجود دارند و میتوانید با زدن دکمه Send Message به تب Chat بروید و تمامی تاریخچه چت شما لود میشود و میتوانید با دوست خود چت نمائید.

#### تب Chat :



پس از زدن دکمه ی Send Message در تب Friends به این تب منتقل شده و میتوانید متن خود را در کادر مشخص شده وارد نموده و با زدن دکمه Send message پیام مورد نظر خود را به دوست خود بفرستید.