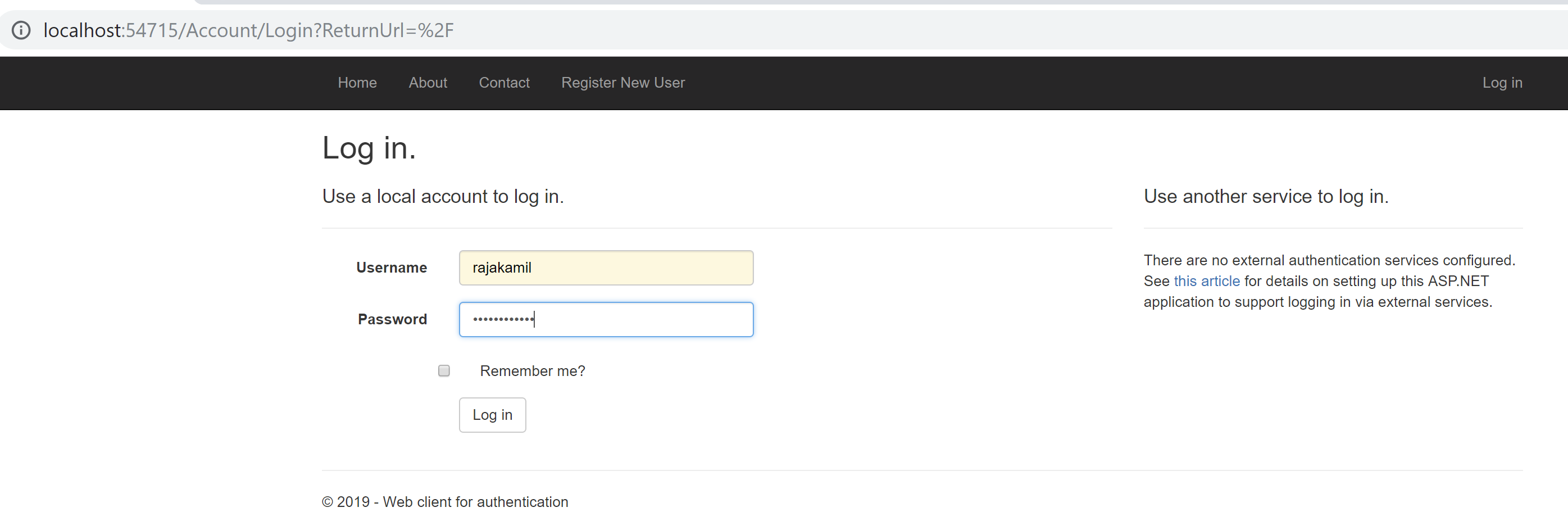
**User manual**



When you run project **SiteCore.Web.Client** from visual studio you should see as in the screenshot above. The credential you can use in this web application are as below

**Credential 1**

Username : rajakamil

Password : rajakamil123

Role : Admin, User

**Credential 2**

Username : rajakamil2

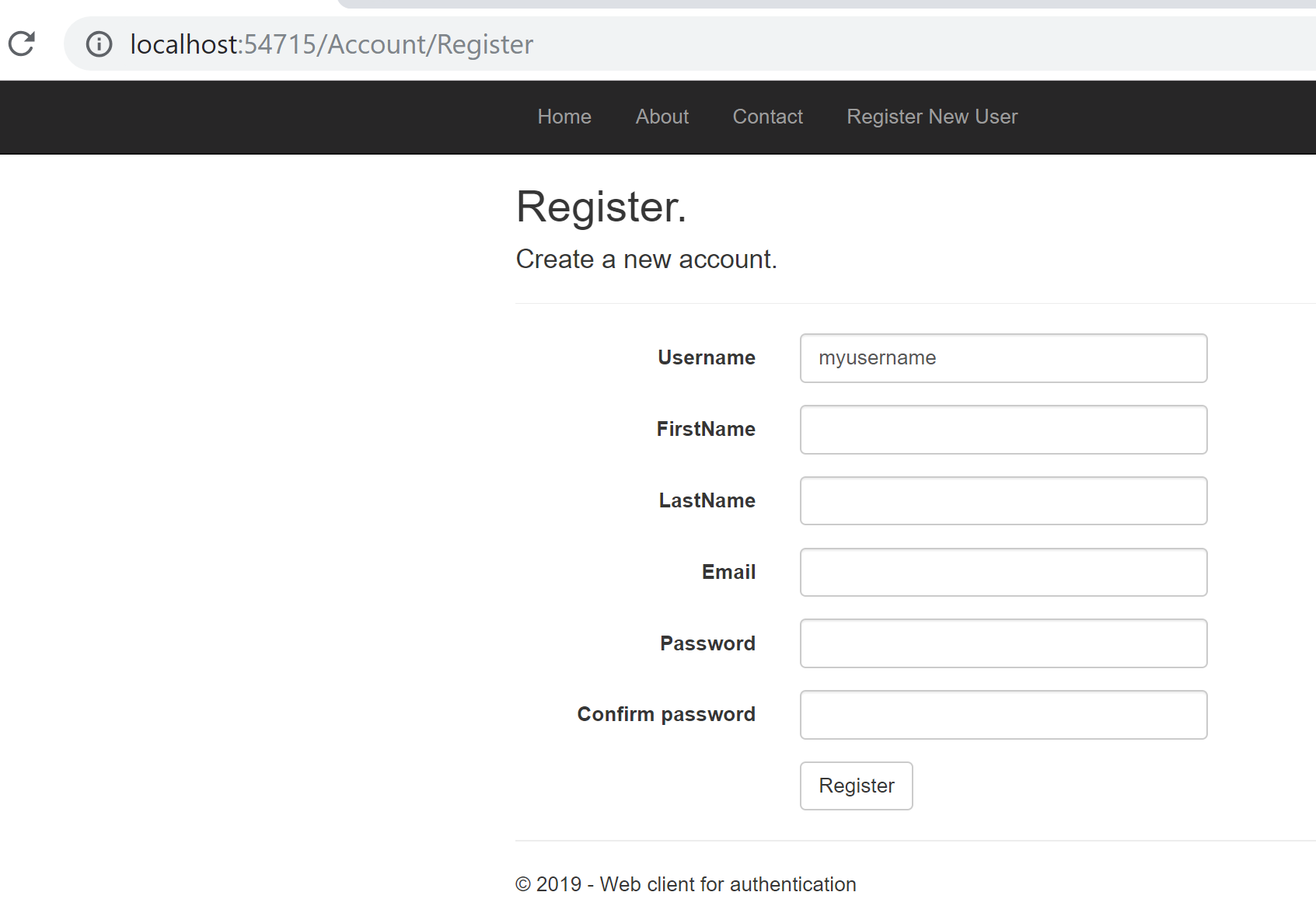
Password : rajakamil123

Role : User

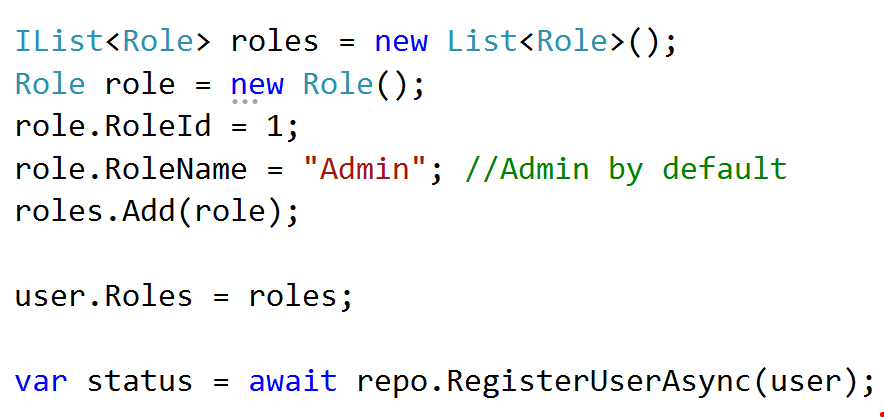
For **Admin** role permission, you can have all access including to navigate and Register New User, Contact, and About page.

But for **User** role permission, you only can navigate main page only, you are not allowed to open the Register New User, Contact, and About page because it will redirect to login page without logoff.

You can register new user if you are with admin role, as below



After fill up all these fields and click register button to register new user. Please take note that by default new user registration is by default as **Admin** role



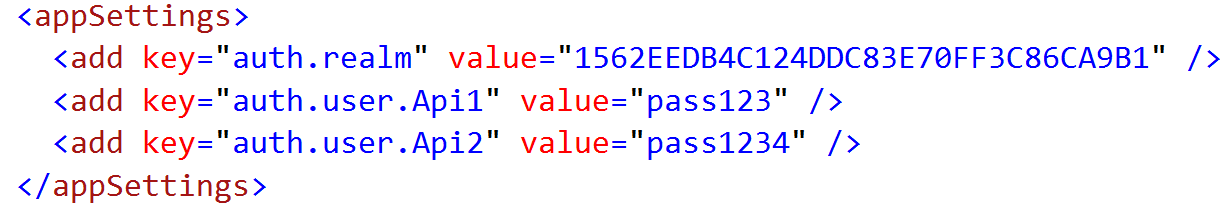
**System Description**

IDE : **Visual Studio 2017**

Database : **MS SQL Server 2012**

This system by far have 2 main function, one is for the web service, and another one is the web client, but in total have 5 layers separated in visual studio. Total amount of layer (project) are 5, SiteCore.Authentication.API, SiteCore.Web.Client, SiteCore.Domain, SiteCore.Helper, and SiteCore.Authentication.Tests to perform SOLID principles.

1) **SiteCore.Authentication.API** is the web service where the central authentication take place. Here the technology used are .NET Framework 4.5 and ASP.NET Web API as web service. To protect from anynomous access from any client to use this web service, im using basic authentication and authorization as below



Every client need to access this web service need to supply basic encoded base64 including username and password as in screenshots above in Header authorization. All this basic authentication and authorization are being handle by engine in Modules\BasicAuthHttpModule.cs

This API used MS SQL Server as a database and CRUD using normal **ado.net** and **stored procedure** to improve the performance. And to improve the performance of this web service interaction, **ASP.NET caching technique** is used to reduce the communication of CRUD with database connection when the same data require for the second time or more.

2) **SiteCore.Web.Client**. For the client web as example in this system is the project **SiteCore.Web.Client,** all the engine that need to access the API webservice need to pass header authorization before allow to access the webservice, and the engine is stored in ClientHttp\ApiRepository.cs file. For the communication with webservice I’m using RestSharp http client to communicate with Web API webservice. All the authentication and role need to communicate with web service first and pass back the relevant data to form authentication in web client, and no database involve.

3) **SiteCore.Domain**. This layer actually to separate models that been used with more than one system, no need to declare many times and improve coding efficiency that only create one model that can be used multiple times in various project and layers.

4) **SiteCore.Helper**. This layer actually to separate ResponseMessage that used as the response by API and maintain standard in response message and been used with more than one system, no need to declare many times and improve coding efficiency that only create one model that can be used multiple times in various project and layers.

5) **SiteCore.Authentication.Tests** This layer/project is to perform unit testing especially for the web service as a critical part in this application exercise. Unit Testing is crucial in avoiding future unforeseen problem during development time and reduced the unnecessary risk as in TDD best practices.