Contents

1 General information

“Identity Login System 1.0” was created in Visual Studio Community by Annice Strömberg, 2020, with Annice.se as the primary download location.

The script is a multi-user role login system based on the ASP.NET Core Identity framework. Furthermore, the user data is stored in an SQL Server database using Entity Framework (EF) Core – code first.

2 License

3 System description

“Identity Login System 1.0” is built in CSS3, HTML5, JavaScript, C# with ASP.NET Core Identity and Entity Framework Core (code first) with SQL Server as the database management system (DBMS). Furthermore, the application is built according to the model-view-controller (MVC) pattern.

4 System requirements

The script can be run on a server that supports C# 8.0 with ASP.NET Core 3.0, e.g. on Azure or on your local computer with the .NET Core 3 platform installed, along with an SQL Server supported database.

However, I will not go into any details of how you can deploy the application to cloud servers such as Azure or similar as this script was implemented and tested locally. Nevertheless, I will go through the necessary steps to run and test this script on-premises.

5 Supported features

The following functions and features are supported by this script:

- Multi-user role login system based on cookies.
- User password encryption (HMAC-SHA256).
- Password recovery function via a generated reset token link.
- Database storage of user details (via EF Core code first).
- Protection against cross-site request forgery.
- Sort and filter function of users.
- Paging function of users on the user list page.
- CRUD function of users for Admins and SuperAdmins.
- Client and server side validation.
- Responsive design.

6 Roles and permissions

In this application, a new user account has by default no roles assigned to it. However, a user account can be assigned to one or many of the following permission roles:

- LimitedAdmin
- Admin
- SuperAdmin

6.1 No roles (default)

A user with no roles yet assigned to the account will only have read permissions of other users. However, a registered user will always be able to update the own user details.

6.2 LimitedAdmin

LimitedAdmin users will have read permissions for all users and roles, but no edit permissions.

6.3 Admin

Admins will have read permissions for all users and roles. Also, the Admin users can edit LimitedAdmins, and users with no roles. However, Admins can only edit LimitedAdmins as long as they donnot also have equivalent or superior roles assigned to them as the Admins.

6.4 SuperAdmin

SuperAdmins have at all times full permissions and access throughout the entire application. One SuperAdmin can also edit another SuperAdmin.

7 ER diagram

This section illustrates the relational database and all its tables supported by the ASP.NET Identity 3.0 framework. Furthermore, the blue marked database tables in the image below are the ones used by this script.

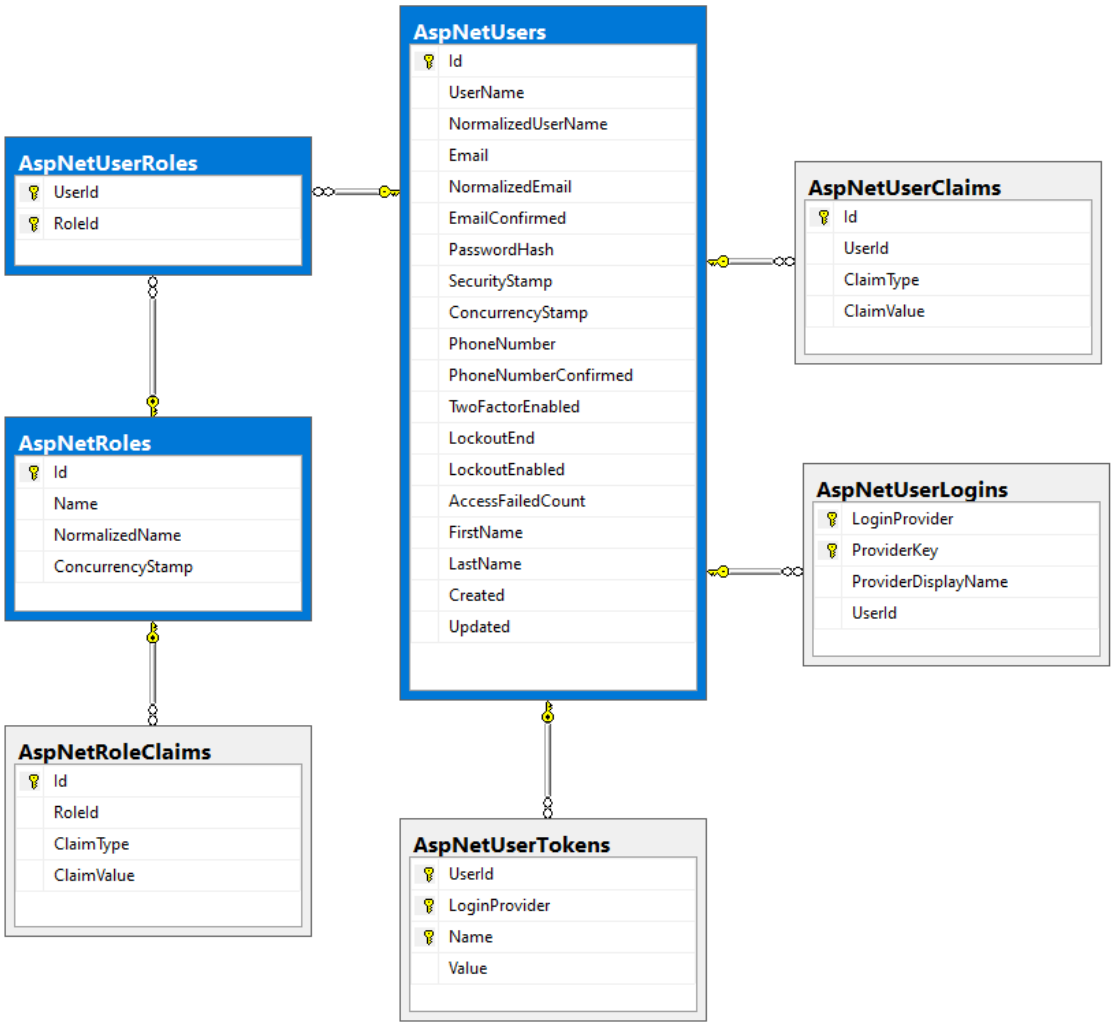


Image 1: *Entity relationship diagram based on ASP.NET Identity.*

8 Graphical user interface

In this section you can see some screenshots of how the application GUI looks like – both in desktop view and mobile device view (responsive view).

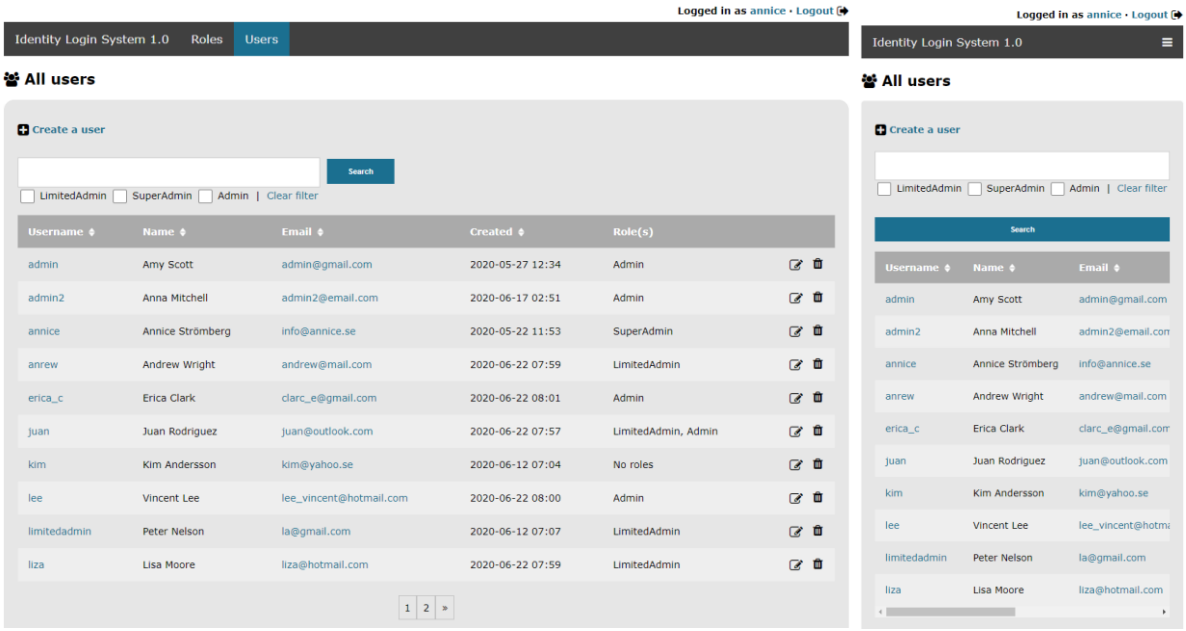


Image 2: *Screenshot of the user list page in desktop vs responsive view for a SuperAdmin user.*

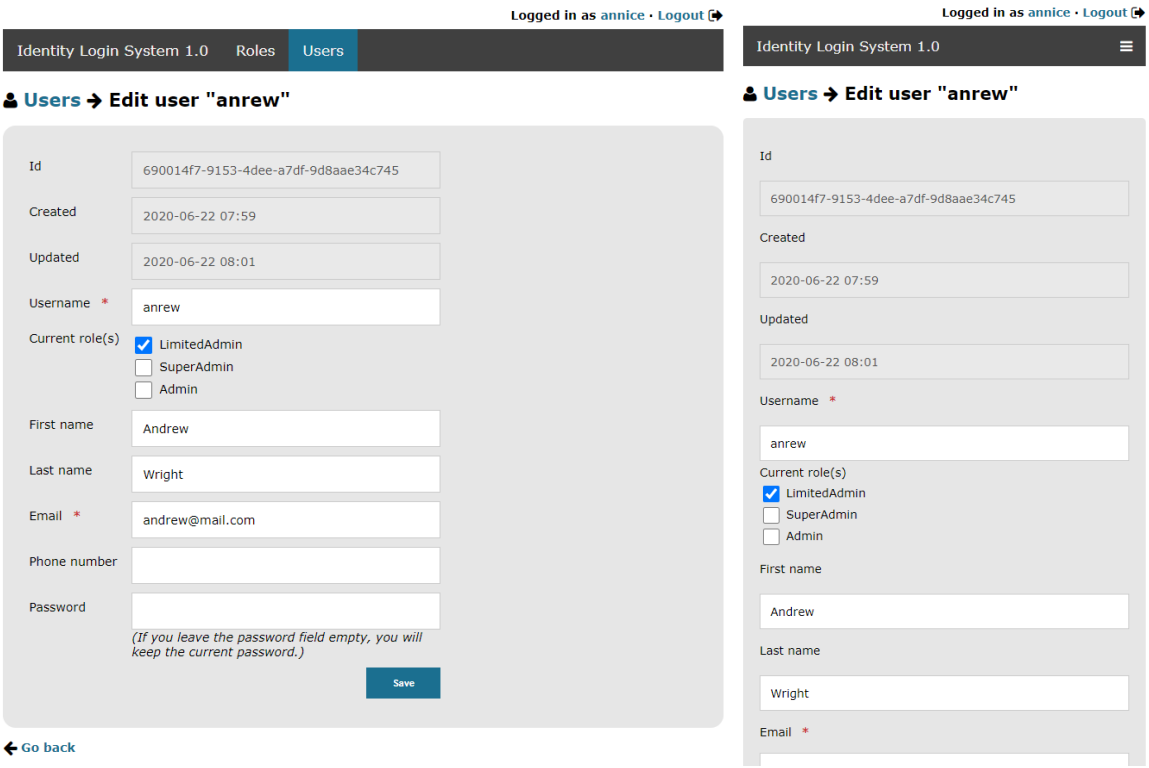


Image 3: *Screenshot of a user edit page in desktop vs responsive view for a SuperAdmin user.*

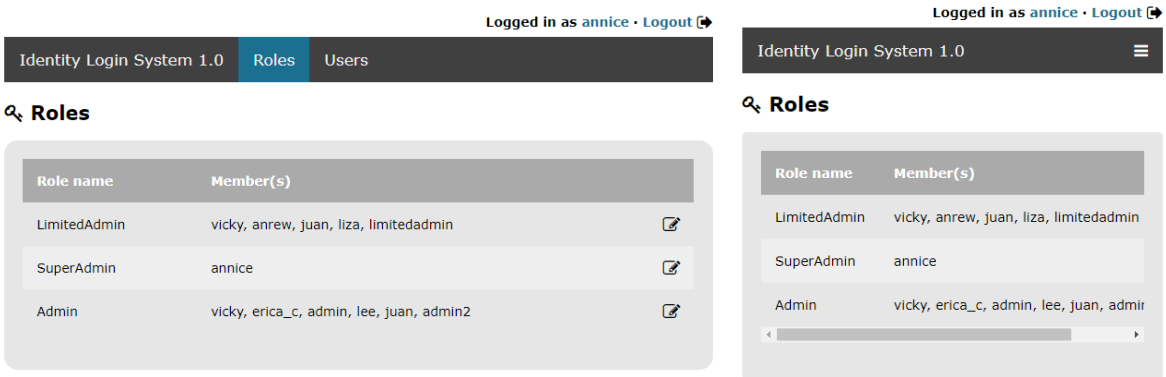


Image 4: *Screenshot of the role list page in desktop vs responsive view for a SuperAdmin user.*

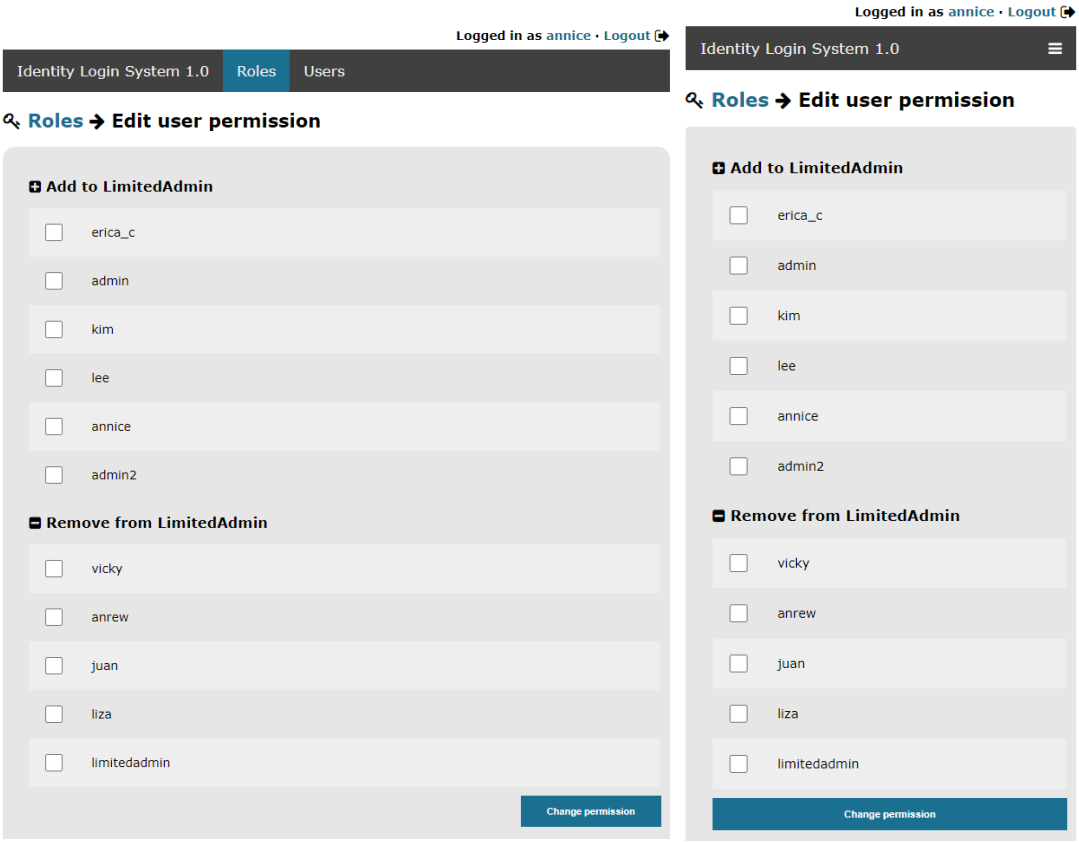


Image 5: *Screenshot of the role edit page in desktop vs responsive view for a SuperAdmin user.*

9 Setup guide

As this script was created in Visual Studio Community with SQL Server, I will go through the necessary setup steps accordingly.

9.1 Prerequisites

- Install SQL Server Express
- Install SQL Server Management Studio (SSMS)
- Install .NET Core 3.1 (SDK)
- Install Visual Studio Community

9.2 Configure the application

1. Select to open the application in Visual Studio by double clicking the "LoginSystem.sln" file via the unzipped script folder path: *IdentityLoginSystem1.0 > LoginSystem > LoginSystem.sln*

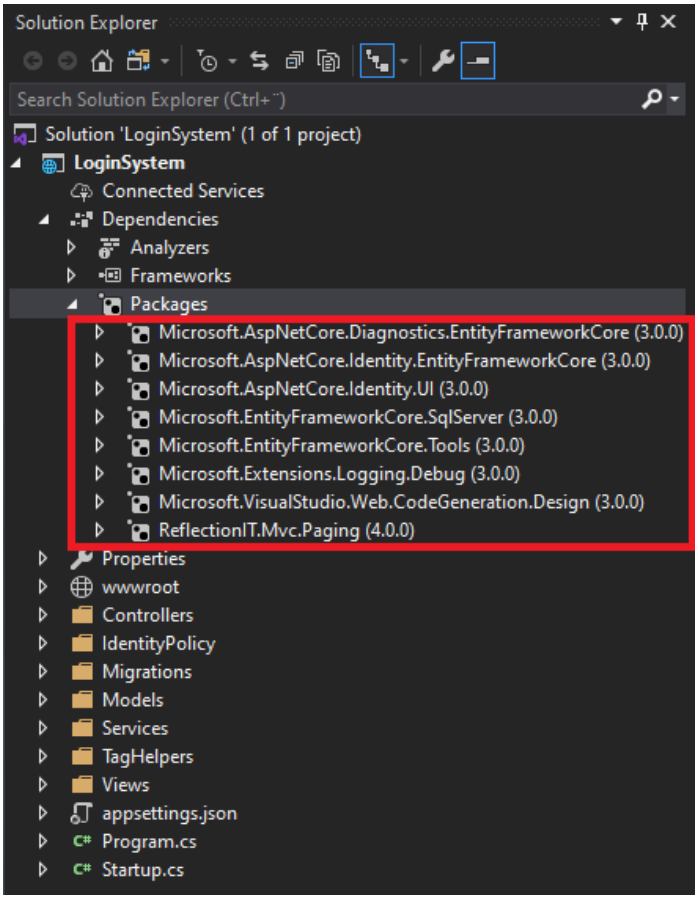
2. Once the LoginSystem solution is open in Visual Studio, select to open its "appsettings.json" file in the "Solution Explorer" window and change the highlighted values below to suit your own settings.
   (**Note!** To enable the ability to send password recovery emails via your Gmail, you will have to configure your Gmail account to enable less secure apps):

```
{
  "AllowedHosts": "*",

  "ConnectionStrings": {
    "LoginSystemConnection":
"Server=.\\SQLEXPRESS;Database=LoginSystem;Trusted_Connection=True;MultipleActi
veResultSets=true" // Only change this if you want to use another DB name!
  },

  "EmailSender": {
    "Host": "smtp.gmail.com", // Keep this setting if you use Gmail.
    "Port": 587, // Keep this setting if you use Gmail.
    "EnableSSL": true,
    "UserName": "your@gmail.com",
    "Password": "YourGmailPassword"
  },

  "Paging": {
    "ItemsPerPage": "50" // The number of records to display per page on the
user list page.
  },

  "UserRoles": {
    "SA": "SuperAdmin", // Keep the role names, otherwise you must ensure they
reflect the ones stored in DB!
    "A": "Admin",
    "LA": "LimitedAdmin"
  }
}
```

3. Save the "appsettings.json" file (Ctrl + S).

9.3 NuGet packages

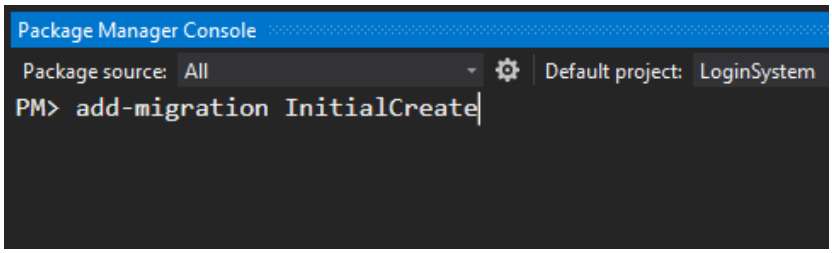
4. Make sure you have the following NuGet packages installed for your solution, otherwise install them:



9.4 Create the ASP.NET Identity database and its tables

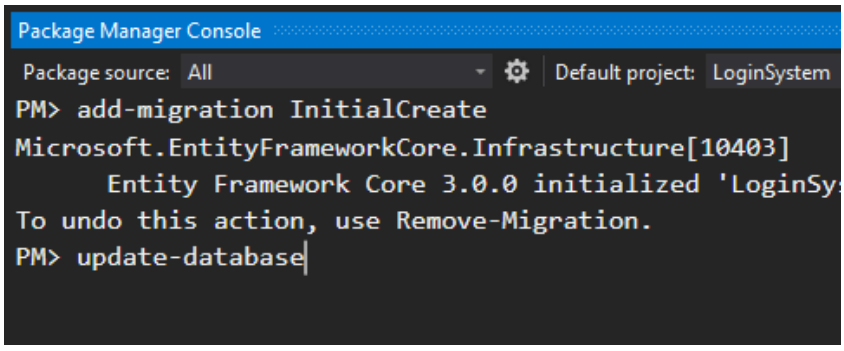
5. In Visual Studio, open the package manager console (PMC) via menu option: *Tools > NuGet Package Manager > Package Manager Console*

6. In the PMC, check that you are in the same folder as your "Startup.cs" file by typing the "dir" command followed by enter. If the startup file is listed, you can create a new database migration by typing the following command and then press enter: *add-migration <a migration name>*



After you have executed the above command, you will notice that a new migration folder named "Migrations" has been added in the "Solution Explorer" window.

7. After you have added the migration, create the database and its tables by typing the following command in PMC followed by enter: *update-database*



8. Once you see the feedback message "Done" in the PMC, the database has been created based on your connection string in "appsettings.json" along with the default database tables supported by the ASP.NET Identity framework. (You can also control this by connecting to your SQLEXPRESS instance via SSMS and check for the database tables there.)

9.5 Run the application and create your first admin

9. When the database and its tables are created, you can select to run the application from Visual Studio via the play button in the top bar.

10. On the initial application launch, you will be redirected to a page to create your first super admin. Fill in the mandatory fields and click to create this user.

11. Once the first super admin is created, you can login with your specified user credentials and start using the app!

10 Contact details

For general feedback related to this script, such as any discovered bugs etc., you can contact me via the following email address: info@annice.se