# Exercises: ASP.NET Core - Part IV

# Assign Roles to Users,Implementing Areas

# Buy Ticket Functionality

Problems for exercises for the ["ASP.NET Core Advanced" course @ SoftUni](https://softuni.bg/trainings/4708/asp-net-advanced-october-2024)

A popcorn and film reels and a movie ticket

Description automatically generated with medium confidence

## Implementing Roles

Roles define the **permissions or responsibilities** assigned to a user in an application (e.g., Admin, Manager, User). Roles are used to **restrict or allow access** to specific parts of an application.

### Configure Identity for role Support

* Add Identity Role Support: Ensure the AddIdentity call in Program.cs includes IdentityRole<Guid>  
  A screen shot of a computer program

  Description automatically generated
  + IdentityRole<Guid> ensures roles are stored in the database
  + AddRoles<IdentityRole<Guid>> registers role management services

### Seed Roles in the Database

Create a **dedicated seeder class** in the Configuration folder  
A screenshot of a computer

Description automatically generated

A computer screen shot of text

Description automatically generated



**Update Program.cs**: Call the seeding method after building the app:

A screen shot of a computer code

Description automatically generated

**Update the routing configuration** to include areas:

A computer code with text

Description automatically generated with medium confidence

* Place the areas route **before** the default route so the application checks for areas first.

### Verify the Seeded Roles Database Provider (SSMS)

After running the application with the SeedRoles method, we need to confirm that the roles have been successfully added to the database. This ensures that the role-seeding logic worked correctly and that the roles (Admin, Manager, User) are available for use in the application.

The roles are stored in the AspNetRoles table, which is part of the default Identity schema created by ASP.NET Core. Checking this table in SQL Server Management Studio (SSMS) helps verify the success of the operation.

A screenshot of a computer

Description automatically generated

### Assign Roles to a User

Now that roles (Admin, Manager, User) **have been seeded into the database**, the next step is to **assign a role to a specific user**. For example, **we can assign the Admin role to a user with the email admin@example.com**. This ensures that the user can access functionality restricted to the Admin role

* **Creating the user** (if they do not already exist)
* **Assigning the desired role to the user**

**Update the DatabaseSeeder Class:**

A computer screen shot of a website

Description automatically generated



Call the AssignAdminRole Method in Program.cs

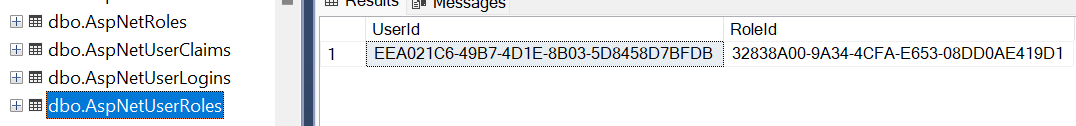
A screenshot of a computer program

Description automatically generated

Restart the Application to execute the updated DatabaseSeeder logic:

A screenshot of a computer

Description automatically generated



## Add Role-Based Redirects

### Update the Middleware in Program.cs

In the Program.cs file, **add middleware to check the user’s role after they authenticate and redirect them**.

* Place this logic **after** app.UseAuthentication() and **before** app.UseAuthorization()

A computer code with text

Description automatically generated

## Implementing Areas

### Allow Admin to Assign Roles to Users

The Admin will be able **manage user roles** directly from the **Admin area**:

* Listing all users in the system
* Selecting a specific user to assign or update roles
* Saving the role assignment in the database

### Create the Admin Area

The **Admin Area** will serve as a dedicated section of the application where **only users with the Admin role can access and perform administrative tasks**, such as assigning roles to users and managing key aspects of the system.

* Inside the existing **Areas folder**, add a **new folder** named **Admin**

A screenshot of a computer

Description automatically generated

### Setup the Admin Area Structure

**Inside the Admin folder, add the following subfolders**

* **Areas/Admin/Controller**
* **Areas/Admin/Views**
  + **/Index**
  + **/UserManagement**

**A screenshot of a computer

Description automatically generated**

### Add \_ViewStart.cshtml in the Admin Area

**Add a new file** named \_ViewStart.cshtml in the **Areas/Admin** folder

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

This ensures that **all views inside the Admin area inherit from the shared layout**

### Add \_ViewImports.cshtml in the Admin Area

A screenshot of a computer

Description automatically generated



### Add the Admin Controllers

* Inside **Areas/Admin/Controllers**, **create a new controller** named HomeController
* Update the controller code to **specify it belongs to the Admin area**

A screenshot of a computer program

Description automatically generated

* Inside **Areas/Admin/Controllers**, **create a new controller** named UserManagementController

**A screenshot of a computer

Description automatically generated**

### HomeController

The HomeController is the entry point for the Admin Area and should provide an **overview** and **navigation to other sections** like UserManagement, CinemaManagement, and MovieManagement.

### Create First View for the Admin Area

**Admin dashboard View:**

* Inside **Areas/Admin/Views/Home**, create Index.cshtml

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Add the following content**

****

**A screenshot of a computer

Description automatically generated**

### Plan the User Management Features

* View a list of users
* Assign or revoke roles for each user
* Delete users if necessary

### Update the UserManagementController

A screenshot of a computer program

Description automatically generated

**Purpose of Index Action**: It retrieves **a list of users from the database** using the UserManager service and   
passes it to the view.

### Create the View Model for User Management

* **Create a new foder** named ViewModels in Areas/Admin folder
* Inside the ViewModels folder, **add a new class** named UserViewModel.cs

A screenshot of a computer program

Description automatically generated

### Update the UserManagementController to Use the View Model

* Update the Index action to use the UserViewModel for displaying user data

A screenshot of a computer program

Description automatically generated

### Create the Index View for User Management

* Navigate to **Areas/Admin/Views/UserManagement**
* Inside the **UserManagement** folder, create a new file named Index.cshtml

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

## UserManager Functionality

### Inject RoleManager into the Controller

* Add the following **RoleManager dependency** in the constructor of the UserManagementController class:

A close-up of a computer screen

Description automatically generated

* **Assign Role Action**

**A screenshot of a computer program

Description automatically generated**

* **Remove Role Action**

**A screenshot of a computer program

Description automatically generated**

* **Delete User Action**

**A screenshot of a computer program

Description automatically generated**

* **Test the Functionalities:**
  + Test that selecting a role and clicking the "**Assign Role**" button correctly assigns the role to the user
  + Test that selecting a role and clicking "**Remove Role**" correctly removes the role from the user
  + Test that clicking the "**Delete**" button successfully deletes the user

## Buy Tickets Functionality

### Add BuyTicketViewModel

* Create a new **ViewModel** to **handle the ticket purchase data:**

A computer screen shot of a program code

Description automatically generated



### Create a BuyTicketRequest Object

This class will **encapsulate the parameters needed for buying tickets**:

A screenshot of a computer program

Description automatically generated

### Extend TicketApiController for Buying Tickets

* We can add an endpoint to allow users to buy tickets
* Check if the user is a manager using the **IsUserManagerAsync** method we already have. If the user is a manager, return an **Unauthorized** **response**

A screenshot of a computer program

Description automatically generated

### Implement the BuyTicketsAsync Method in the TicketService (+interface)

A screen shot of a computer program

Description automatically generated

Now that the method for buying tickets is working, we can proceed to implement the **View** for the "Buy Ticket" functionality. This view will allow users to select the quantity of tickets they want and confirm the purchase

A screen shot of a computer

Description automatically generated

### Attach Click Event to the "Buy Ticket" Button

* We will change the asp-action attributes in the "**Buy Ticket**" button to data-\* attributes, so we can fetch the required data using jQuery
* **Update the Cinema ViewProgram View:**
  + **Updated Button Code:**A computer screen shot of a program code

    Description automatically generated
  + **Add a Modal for the Purchase Form  
    A screenshot of a computer code

    Description automatically generated  
    **
  + **Create the Modal Partial View**

**A screenshot of a computer

Description automatically generated**

### Include the Modal in the Cinema ViewProgram View

In the **Cinema/Program** view (ViewProgram.cshtml), include the modal partial view at the end of the file:

A computer code with text

Description automatically generated

### Modify the Buy Ticket Button to Trigger the Modal

A screen shot of a computer code

Description automatically generated

### Create an External JavaScript File for jQuery Logic

A screenshot of a computer

Description automatically generated A screen shot of a computer code

Description automatically generated



### Reference buyTickets.js in \_Layout.cshtml

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

