

# Role-based Security Exercise

## Table of Contents

<b>Outline.....</b>	<b>2</b>
Resources	2
Scenario	2
<b>How-To.....</b>	<b>6</b>
Getting Started	6
Create the HR Manager Role	9
Create two new Users	10
Assign the HRManager Role to a User	14
Add Logic to Check for the HRManager Role	16
Provide Feedback to the End-User and Test the Application	28
Control the Access to Screens	35

# Outline

In this exercise, we will create a Role and use it to control the authorization on an application that manages Employees and their Projects. This exercise will require a bit of work in Service Studio, but also some testing in the browser and usage of the Users application, the OutSystems default user provider.

In the application we have a set of projects and employees. Besides other things, the application allows adding employees to projects as ProjectMembers. In this exercise, we want to:

- Create a new HRManager Role.
- Create Users for our application and grant the new Role to one of them.
- Restrict the access to all Screens of the application to users that are only registered (have a username and password) and restrict the Screen to add new members only to HRManagers.
- Adapt the logic to add employees to a project to guarantee that only HRManagers can perform that task.

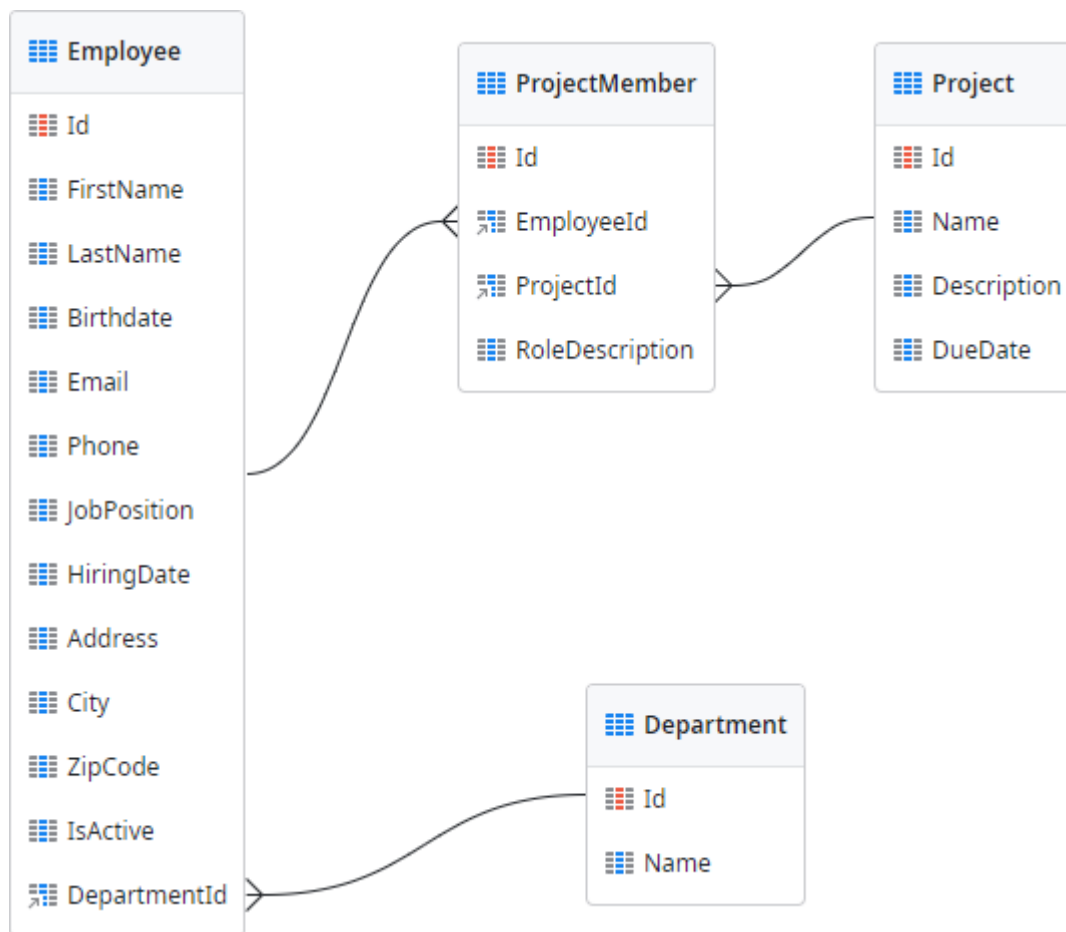
## Resources

We will start with a Quickstart application already created. This application has everything needed to start the exercise. This quickstart application can be found in the Resources folder of this exercise, with the name **Role-based Security Exercise.oap**.

## Scenario

Let's start from an existing application with one module. Inside that module, we have a data model already defined with four Entities. The module has the logic to import data

from Excel to populate these Entities. So, when the application (and its module) is installed, the data will be automatically bootstrapped.



The module also has four screens defined

Employees



AddProjectMember



Projects



ProjectDetail



The **Employees** screen shows a list of all the employees. The **Projects** screen displays all existing projects, while the **ProjectDetail** shows the detailed information about a particular project. In the ProjectDetail screen, there's an option to add a new member to a project, meaning associating an existing employee with an existing project. This option opens the **AddProjectMember** screen which has a Form to fill the relevant information needed to associate the employee with the project.

## Add Employee to Project

Employee \*

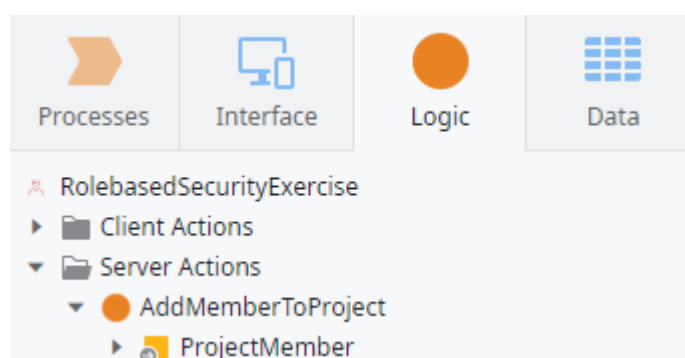
Patricia Wesley

Role Description \*

Senior Developer

Save

The Save Button triggers an Action that calls the already created AddMemberToProject Server Action, which has the logic to create a new ProjectMember in the database.



In this exercise, we will create two users, John Employee and Julie Manager, where Julie will be an **HRManager**, a new Role that we need to add to our application.


Then, we'll filter the access to our screens, making sure that only **Registered** users access the Employees, Projects and ProjectDetail screens. The AddProjectMember screen should only be accessible by **HRManagers**.

Then, we'll change our application again, in particular the AddMemberToProject Action, to check if the user currently logged in has the **HRManager** Role, **before** we actually create the new member in the database. The **CheckHRManagerRole** Action will be helpful for that.

Finally, we will change the access control of the AddProjectMember Screen to accept registered users as well, and make sure that the change in the AddMemberToProject Action works fine. When a simple registered user tries to add a new member to a project, the following message should appear:

 You don't have permissions to add a new member to a project

If it is an HRManager, then the following message is expected:

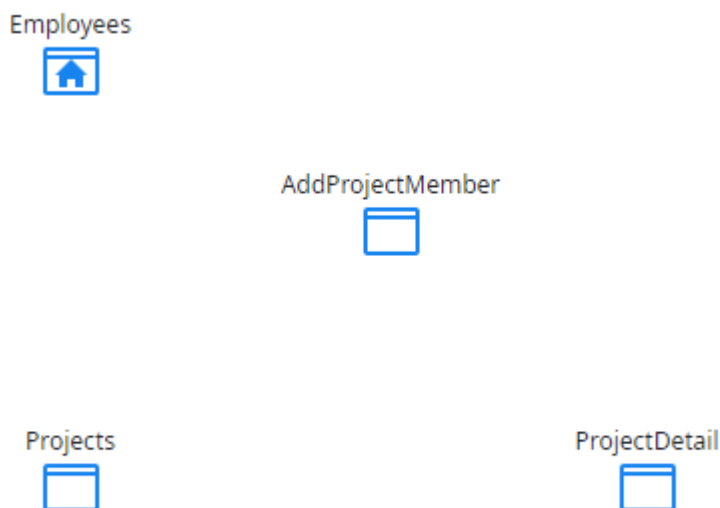
 New member added to the project

# How-To

In this section, we'll show you how to do this exercise, with a thorough step-by-step description. **If you already finished the exercise on your own, great! You don't need to do it again.** If you didn't finish the exercise, that's fine! We are here to help you.

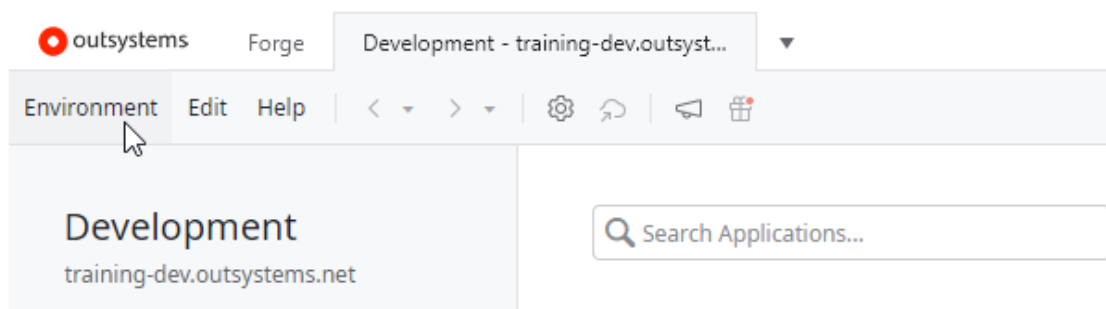
## Getting Started

To start this exercise, we need to install the Quickstart file, **Role-based Security.oap**. This file has an application with four Screens, one for the Employees (List), two for the Projects (List and Detail) and one for adding members to projects.

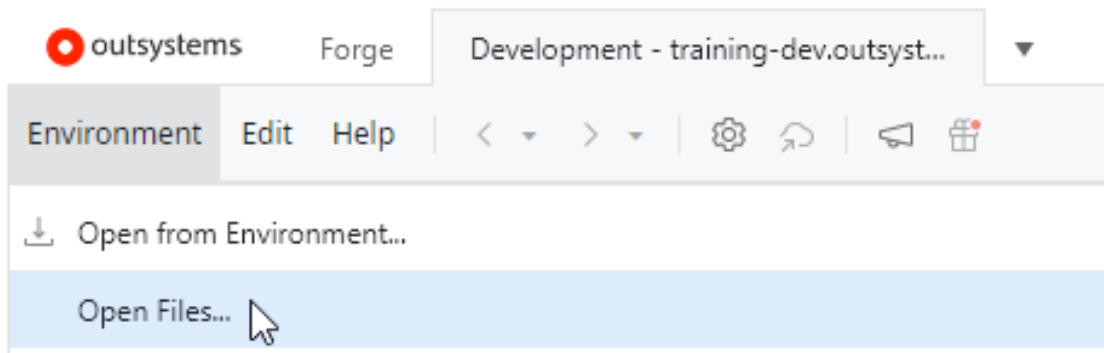


The first step that we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

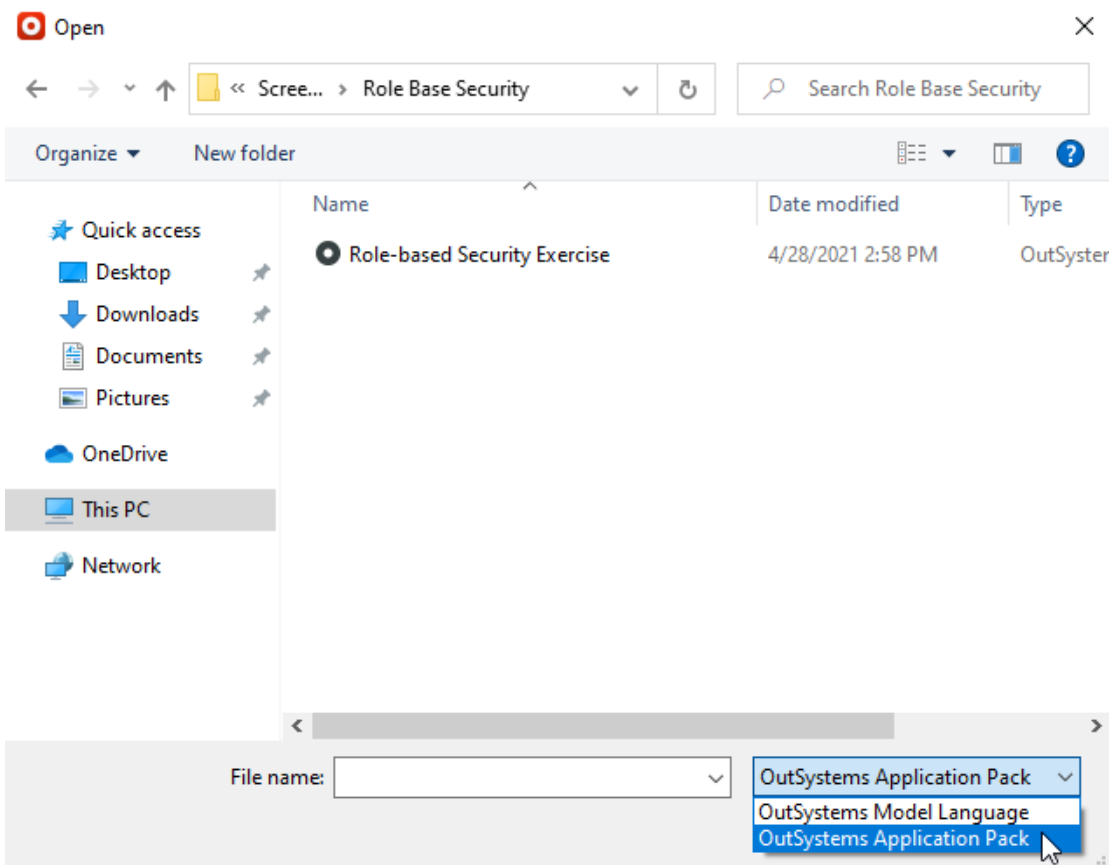
- 1) In Service Studio's main window, select the **Environment** menu on the top left.



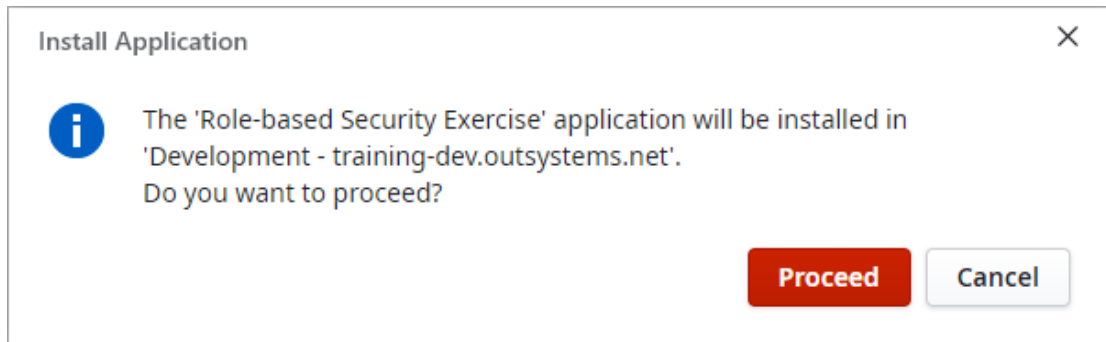
- 2) Select **Open Files...**



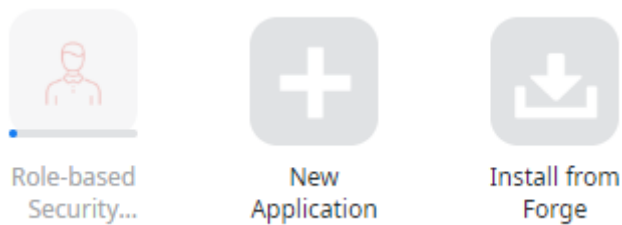
- 3) In the following dialog, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart and open the file named **Role-based Security.oap**.



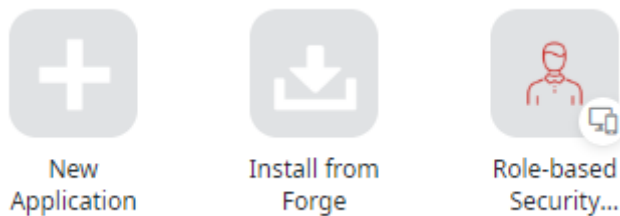
- 4) In the new confirmation dialog, select **Proceed**.



- 5) The application will begin installing automatically. When it's finished, we're ready to start!

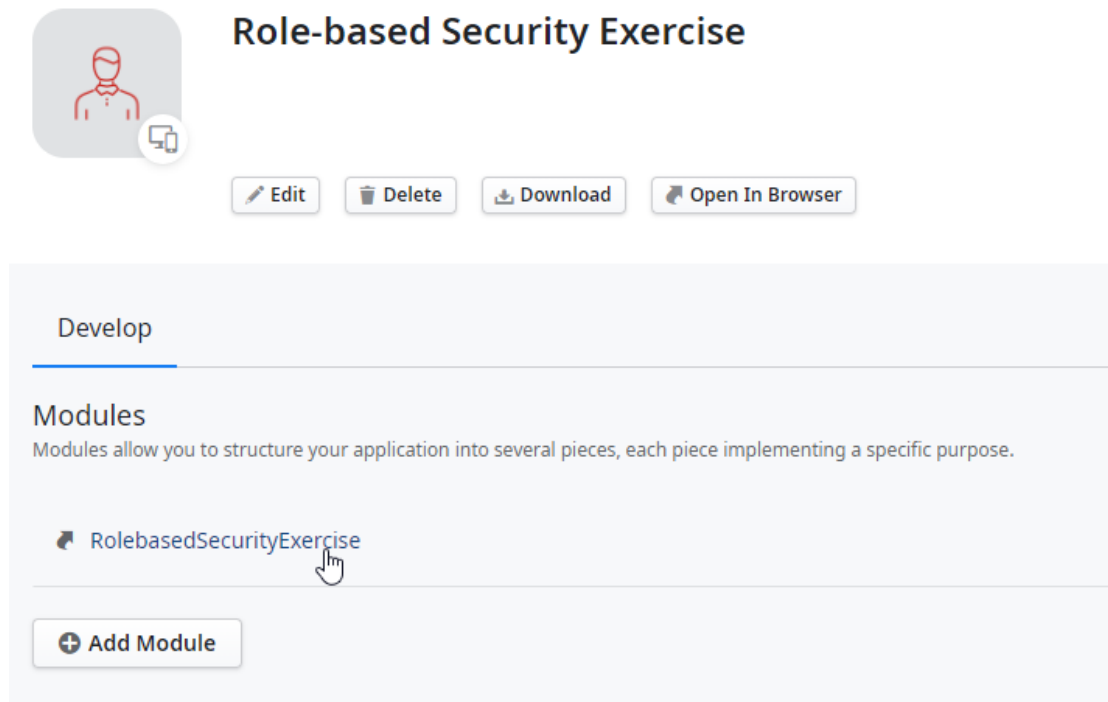


- 6) Open the application in Service Studio.





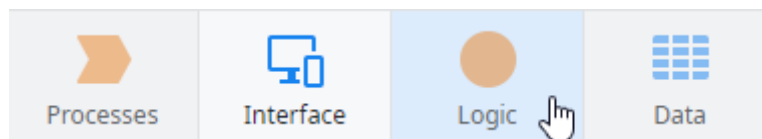
- 7) The application has only one module. Let's open it!



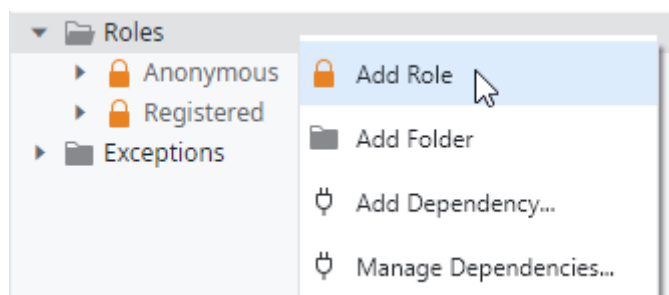
## Create the HR Manager Role

In this section, we'll create a Role in the application. This Role is called HRManager.

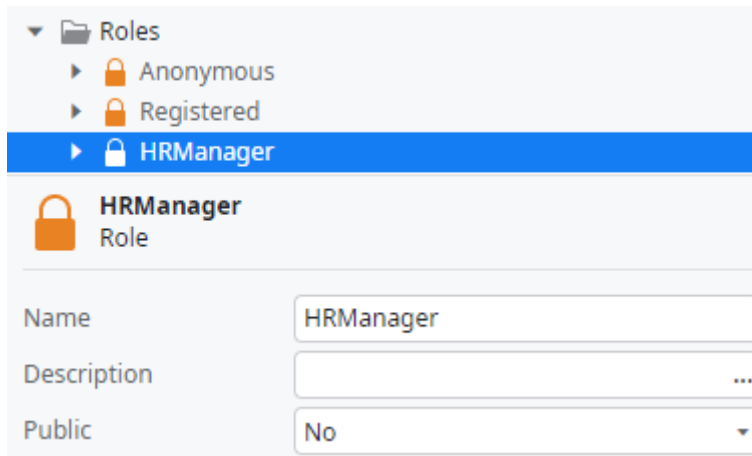
- 1) Switch to the **Logic** tab on the upper right of Service Studio.



- 2) Locate the **Roles** folder under the Logic tab, right-click the folder and select the **Add Role** option.



- 3) Set the Name of the Role to HRManager.



Roles

- Anonymous
- Registered
- HRManager**

**HRManager**  
Role

Name:

Description:

Public:

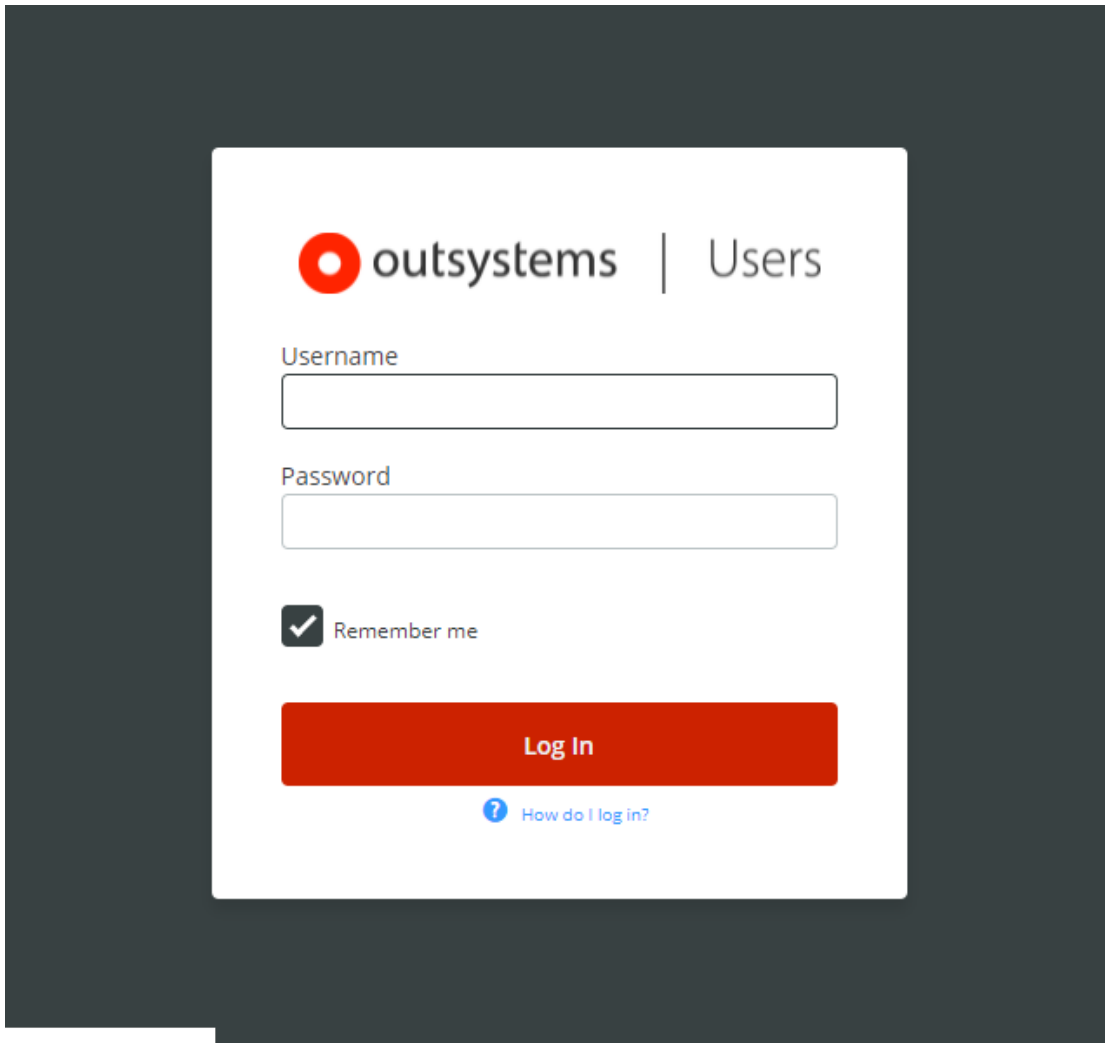
- 4) Publish the module to save the application in the server.

## Create two new Users

In this section, we'll create two new users in our environment. Note that the two new users that will be created in this section are end-users that can log in into the app. On the other hand, the Employees are simply data shown in the application, and in this exercise are not related to the end-users.

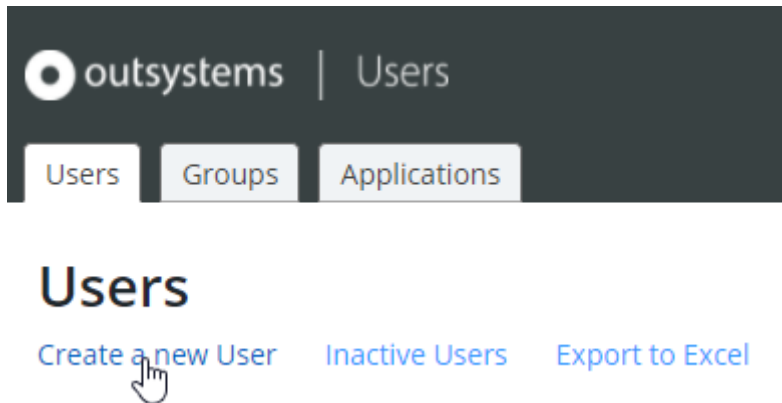
- 1) Switch to your browser and access the Users application through the following URL: [https://<your\\_development\\_environment>/Users](https://<your_development_environment>/Users). For instance, if you are

logged in to the environment `os11training.outsystems.net` in Service Studio, the URL should be: `https://os11training.outsystems.net/Users`.

A screenshot of the Outsystems Users login page. The page has a dark grey background. In the center is a white rectangular box containing the login form. At the top of the box is the Outsystems logo (a red circle with a white dot) followed by the text "outsystems | Users". Below this are two input fields: "Username" and "Password". Under the "Password" field is a checkbox with a checkmark and the text "Remember me". At the bottom of the box is a large red button with the text "Log In". Below the button is a small blue question mark icon followed by the text "How do I log in?".

- 2) Log into the Users app using your credentials. If you are using a Personal Environment, the credentials are the same that you used for Service Studio. For other types of environments you should have your credentials or will need to ask your administrator.
- 3) In the following page, we have the full list of Users registered in the environment. Let's add a new one.

- 4) Click the **Create a new User** link to start adding a new user to the environment.



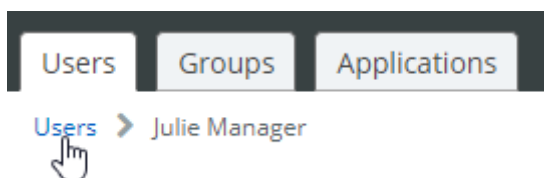
- 5) Fill in the user information with the following data and click the **Save** button.

**Name:** Julie Manager  
**Username:** jmanager  
**Password:** outsystems

## New User

<b>Name *</b>	<input type="text" value="Julie Manager"/>
<b>Username *</b>	<input type="text" value="jmanager"/>
<b>Email</b>	<input type="text"/>
<b>Phone</b>	<input type="text"/>
<b>Password *</b>	<input type="password" value="....."/>
<b>Confirm New Password *</b>	<input type="password" value="....."/>

- 6) Go back to the main page of the Users application.



- 7) Select the **Create a new User** option.

## Users

[Create a new User](#) [Inactive Users](#)



- 8) Fill in the user information with the following data and click the Save button.

**Name:** John Employee

**Username:** jemployee

**Password:** outsystems

## New User

**Name \***

**Username \***

**Email**

**Phone**

**Password \***

**Confirm New Password \***

**Save**

Cancel



- 9) Go back to the main Users page and confirm that the two users are there.

## Assign the HRManager Role to a User

In this section, we'll assign the **HRManager** Role to the user Julie Manager.

- 1) In the Users app, find the user called Julie Manager and click on the name to open the details page.

[Users](#) > Julie Manager

## Julie Manager

[Edit this User](#)

[Set as Inactive](#)

[Set Password](#)

**Username** jmanager

**Email**

**Phone**

**Creation Date** 15:05 (4 minutes ago)

**Last Login**

## Groups

0 records

Group Name

No Groups to show...

Type or double-click for list

Add

0 records

## Roles

0 records

Role Name

No Roles to show...

Type or double-click for list

Add

0 records

- 2) In the Roles section of this page, type *HRManager* and select the Role from the **RolebasedSecurityExercise** module. Click on **Add** to assign that Role to the user.

## Roles

0 records

Role Name

No Roles to show...

HRManager (RolebasedSecurityExe

Add

0 records

- 3) Make sure the Role is associated with the user.

## Roles

1 record

Role Name



HRManager (RolebasedSecurityExercise)

Type or double-click for list

Add

1 record

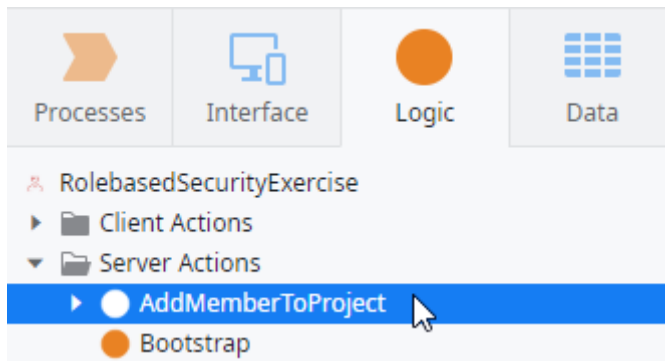
## Add Logic to Check for the HRManager Role

In this section, we add some logic to the Action that adds a new member to an existing project. The AddMemberToProject Server Action is used by the SaveOnClick Screen Action of the AddMemberToProject Screen. Inside the Server Action we'll add some logic that will allow us to validate this operation, guaranteeing that only users with the HRManager Role can actually perform the operation. This is actually a best practice when code runs on the device (as it happens with screens and screen actions) and also



on the server. Operations that modify or access sensitive data should always be validated on the server side (e.g. Server Action).

- 1) Switch to the Logic tab and open the **AddMemberToProject** Action by double-clicking it.

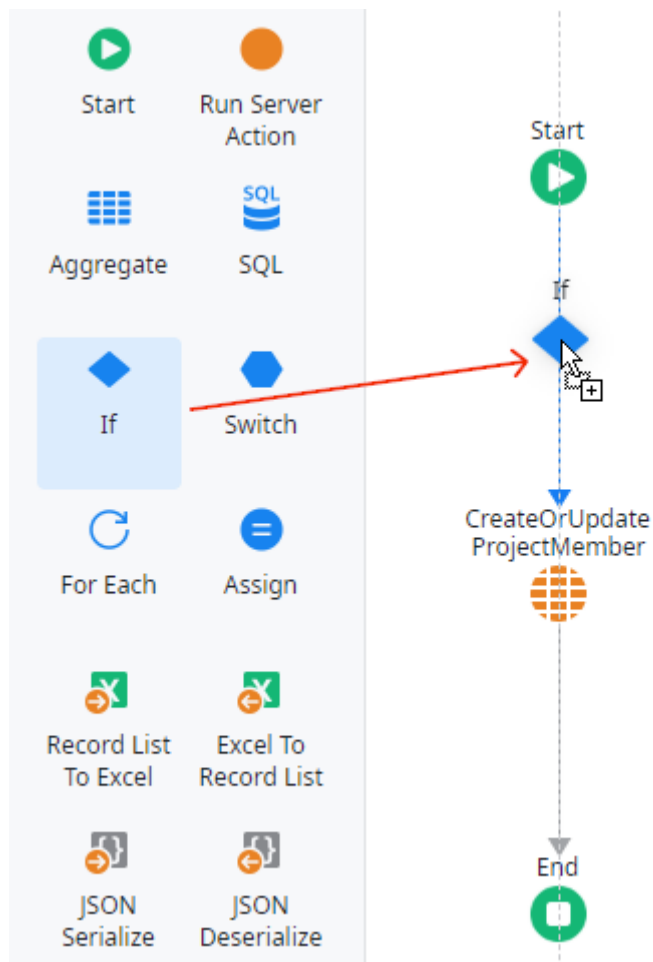


- 2) You should see the flow of the Action with the **CreateOrUpdateProjectMember** Entity Action, to add the employee to the project.

#### AddMemberToProject



- 3) Drag an **If** node and drop it between the Start and the CreateOrUpdateProjectMember Action.

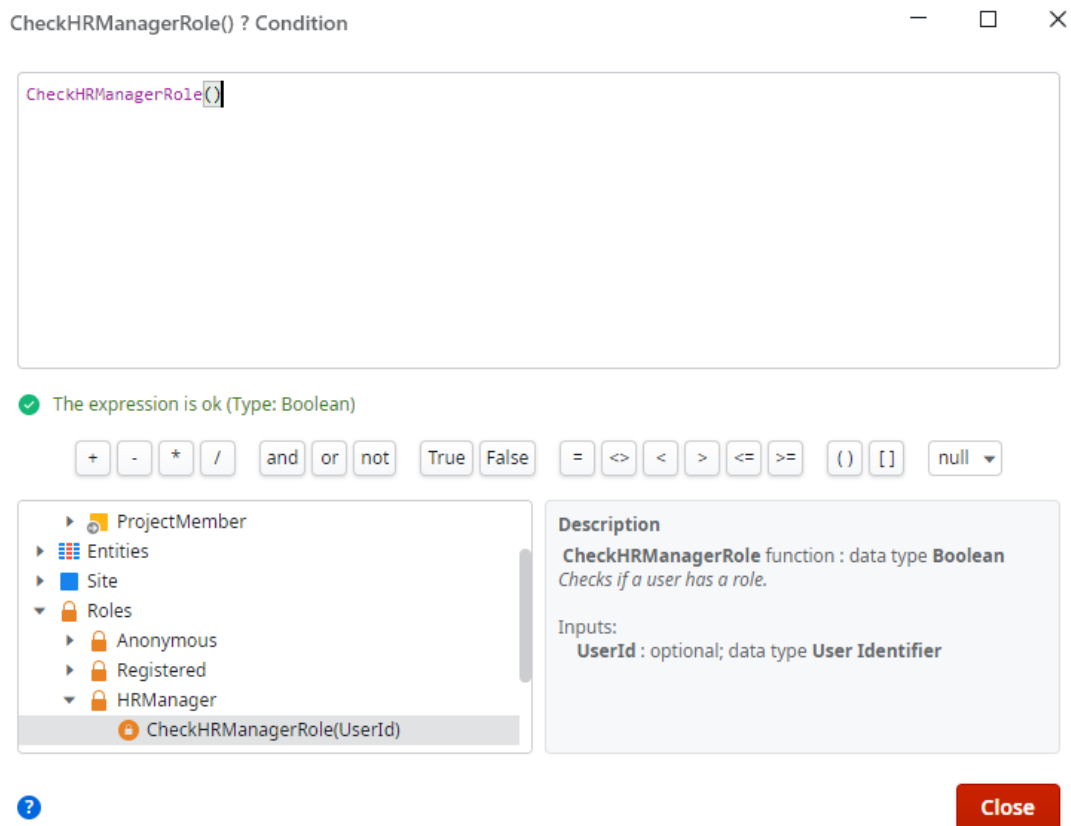


- 4) Double-click the **Condition** property of the If node to open the Expression Editor.

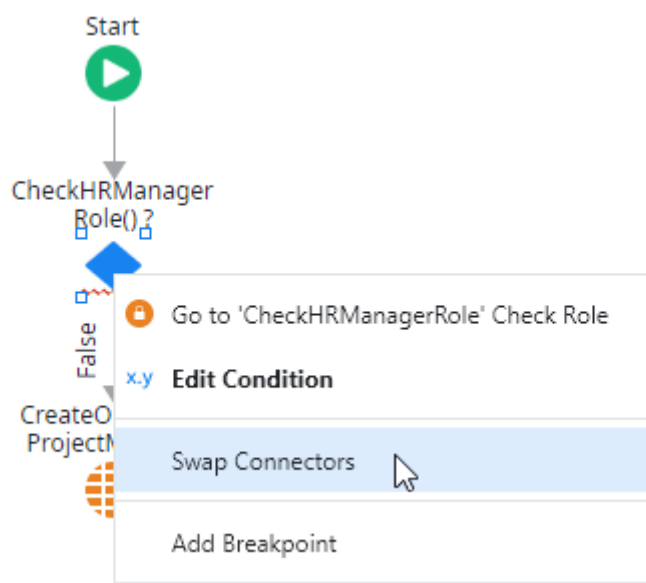
- 5) Set the Expression to be:

```
CheckHRManagerRole()
```

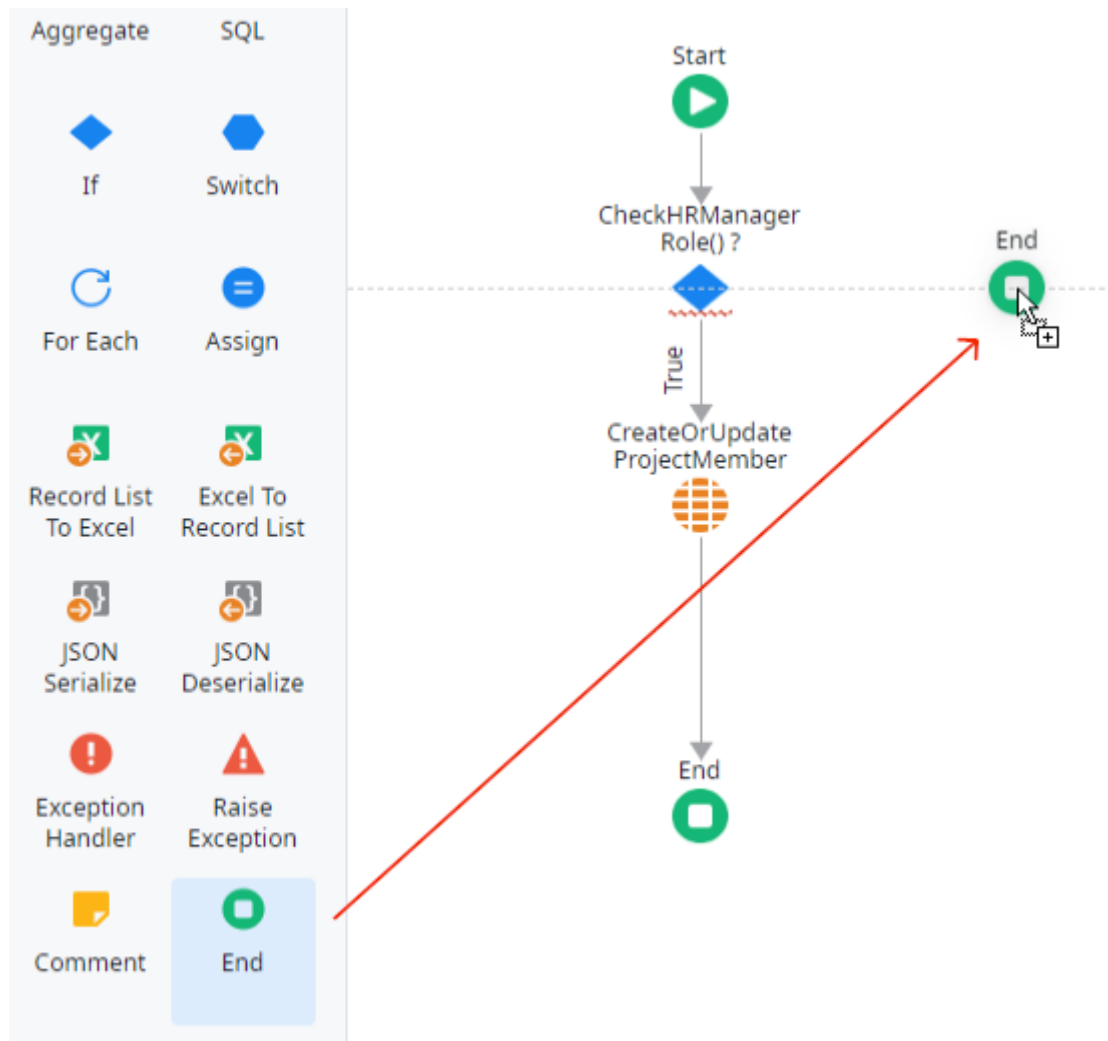
- 6) This expression checks if the user currently logged in has the HRManager Role associated to it. Click on **Done** to close the Expression Editor.



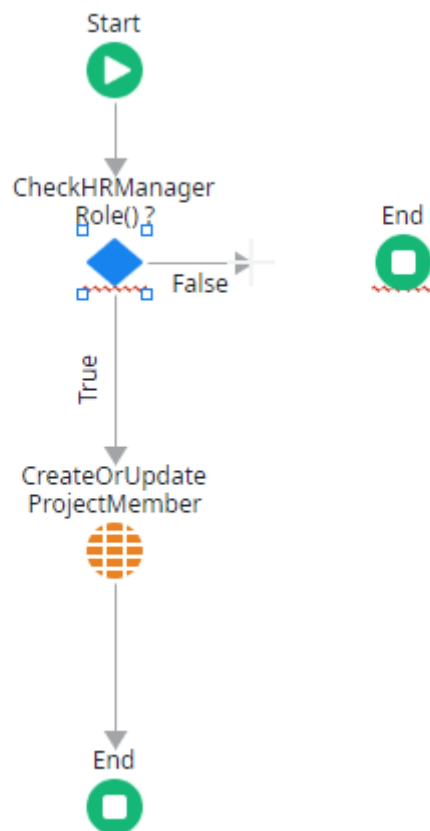
- 7) Right-click the If in the Action flow and select **Swap Connectors**. This will make sure that the flow will continue to the CreateOrUpdate Action when the If **Condition** is evaluated to **True**.



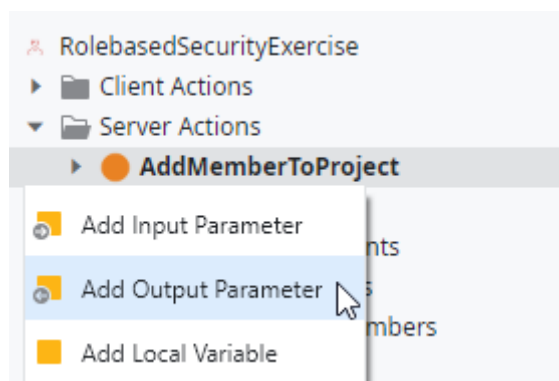
- 8) Drag an **End** node and drop it to the right of the If.



- 9) Create the connector between the If and the new End to create the **False** branch.



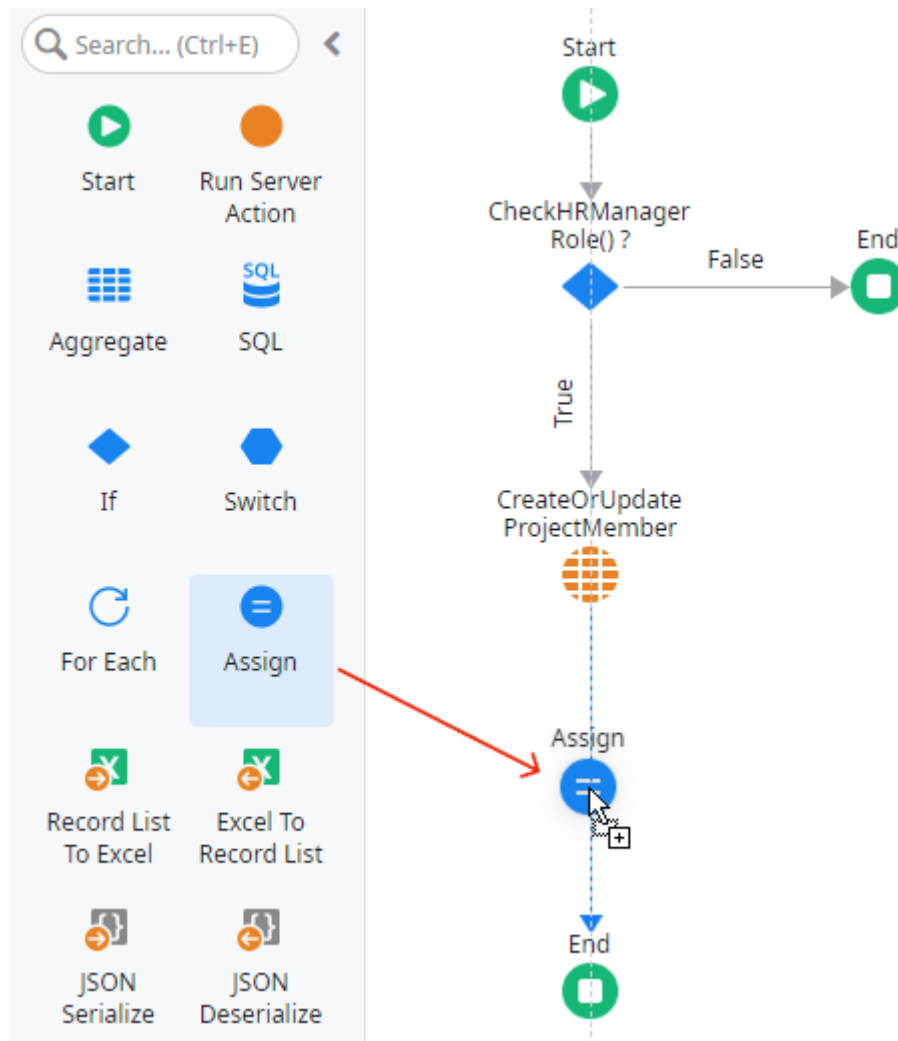
- 10) Right now, we only create a ProjectMember in the database if the user currently logged in is an HRManager. Now, whenever this Action is called, we need to make sure that it returns the information that the ProjectMember was created or not. For that, we will need an Output Parameter. Right-click the AddMemberToProject Action on the right and select **Add Output Parameter**.



- 11) Select the **Out1** Output Parameter and change its name to *MemberAdded*. Set its **Data Type** to *Boolean*.

The screenshot shows the OutSystems console interface. On the left, a tree view displays the project structure: 'RolebasedSecurityExercise' (expanded) contains 'Client Actions' and 'Server Actions'. Under 'Server Actions', 'AddMemberToProject' is expanded, showing 'ProjectMember' and 'MemberAdded'. The 'MemberAdded' element is selected and highlighted in blue. On the right, the configuration panel for 'MemberAdded' is shown. It is identified as an 'Output Parameter'. The configuration fields are: 'Name' (text input with 'MemberAdded'), 'Description' (text input with a placeholder '...'), 'Data Type' (dropdown menu with 'Boolean' selected), and 'Default Value' (dropdown menu).

12) Drag an **Assign** node between the CreateOrUpdate Action and the End node.

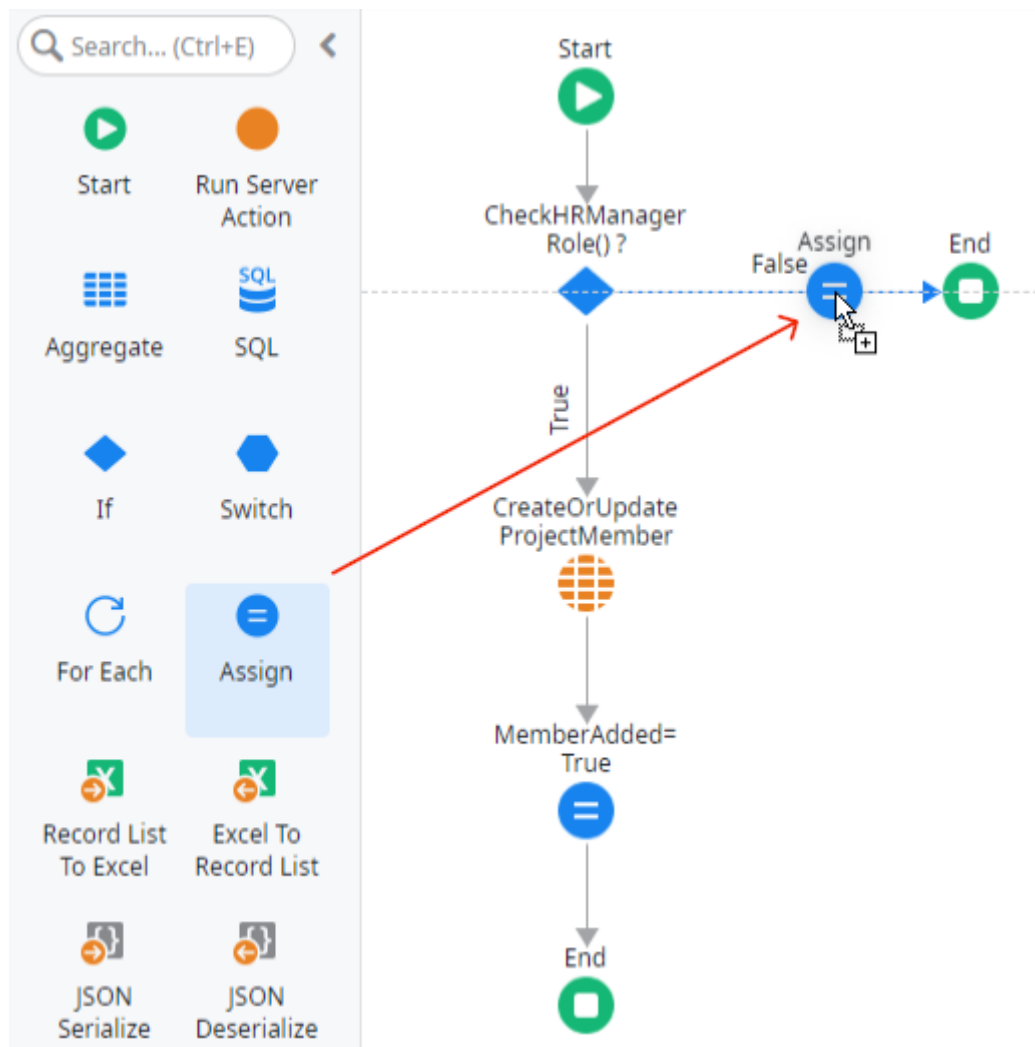


13) In the properties of the node, set the assign to be:

```
MemberAdded
= True
```

14) Set its **Label** to be *MemberAdded=True*.

- 15) Drag a new **Assign** node and drop it between the If and the End on the False branch.

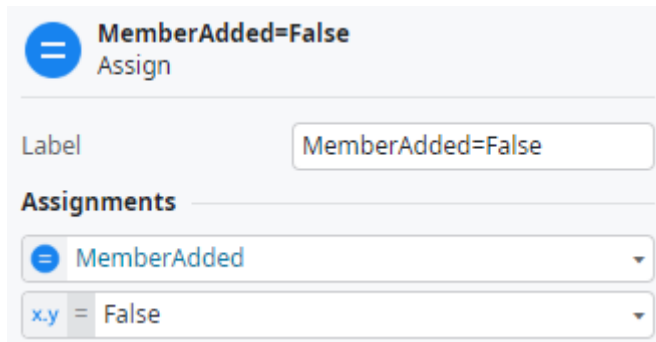


- 16) Set the new assign to be:

```
MemberAdded
= False
```



17) Set its **Label** to *MemberAdded=False*.



The screenshot shows a configuration panel for a widget named 'MemberAdded=False'. The panel has a blue header with a minus icon and the text 'MemberAdded=False' and 'Assign'. Below the header, there is a 'Label' property set to 'MemberAdded=False'. Under the 'Assignments' section, there is a dropdown menu set to 'MemberAdded' and a property 'x.y' set to 'False'.


18) Publish the module to save the application in the server.



19) Open the application in the browser, by clicking on the blue **Open in Browser** button.



20) Log in with the John Employee user (Username: *jemployee*, Password: *outsystems*)



### Role-based Security Exercise

Username  
*jemployee*

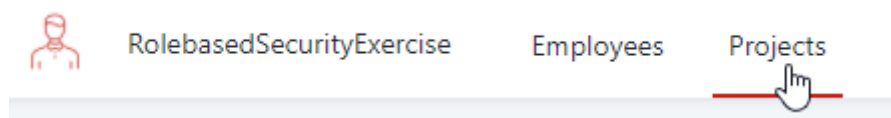
Password  
.....

☐ Remember me [Forgot password?](#)

**Login**

21) The application opens in the Employees screen, which is accessible by Registered users, just like John Employee.

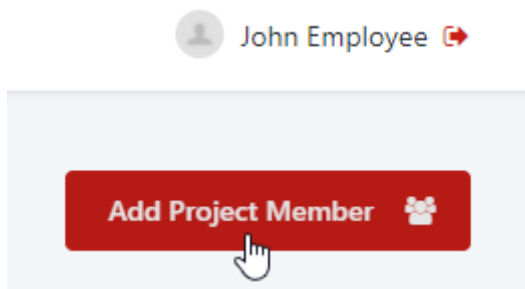
22) On the top menu of the app, select the Projects option.



23) On the Project List, select the Directory to open its details page.

Projects		
<input type="text" value="Search"/>		<a href="#">Add Project +</a>
Name ▾	Description ▾	Due Date ▾
<a href="#">Directory</a>	Internal web application to manage all the employees, including their department, birthdays and benefits.	31 Oct 2020
<a href="#">Expenses</a>	Web application to allow internal employees and subcontractors to submit their expenses and the managers to approve / reject them.	31 Jan 2021
<a href="#">Secret Mobile B2C</a>	Top secret B2C mobile application. This project is need to know basis.	28 Feb 2021
<a href="#">Task Manager</a>	Web application for customers to manage their tasks. This application will allow people to submit tasks, set priorities and manage the teams working on those tasks.	1 Dec 2020
<a href="#">Travel Portal</a>	Web application for managing travel requests.	31 May 2020

24) On the new screen, click on the **Add Project Member** button to access the **AddProjectMember** screen.



25) Fill out the form and press **Save**.

26) On the Details screen of the Directory project validate that the Employee was **not** added to the project.

27) On the top right side of the screen, click the Logout icon.



28) Login with the **Julie Manager** user (Username: *jmanager*, Password: *outsystems* ).

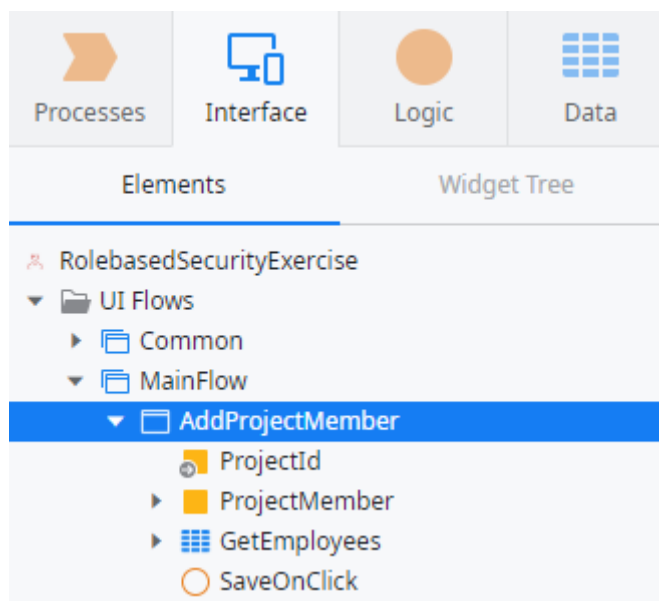
29) Add the same employee to the same project and validate that the project member was successfully added.

## Provide Feedback to the End-User and Test the Application

In this section, we'll add a feedback mechanism to make sure that a message is displayed when the employee is added to the project.

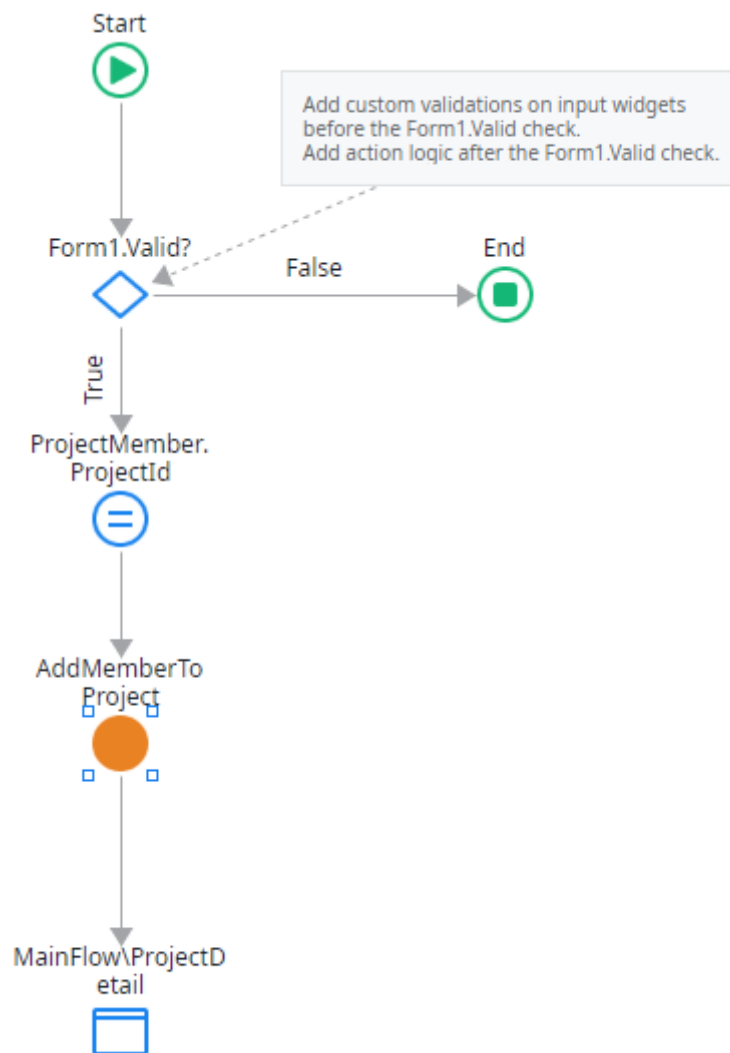
We'll change the access to the AddProjectMember Screen and see the Action working with the new logic.

- 1) Switch to the Interface tab in Service Studio and expand the **AddProjectMember** Screen.

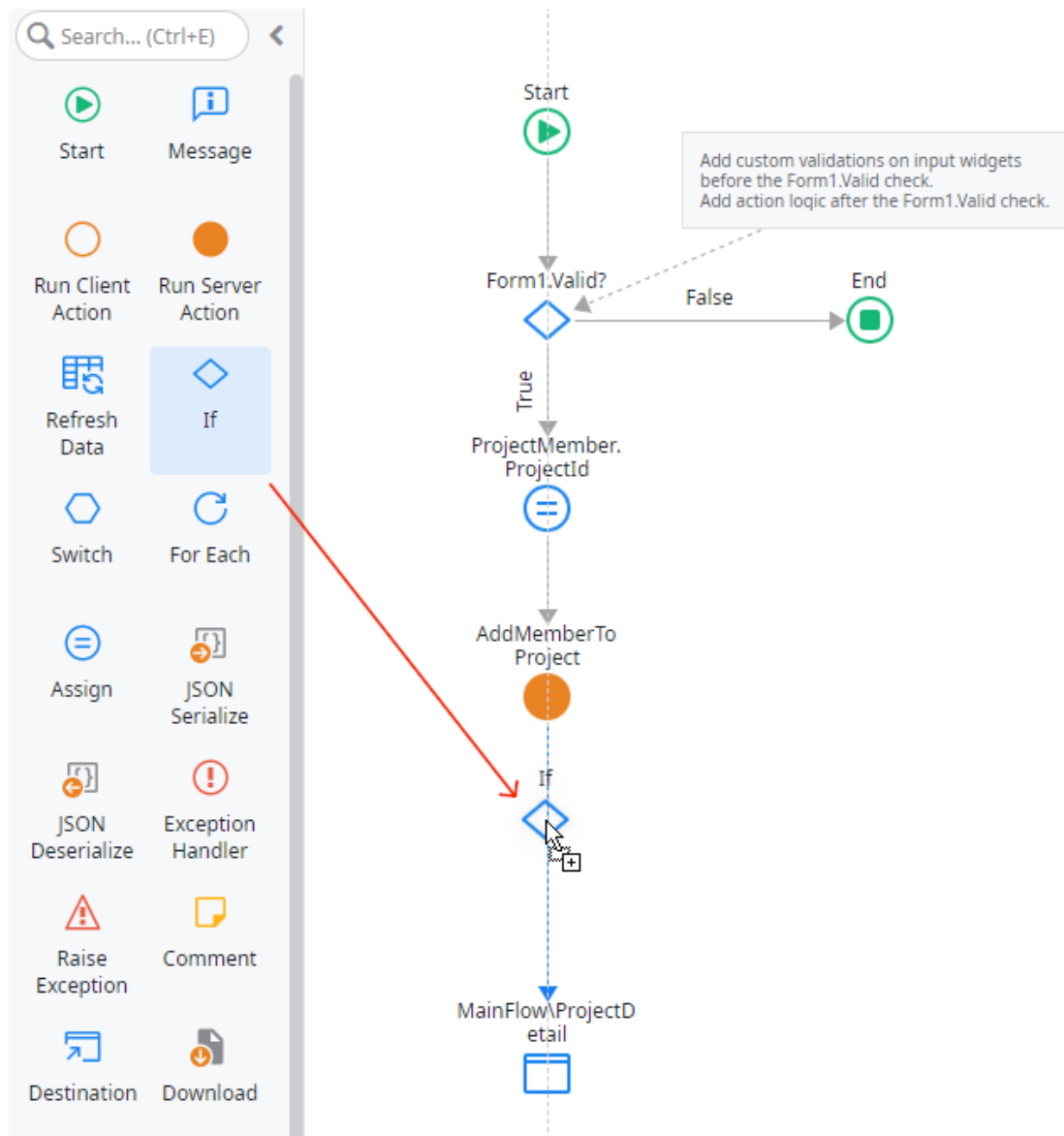


- 2) Double-click the **SaveOnClick** Action to open it and to see its logic flow. We can see that the AddMemberToProject Action is used in the flow. We now need to

adapt this Action to use the Output Parameter of the AddMemberToProject Action to give some feedback to the end-user.



- 3) Drag an **If** and drop it between the AddProjectToMember Action and the MainFlow\ProjectDetail destination.



- 4) Set the Condition of the If to be

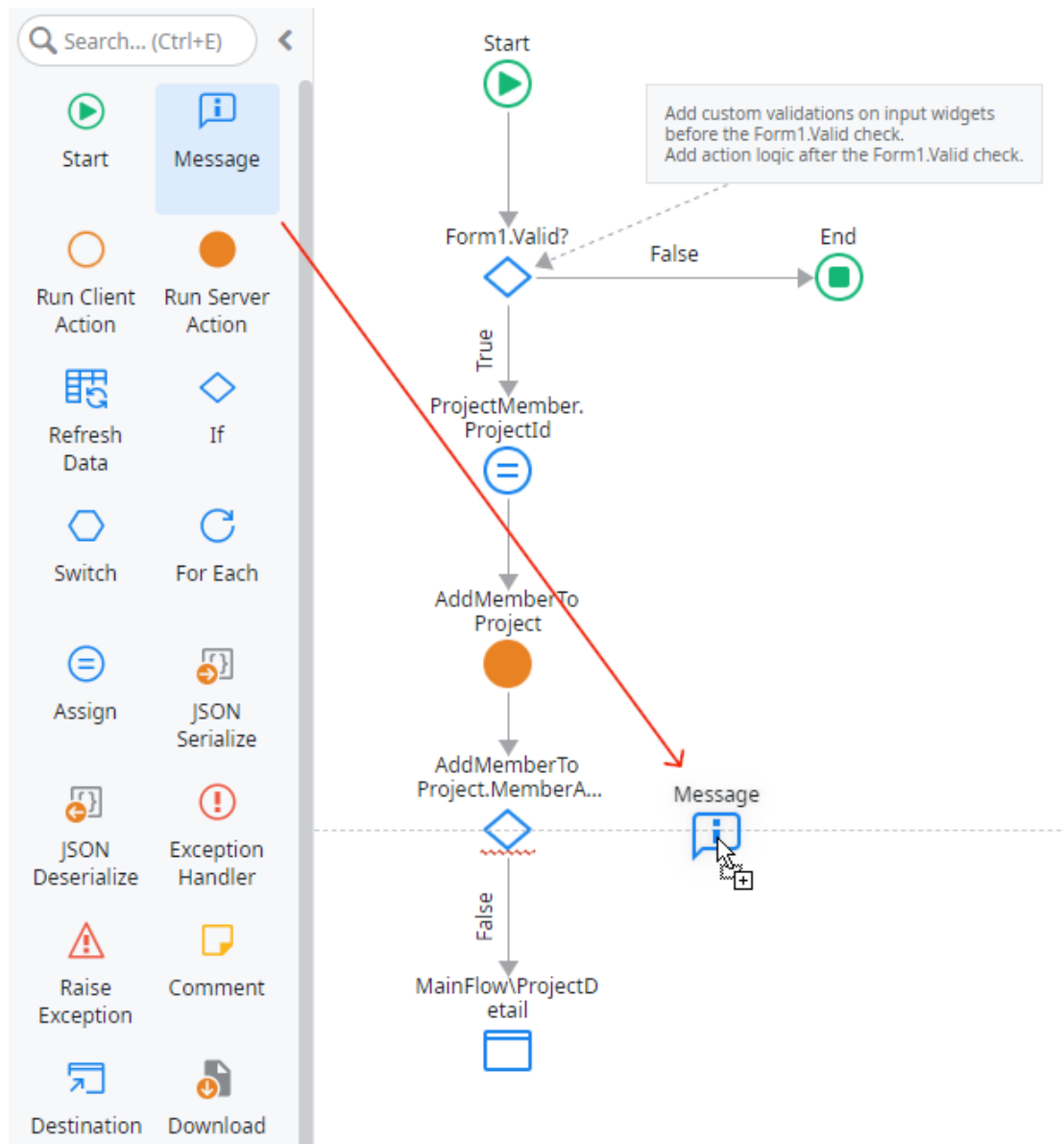
`AddMemberToProject.MemberAdded`

**AddMemberToProject.MemberAdded ?**  
 If

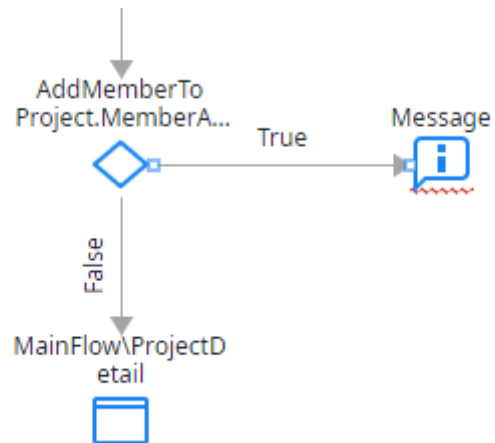
Label

Condition


- 5) Drag a **Message** node and drop it on the right of the recently added If.



- 6) Create the **True** branch between the If and the Message by connecting the two.

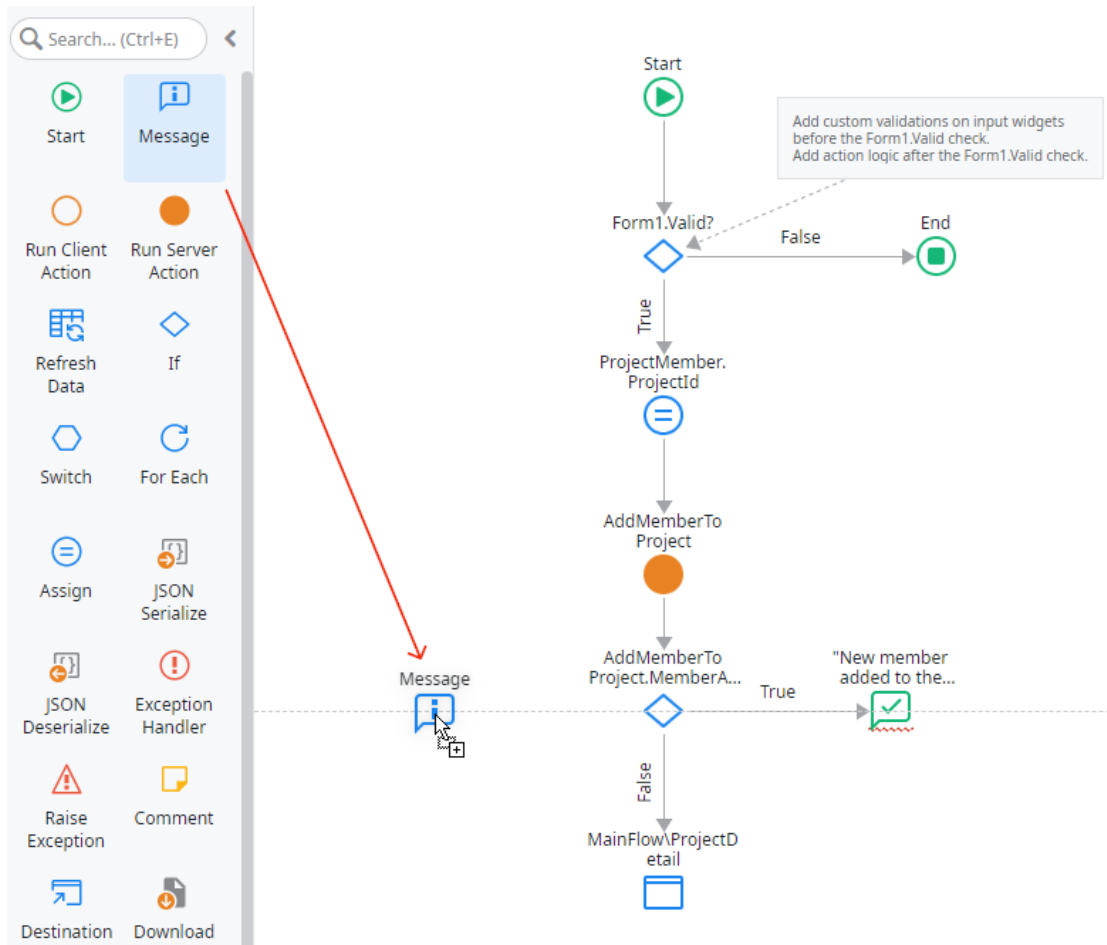


- 7) Select the Message node, and set its properties as the following:

	<b>"New member added to the project"</b> Message
Label	<input type="text"/>
Message	



- 8) Drag another **Message** node to the left of the If.



- 9) Select the most recently added **Message** and set the properties as following

**"You don't have permissions to add a new member to a project"**

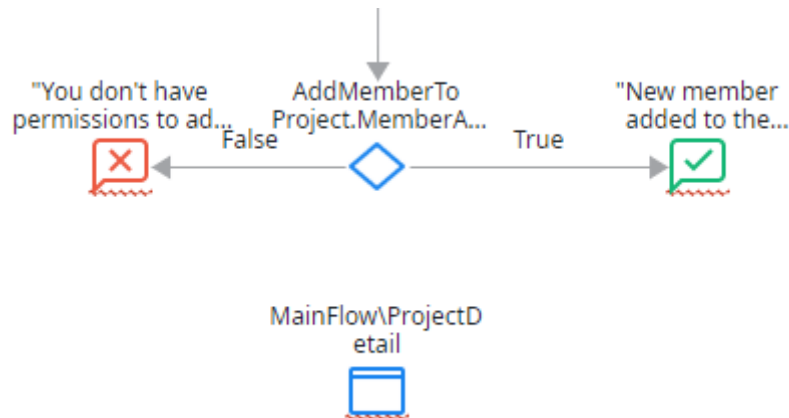
Message

Label

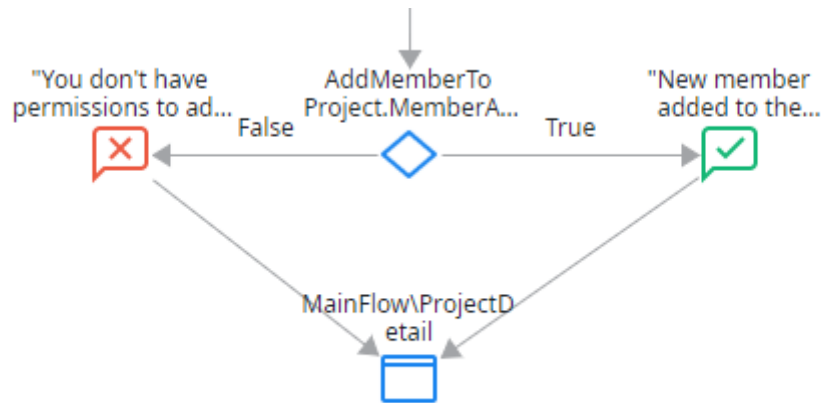
Message

Type

10) Drag the **False** branch of the If to connect to the new Message.



11) Connect both **Messages** to the **MainFlow\ProjectDetail** destination.



12) Publish the module to save the application in the server.



13) Open the application in the browser, by clicking on the blue **Open in Browser** button.



14) Log out, and then log in with the **John Employee** user (Username: *jemployee*, Password: *outsystems*). Navigate to the Directory project and click the Add Project Member button.

- 15) Fill in the information of the Employee, their Role Description, and click on Save. What happened? The following feedback message should be displayed at the top of the Screen.

✖ You don't have permissions to add a new member to a project

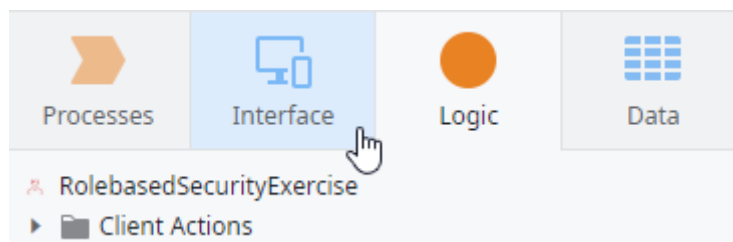
- 16) Log out and repeat the process for Julie Manager. Was the operation successful? Then, we're done with this section!

✔ New member added to the project

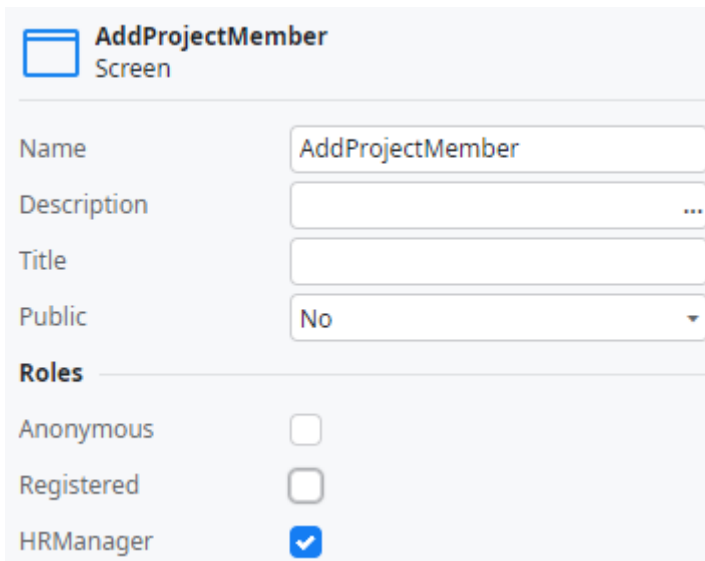
## Control the Access to Screens

In this section, we'll limit the access to all Screens to **Registered** users and the AddProjectMember screen to **HRManager** users. At the end, we'll test the application to confirm it works.

- 1) Switch to Service Studio and open the **Interface** tab on the upper right corner.



- 2) Select the **AddProjectMember** screen and untick the **Anonymous** and **Registered** Roles.



The screenshot shows the configuration interface for the 'AddProjectMember' screen. It includes fields for Name, Description, Title, and Public status, followed by a section for selecting roles.

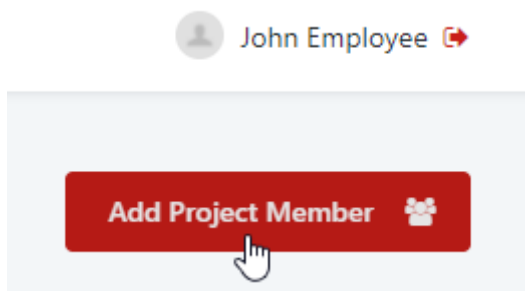
AddProjectMember Screen	
Name	AddProjectMember
Description	
Title	
Public	No
<b>Roles</b>	
Anonymous	<input type="checkbox"/>
Registered	<input type="checkbox"/>
HRManager	<input checked="" type="checkbox"/>

- 3) Repeat the previous step for the remaining three Screens. Guarantee that the Screens have the **Registered** and **HRManager** Roles selected.
- 4) Publish the module to save the application in the server.

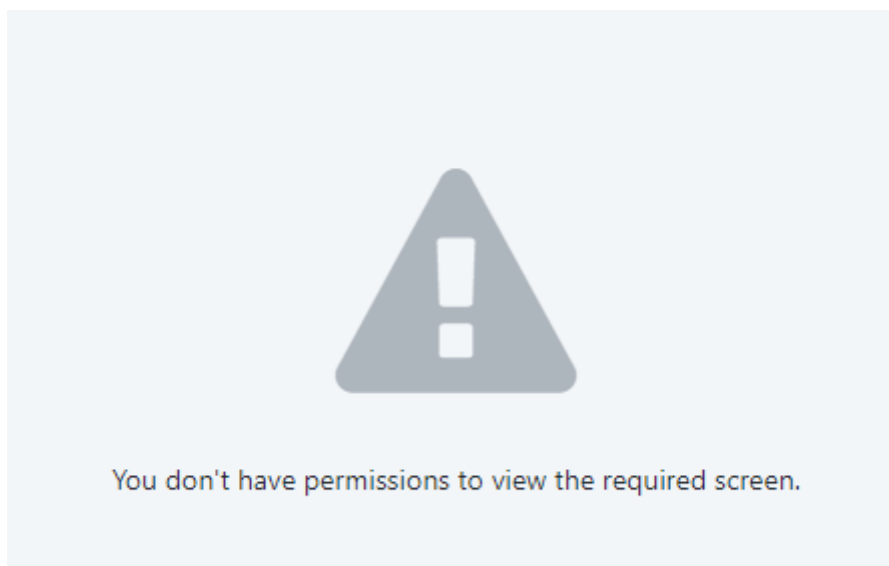
- 5) Open the application in the browser, by clicking on the blue Open in Browser button.

- 6) Make sure you are logged in with the **John Employee** user (Username: *jemployee*, Password: *outsystems*)
- 7) Navigate to the Directory details screen.

- 8) Click the **Add Project Member** Button to access the **AddProjectMember** Screen.



- 9) Since the Screen is not accessible by users that are not HRManagers, we get an invalid permissions message.



- 10) Click on the **Login** button and use the **Julie Manager** user credentials (Username: *jmanager*, Password: *outsystems*).

- 11) Follow the same steps as before, to try adding a new member to the Directory project. This time we are able to access the Screen and add a new employee to a project.

## Add Employee to Project

Employee \*

Patricia Wesley ▼

Role Description \*

Senior Developer

Save