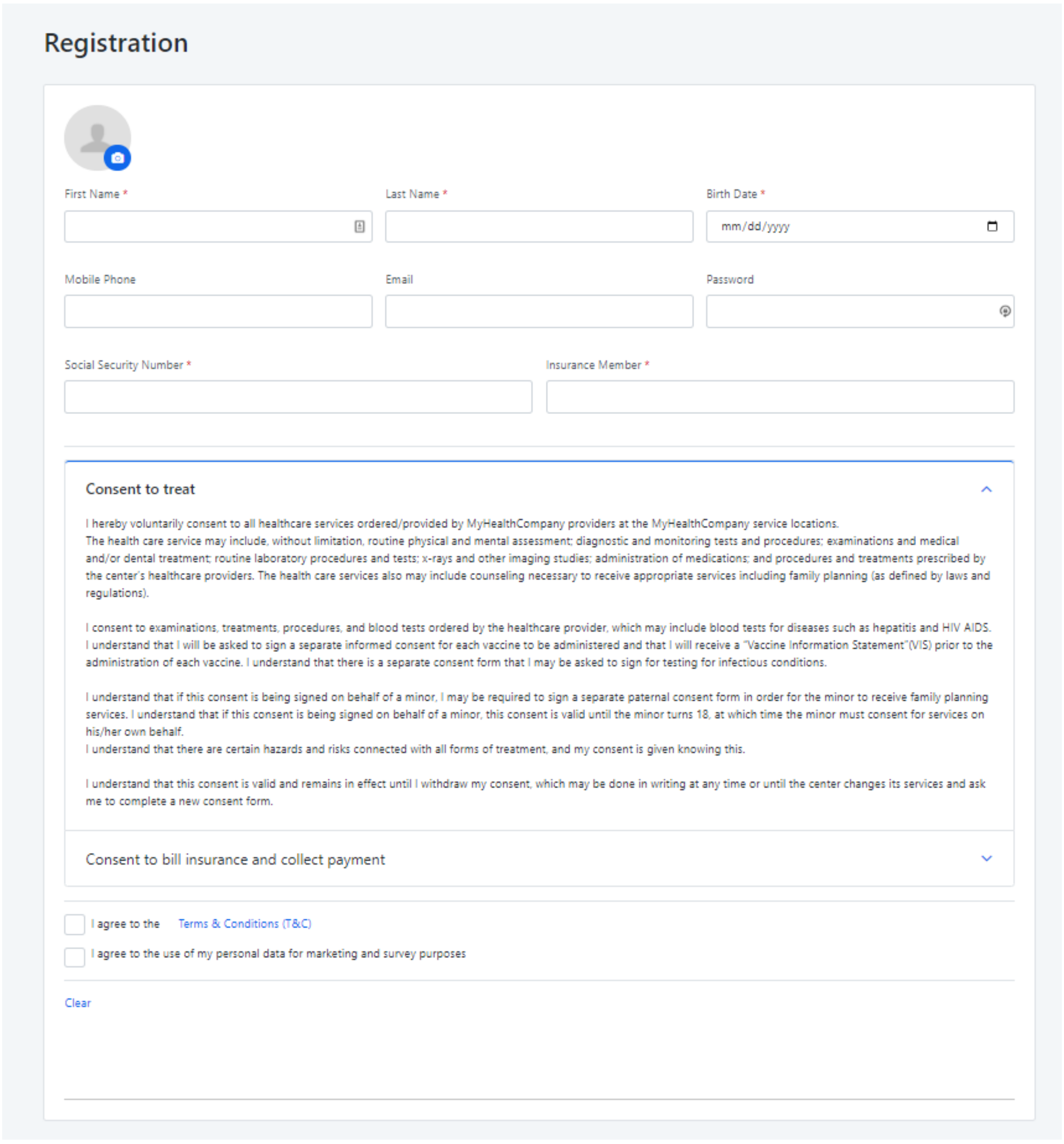
Registration Logic

Table of Contents

Scenario

In the previous tutorials, you have created the Registration Screen for users to register themselves in the app as patients. However, you have only worked on the interface, so the information is not being submitted and processed yet!



Registration

First Name *	Last Name *	Birth Date *
		mm/dd/yyyy
Mobile Phone	Email	Password
Social Security Number *	Insurance Member *	

Consent to treat

I hereby voluntarily consent to all healthcare services ordered/provided by MyHealthCompany providers at the MyHealthCompany service locations.
The health care service may include, without limitation, routine physical and mental assessment; diagnostic and monitoring tests and procedures; examinations and medical and/or dental treatment; routine laboratory procedures and tests; x-rays and other imaging studies; administration of medications; and procedures and treatments prescribed by the center's healthcare providers. The health care services also may include counseling necessary to receive appropriate services including family planning (as defined by laws and regulations).

I consent to examinations, treatments, procedures, and blood tests ordered by the healthcare provider, which may include blood tests for diseases such as hepatitis and HIV AIDS.
I understand that I will be asked to sign a separate informed consent for each vaccine to be administered and that I will receive a "Vaccine Information Statement"(VIS) prior to the administration of each vaccine. I understand that there is a separate consent form that I may be asked to sign for testing for infectious conditions.

I understand that if this consent is being signed on behalf of a minor, I may be required to sign a separate paternal consent form in order for the minor to receive family planning services. I understand that if this consent is being signed on behalf of a minor, this consent is valid until the minor turns 18, at which time the minor must consent for services on his/her own behalf.
I understand that there are certain hazards and risks connected with all forms of treatment, and my consent is given knowing this.

I understand that this consent is valid and remains in effect until I withdraw my consent, which may be done in writing at any time or until the center changes its services and ask me to complete a new consent form.

Consent to bill insurance and collect payment

☐ I agree to the Terms & Conditions (T&C)

☐ I agree to the use of my personal data for marketing and survey purposes

Clear

In this tutorial you will work on three different Actions that will have the logic you need to register a patient:

- **OnClick_Registration**: This Action is executed when the user clicks on the Register button (that will also be created in this tutorial) on the Registration

Screen. It will check if the mandatory fields of the Form are filled out. If they are, the Action proceeds by executing the logic to save the patient information in the database, which will be created on the next Action. If everything goes well, the user is redirected to the Login Screen to insert the login information.

- **Registration_Create**: This Action has all the logic to add the patient information to the database. This start by checking if a user with the inserted username already exists, and create one if it doesn't (defined in the next Action). Then, it should add the Patient, PatientImage and PatientOnboarding information to the database. The Action ends with the grant of the Patient Role to the user being created.

- **User_VerifyAndRegister**: This Action has all the logic related to the user. First, it checks if the user exists. If it does, it creates the user in the database. This Action is used in the **Registration_Create** Action flow.

How to

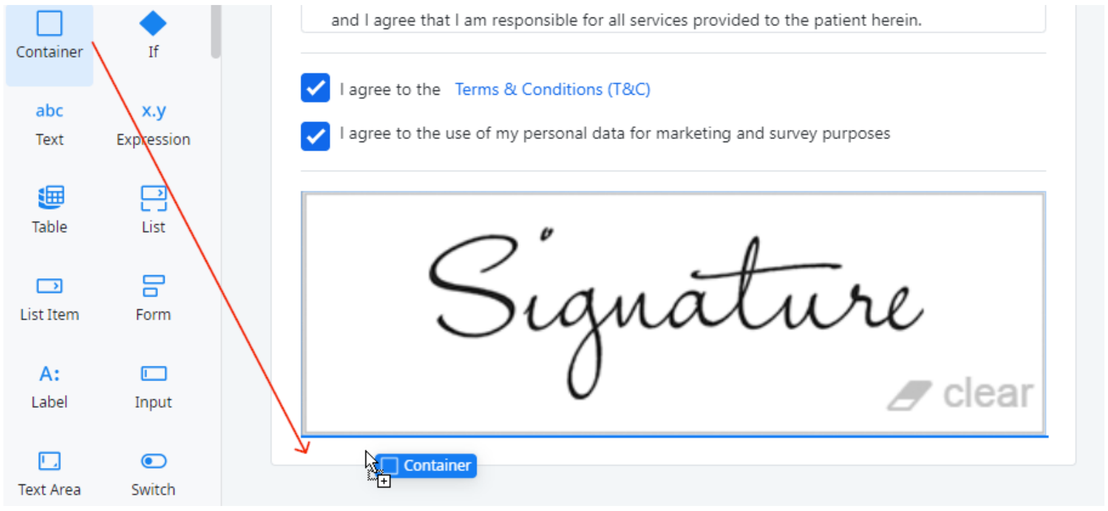
Now that you know the scenario, let's implement it by following this how-to guide! Are you ready? Let's do it together!

Adding a Register Button

As a first step, you need to create the interface so the user can trigger the registration logic. So let's add a registration Button.

1) Open the **Registration** Screen by double-clicking on it, if it is not open yet.

2) Drag and drop a **Container** after the Signature Block.

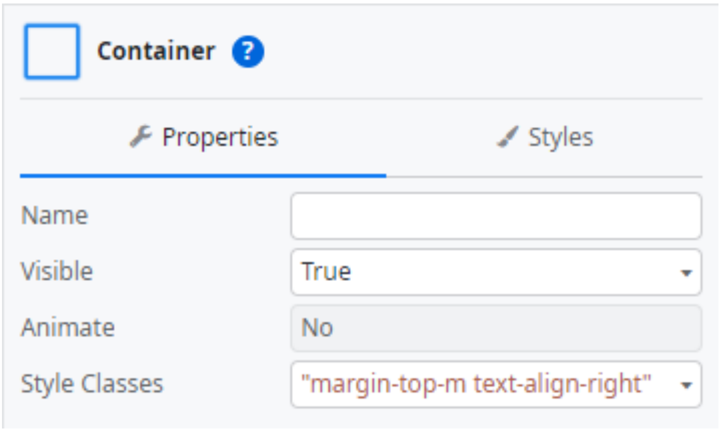


and I agree that I am responsible for all services provided to the patient herein.

I agree to the Terms & Conditions (T&C)

I agree to the use of my personal data for marketing and survey purposes

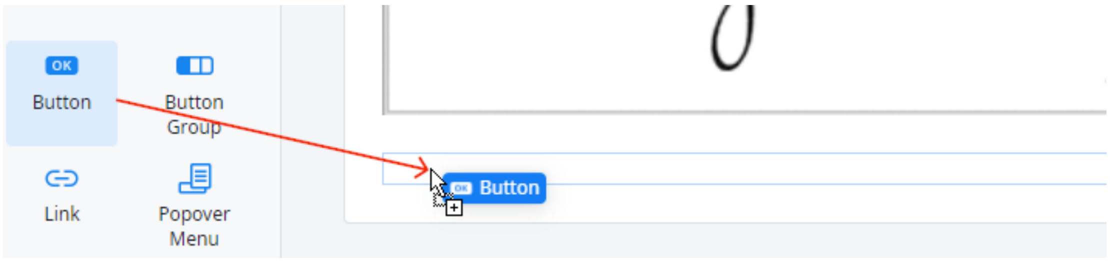
3) Set its **Style Classes** property to `"margin-top-m text-align-right"`.



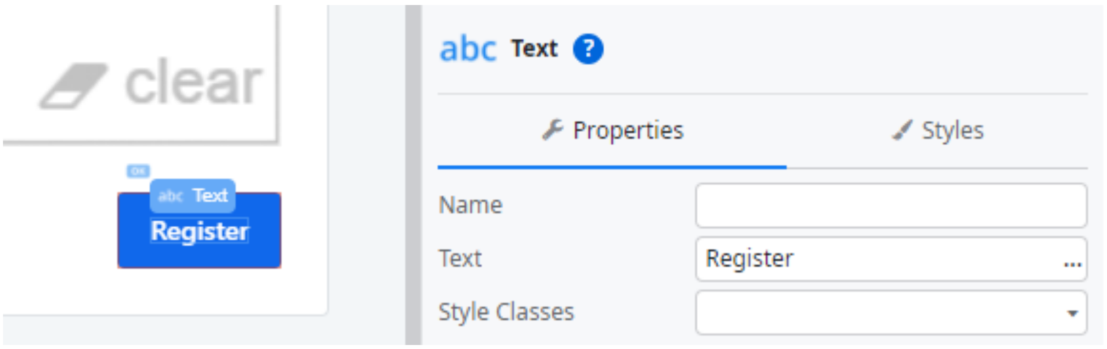
Name	
Visible	True
Animate	No
Style Classes	"margin-top-m text-align-right"

This will align the content of the Container to the right, while it adds a margin to the top.

4) Drag a **Button** from the left sidebar and drop it inside the Container.

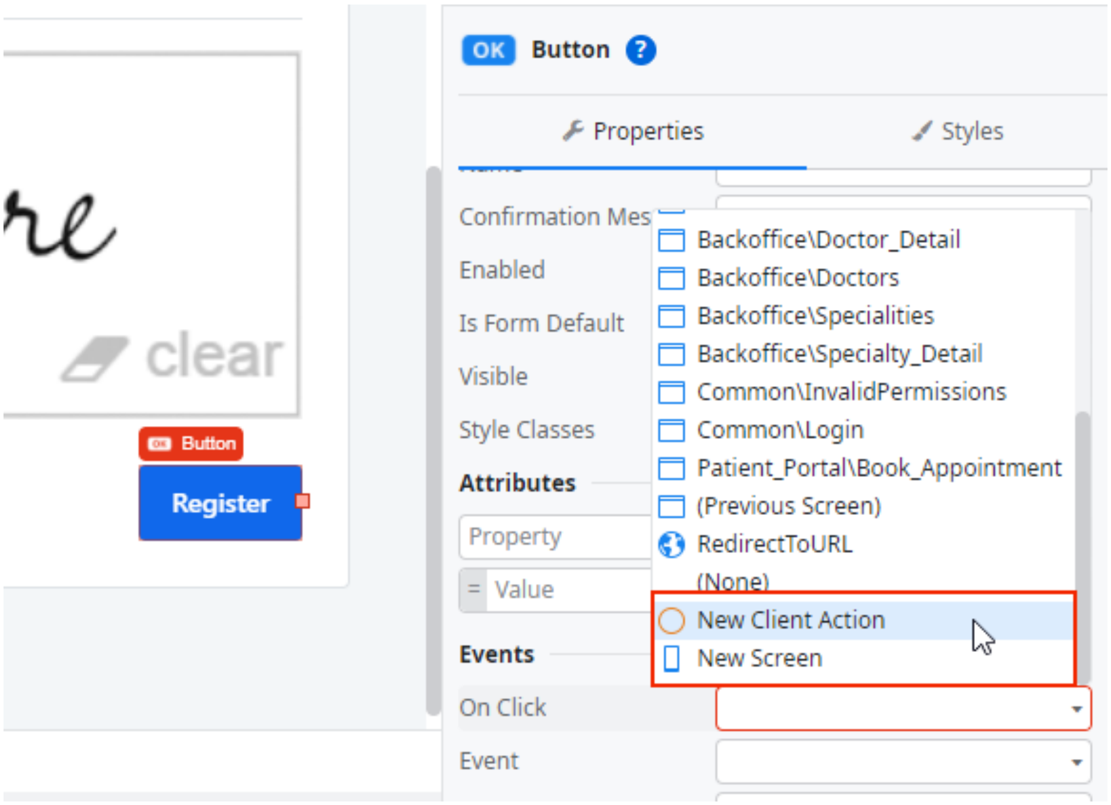


5) Click on the Button and type *Register* in the text.

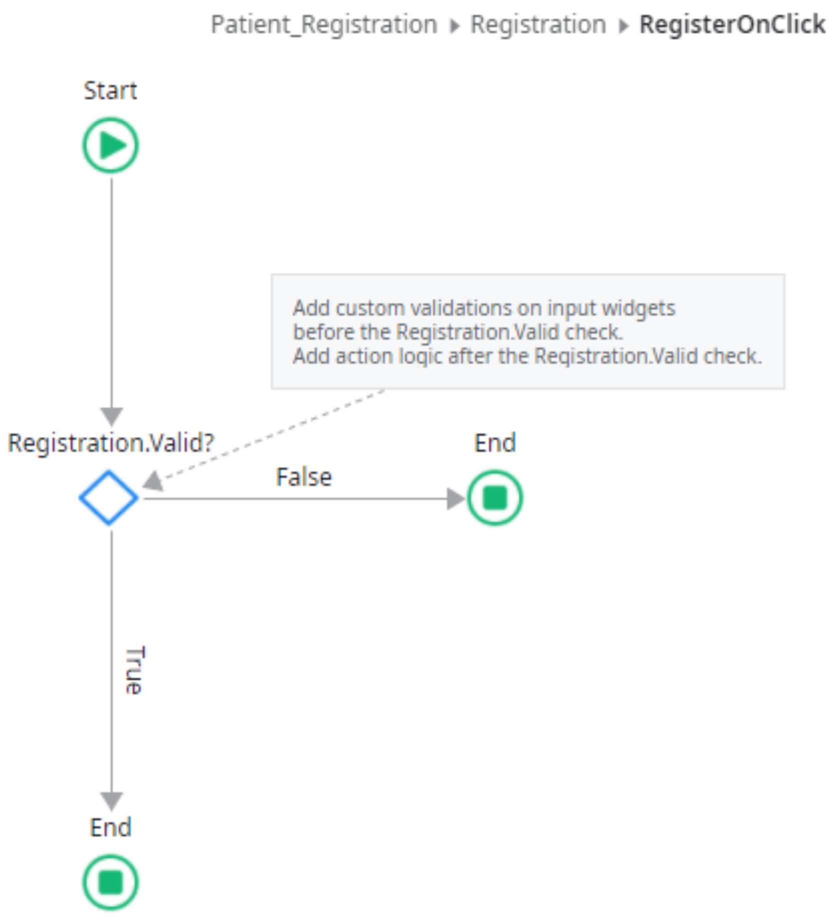


Name
Text: Register
Style Classes

6) The Button has an error because its behavior has not been defined yet. Make sure the Button its selected, and on the properties area, set its **On Click** property to a **New Client Action**.



A new Client Action will be created containing only an If. This If exists to help us validate the data on the Form. In case the Form is filled out correctly (for instance, all mandatory fields are filled), the Action will follow the flow on the True branch. Otherwise, it will follow the flow on the False branch.


Patient_Registration ▸ Registration ▸ RegisterOnClick
Start
Add custom validations on input widgets before the Registration.Valid check.
Add action logic after the Registration.Valid check.
Registration.Valid?
False
End
True
End

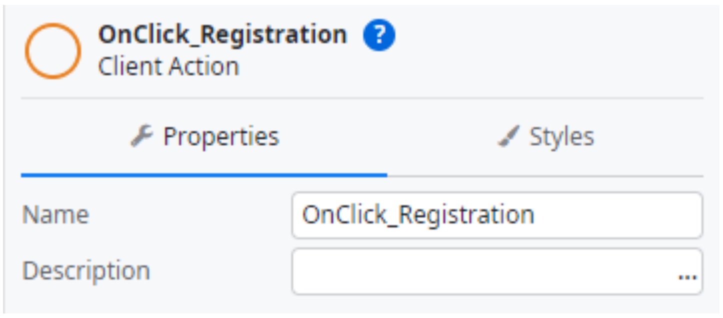
Starting the Registration Logic

When a user clicks on the Registration button, the Action must have all the logic to save the patient information in the database, which involves the Patient, PatientImage, PatientOnboarding and User Entities. Let's do this one step at a time, starting to define what happens when the data in the Form is not valid!

Data is Not Valid

If the data is not valid, the user should see an error message on the Screen.

1) Set the Action's **Name** to *OnClick_Registration*.

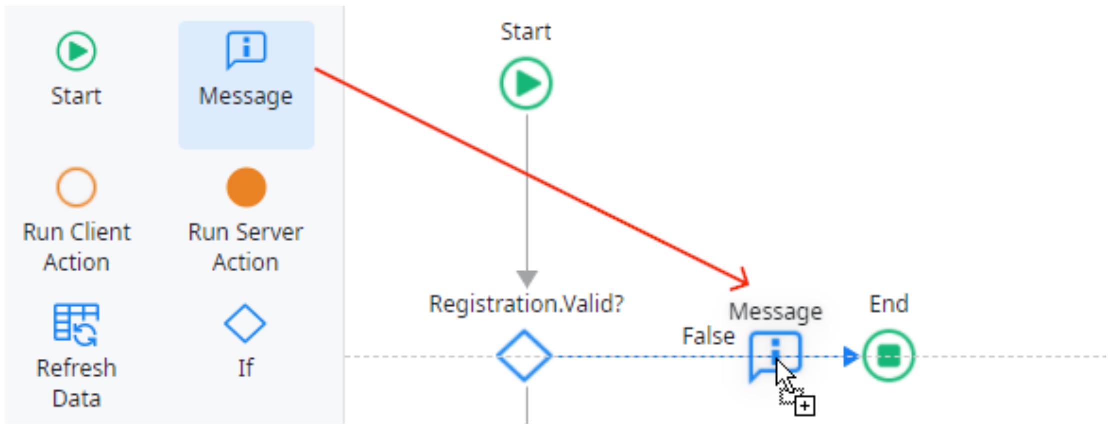


OnClick_Registration
Client Action

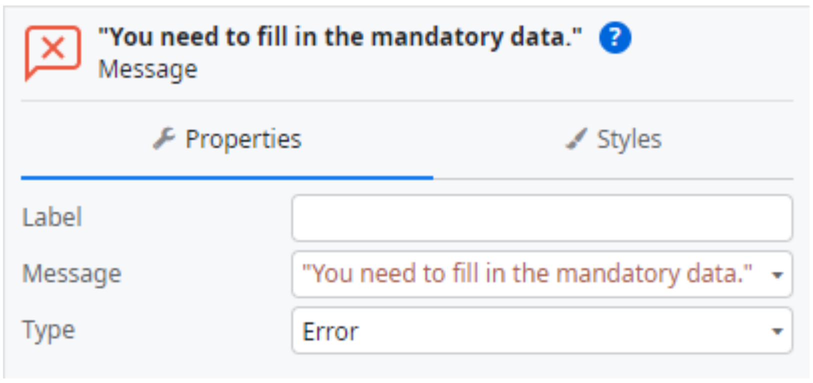
Properties | Styles

Name	OnClick_Registration
Description	

2) Drag a **Message** from the left sidebar and drop it in the False branch.



3) Set the **Message** property to *"You need to fill in the mandatory data."* and the **Type** property to **Error**.



"You need to fill in the mandatory data."
Message

Properties | Styles

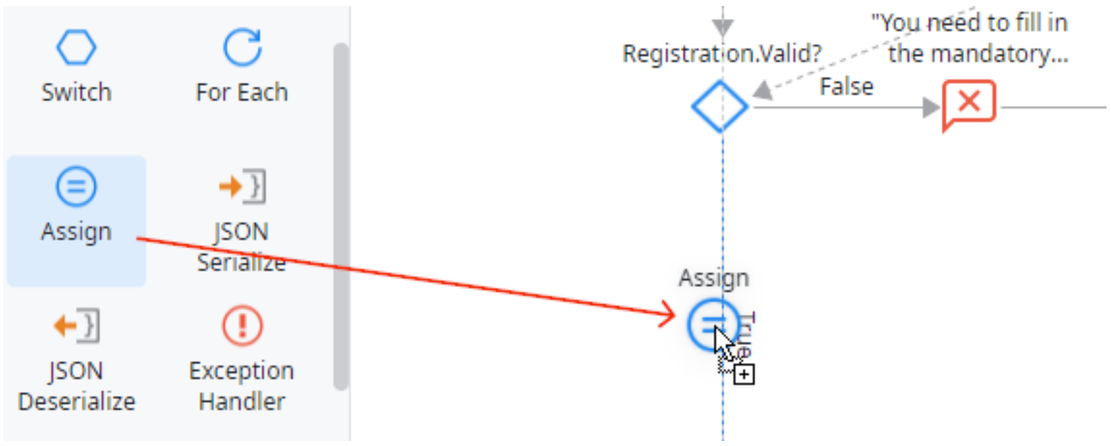
Label	
Message	"You need to fill in the mandatory data."
Type	Error

Ok, so now, let's focus on the True branch of the If, meaning that the Form data submitted by the user is valid.

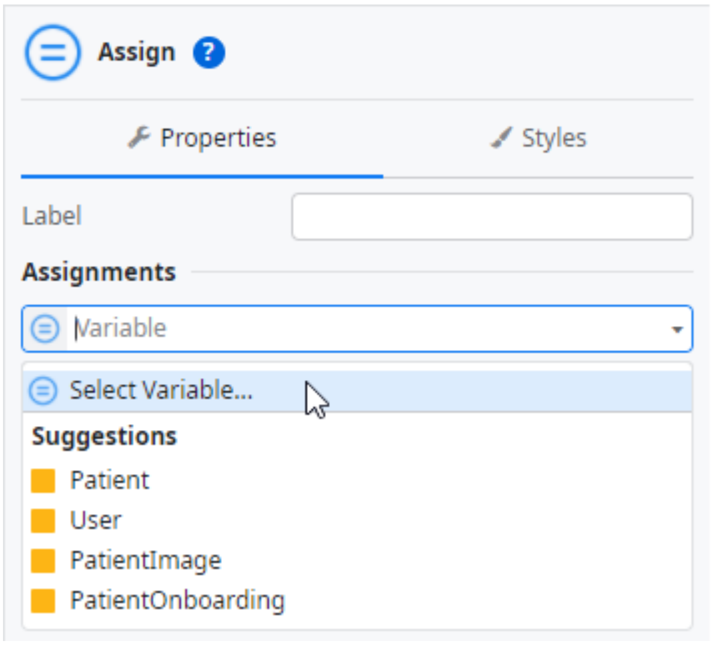
User Entity Fields

The first data that you will save in the database concerns the User. Don't forget that a patient is also a User of the application. Whenever you want to save data in the database, you should determine if you already have all the information you need. In this case, the user of the app fills the mobile phone, email and password, so you need to programmatically set other values, such as the Name, Username, CreationDate and Is_Active, before saving the User in the database.

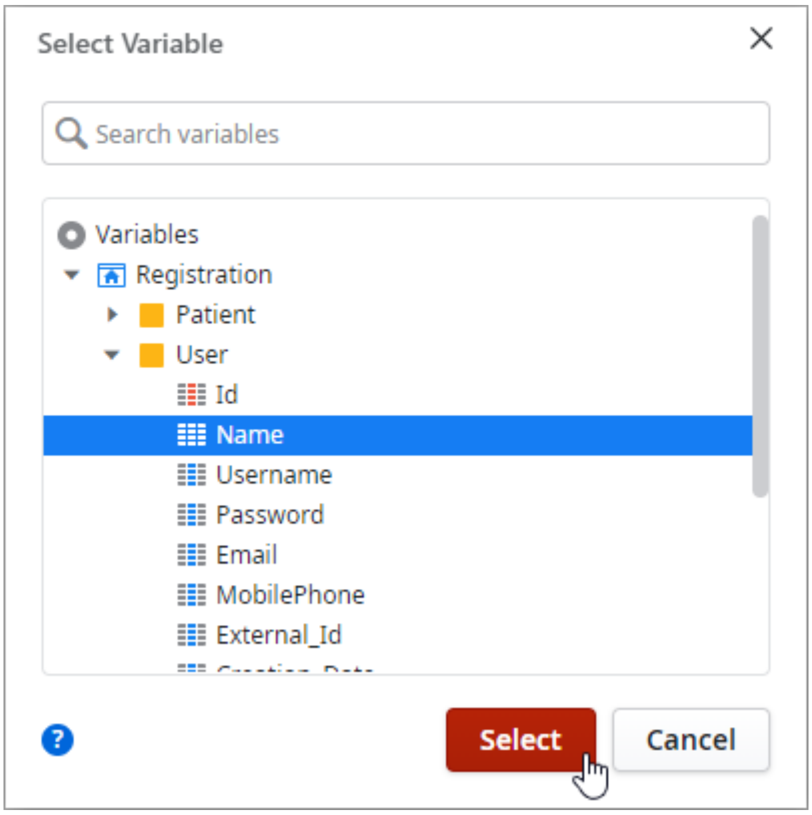
1) Drag and drop an **Assign** to the True branch of the If, to set the missing fields for the user record.



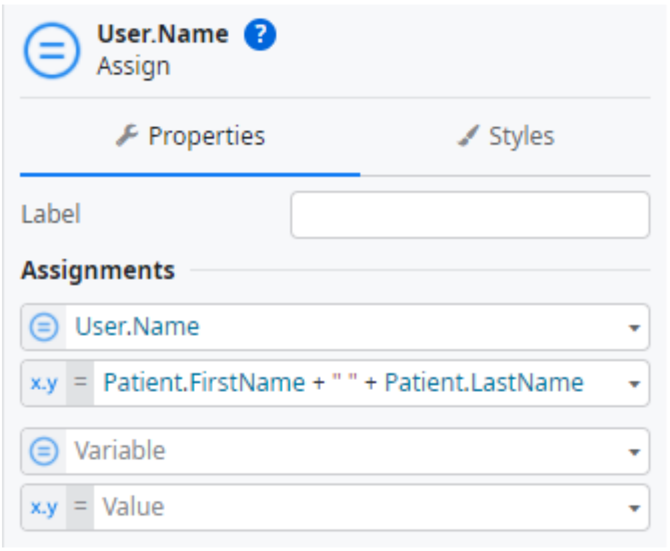
2) Click on the dropdown of the **Variable** property and click on the **Select Variable...** option.



3) Expand the **User** Local Variable and select the **Name**.



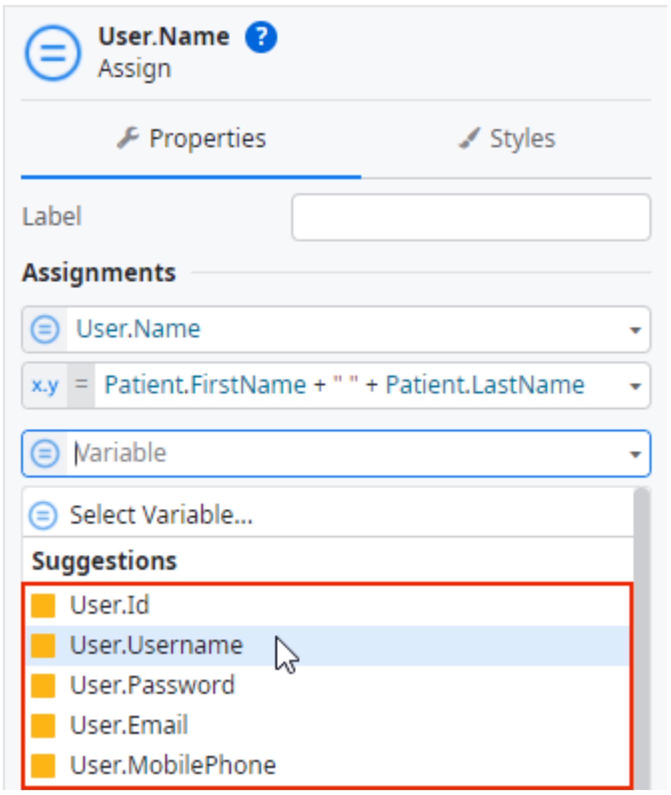
4) Since the User has only a Name field, we need to join together the patient's first name and last name. So, set the **Value** to `Patient.FirstName + " " + Patient.LastName`



User.Name = Patient.FirstName + " " + Patient.LastName

You still need to Assign the rest of the User's fields. You can use the same Assign for that.

5) Select the **User.Username** in the dropdown of the second Variable.



User.Name
Assign

Properties | Styles

Label:

Assignments

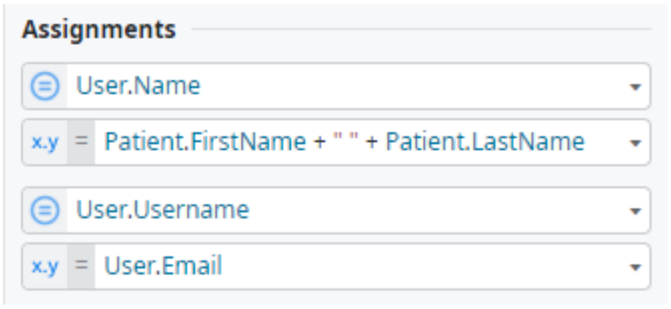
User.Name
x.y = Patient.FirstName + " " + Patient.LastName

Variable

Select Variable...
Suggestions
- User.Id
- User.Username
- User.Password
- User.Email
- User.MobilePhone

Did you notice that now the attributes of the User variable appear in the suggestions? This happens because you already used one of the attributes in the first Assign. OutSystems improves the suggestions based on what you did before.

6) Add the value **User.Email** to the **Value** property. Your assignments should look like the image below:

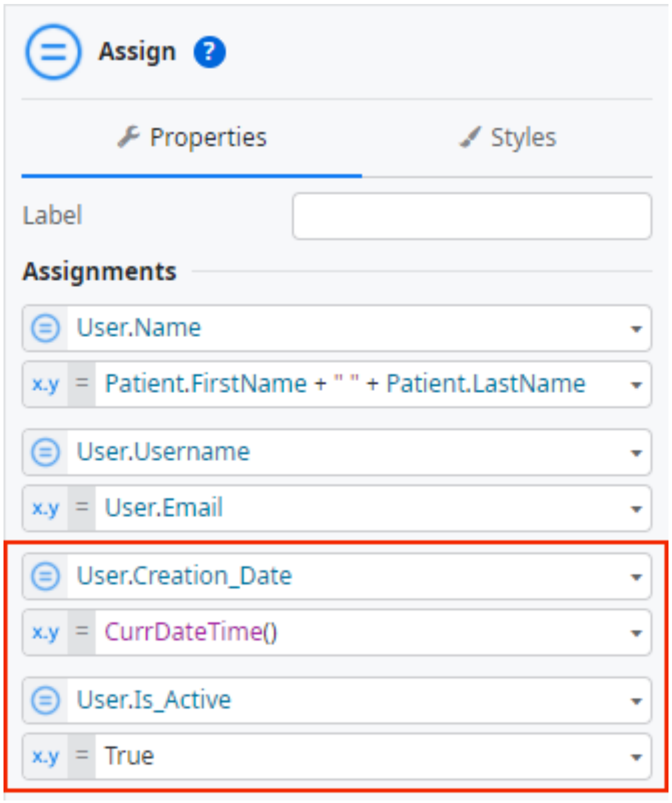


Assignments

User.Name
x.y = Patient.FirstName + " " + Patient.LastName

User.Username
x.y = User.Email

7) To finish, repeat the previous steps and select the variables **User.Creation_Date** and **User.Is_Active**. Set their values to *CurrDateTime()* and *True*, respectively.



User.Name = Patient.FirstName + " " + Patient.LastName
User.Username = User.Email
User.Creation_Date = CurrDateTime()
User.Is_Active = True

The CurrDateTime() function already exists in OutSystems and returns the current date and time.

Writing the User Information in the Database

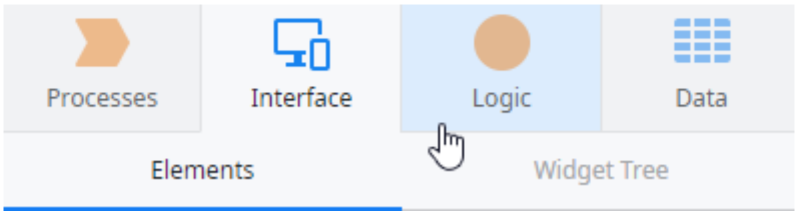
The Registration Action you just created needs to have the logic to save the information in the database. Since the database is hosted in a server, that logic is defined in what we call Server Actions in OutSystems. They are like a regular Action with a flow, like the one you're working now, but it runs server-side logic.

So, to define the logic that you need to save the information in the database, you will create two Server Actions: **User_VerifyAndRegister** and **Registration_Create**. The first one will check if the user being registered already exists, and creates one if it doesn't, while the second will have the logic to save the Patient, PatientImage and PatientOnboarding data in the database.

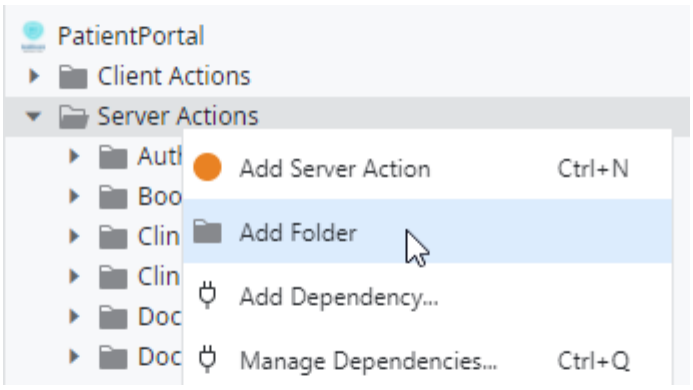
Let's start by creating the User_VerifyAndRegister Action!

Creating a new Server Action

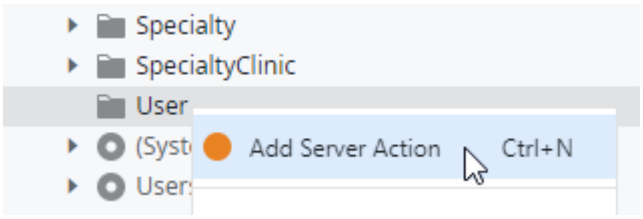
1) Switch to the **Logic** tab.



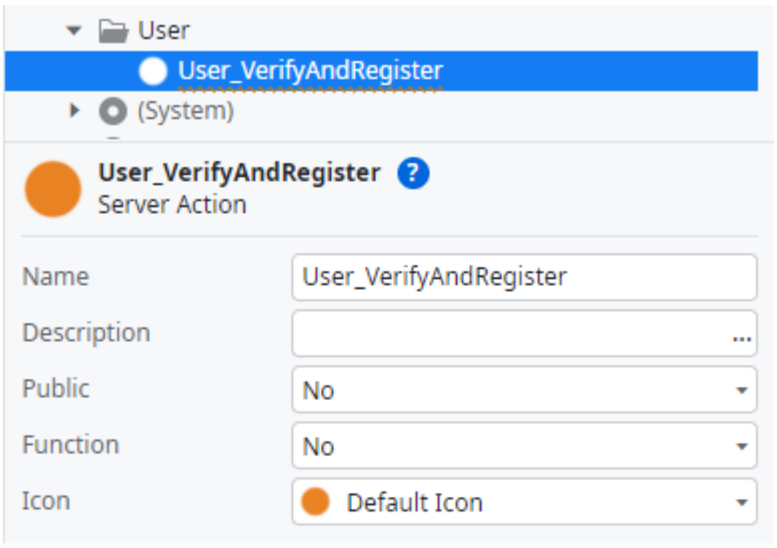
2) Right-click on the **Server Actions** folder and select **Add Folder**. Set its name to *User*.



3) Right-click on the new folder and select **Add Server Action.**



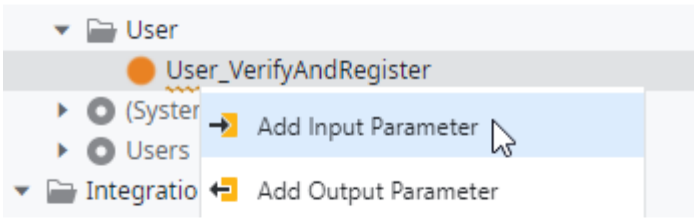
4) Set its **Name** to *User_VerifyAndRegister*.



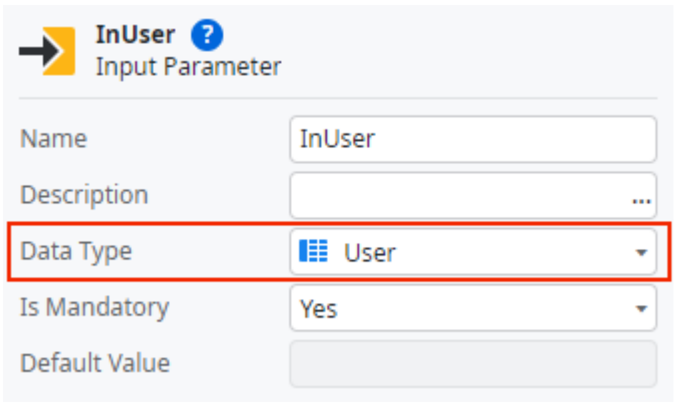
User_VerifyAndRegister
Server Action

Name	User_VerifyAndRegister
Description	
Public	No
Function	No
Icon	Default Icon

5) Right-click on the Action and select **Add Input Parameter**.



6) Name it *InUser* and make sure the **Data Type** is set to **User**.



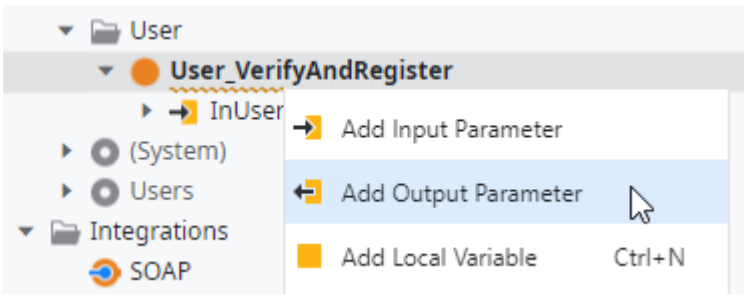
InUser
Input Parameter

Name	InUser
Description	
Data Type	User
Is Mandatory	Yes
Default Value	

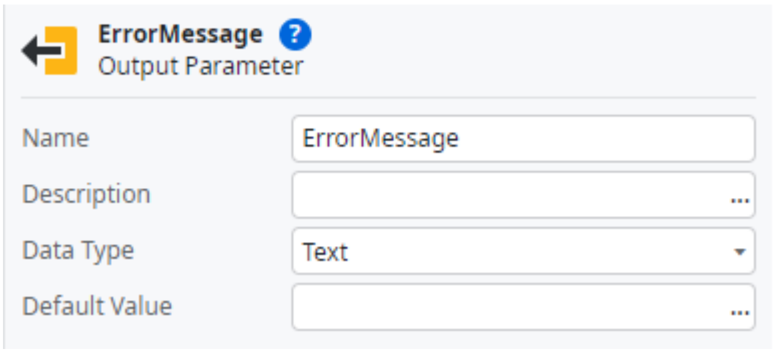
You know have an Action that expects a User record as input. What is this input for? Well, it will receive the information related to the user from the Registration Form, so it can be saved on the database.

We also need two output parameters. One to return the UserId, when the user is created, and one to return an error message, if the user already exists.

7) Right-click on the User_VerifyAndRegister Action on the right sidebar, then select **Add Output Parameter**.

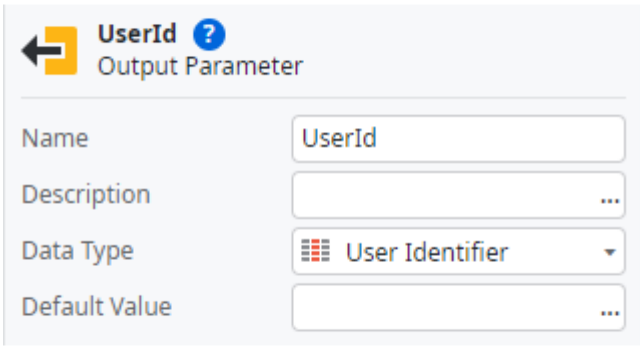


8) Set its **Name** to *ErrorMessage* and change the **Data Type** to **Text**.


ErrorMessage
Output Parameter

Name	ErrorMessage
Description	
Data Type	Text
Default Value	

9) Repeat the previous steps to create another Output Parameter. Set its **Name** to *UserId* and the **Data Type** should automatically change to **User Identifier**.


UserId
Output Parameter

Name	UserId
Description	
Data Type	User Identifier
Default Value	

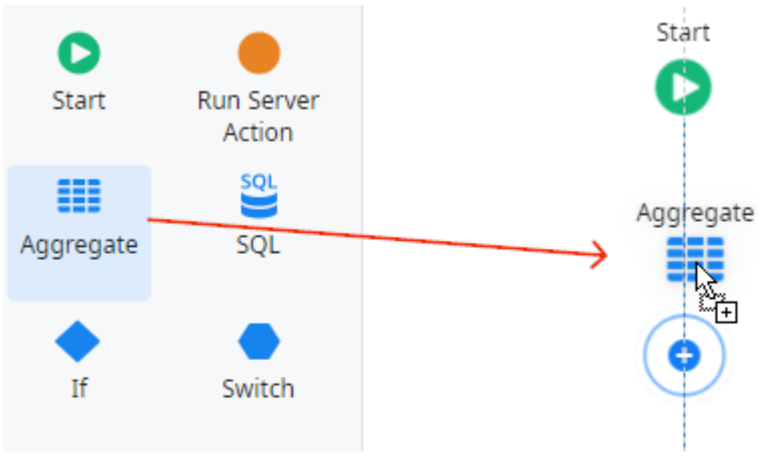
You're good to go. Let's start building the logic.

Fetching the User

The first thing you need to do about the logic is to search for the user in the database, using a query.

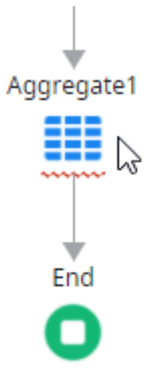
1) Double-click the new User_VerifyAndaRegister Action to open its flow.

2) Drag an **Aggregate** from the left sidebar and drop it inside the Action Flow.

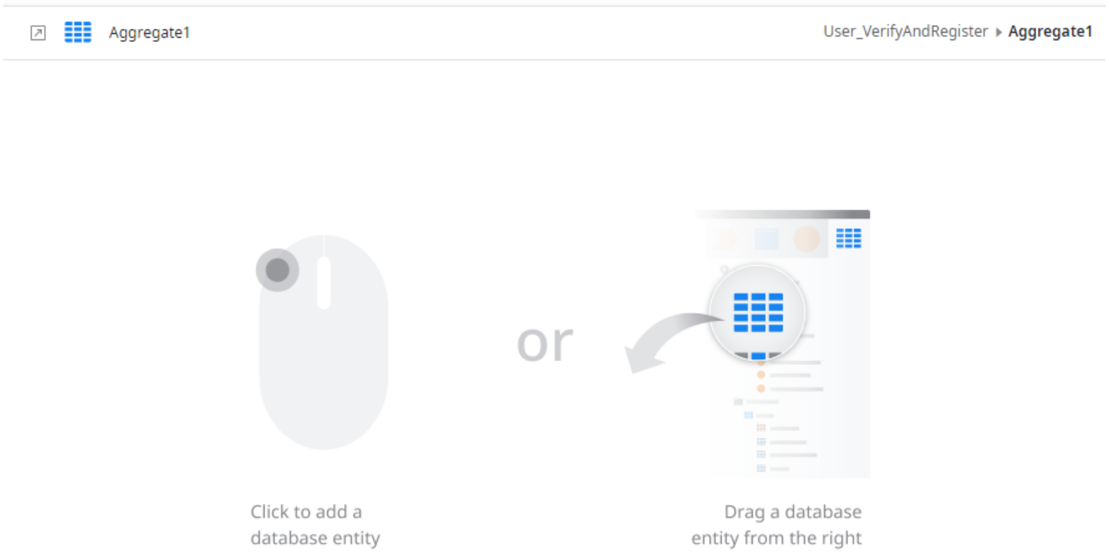


An Aggregate is a visual way to represent a query to the database.

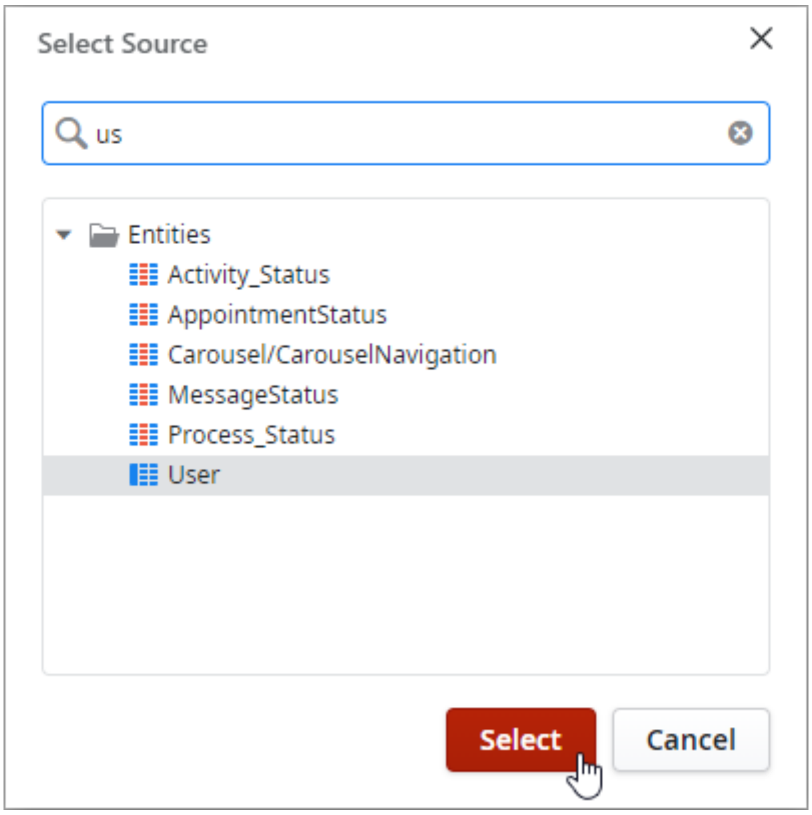
3) The Aggregate will have an error because it wasnt defined yet. Double-click on it to open.



4) When the Aggregate is open, click on the center of the screen to select a Source.

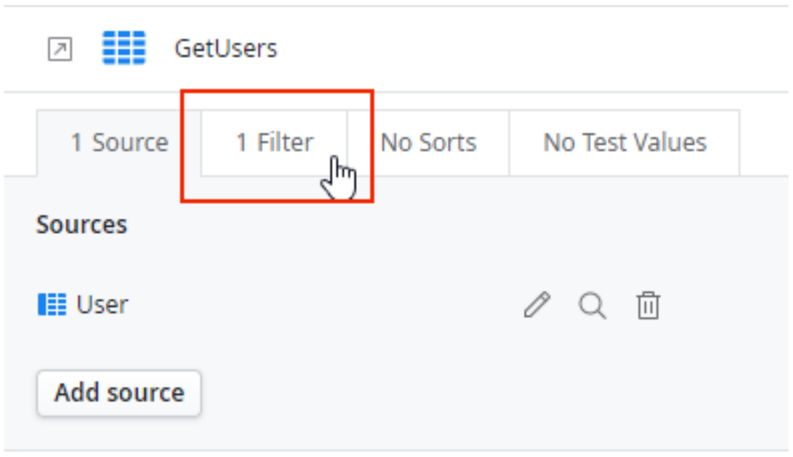

Aggregate1
User_VerifyAndRegister ▸ Aggregate1
or
Click to add a database entity
Drag a database entity from the right

5) Find the **User** Entity in the dialog, choose it and click **Select** to close the dialog.

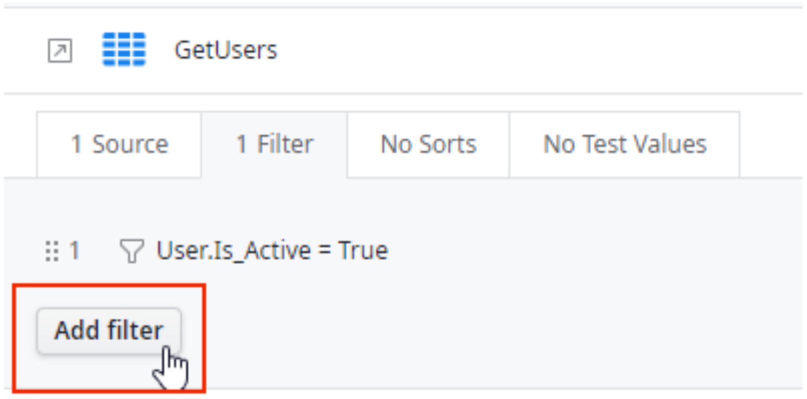


With this, you're saying that the Aggregate is fetching all the users saved in the User Entity. But that's not what you want. You want to search for a specific User.

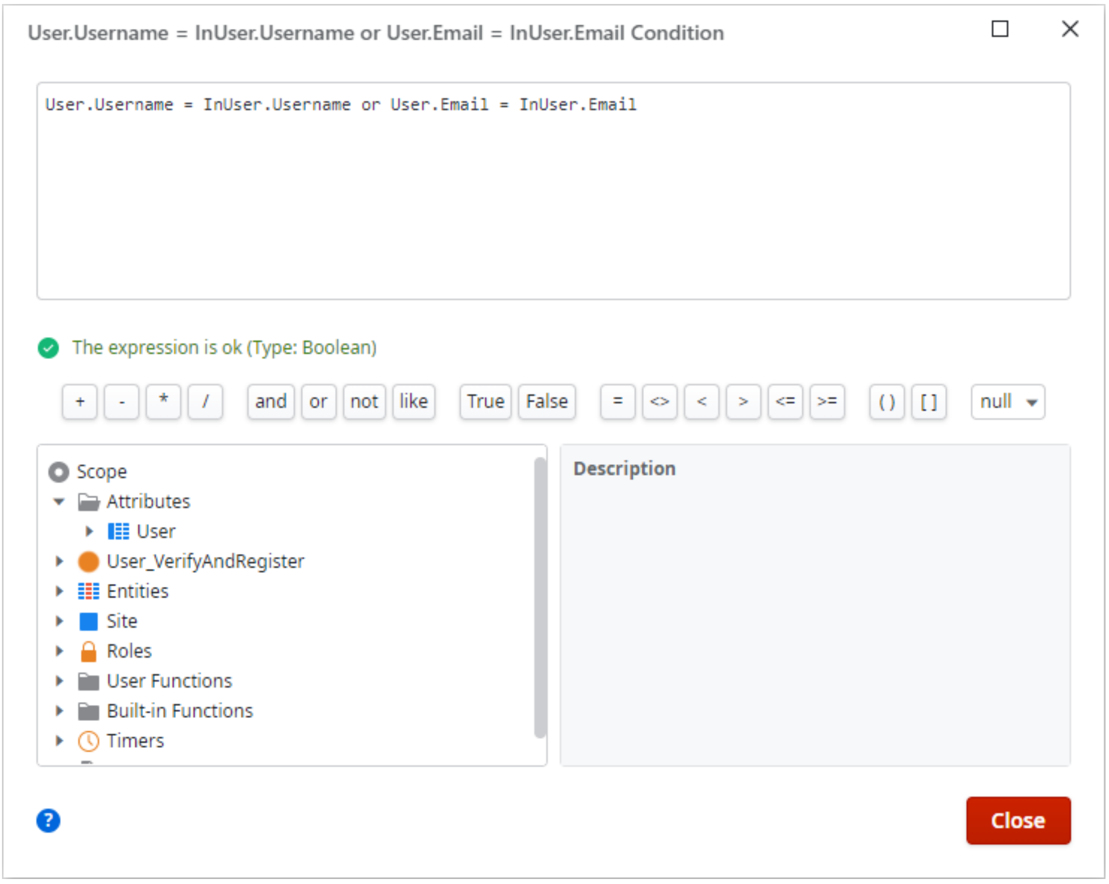
6) Click on the **Filter** tab of the Aggregate.



7) Click on **Add filter**



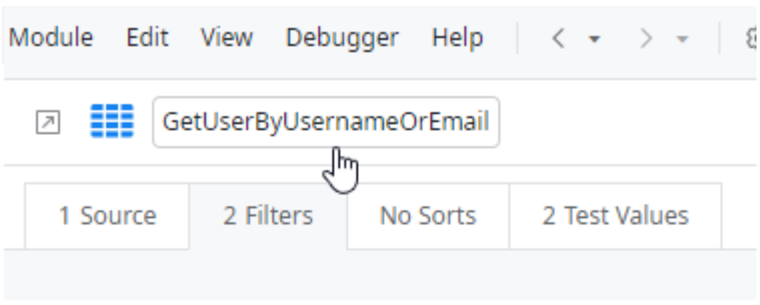
8) Add the following condition: `User.Username = InUser.Username or User.Email = InUser.Email`


User.Username = InUser.Username or User.Email = InUser.Email Condition
The expression is ok (Type: Boolean)
+ - * / and or not like True False = <> < > <= >= ( ) [ ] null
Scope
Attributes
User
User_VerifyAndRegister
Entities
Site
Roles
User Functions
Built-in Functions
Timers
Description
Close

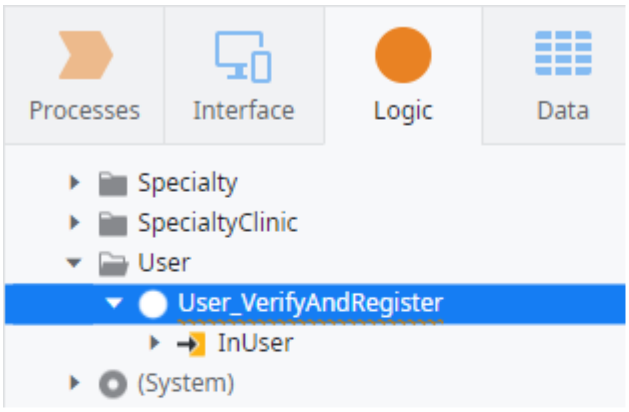
```
User.Username = InUser.Username or User.Email = InUser.Email
```

Like the name of the tab says, you are filtering the result of the Aggregate. With this expression, you're searching for a user with a username or email that matches the data passed from the input parameter InUser, that you created earlier in this tutorial. Don't forget that the InUser Input Parameter will receive the User information submitted in the Registration Form.

9) Rename the Aggregate by clicking on its name and by typing *GetUserByUsernameOrEmail*.



10) Go back to the **User_VerifyAndRegister** Server Action by double-clicking on it.

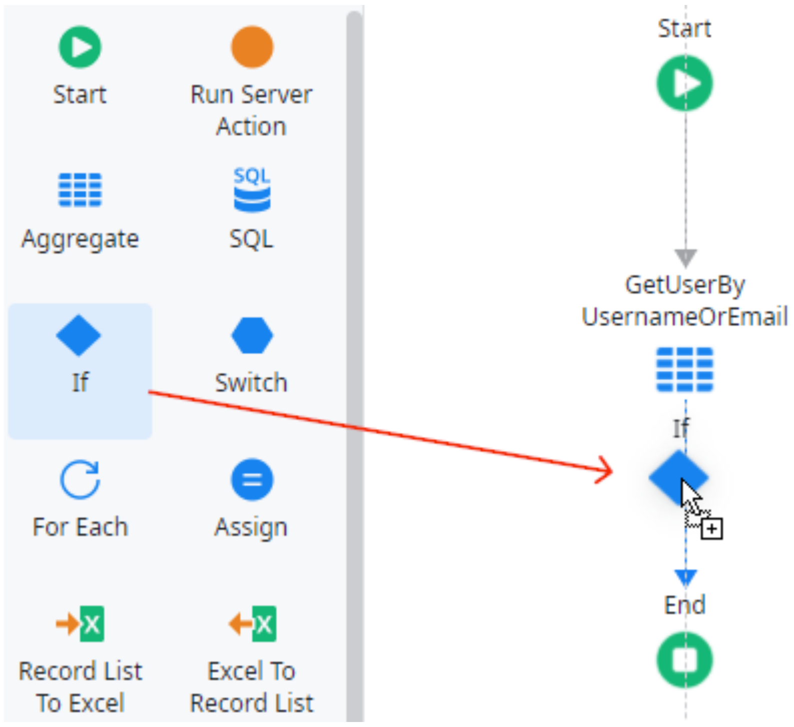


So, now you are querying the database for a user with that username or email. What happens if the user exists?

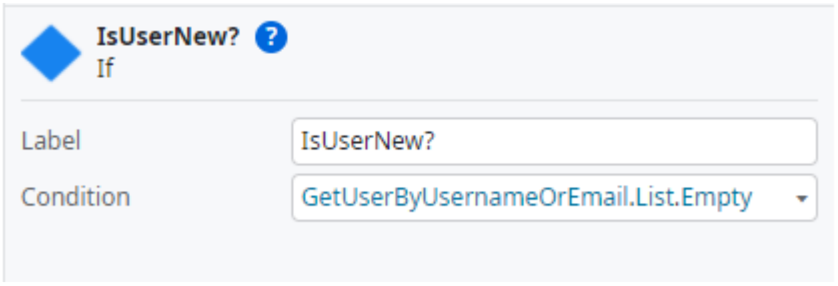
If the User Exists

If the user already exists, the Action should return an error stating that the username or email is already in use.

1) Drag and drop an **If** below the Aggregate.



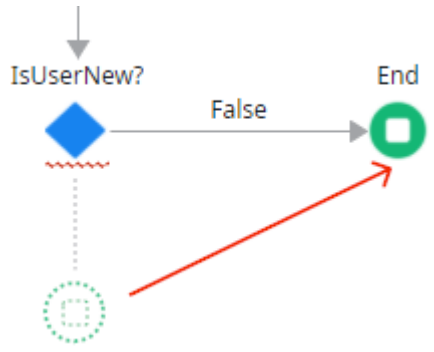
2) Set the **Label** property to *IsUserNew?* to help us identify what the If is doing. Then, set the **Condition** to *GetUserByUsernameOrEmail.List.Empty*.



Label	IsUserNew?
Condition	GetUserByUsernameOrEmail.List.Empty

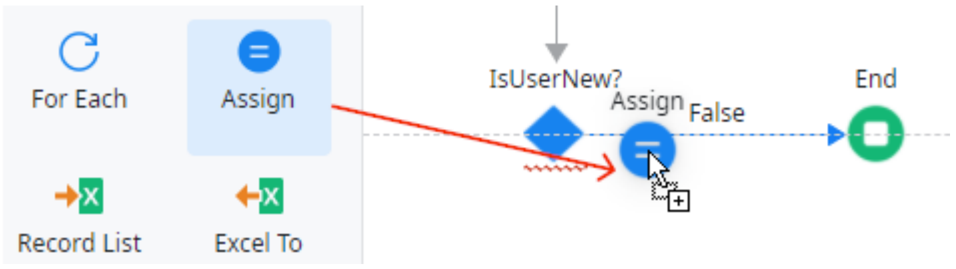
This If asks if the Aggregate is empty. If it is, the Condition is evaluated to true, meaning the user does not exist. If it is not, the Condition is evaluated to false, meaning the user already exists.

3) Move the **End** node of the False branch to the right of the If.

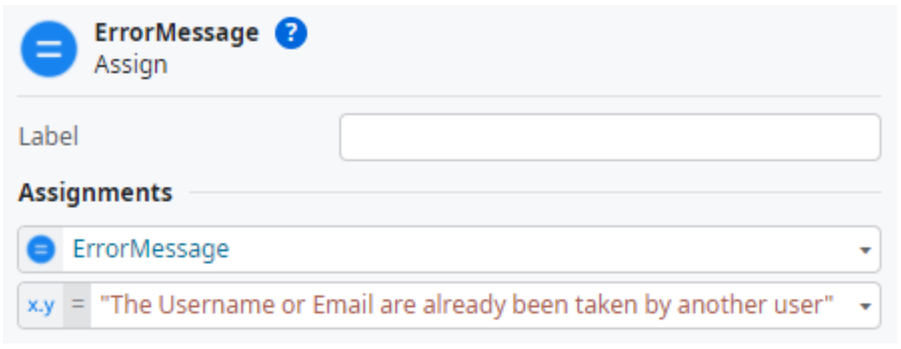


This is optional, but since the If as a true and false branch, it's a good practice to expand one branch down and another one to the right, to keep the flow easy to read.

4) Drag and drop an **Assign** between the If and the End node in the False branch.



5) Select the **ErrorMessage** output parameter in the **Variable** property and add the following message *"The Username or Email are already been taken by another user"* as its **Value**.



ErrorMessage
Assign

Label

Assignments

ErrorMessage

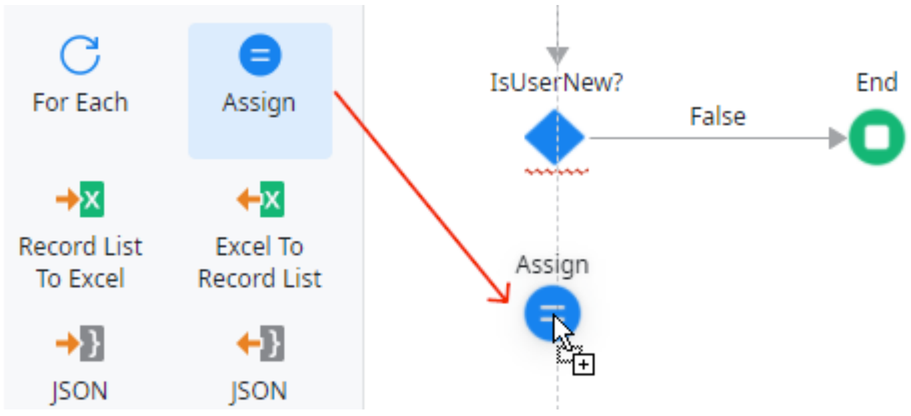
x.y = "The Username or Email are already been taken by another user"

Now let's work on the True branch, meaning you will build the logic to save the User in the database.

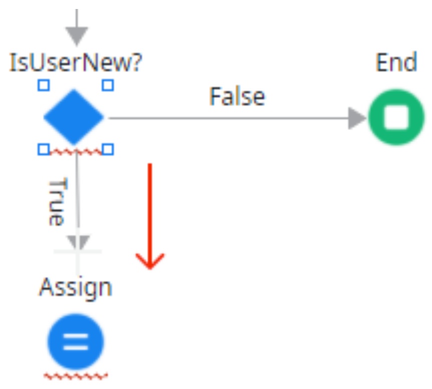
Encrypting the Password

If the user does not exist, the first step is to encrypt the password. You don't want to store a password as open text in the database right?

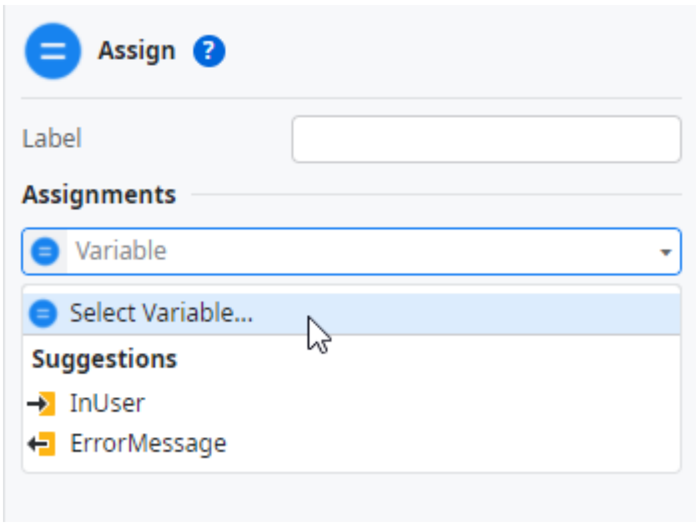
1) Drag another **Assign** under the If.



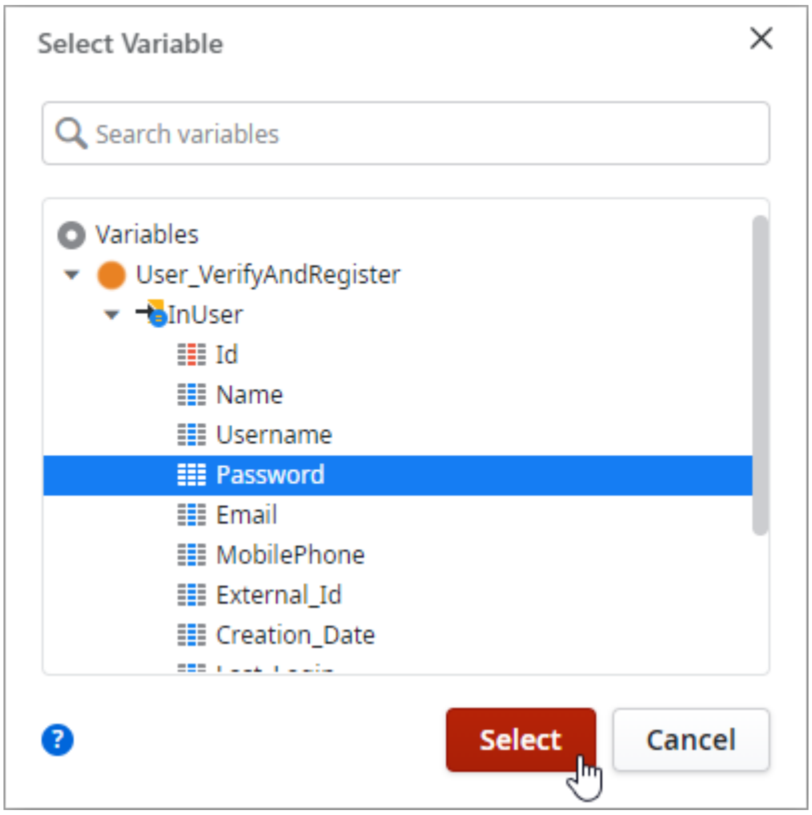
2) Connect the **If** to the Assign to create the True branch.



3) Click on the Assign, then open the **Variable** dropdown and choose the **Select Variable** option.

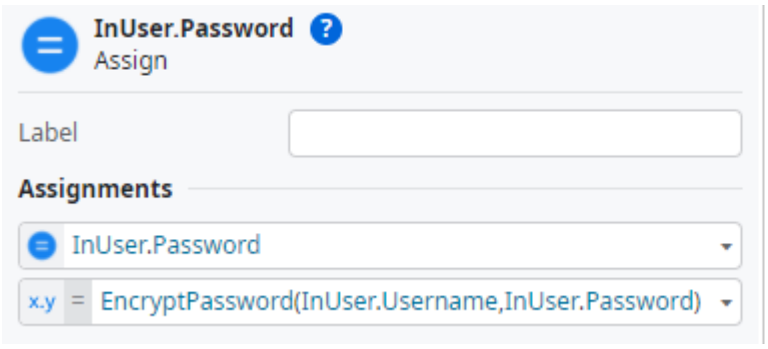


4) Expand the **InUser** Local Variable, select the **Password** attribute, then click on **Select** to close.



5) Add the following expression to the **Value**:

```
EncryptPassword(InUser.Username,InUser.Password)
```



InUser.Password

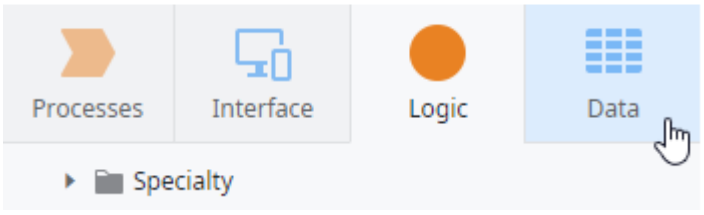
EncryptPassword(InUser.Username,InUser.Password)

And the password is encrypted!

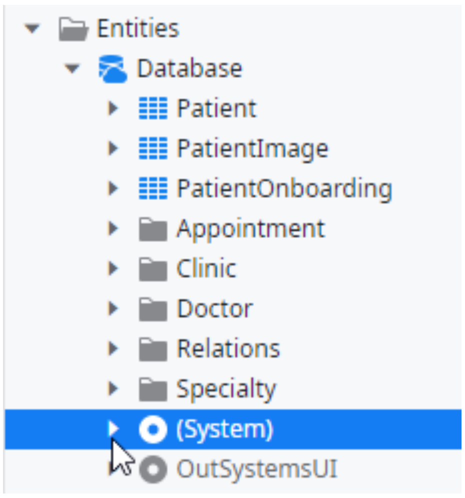
Creating a New User

Now, you can create the user in the database. It would be nice to have something that you can just simply drag and drop, and the platform does the rest for you, right? Well, and there is!

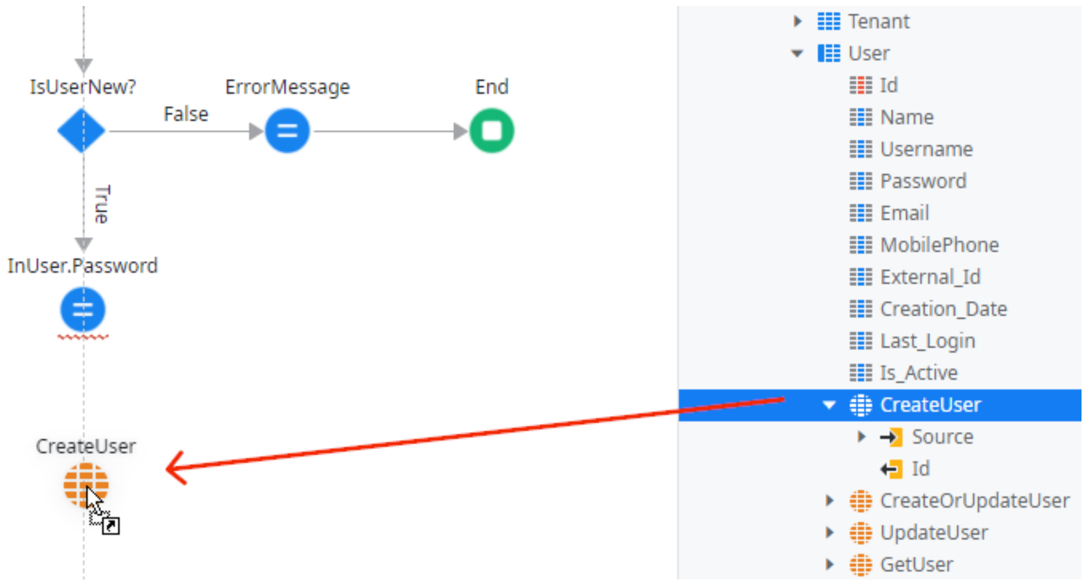
1) Switch to the **Data** tab.



2) Expand the **System** folder, inside the **Database** section, to find the User Entity. below the Entities in the Database folder so you can find the User Entity.

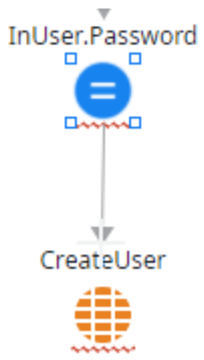


3) Expand the **User** Entity, then select the Action **CreateUser** and drag and drop it under the Assign for the password.

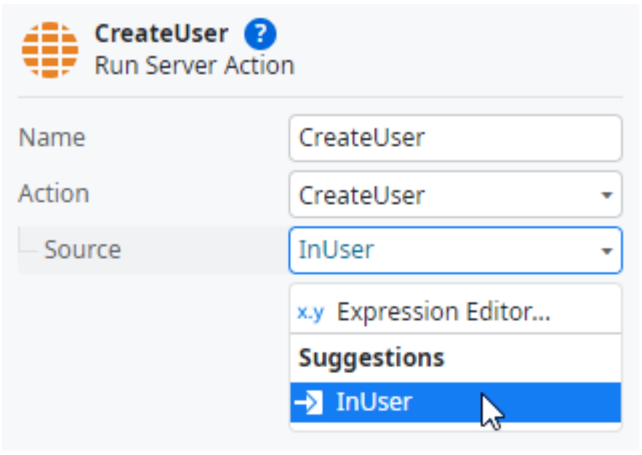


This Action creates a new user in the database. You just need to pass an input parameter with the User information (remember the InUser Input Parameter?).

4) Connect the Assign to the CreateUser Action.



5) Click on the **CreateUser** Action and select the **InUser** Input as the **Source**.

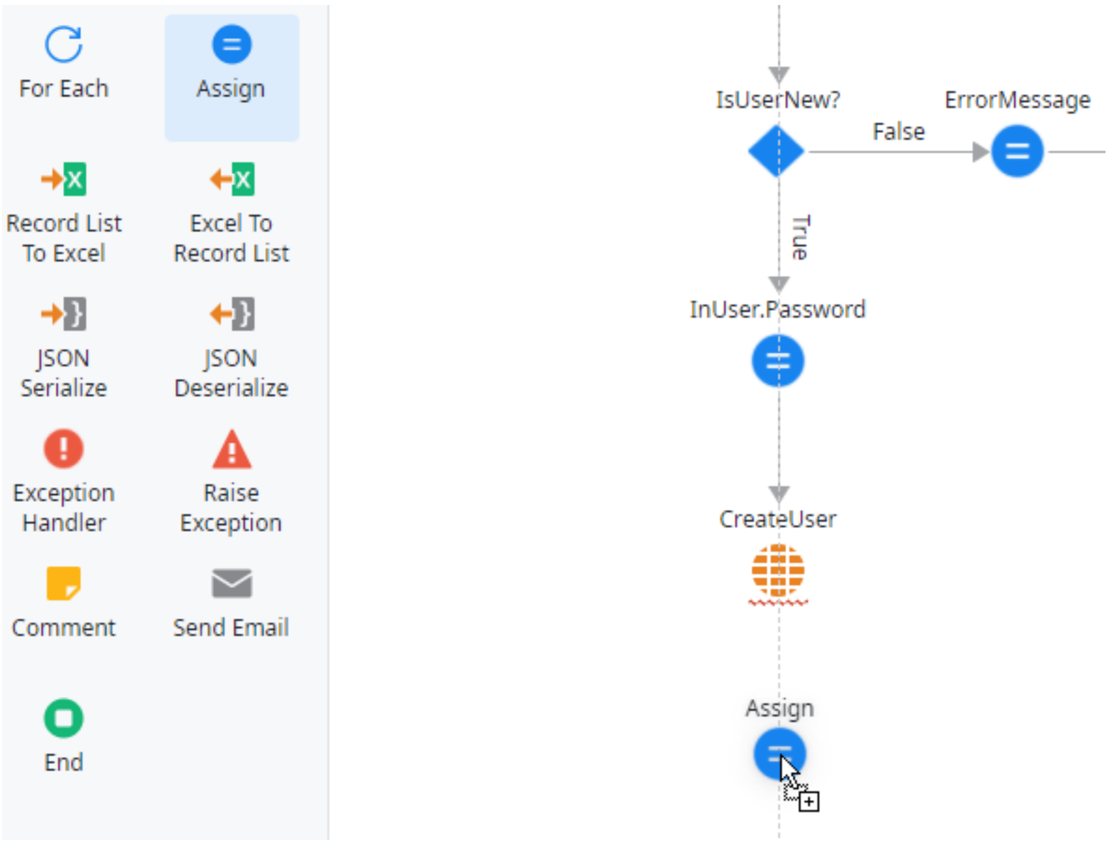


CreateUser
Run Server Action

Name	CreateUser
Action	CreateUser
Source	InUser

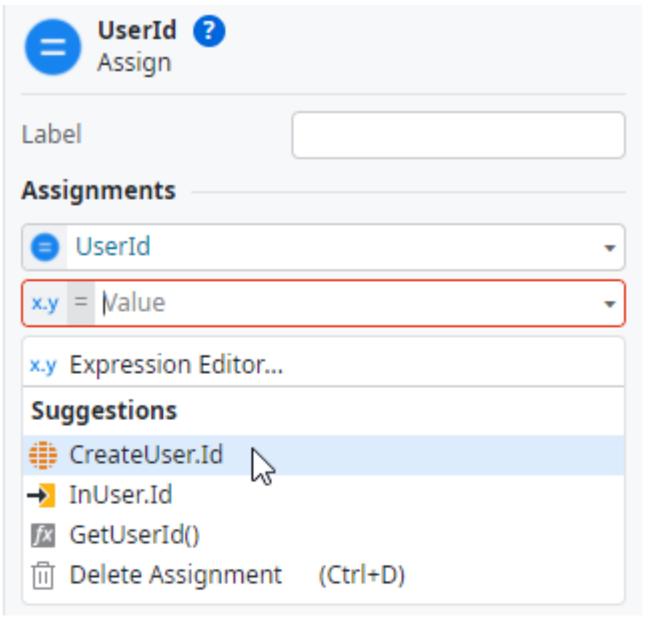
Since we know the InUser will have the information of the user inserted in the Registration Form, you can leverage it and pass it to the CreateUser Action. The Action will do the rest.

6) Drag and drop an **Assign** under the CreateUser Action and connect them.

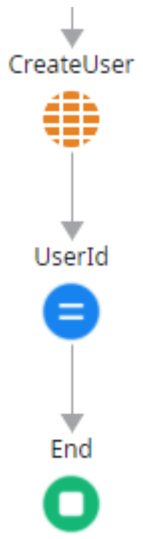


Let's now set the UserId output parameter with the Id of the user that was just created in the database (and that the CreateUser Action gives you for free)!

7) Set the **Variable** field of the Assign to the **UserId** Output Parameter and the **Value** field to **CreateUser.Id**, the result of the CreateUser Action.



8) Drag an **End** node under the last Assign and connect it to the flow.



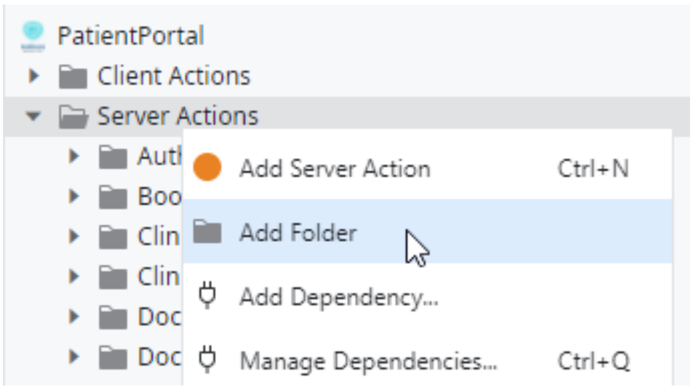
The User part is done. Let's move to the logic to save the Patient, PatientOnboarding and PatientImage in the Database, using the Registration_Create Server Action.

Registration Server Action

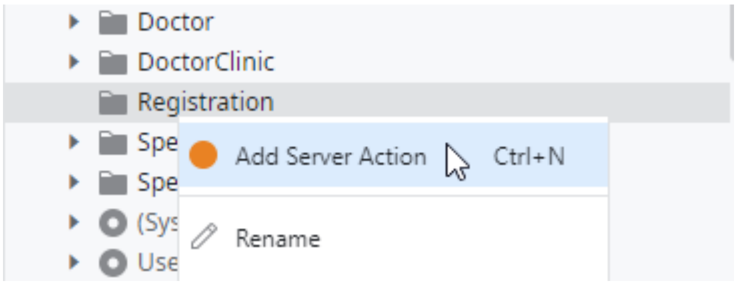
You have arrived to the second Action you need to create: the Registration_Create Action, where the remaining logic to save the data in the database will live.

Creating the Server Action

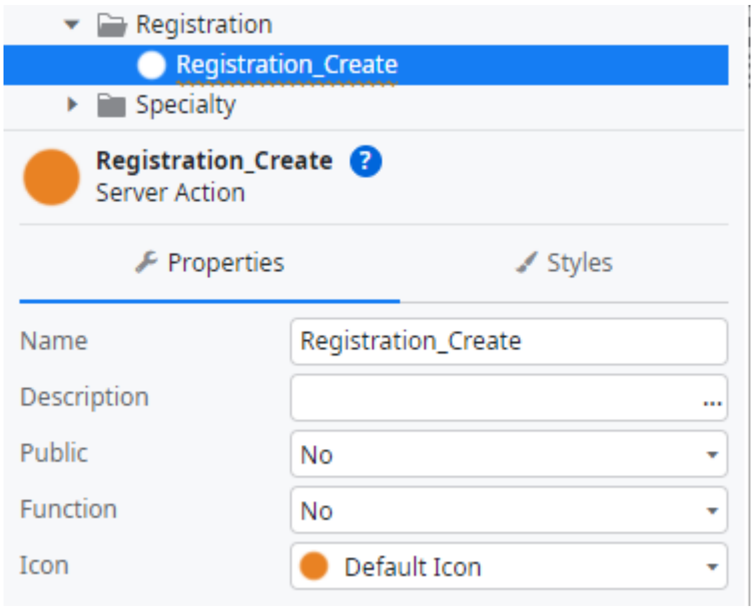
1) Still in the **Logic** tab, right-click on the Server Actions folder and create a new folder called *Registration*.



2) Right-click on the Registration folder and select **Add Server Action**.



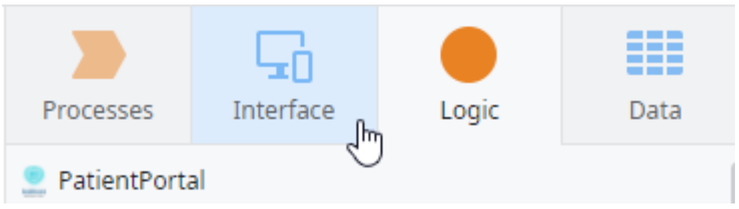
3) Name the Server Action *Registration_Create*.



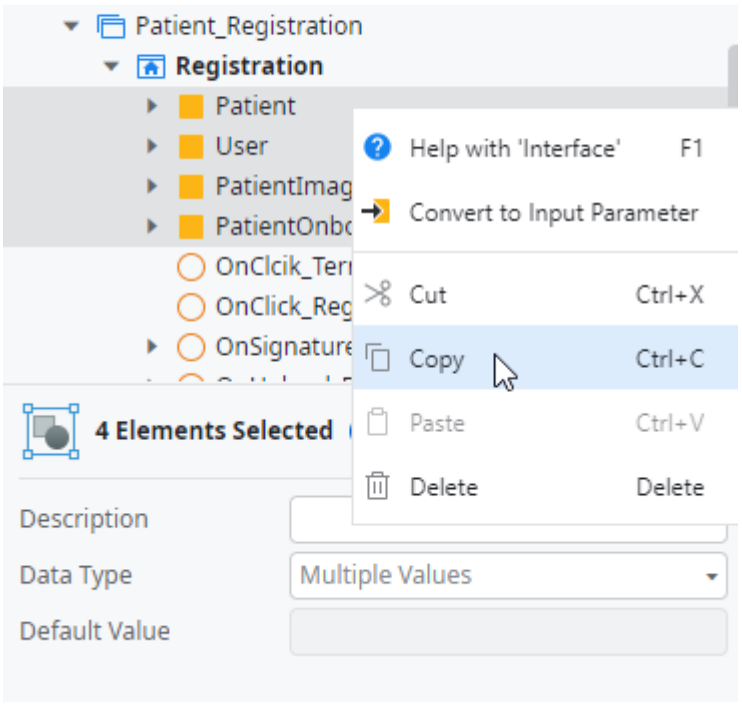
Name	Registration_Create
Description	
Public	No
Function	No
Icon	Default Icon

This Action needs to receive all the data that is stored in the Registration Screen local variables, which have the data that the user inserted and submitted in the Form. So, you need to create some input parameters.

4) Click on the **Interface** tab and expand the **Registration** Screen on the right sidebar.

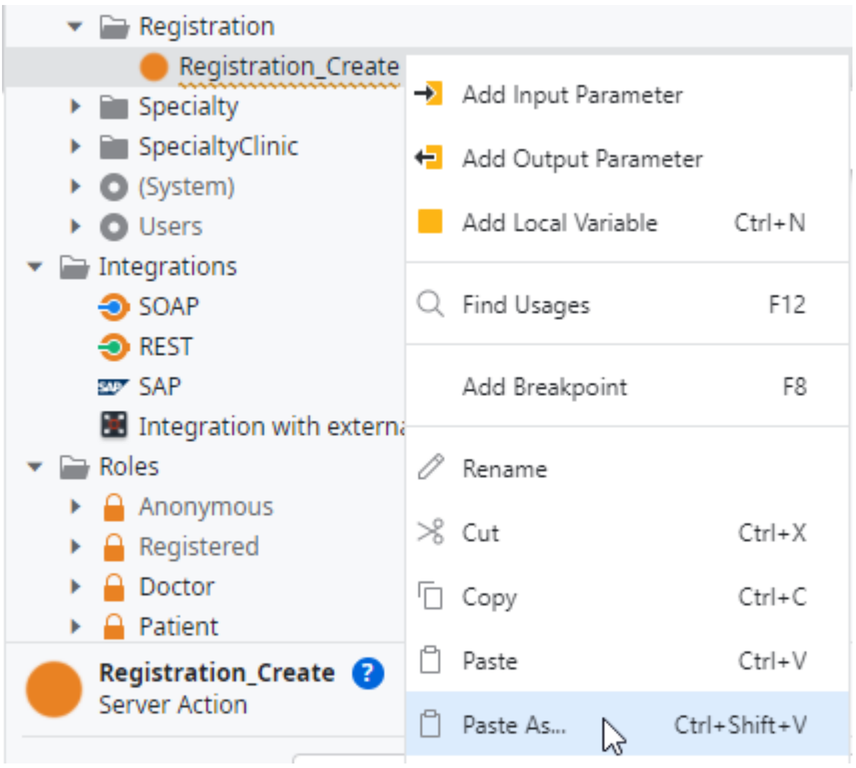


5) Select the **Patient**, **User**, **PatientOnBoarding**, and **PatientImage** Local Variables, then right-click on them and select Copy.

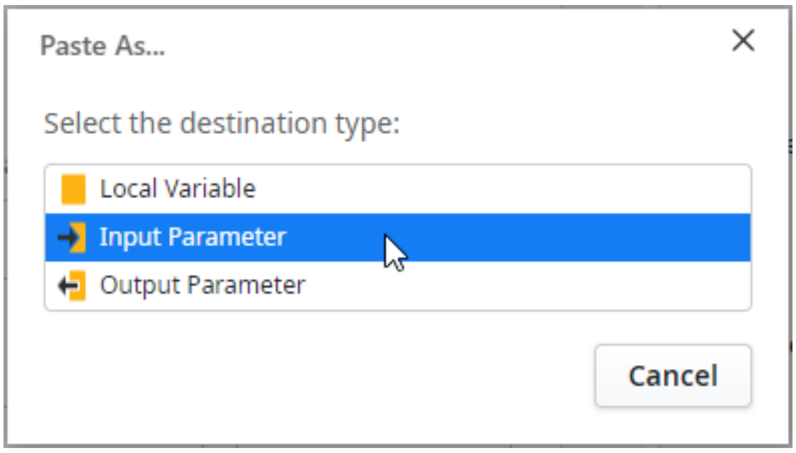


Description	
Data Type	Multiple Values
Default Value	

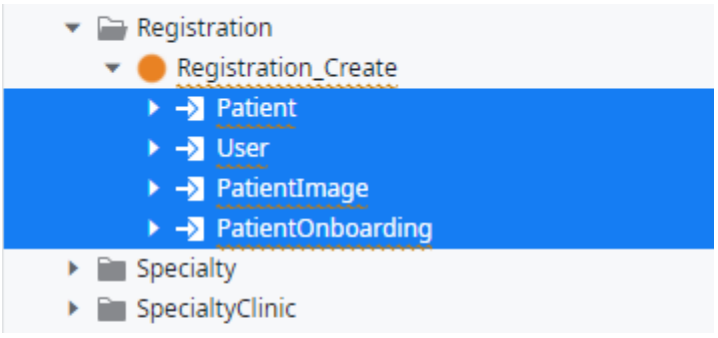
6) Then go back to the Logic tab, right-click on the **Registration_Create** Server Action, and select **Paste As**.



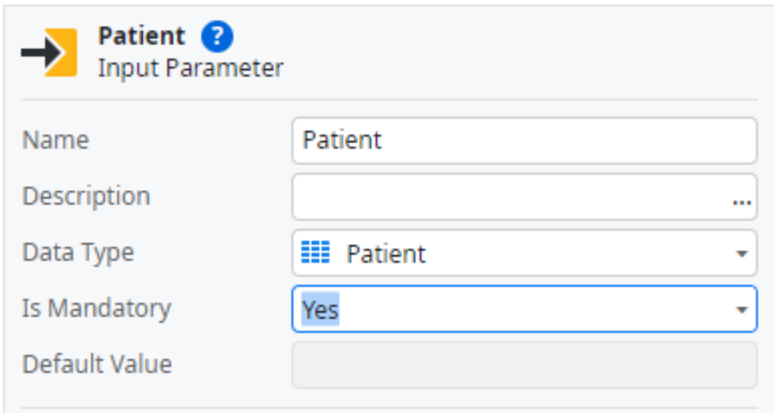
7) Select **Input Parameter** in the dialog that appears.



The Input Parameters will be created inside the Server Action. You just saved a bunch of time!



8) Click on each one of the Input Parameters and change their **Is Mandatory** property to **Yes**.

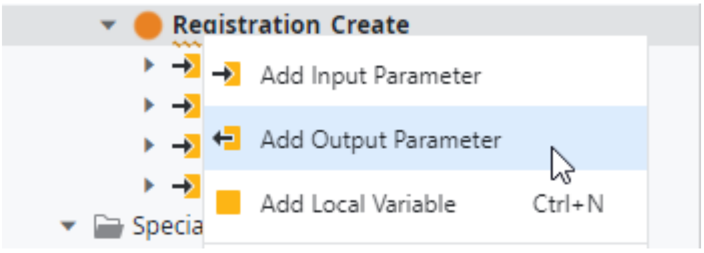

Patient
Input Parameter

Name	Patient
Description	
Data Type	Patient
Is Mandatory	Yes
Default Value	

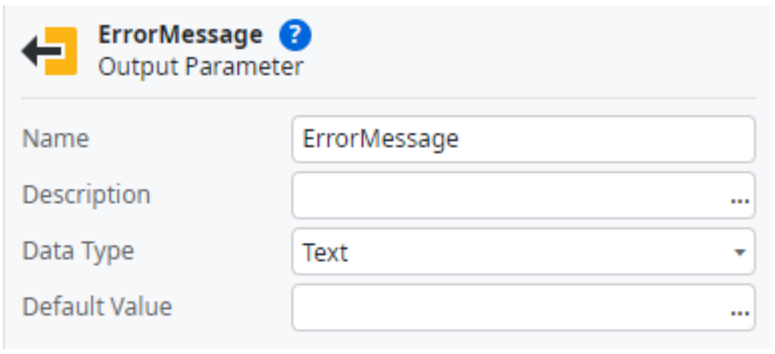
We also need an output parameter to return an error message, when applicable.

9) Right-click on the Registration_Create Action and select **Add Output Parameter**

.



10) Set the **Name** of the Output Parameter to *ErrorMessage* and make sure is **Data Type** is set to **Text**.



ErrorMessage
Output Parameter

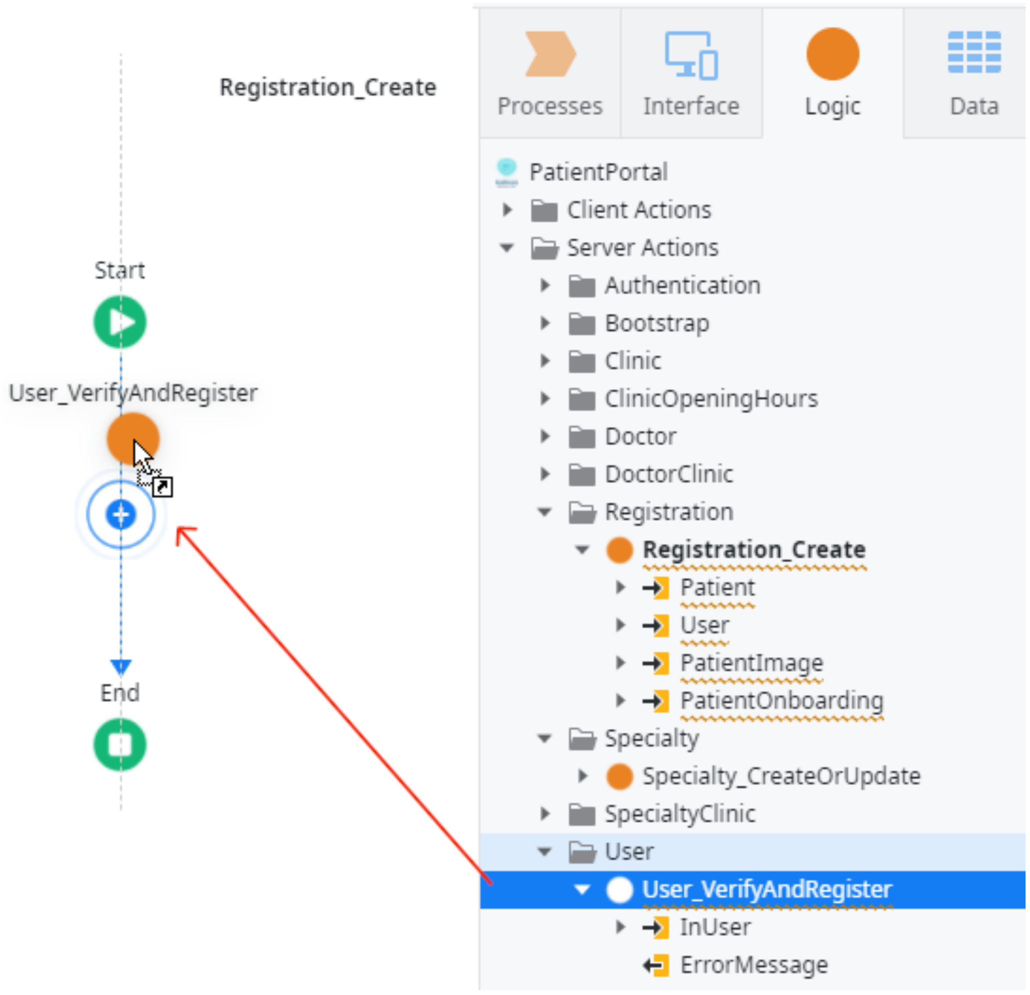
Name	ErrorMessage
Description	...
Data Type	Text
Default Value	...

Now that the Input and Output Parameters are created, let's work on the Action flow!

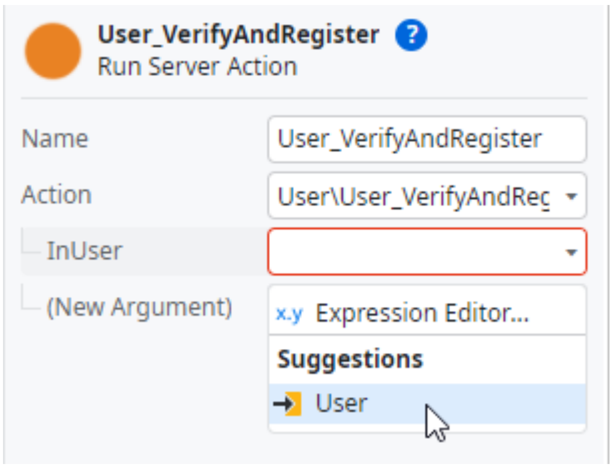
Add the User and Patient to the Database

You already created the logic to verify and add the user. Now, it's time to use it in this new Action. After that, you will add the Patient information to the database.

1) Drag the **User_VerifyAndRegister** Server Action from the right sidebar and drop it in the Registration_Create flow.



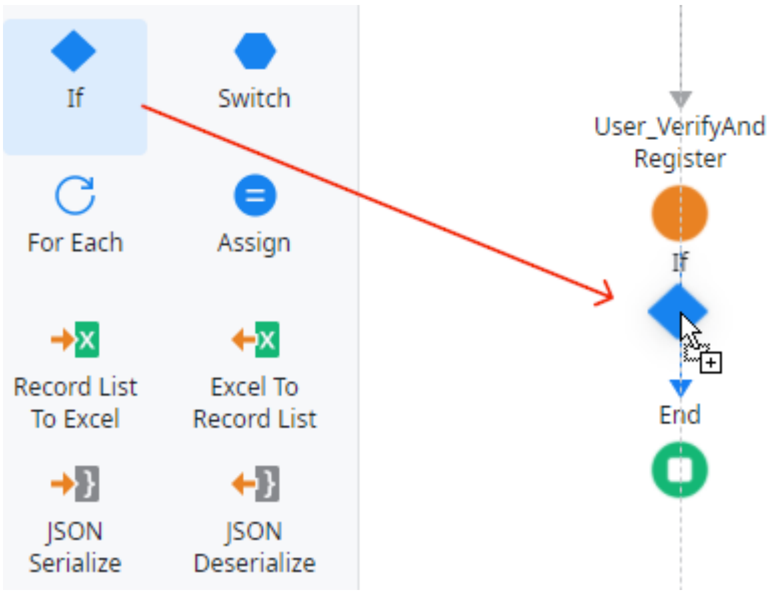
2) This Server Action has a mandatory Input parameter called InUser, remember? So, in the properties area, set the **InUser** value with the **User** Input Parameter of the Registration_Create Action.



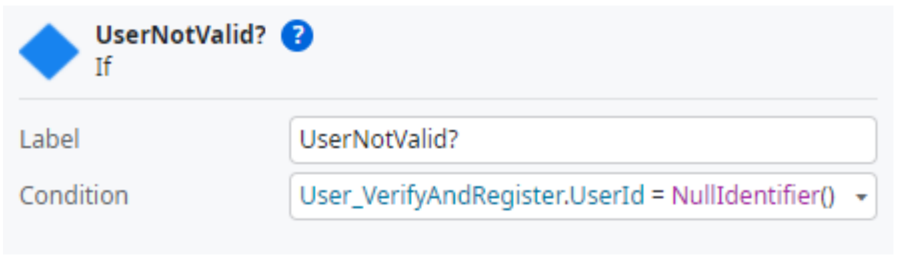
User_VerifyAndRegister
Run Server Action

Name	User_VerifyAndRegister
Action	User\User_VerifyAndReg
InUser	
(New Argument)	

3) Drag and drop an **If** under the User_VerifyAndRegister Action.



4) Set the **Label** of the If to *UserNotValid?* and the **Condition** to `User_VerifyAndRegister.UserId = NullIdentifier()`.

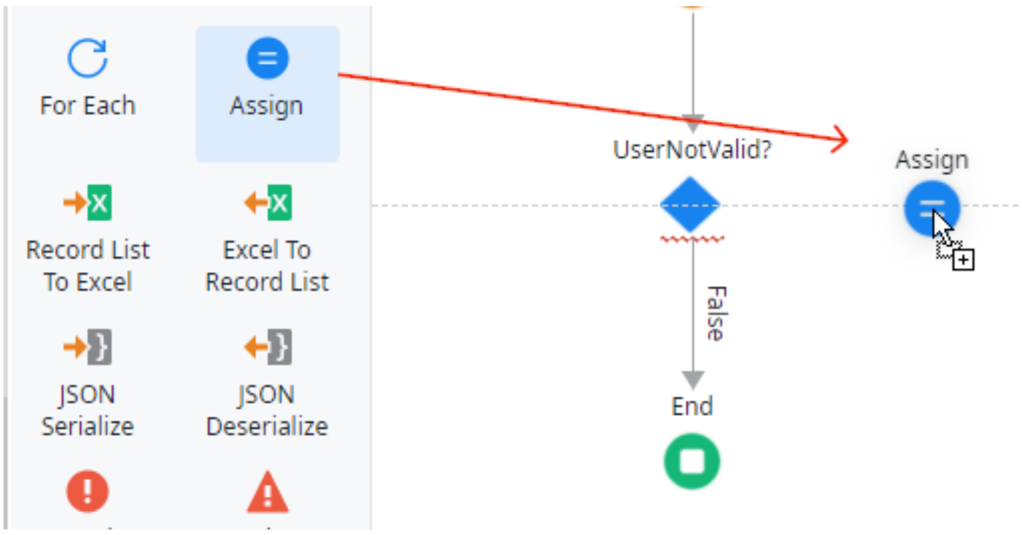


UserNotValid?
If

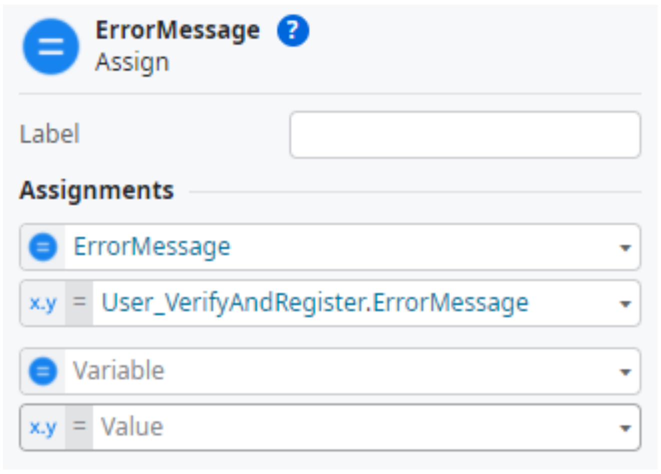
Label	UserNotValid?
Condition	User_VerifyAndRegister.UserId = NullIdentifier()

This checks if the user was created or not. Let's define what happens if the User was not created: the error message will be displayed. Which one? The one you defined earlier in the User_VerifyAndRegister Action and that is one of its output parameters.

5) Drag and drop an **Assign** on the right side of the If and connect the two to create the **True** branch.



6) Select the Assign element by clicking on it. Then, in the properties area on the right sidebar, set the **Variable** to **ErrorMessage** and the **Value** to `User_VerifyAndRegister.ErrorMessage`.



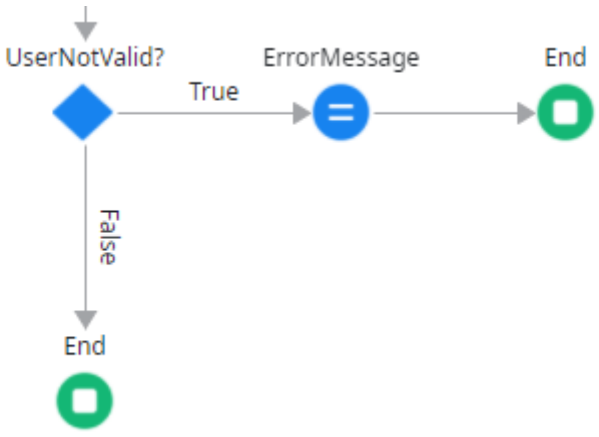
ErrorMessage
Assign

Label	

Assignments

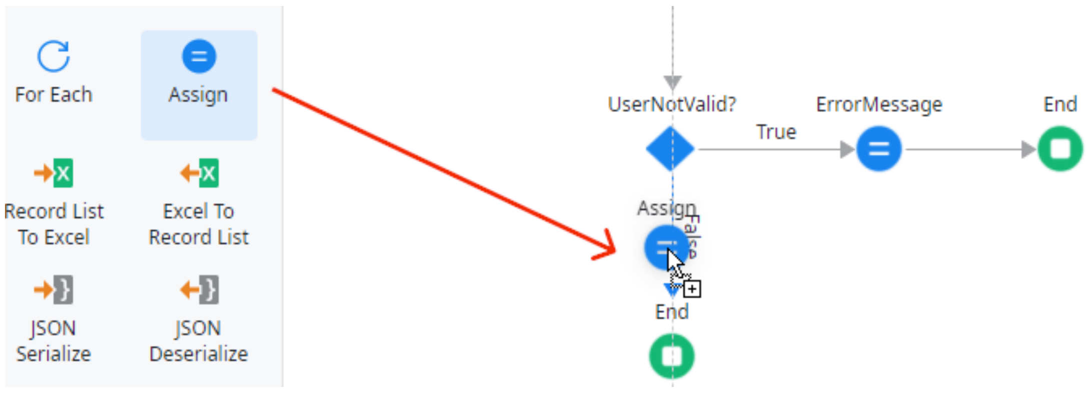
ErrorMessage
x.y = User_VerifyAndRegister.ErrorMessage
Variable
x.y = Value

7) Drag and drop an **End** after the Assign and connect them.

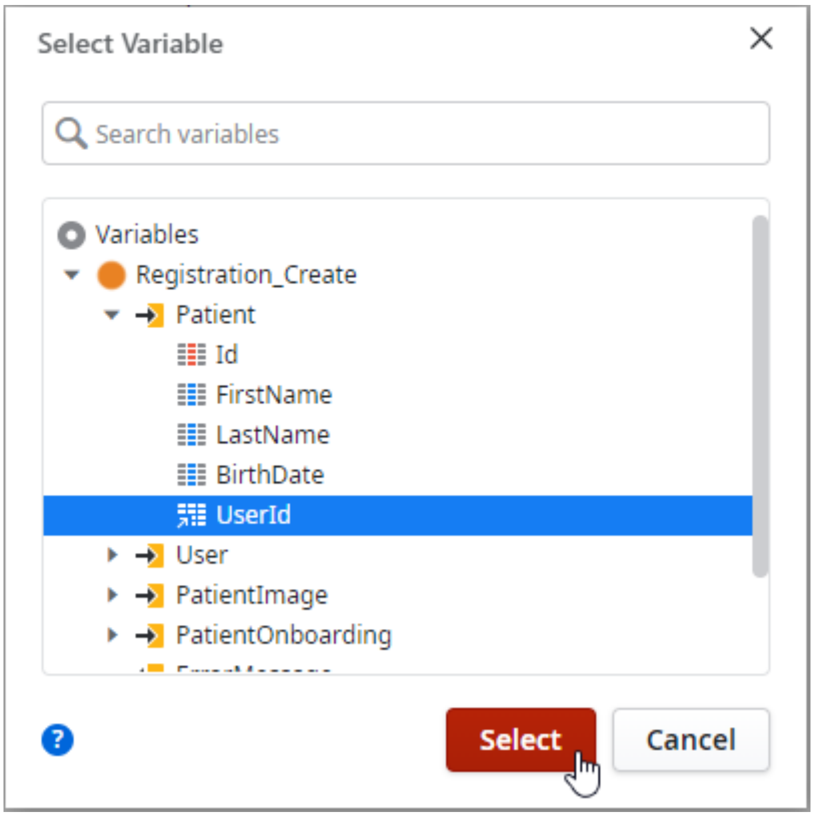


If the user is valid, we can proceed with the regular flow and assign the User Id to the Patient and create the Patient in the database.

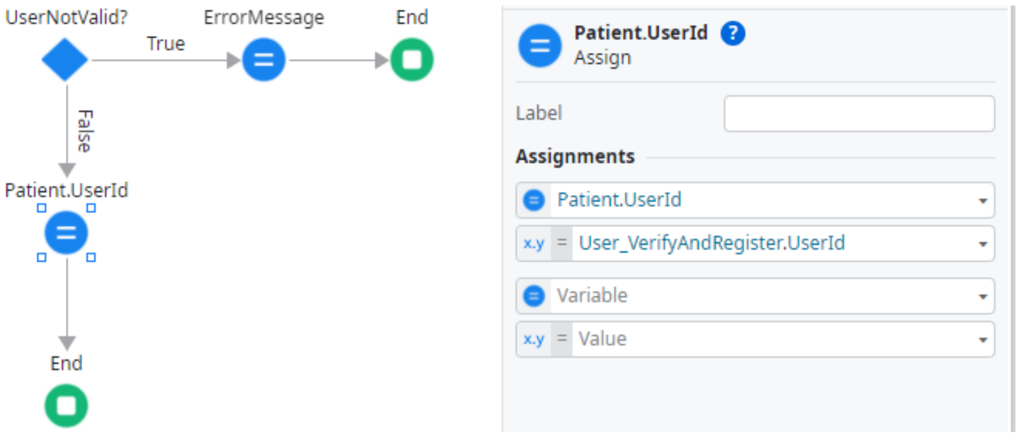
8) Drag and drop another **Assign**, but this time in the **False** branch.



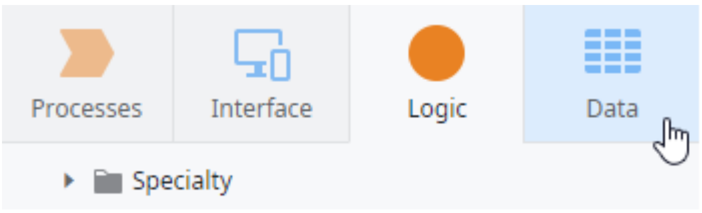
9) In the **Variable** property, click on the option **Select Variable**, then select the `Patient.UserId` in the dialog.



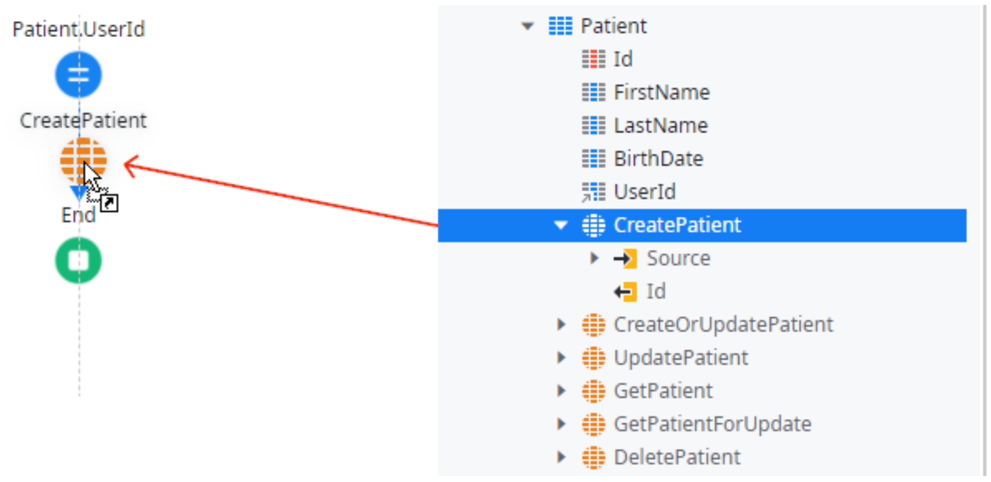
10) Set the **Value** to `User_VerifyAndRegister.UserId`.



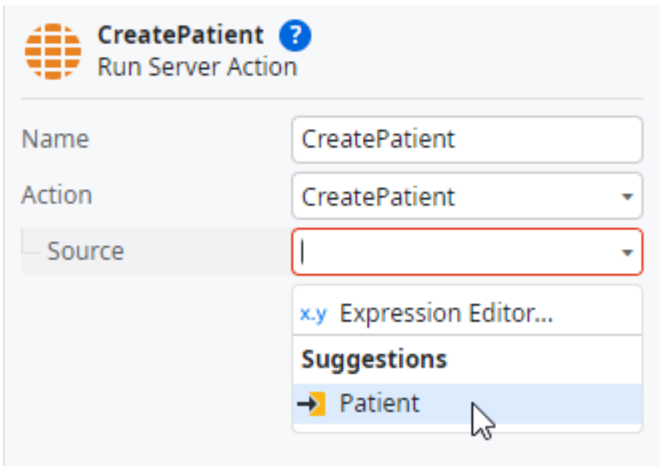
11) Switch to the **Data** tab.



12) Expand the **Patient** Entity, click on the **CreatePatient** Action, then drag and drop it under the Assign.



13) Select the **Patient** Input Parameter as the **Source** of the CreatePatient Action.



Name	CreatePatient
Action	CreatePatient
Source	

Let's proceed for the other Entities.

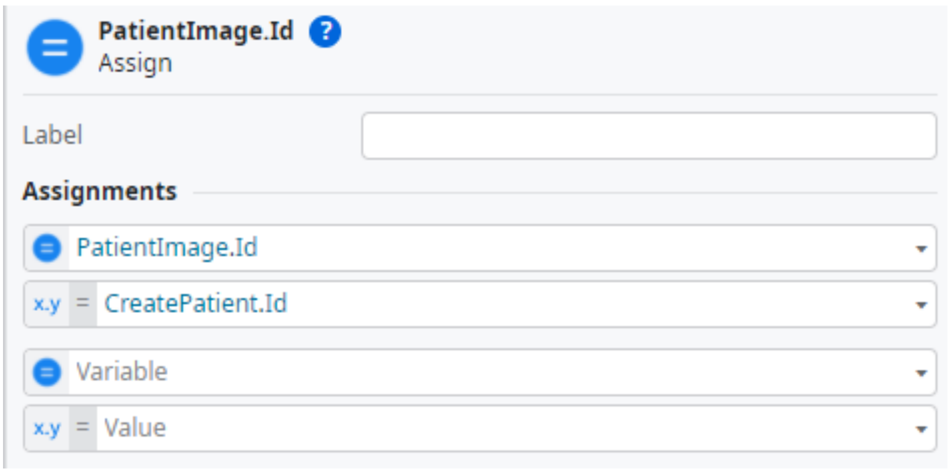
Add the Patient Image and Onboarding Info to the Database

The Action flow will continue with the creation of the image and onboarding information in the database. The Patient Image information is almost complete with the picture and filename uploaded by the user. The same thing happens with the Patient Onboarding, since the user already provided the consent and added a signature.

So, the only thing missing, is to set the Id attributes of the PatientImage and PatientOnboarding. Remember that these Entities have an Id that is not an auto number, since we want them to have the same Id of the Patient. So, that's what you need to do: use the Id of the Patient to complete the PatientImage and PatientOnboarding data.

1) Drag and drop another **Assign** under the CreatePatient Action.

2) In the properties area on the right sidebar, set the **Variable** to **PatientImage.Id** and the **Value** to **CreatePatient.Id**.



PatientImage.Id
Assign

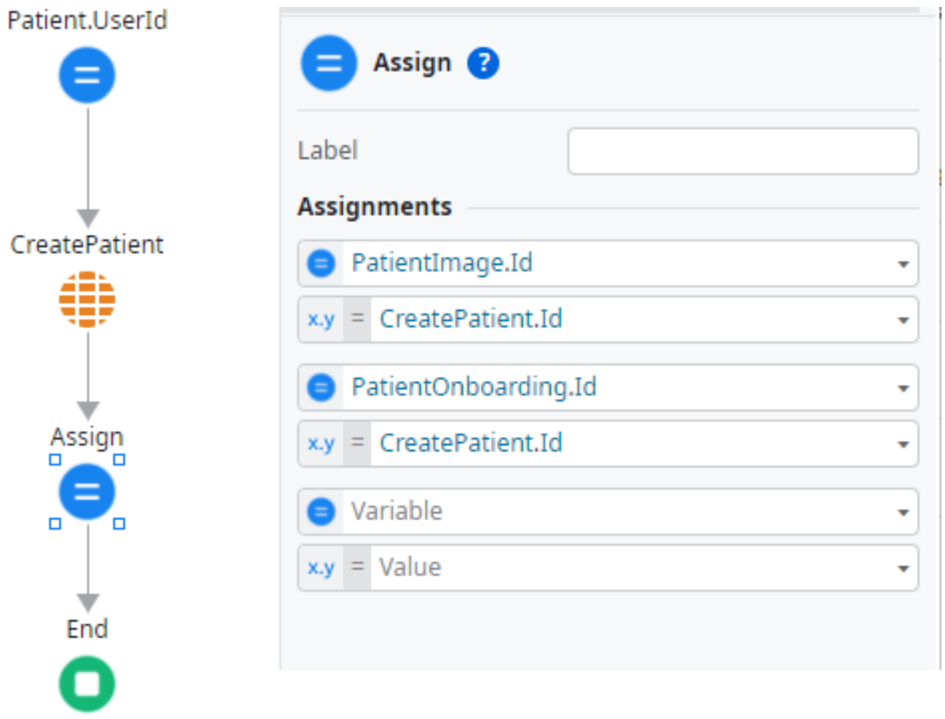
Label

Assignments

PatientImage.Id = CreatePatient.Id

Variable = Value

3) Just below the assignment you just created, let's create a new one. Set the **Variable** to **PatientOnboarding.Id** and the **Value** to **CreatePatient.Id**.



Assign

Label

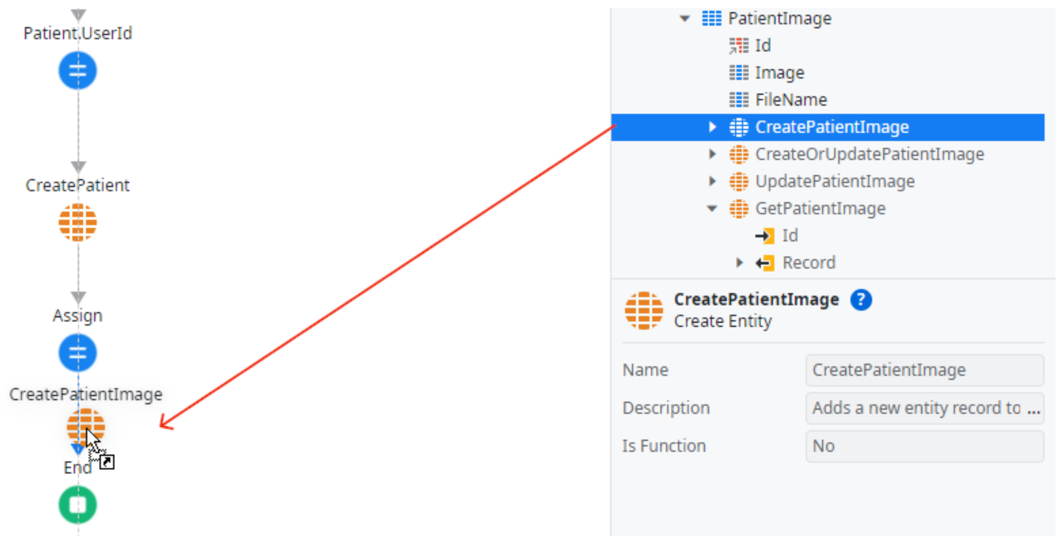
Assignments

PatientImage.Id = CreatePatient.Id

PatientOnboarding.Id = CreatePatient.Id

Variable = Value

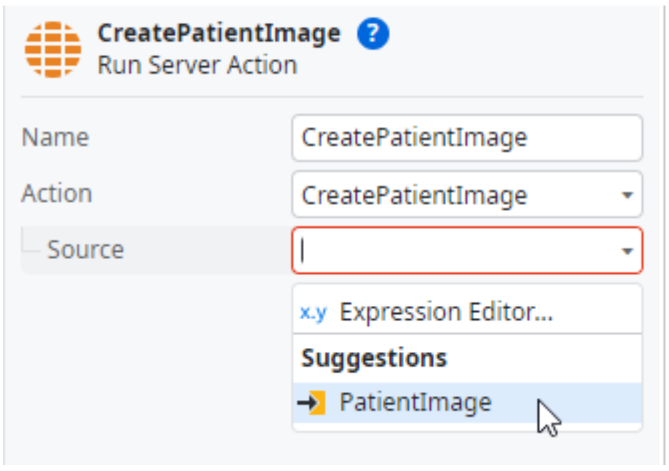
4) Still in the **Data** tab, expand the **PatientImage** Entity. Select the **CreatePatientImage** Action, then drag and drop it under the last Assign.



CreatePatientImage
Create Entity

Name	CreatePatientImage
Description	Adds a new entity record to ...
Is Function	No

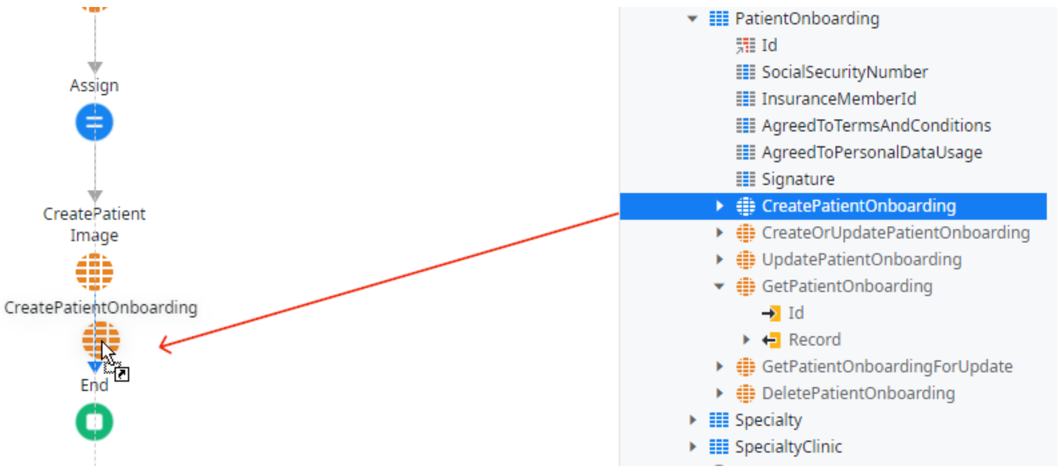
5) In the properties section, set the **Source** to the the **PatientImage** input parameter.



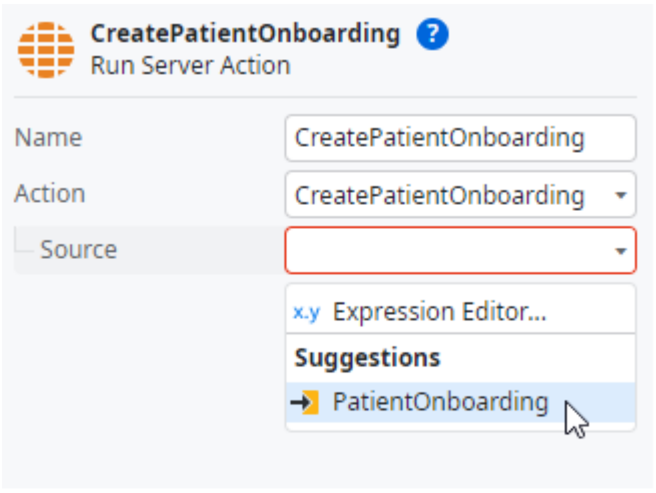
CreatePatientImage
Run Server Action

Name	CreatePatientImage
Action	CreatePatientImage
Source	

6) Similarly, expand the **PatientOnboarding** Entity. Select the **CreatePatientOnboarding** Action, then drag and drop it under the CreatePatientImage Action.



7) Set the **Source** to the the **PatientOnboarding** input parameter.

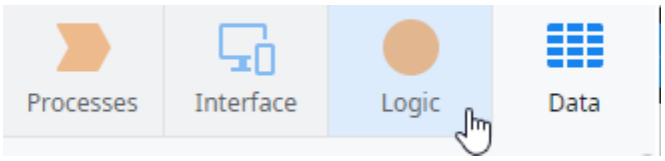


And you're set in terms of the logic to add the info to the database. But you're not ready yet. Since the Patient will be a user of the app, it is also important to grant them a Role. The Role will help define what the user can or cannot do in the app. And you already have the Patient Role available. You just need to grant the role to the patient.

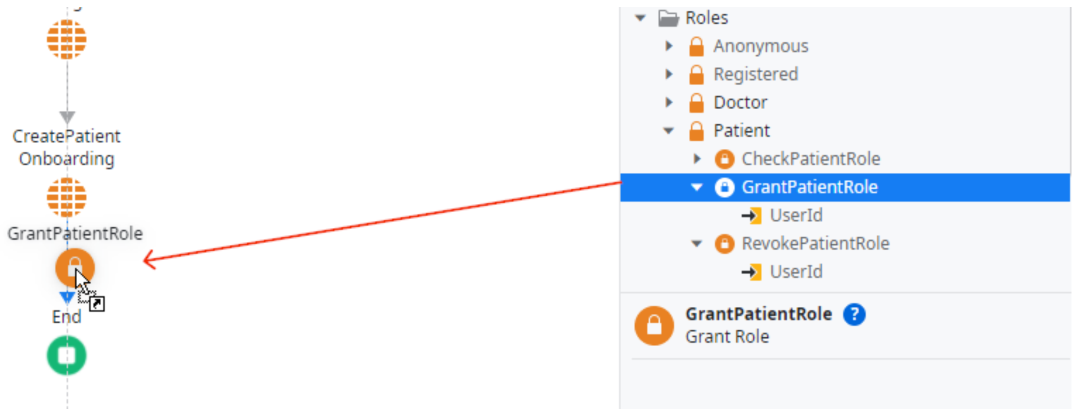
Granting the Patient Role

The Roles exist in the Logic tab and have already a GrantRole Action available that grants a role to a user. Let's leverage that!

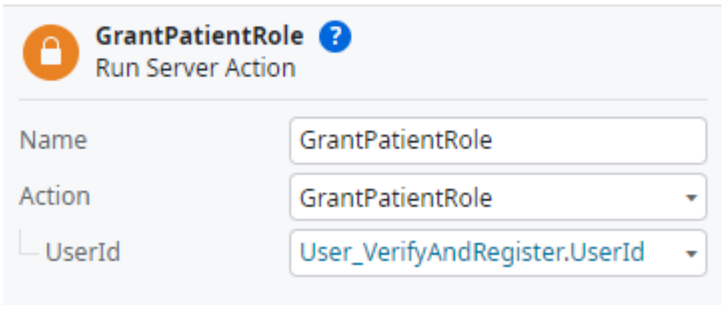
1) Open the **Logic** tab.



2) Expand the **Roles** folder, then the **Patient** Role and click on the **GrantPatientRole** Action. Drag and drop it under the CreatePatientOnboarding Action.



3) Set the **UserId** to **User_VerifyAndRegister.UserId**.



GrantPatientRole
Run Server Action

Name	GrantPatientRole
Action	GrantPatientRole
UserId	User_VerifyAndRegister.UserId

Again, here's an action that does all the hard work for you. You just drag it, pass the identifier of the User that you want to have the Patient Role, and that's it!

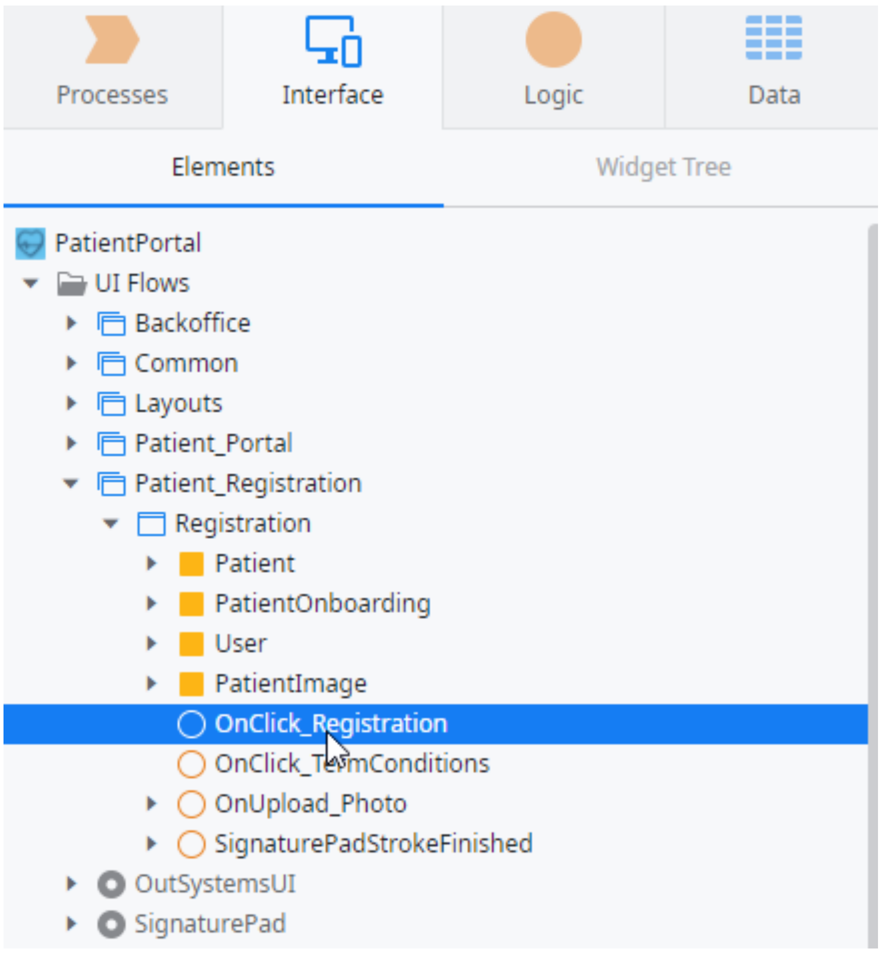
The logic to add data to the database is done, so it's time to go back at the beginning and finish the OnClick_Registration Action on the Registration Screen.

Finishing the OnClick_Registration Action

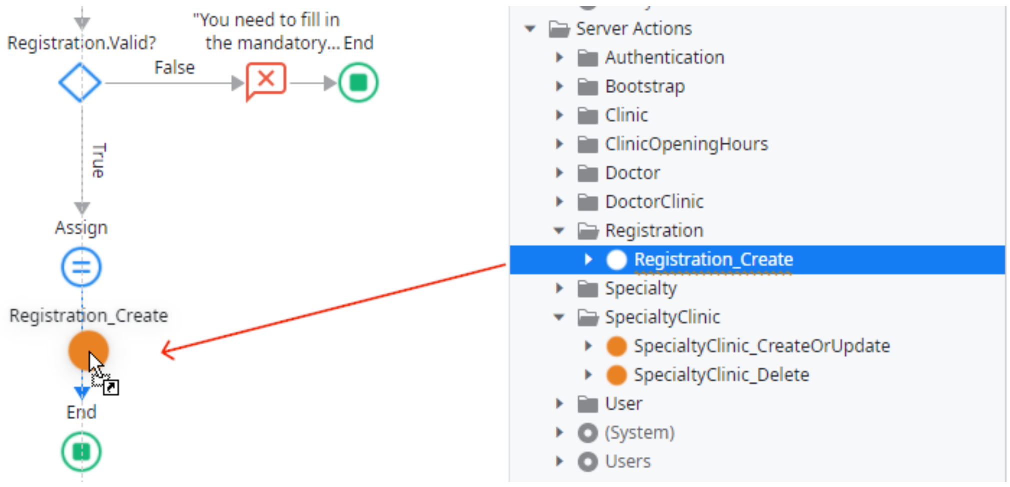
So, to complete the action flow you need to:

- Call the Registration_Create Action.
- If the registration was successful, the user is redirected to the Login Screen.
- Otherwise, an error message is displayed to the user.

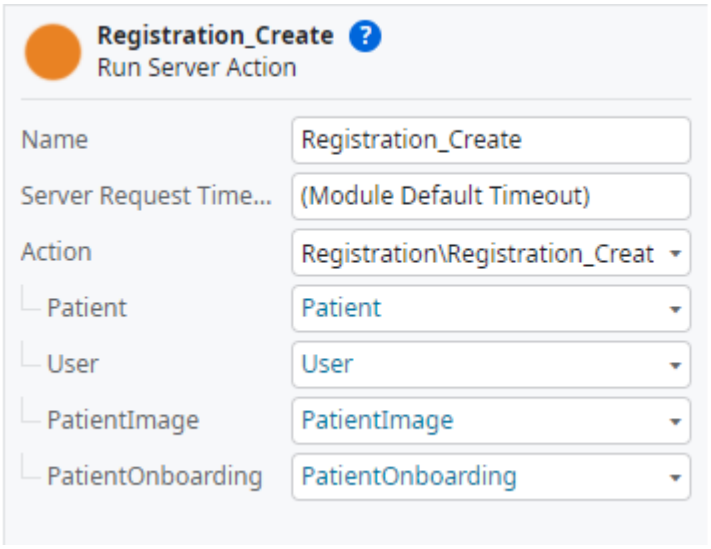
1) Switch back to the Interface tab, expand the **Registration** Screen and open the **OnClick_Registration** Client Action by double-clicking ot it.



2) Switch to the **Logic** tab, then drag and drop the **Registration_Create** Server Action under the Assign.



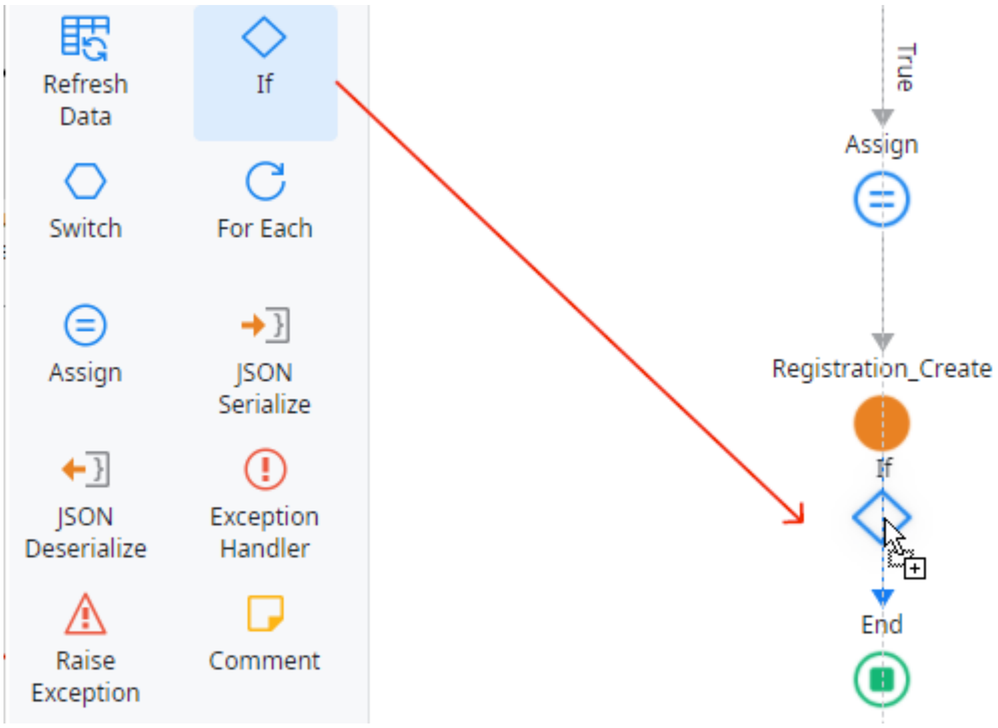
3) There's a couple of errors. That's because the Registration_Create Action is waiting for values for its input parameters. Set the values to the **Patient**, **PatientImage**, **User** and **PatientOnboarding** input parameters.



Registration_Create
Run Server Action

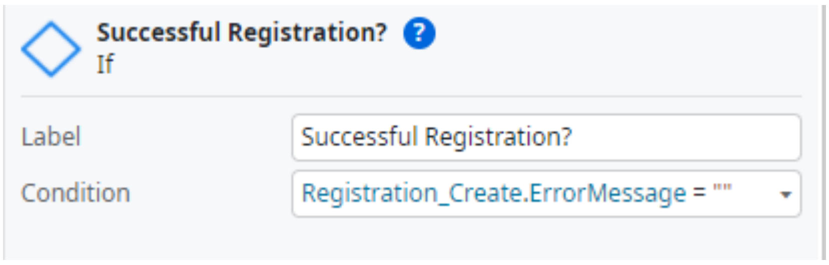
Name	Registration_Create
Server Request Time...	(Module Default Timeout)
Action	Registration\Registration_Creat
Patient	Patient
User	User
PatientImage	PatientImage
PatientOnboarding	PatientOnboarding

4) Drag and drop an **If** under the Action. This if should validate if there were any issues with the registration.



This If will evaluate if the error message in the Registration_Create Action is empty. If yes, it means everything went well and you can continue the Action. If not, the user must get an error message.

5) Set the **Condition** to `Registration_Create.ErrorMessage = ""` and the **Label** to *Successful Registration?* .

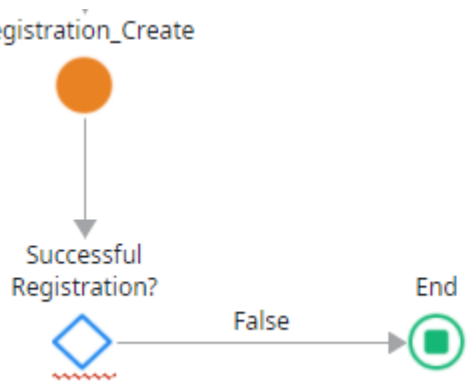


Successful Registration?
If

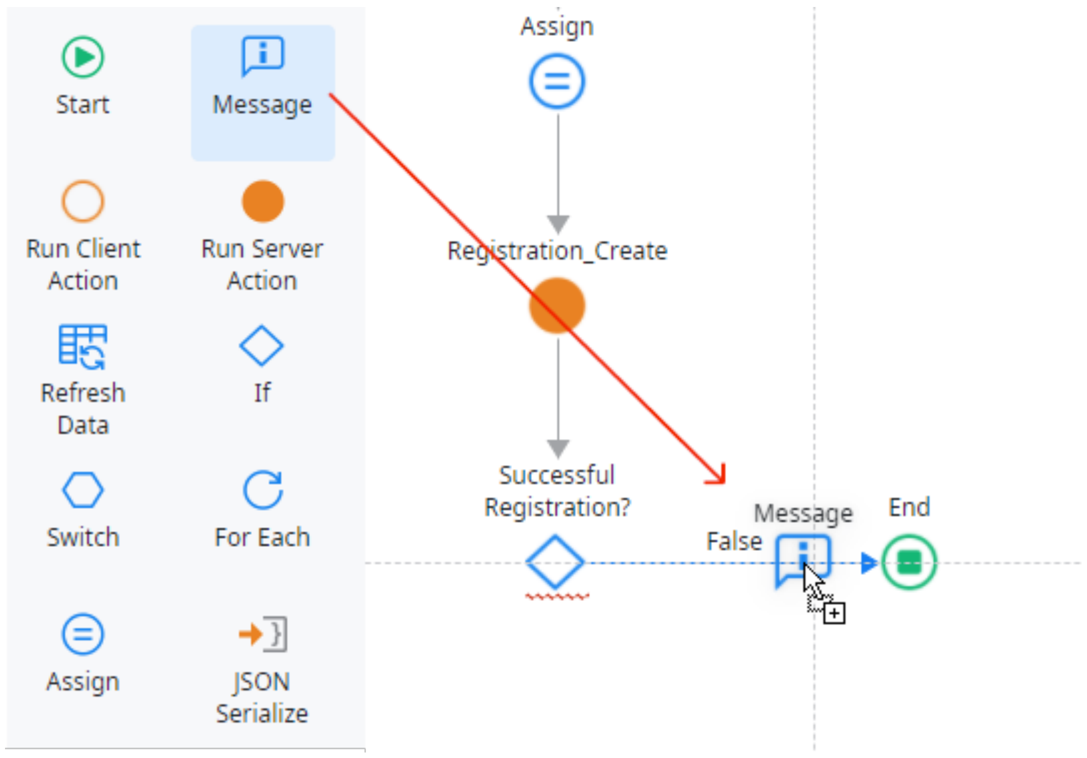
Label	Successful Registration?
Condition	Registration_Create.ErrorMessage = ""

Now let's define what happens if the registration was not successfull.

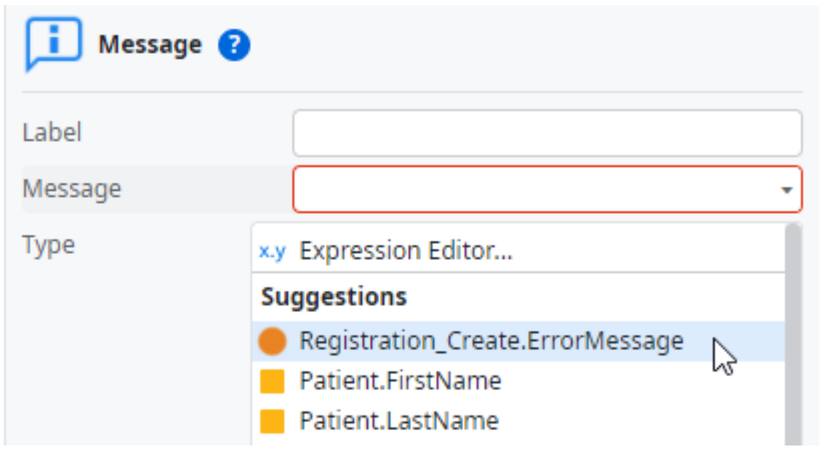
6) Move the **End** element with the False branch to the right side of the If.



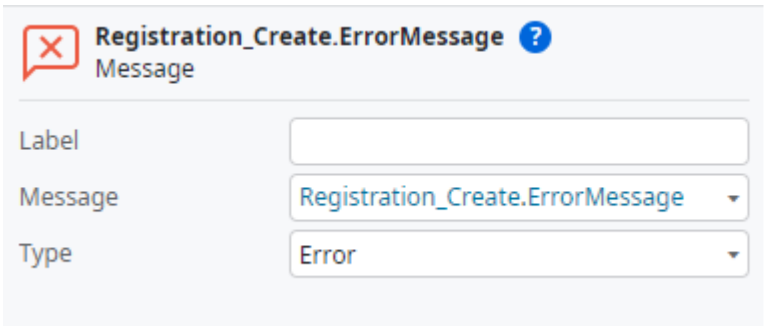
7) Drag and drop a **Message** in the False branch.



8) In the Message's properties, set the **Variable** to the output of the Registration_Create Action: **Registration_Create.ErrorMessage**.



9) Also, set the **Type** property to **Error**.

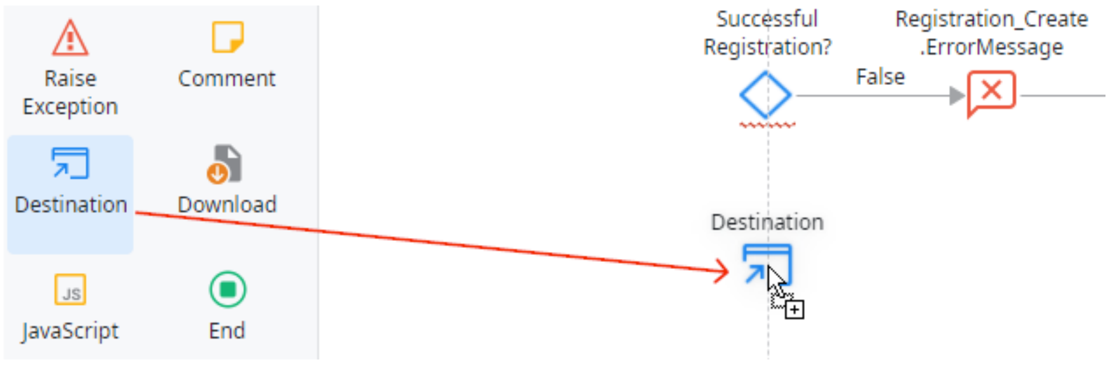


Registration_Create.ErrorMessage
Message

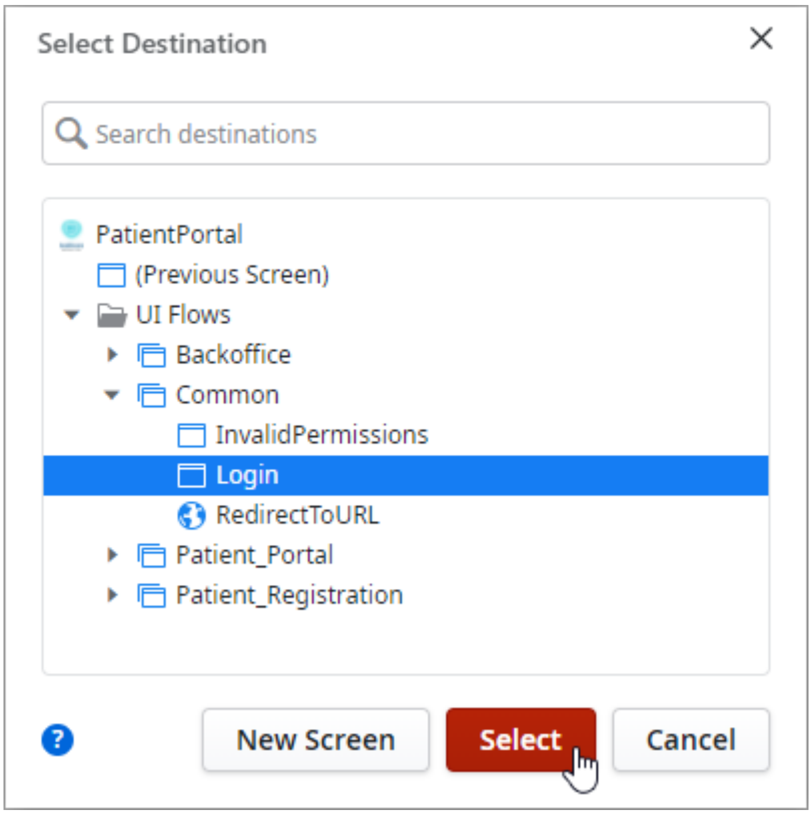
Label	
Message	Registration_Create.ErrorMessage
Type	Error

Now let's work on the True branch.

10) Drag a **Destination** element from the left sidebar and drop it under the If.

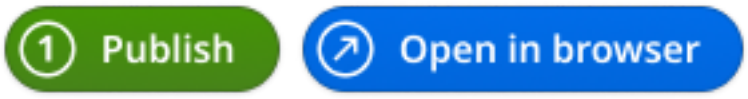


11) Select the **Login** Screen in the dialog.



This will automatically redirect the user to the Login Screen, when this Action finishes its execution.

12) Publish the module and open it in the browser.

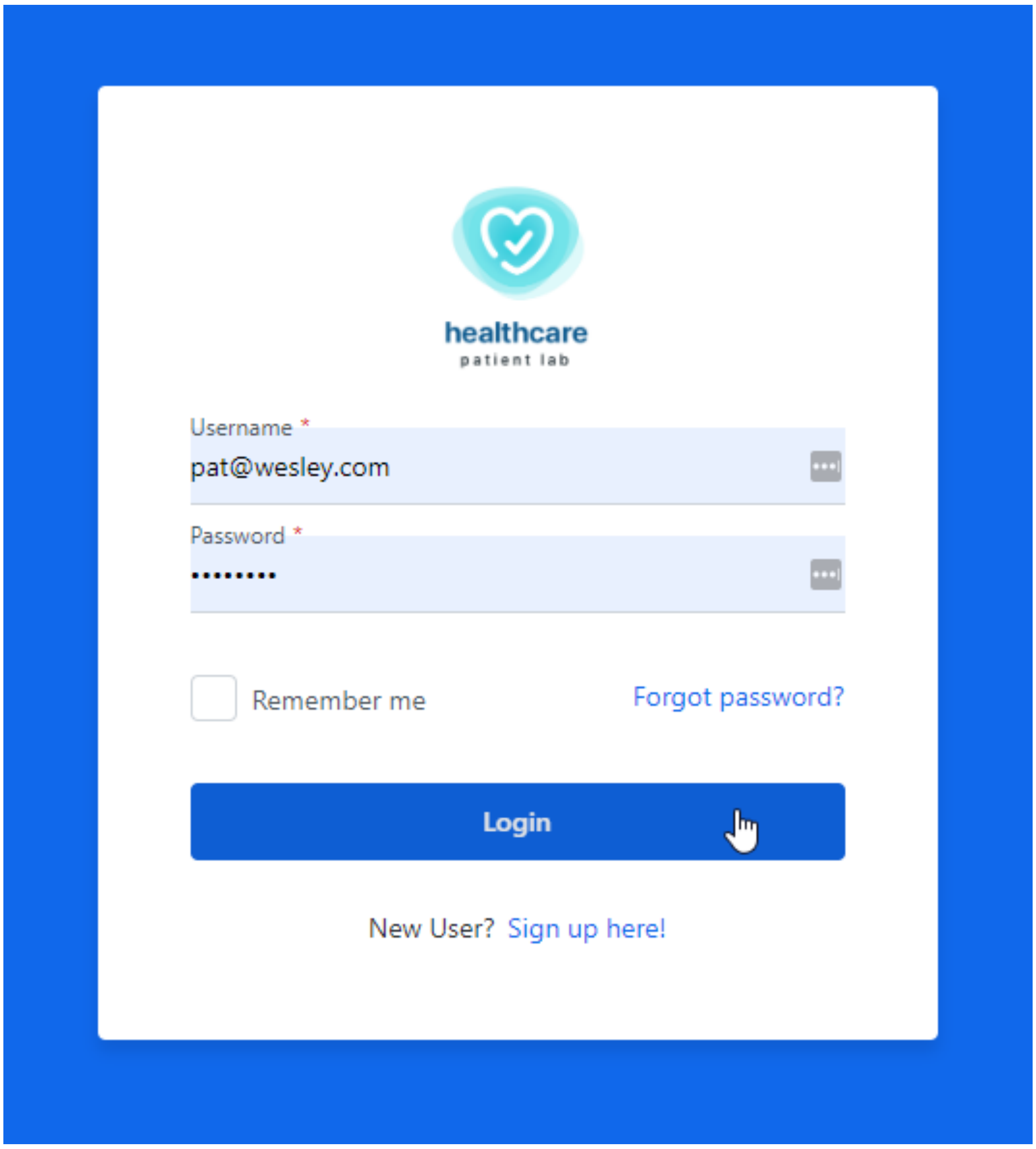


Testing

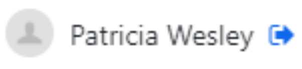
Now that the registration logic is finally ready, you can test the registration flow end-to-end.

1) Fill out the registration form with any information you want. Don't forget the password and the username, to make sure you can login later.

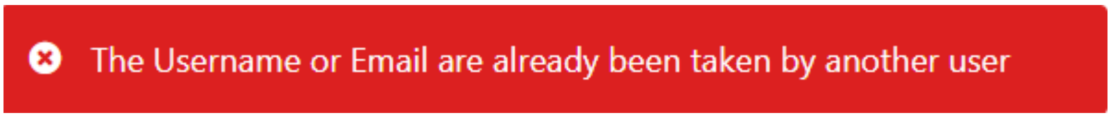
2) When you are back to the Login Screen, try to login with the new credentials you created.



At this point, you're back to the Registration Screen, but you should see the login information on the top right of your Screen.

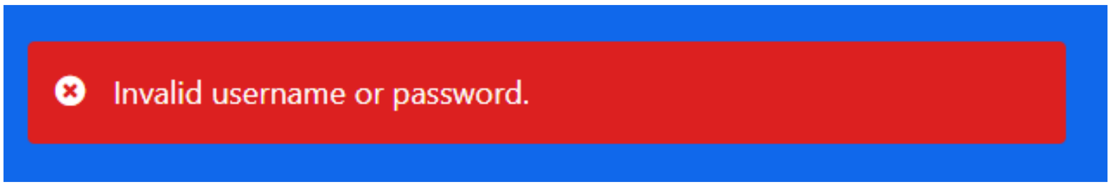


3) Now, try registering the same user again. You should get the error below:



The Username or Email are already been taken by another user

4) Try to login with an invalid username or incorrect password. You should also get an error.



Invalid username or password.

Wrapping up

Congratulations on finishing this tutorial. With this exercise, we had the chance to go through some essential aspects of OutSystems, especially Action's logic, and get to know more about the platform.

References

If you want to deep dive into the subjects that we have seen in this exercise, we have prepared some useful links for you:

1) Actions in Reactive Web and Mobile Apps
2) Input Parameter
3) Output Parameter
4) Assign Tool
5) Message Tool

See you in the next tutorial!