Patient Registration

Table of Contents

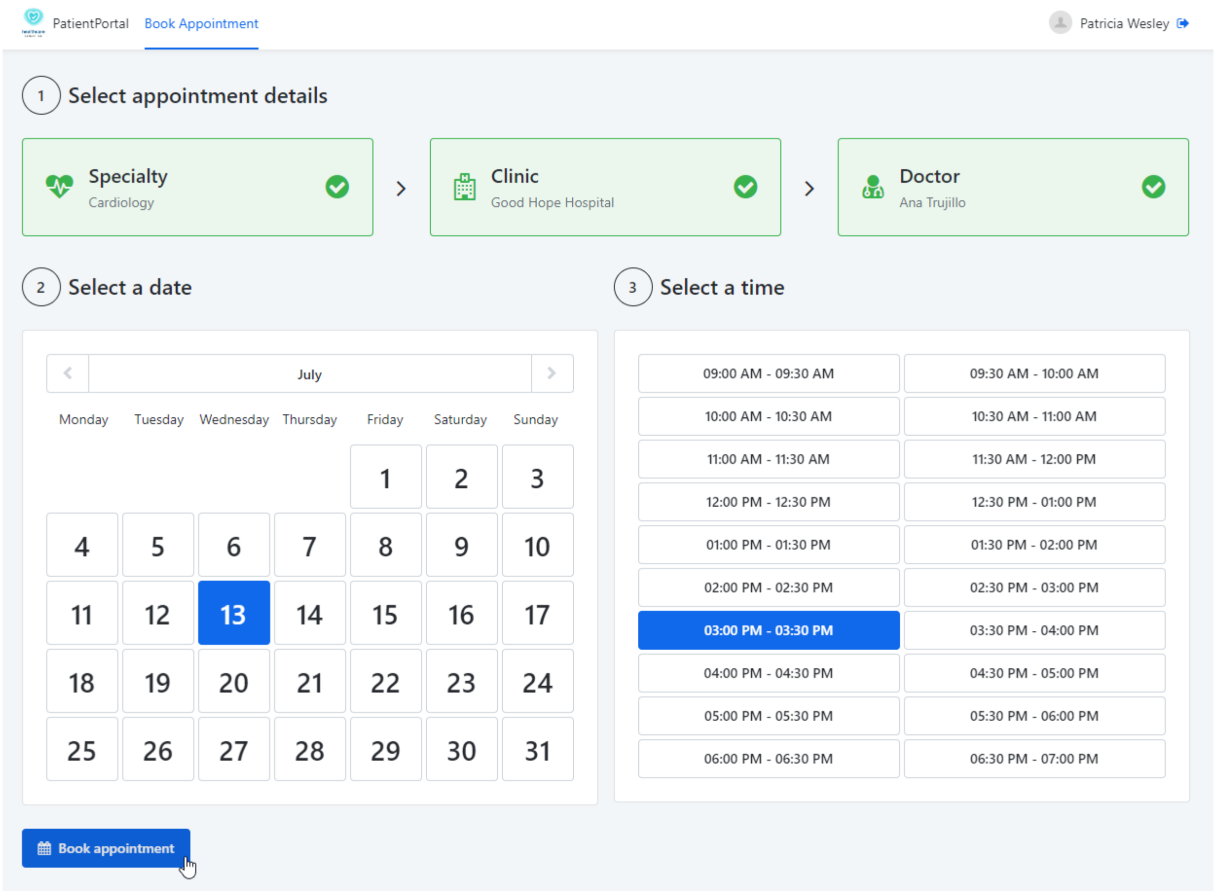
Scenario

In this tutorial, you will start building a Healthcare Self-Service Portal, where patients can book appointments, view appointment schedules, as well as medical information added by their doctors. Also, doctors will be able to check their scheduled appointments, prescribe medication and edit diagnosis for a patient.

You will start by a quickstart application that already has some functionality defined. We will talk about it when needed during this set of tutorials.

Booking Appointment

One of the features already created is the UI and logic to book an appointment.



PatientPortal Book Appointment Patricia Wesley

1 Select appointment details

Specialty: Cardiology
Clinic: Good Hope Hospital
Doctor: Ana Trujillo

2 Select a date: July 13

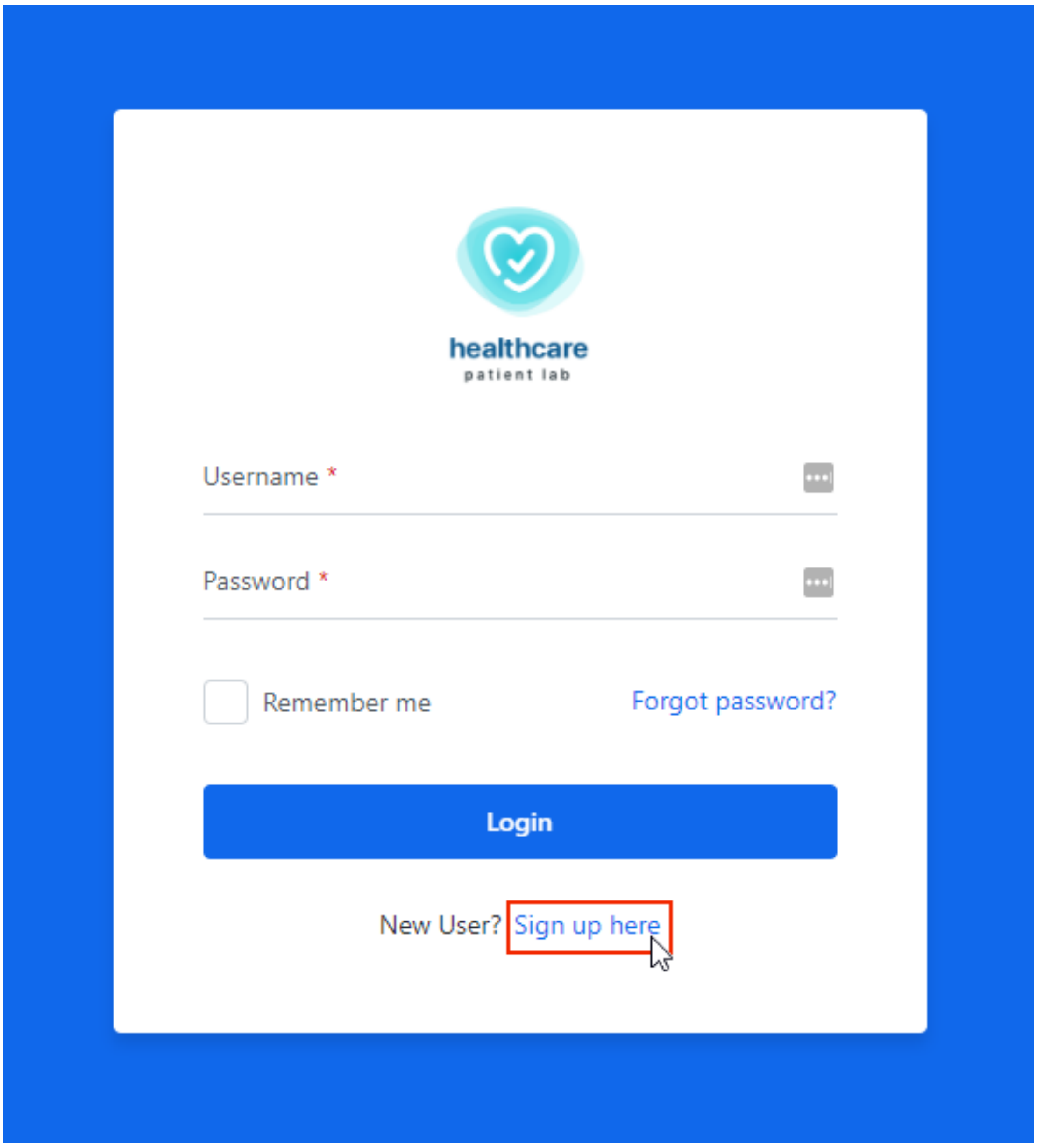
3 Select a time: 03:00 PM - 03:30 PM

A patient can choose a specialty, then a clinic, and then the doctor (that works in the clinic with that specialty). Then, its just a matter of choosing the date and time. After that, the patient must choose one of the available dates and times. You don't need to worry about this part for now, but once you are more familiar with the platform, feel free to explore it and see how it works!

Patient Registration

What we don't have yet on the app is the patient experience. And that starts by the patient registration, where a patient adds their information and register on the app. Without that, the patient cannot book an appointment.

And that's what you will do on this tutorial! Create a Registration Screen (without the UI yet) and make sure there's an option on the Login Screen to navigate to that Screen:

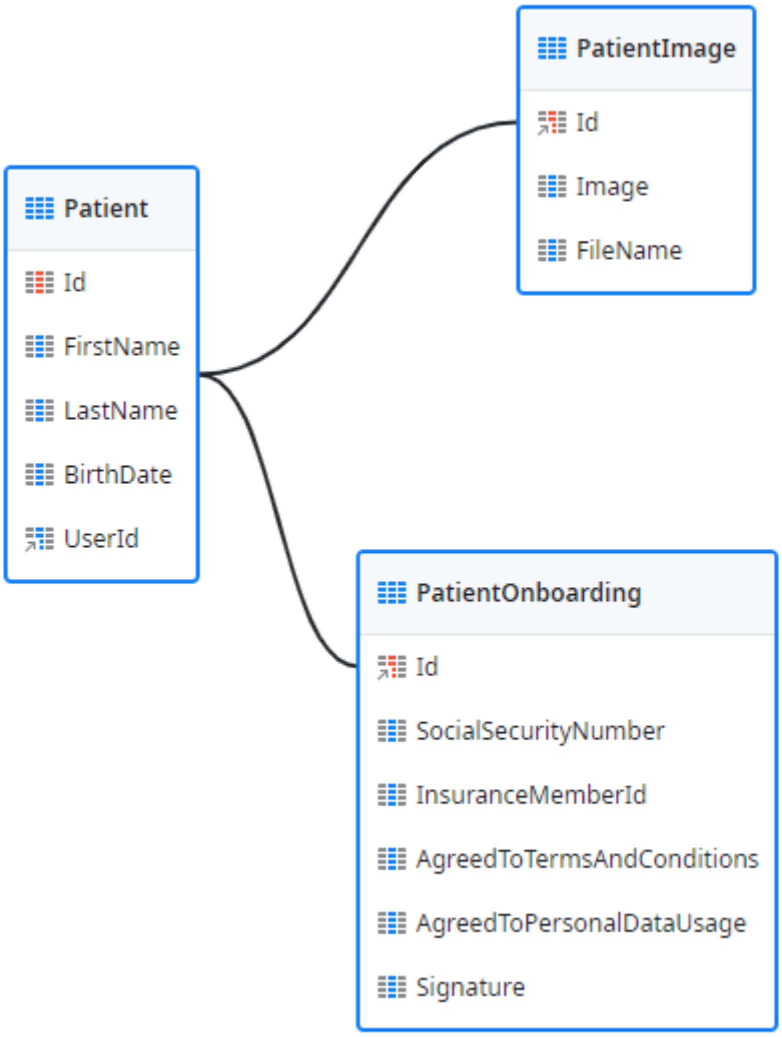


Data Model

The functionality that already exists in the quickstart app, plus the one that you will add needs to be supported by data. Every application needs to store and retrieve data that represents various business concepts. In OutSystems, these concepts are modeled and referred to as Database Entities.

This application already have some Entities created, such as Doctor and Clinic. In this tutorial, you will create some new Entities for the Patient and some of its relevant information:

- **Patient**, which will store the information provided by the user in the registration form.
- **Patient Onboarding**, which contains relevant patient information for the patient onboarding, such as their signature and if they agreed to terms and conditions;
- **PatientImage**, with the profile picture of the patient.



How-To

Now that you know the scenario and the objectives, let's do this together!

Extend the Data Model

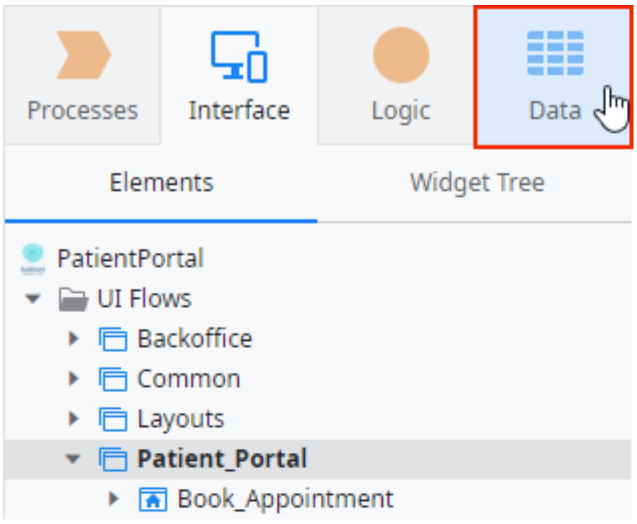
Now that you have the quickstart app installed, let's start by extending the existing Data Model with the patient information.

The first Entity being created is the Patient Entity.

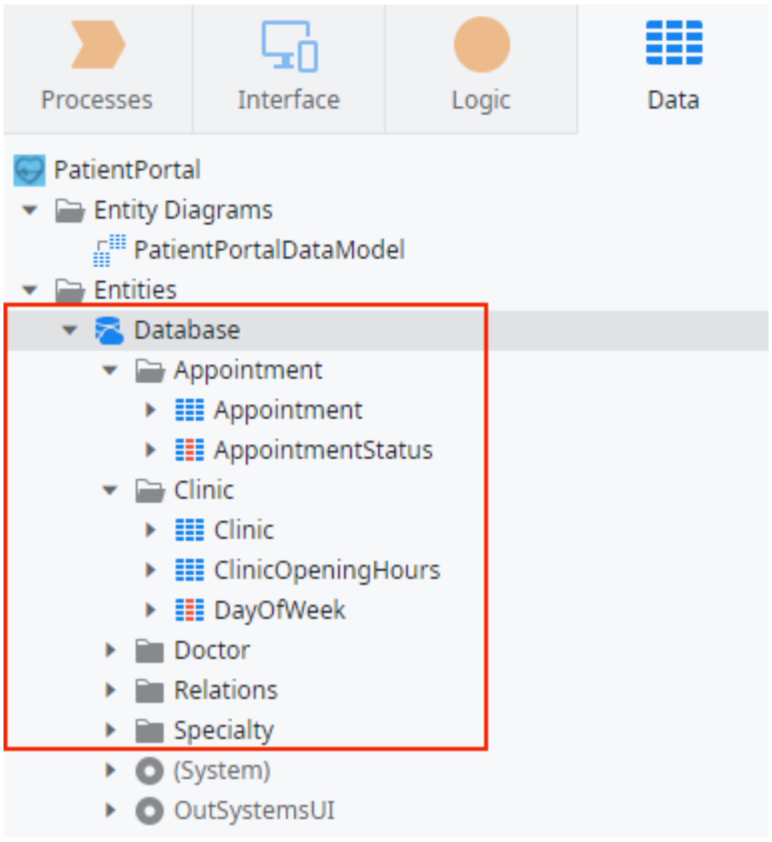
Create the Patient Entity

The Patient Entity will have the patient's first name and last name, as well as the birth date.

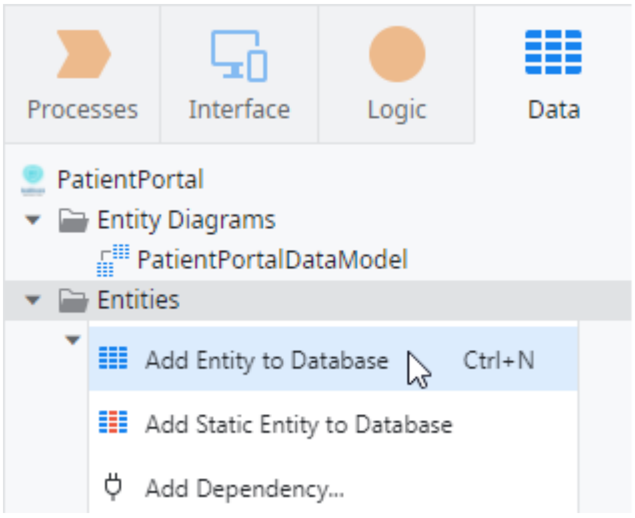
1) In Service Studio, select the **Data** tab on the top-right corner. This will open an area of Service Studio where you can see and define the data for your apps.



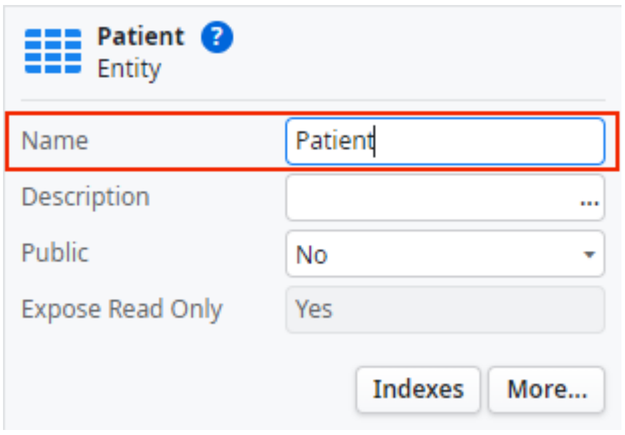
When you open the Data tab you will see many folders with elements that are automatically created when you create a new application, as well as elements that are part of the quickstart app that you have just installed. The Entities for doctors, clinics, specialties, etc. were already created to speed up the process.



2) Right-click the Entities folder and select **Add Entity to Database**.



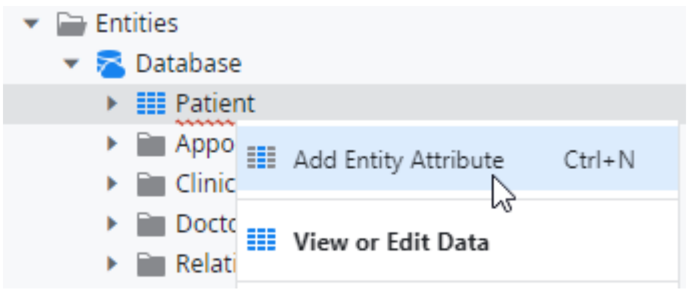
3) Set the **Name** of the Entity to *Patient*, in the Properties editor on the right sidebar of Service Studio.



Patient
Entity

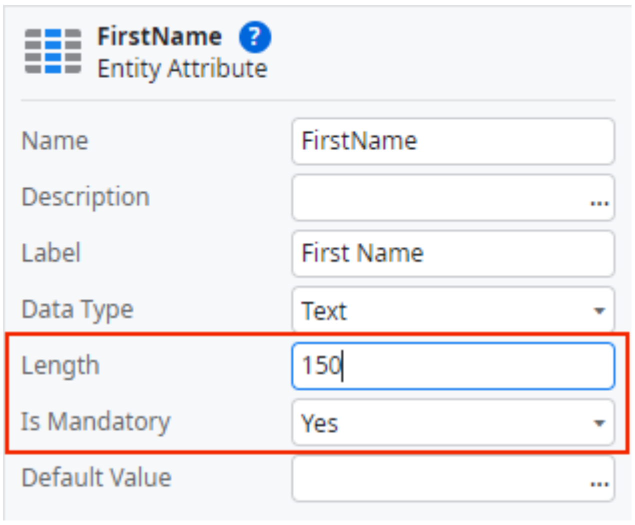
Name	Patient
Description	...
Public	No
Expose Read Only	Yes

4) Right-click on the newly created Entity and select **Add Entity Attribute**.



An Entity is defined by Attributes. The attributes define the element represented by the Entity and they represent a column in the database table. Examples of Entity attributes are: Name, Address, Zip Code, City, and so on.

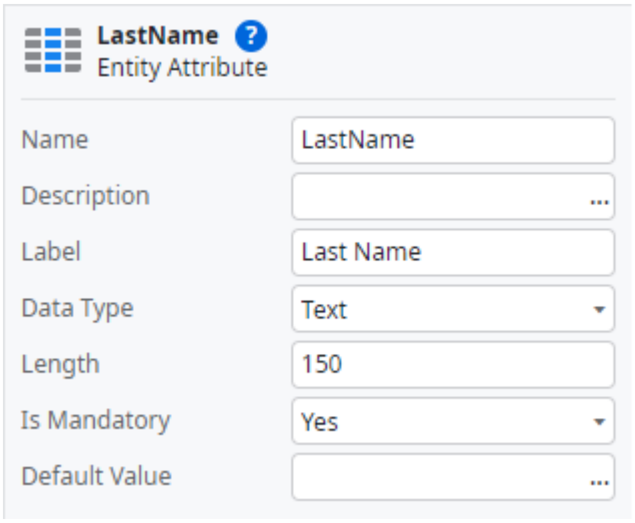
5) Set the attribute's **Name** to *FirstName* in the Properties editor on the right sidebar of Service Studio. Then, change the **Length** to *150*, and the **Is Mandatory** property to **Yes**.



FirstName
Entity Attribute

Name	FirstName
Description	...
Label	First Name
Data Type	Text
Length	150
Is Mandatory	Yes
Default Value	...

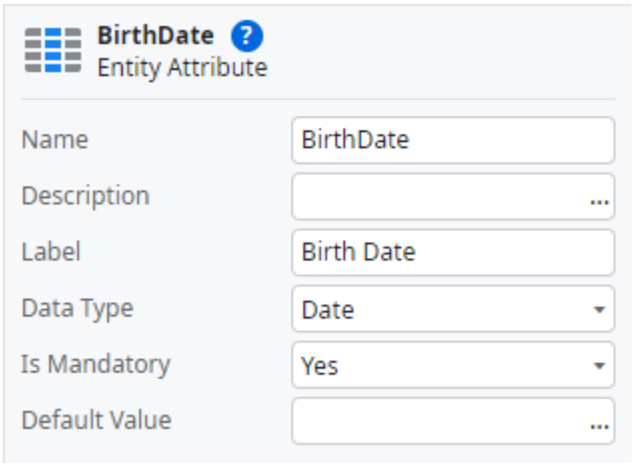
6) Add another attribute just like the previous one and name it *LastName*. Use the same Data Type and Length, and don't forget to set the **Is Mandatory** property to **Yes**.



LastName
Entity Attribute

Name	LastName
Description	...
Label	Last Name
Data Type	Text
Length	150
Is Mandatory	Yes
Default Value	...

7) Create a new attribute called *BirthDate* and make it **mandatory** as well.

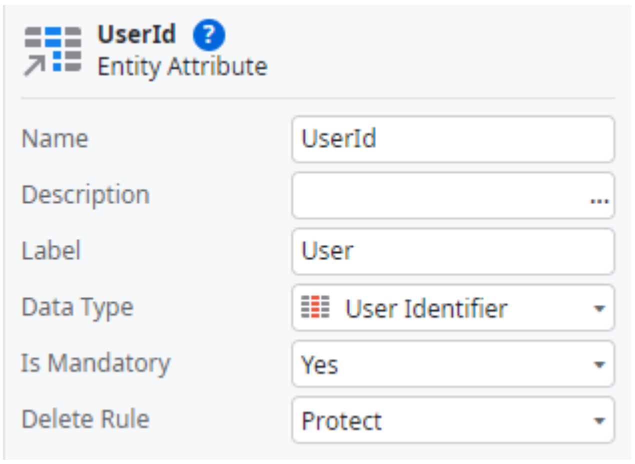


BirthDate
Entity Attribute

Name	BirthDate
Description	...
Label	Birth Date
Data Type	Date
Is Mandatory	Yes
Default Value	...

Note: the Data Type was automatically changed to Date. This happens because OutSystems tries to predict the desired Data Type based on its name. This is a good reason to use a naming convention that makes sense!

8) Last but not least, create another attribute, name it *UserId* and make the attribute **mandatory**.



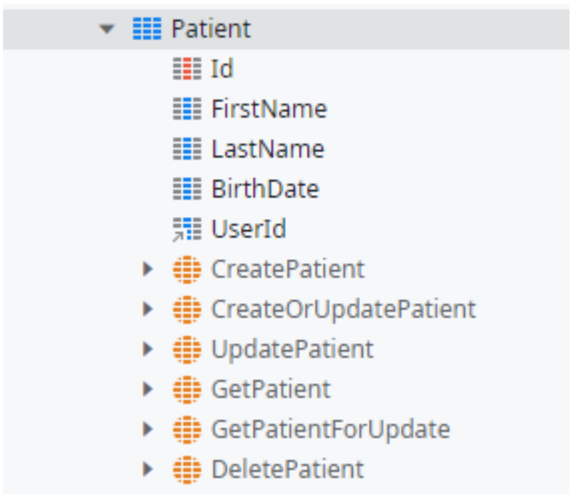
UserId
Entity Attribute

Name	UserId
Description	
Label	User
Data Type	User Identifier
Is Mandatory	Yes
Delete Rule	Protect

The Data Type changed to **User Identifier**. This is a special attribute, since it is linking this Patient Entity with the already existing User Entity, creating a relationship between them. This is a foreign key attribute.

The User Entity represents the users of the application. So, since a patient will use the app, we create a relationship between the two Entities.

Your Entity should look like the image below:

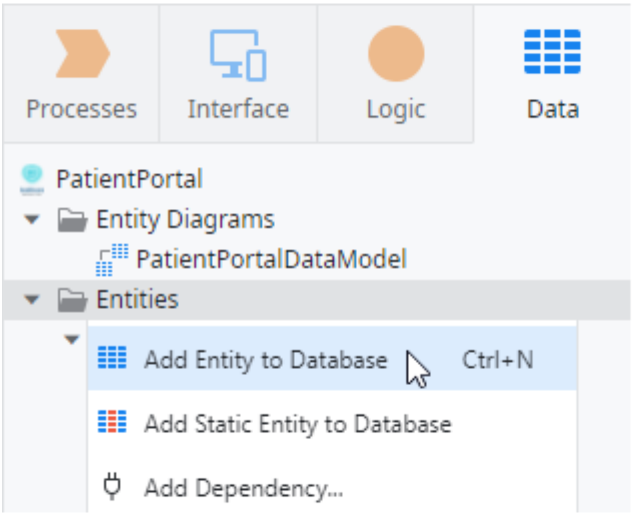


```
Patient
    Id
    FirstName
    LastName
    BirthDate
    UserId
    CreatePatient
    CreateOrUpdatePatient
    UpdatePatient
    GetPatient
    GetPatientForUpdate
    DeletePatient
```

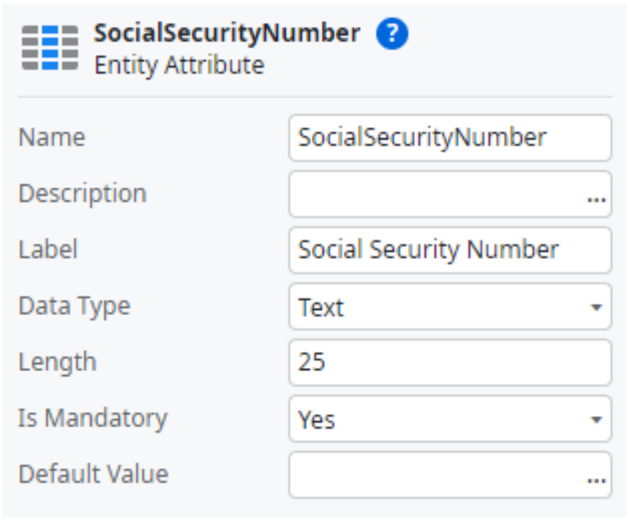
Create the PatientOnboarding Entity

The PatientOnboarding Entity will have some relevant information about the patient registration, such as the social security number, the insurance id number, the patient signature and the information regarding if the patient accepted the terms and conditions and the usage of personal data.

1) Right-click the Entities folder and select **Add Entity to Database**. Set the Entity's **Name** to *PatientOnboarding*.



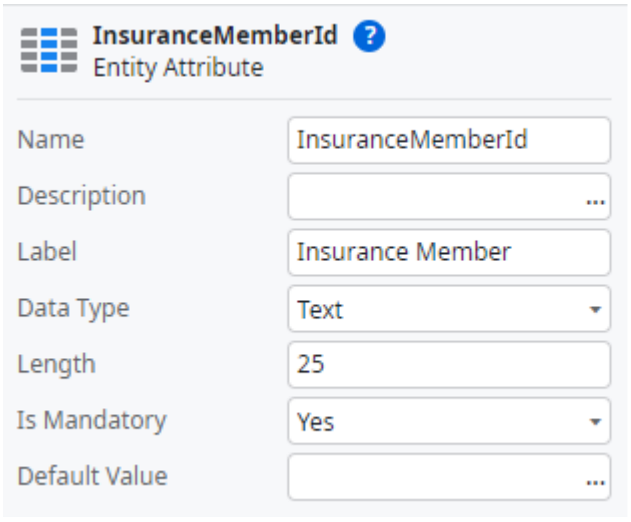
2) Add a new attribute to the Entity. Set its **Name** to *SocialSecurityNumber*, the **Data Type** to **Text**, the **Length** to *25* and make it **mandatory**.



SocialSecurityNumber
Entity Attribute

Name	SocialSecurityNumber
Description	
Label	Social Security Number
Data Type	Text
Length	25
Is Mandatory	Yes
Default Value	

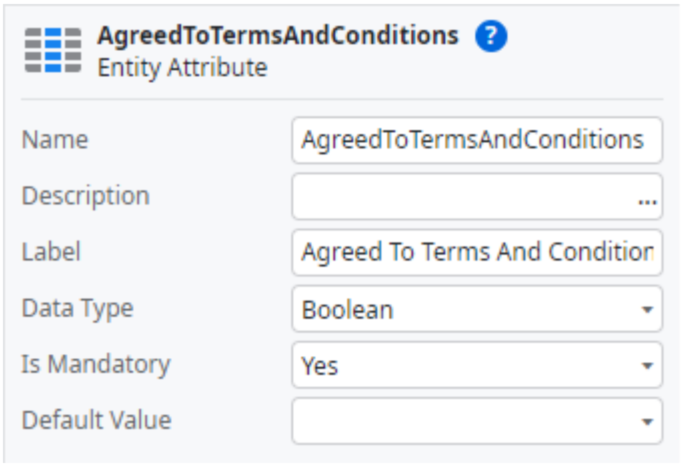
3) Add a new attribute, *InsuranceMemberId*. This attribute should be a **Text**, with **Length** set to *25* and **mandatory**.



InsuranceMemberId
Entity Attribute

Name	InsuranceMemberId
Description	...
Label	Insurance Member
Data Type	Text
Length	25
Is Mandatory	Yes
Default Value	...

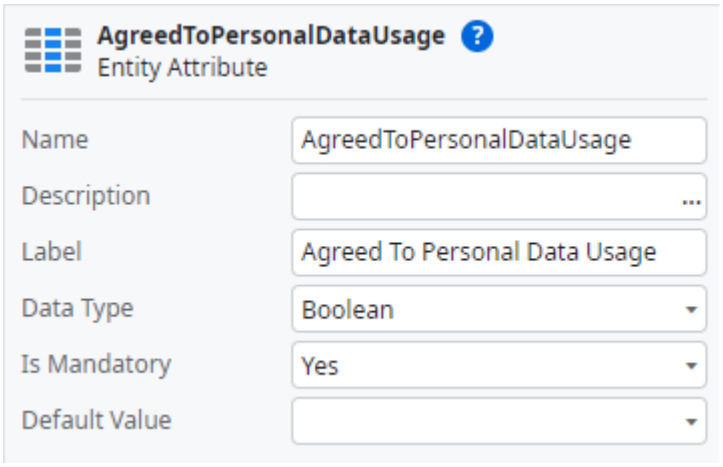
4) Add a new attribute, *AgreedToTermsAndConditions*. This attribute should be a **Boolean** and **mandatory**.



AgreedToTermsAndConditions
Entity Attribute

Name	AgreedToTermsAndConditions
Description	...
Label	Agreed To Terms And Condition
Data Type	Boolean
Is Mandatory	Yes
Default Value	

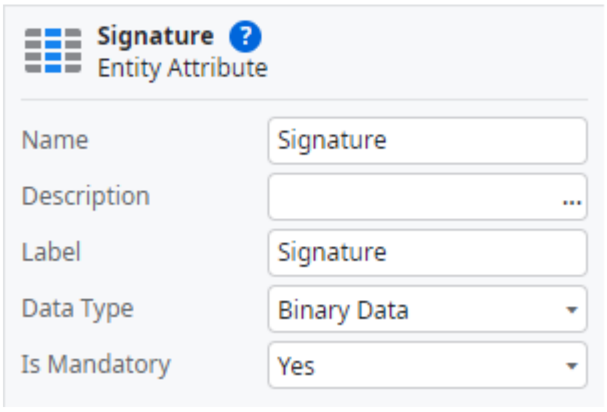
5) Add a new attribute, *AgreedToPersonalDataUsage*. This attribute should be a **Boolean** and **mandatory**.



AgreedToPersonalDataUsage
Entity Attribute

Name	AgreedToPersonalDataUsage
Description	...
Label	Agreed To Personal Data Usage
Data Type	Boolean
Is Mandatory	Yes
Default Value	

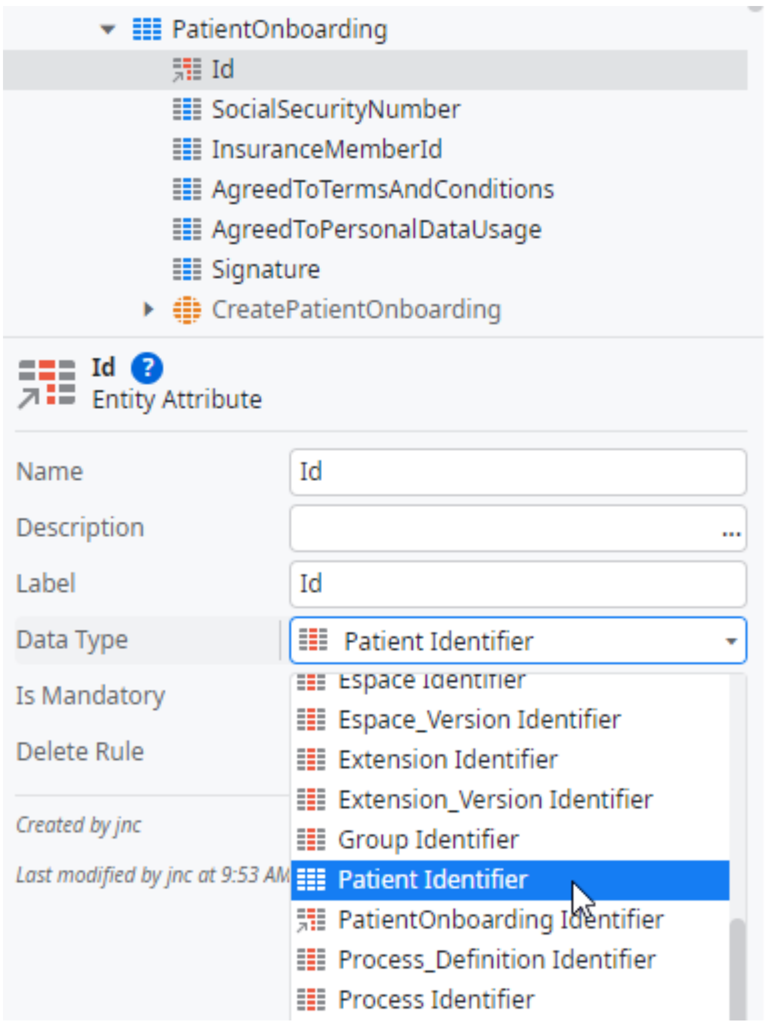
6) Add one last attribute called *Signature*. Set its **Data Type** to **Binary Data** and make it **mandatory**.



Signature
Entity Attribute

Name	Signature
Description	...
Label	Signature
Data Type	Binary Data
Is Mandatory	Yes

7) Click on the **Id** attribute to select it. Expand the **Data Type** dropdown and select the **Patient Identifier** data type.



Id
Entity Attribute

Name	Id
Description	...
Label	Id
Data Type	Patient Identifier
Is Mandatory	
Delete Rule	

Created by jnc

Last modified by jnc at 9:53 AM

You just turned the PatientOnboarding Entity into an extension of the Patient Entity. This means that one Patient will only have one Patient Onboarding record associated with it. This is how you create a one-to-one relationship between two Entities in OutSystems.

You can learn more about Entity relationships in our documentation.

Create Patient Image Entity

Finally, you need to create a new Entity to save the picture of the Patient, which includes the image itself and its file name. This Entity should also have a one-to-one relationship with the Patient, since a Patient has only one image associated with it.

1) Create another Entity and set its **Name** to *PatientImage*.

This entity will be used similar to the Patient Onboarding Entity, but instead of the Onboarding information, it will store the Patient Image. Having a separate Entity for the Image is one of the best practices that can help with performance.

2) Add a new attribute and set its **Name** to *Image* and make sure its **Data Type** is set to **Binary Data**.

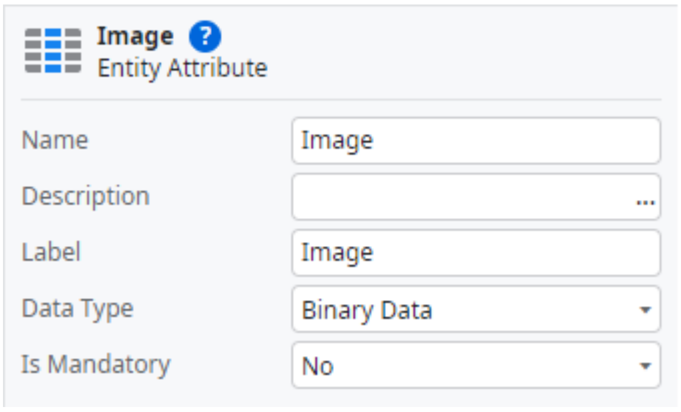
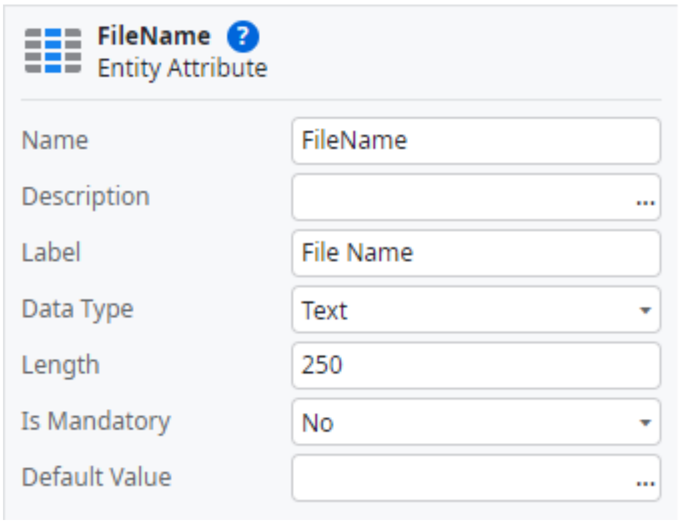


Image
Entity Attribute

Name	Image
Description	
Label	Image
Data Type	Binary Data
Is Mandatory	No

Let's not change the Is Mandatory property, so keep it as **No**.

3) Add a new attribute and set its **Name** to *FileName* and make sure its **Data Type** is set to **Text**.

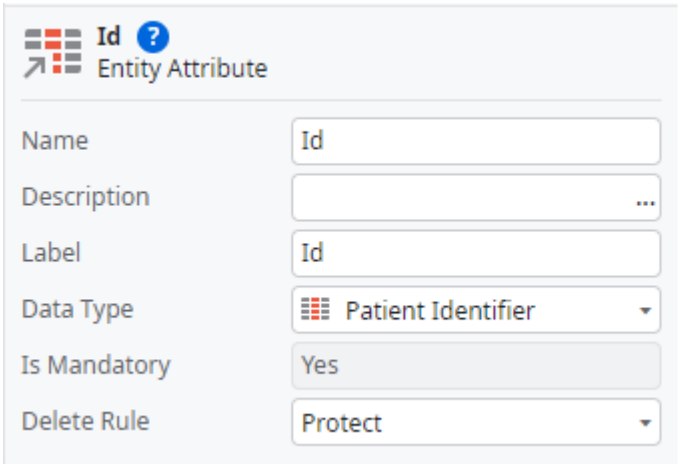


FileName
Entity Attribute

Name	FileName
Description	
Label	File Name
Data Type	Text
Length	250
Is Mandatory	No
Default Value	

Let's not change the Is Mandatory property, so keep it as **No**.

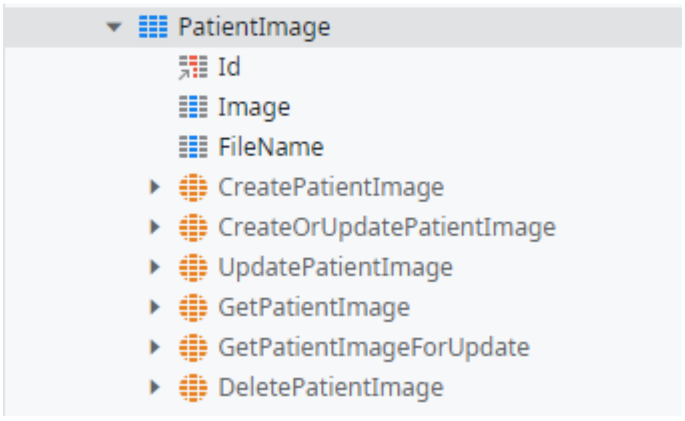
4) Select the **Id** attribute and open the **Data Type** dropdown and select **Patient Identifier**.



Id
Entity Attribute

Name	Id
Description	
Label	Id
Data Type	Patient Identifier
Is Mandatory	Yes
Delete Rule	Protect

Your Entity should look like this:

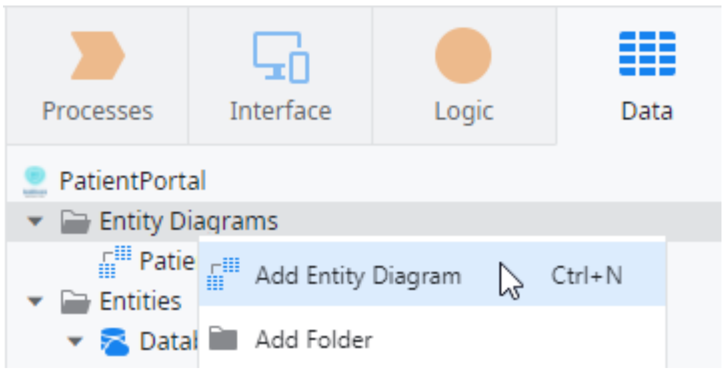


```
PatientImage
    Id
    Image
    FileName
    CreatePatientImage
    CreateOrUpdatePatientImage
    UpdatePatientImage
    GetPatientImage
    GetPatientImageForUpdate
    DeletePatientImage
```

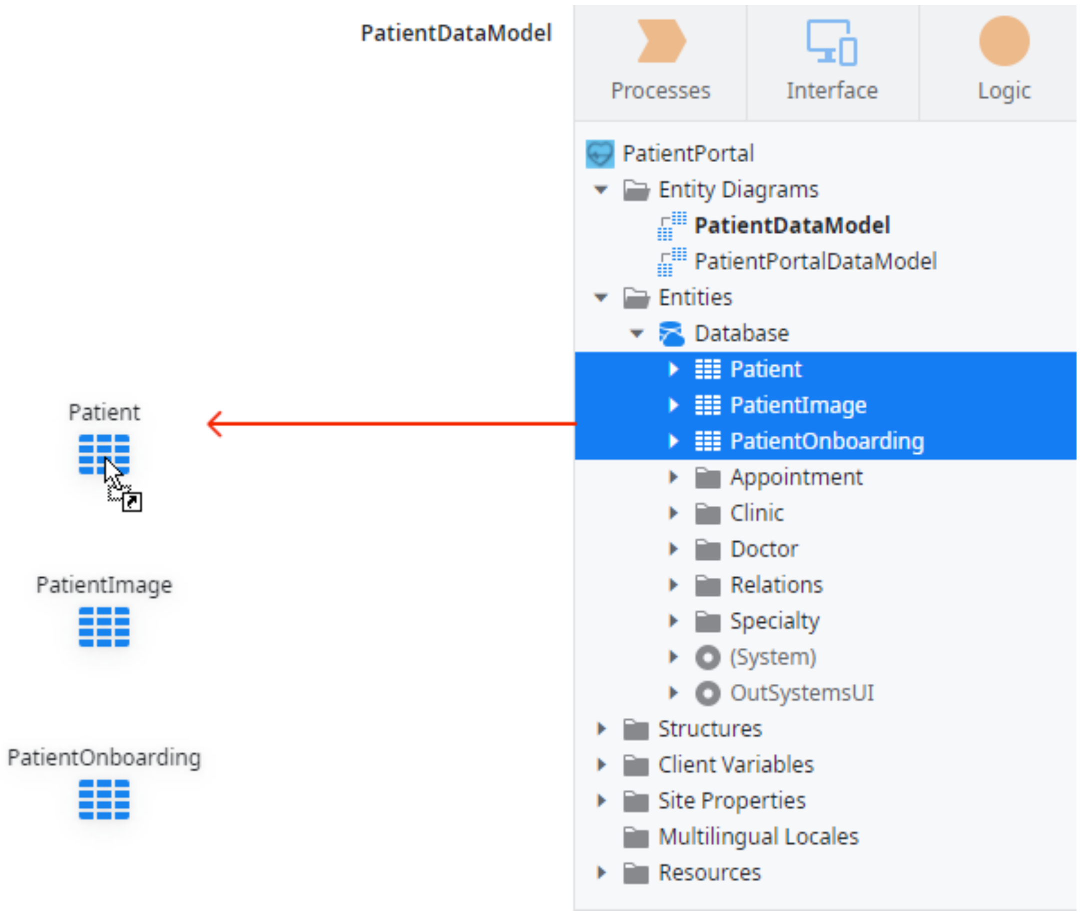
Data Model Diagram (Optional)

This section is optional, but will help you understand the relationship between the Entities you just created. So you will create a visual represantation of the data model.

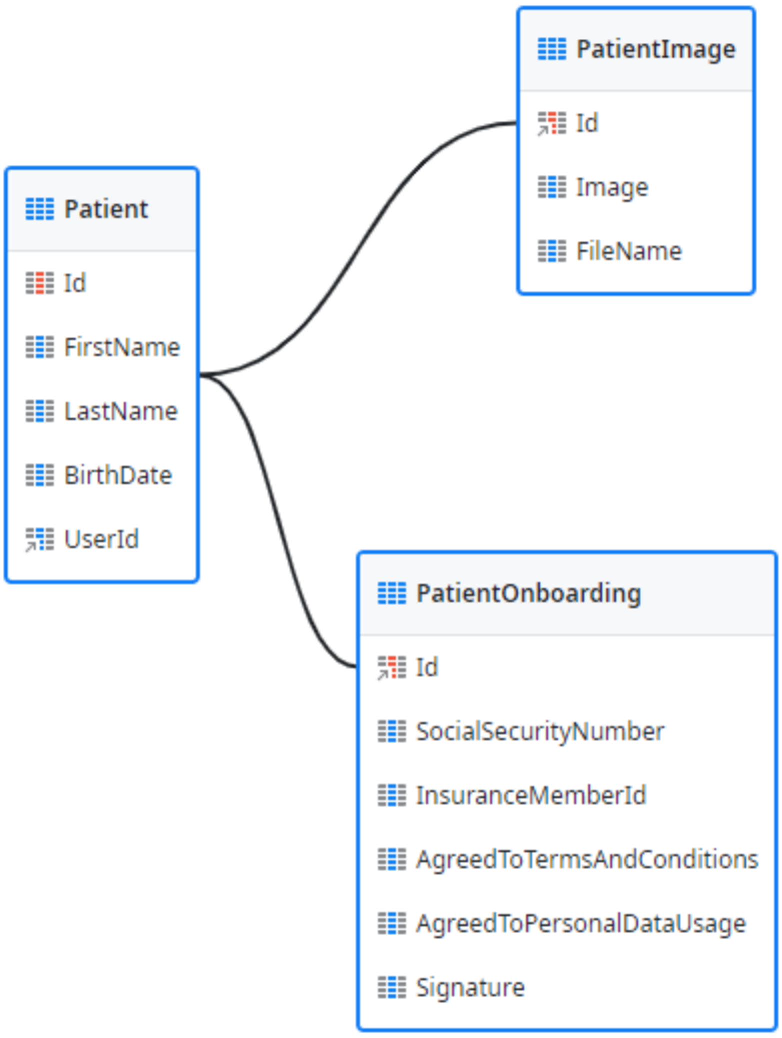
1) Right-click on the Entity Diagram folder and select **Add Entity Diagram**. Set its **Name** to *PatientDataModel*.



2) Select the **Patient**, **PatientImage**, and **PatientOnboarding** Entities (you can use CTRL+click to select several elements). Drag and drop the selected Entities to the empty data model you have just created.



Your diagram will look something like the image below.



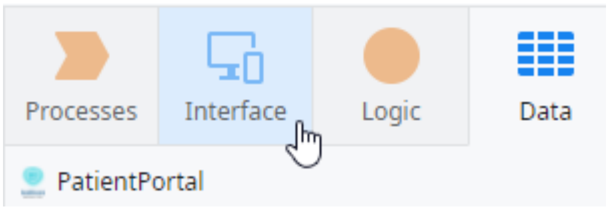
Creating the Registration Navigation Flow

In this section, you will create the navigation flow for registering a new patient in the app. You are not going to build the interface yet, just the navigation between Screens.

Let's start by creating the Registration Screen.

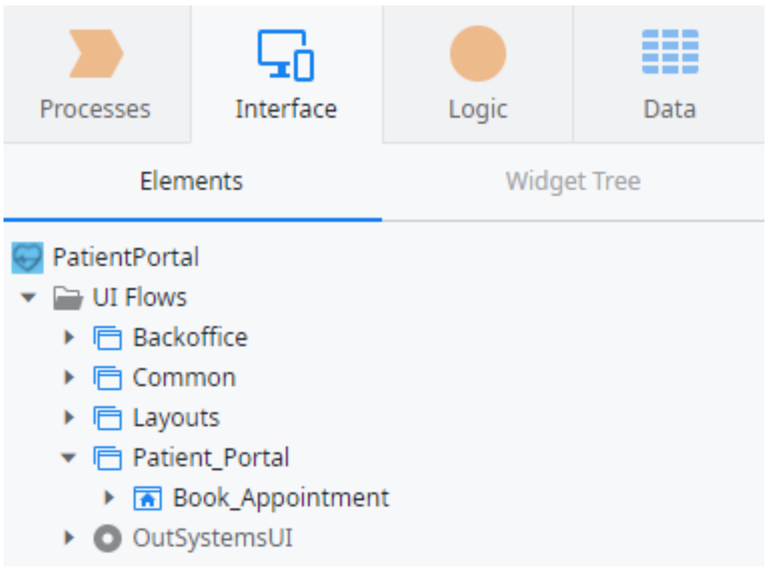
Create the Registration Screen

1) Click on the **Interface** tab.

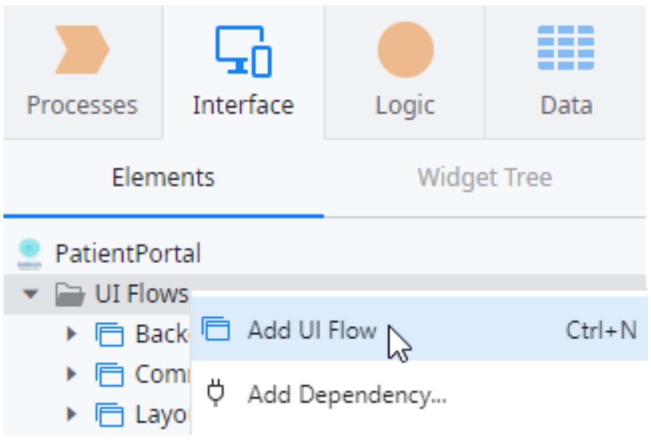


You will find a folder with UI Flows. A UI Flow is an element that groups visual elements, such as Screens. A Screen will represent a page from the app that the user will be able to access and interact with.

For instance, we have the **Patient_Portal** UI flow with the **Book_Appointment** Screen and logic ready.

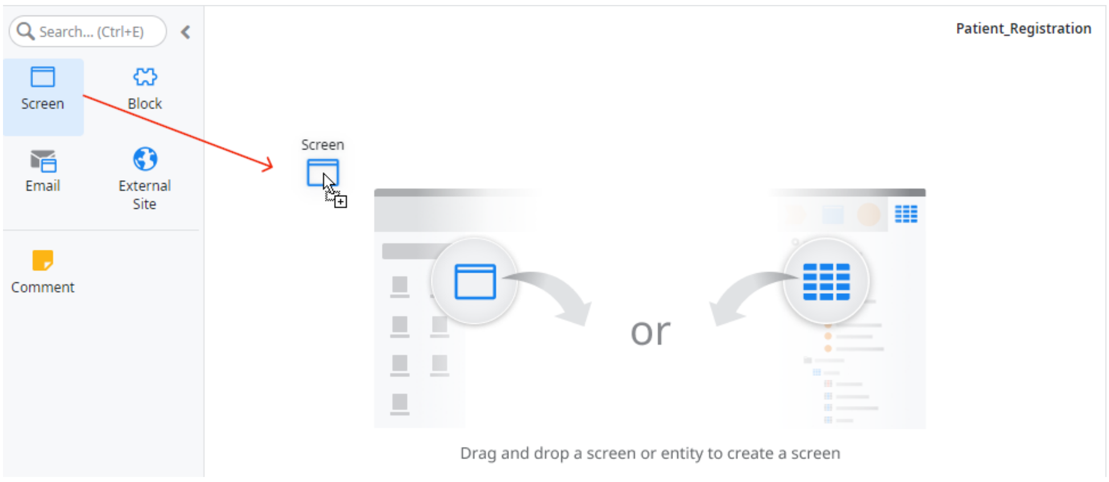


2) Right-click on the UI Flows folder and select **Add UI Flow**.

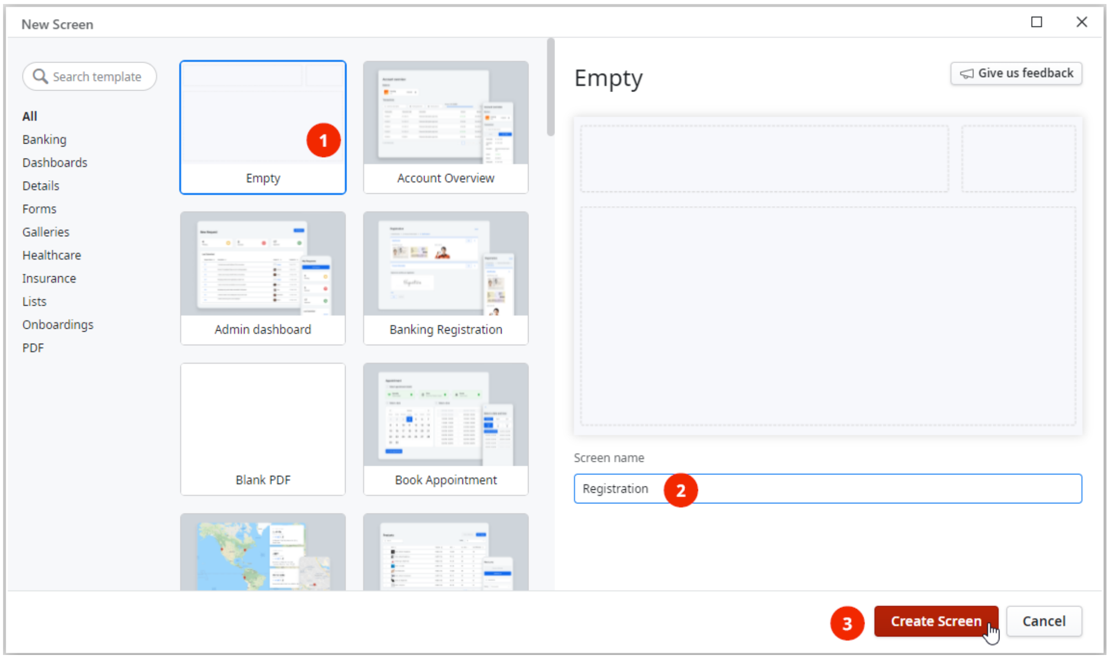


3) Type *Patient_Registration* as the name of the new flow.

4) You just created a new UI flow that has no elements inside it. You can now see at the center of Service Studio that you can drag and drop a Screen to the empty area to create a new Screen. Let's do that! Drag a Screen from the Toolbox (left sidebar) and drop it on the empty area.

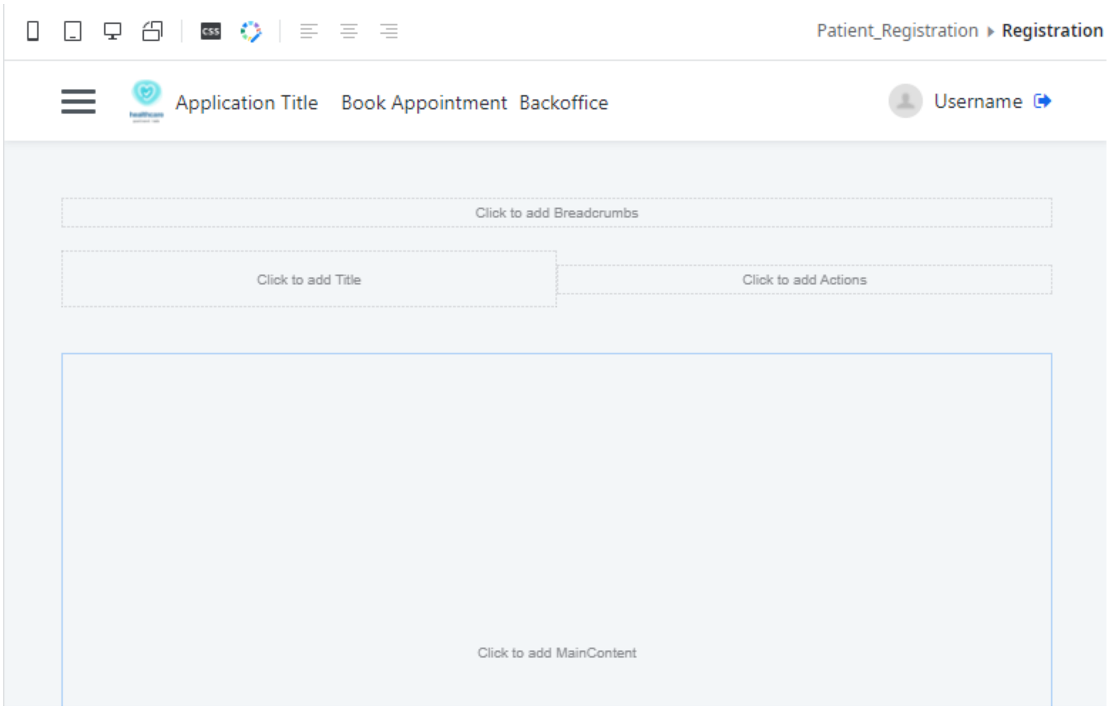


5) Select **Empty** in the new dialog, set the Screen name to *Registration* and click on **Create Screen**.

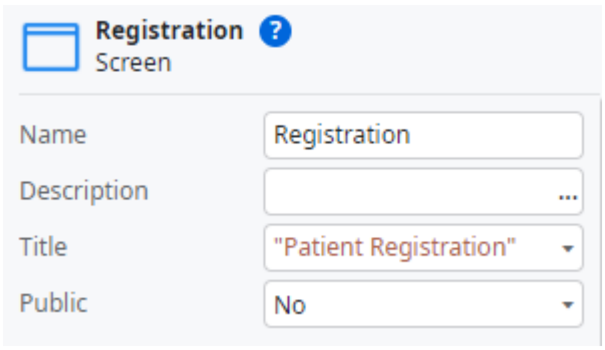


OutSystems offers Screen Templates that can be used to create Screens with predefined layouts, widgets, components, styles, and logic. The Screens you create from built-in Screen Templates have sample data that allow you to see how the Screen is designed and to get inspired for developing your apps. You can check the OutSystems UI website for the list of templates and their previews.

At this point you have an empty Screen, but OutSystems already gives you a basic structure where you can add elements later.



6) In the Screen properties on the right, set the **Title** property to *"Patient Registration"*.

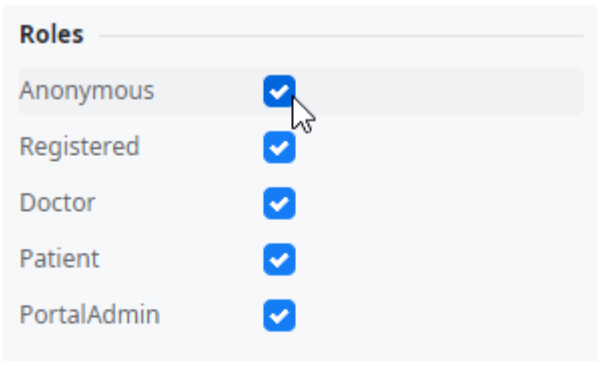


Registration
Screen

Name	Registration
Description	
Title	"Patient Registration"
Public	No

This property defines what will be displayed in the title bar of the browser.

7) Still in the properties, click on the **Anonymous** checkbox in the Roles area.

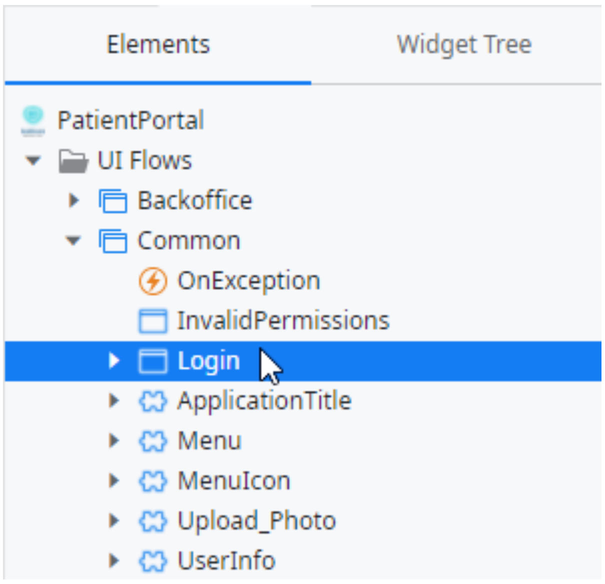


This is a very easy way to define who can access the Screens. In this case, any end-user will have access to the Screen.

Redirect to Registration Screen from Login

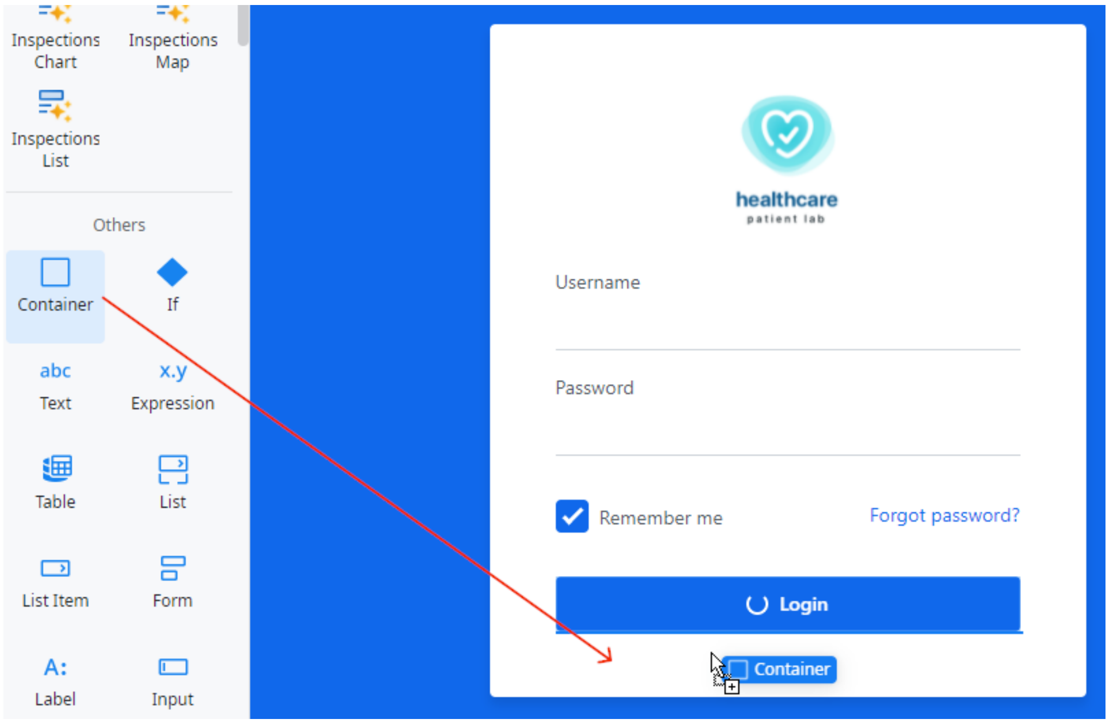
Now, it's time to focus on what happens during login. So, if the user is not registered yet on the app, they need to be redirected to the Registration Screen you just created.

1) Expand the **Common** UI Flow and double-click on the **Login** Screen to open it.

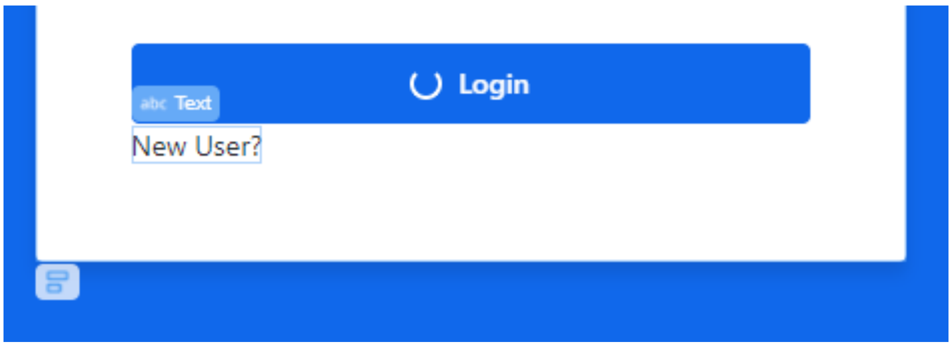


Applications with UI created with OutSystems already have some elements that can help you, such as the Login Screen and logic. And they can be customized to your needs!

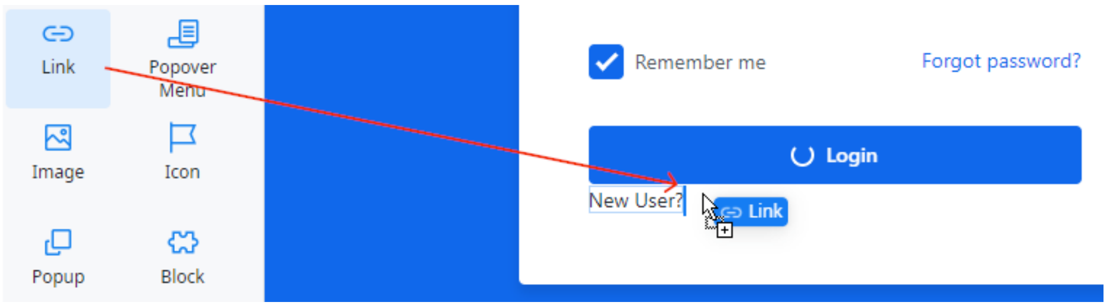
2) Drag a **Container** from the Toolbox on the left sidebar and drop it under the Login button, just like shown in the picture.



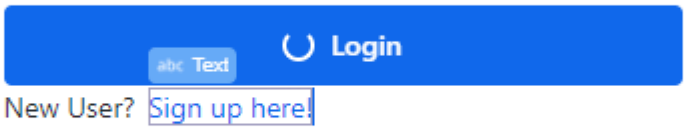
3) Type *New User?* in the Container



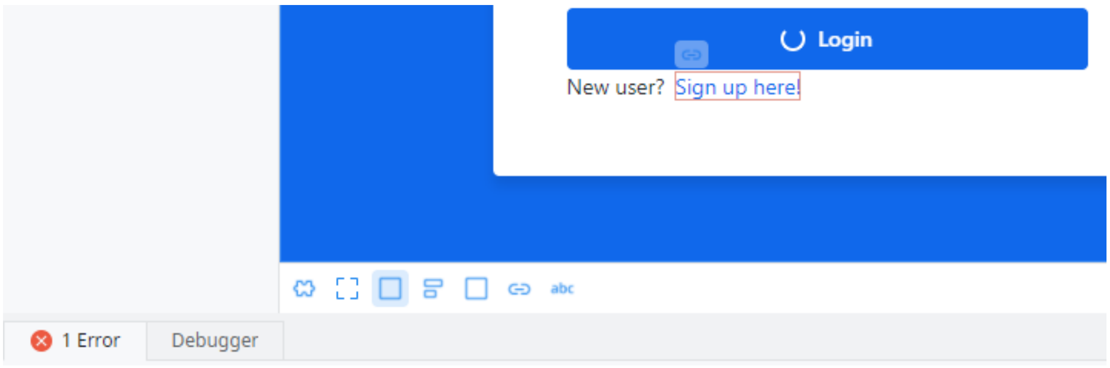
4) Drag a **Link** from the left sidebar and drop it right after the New User text.



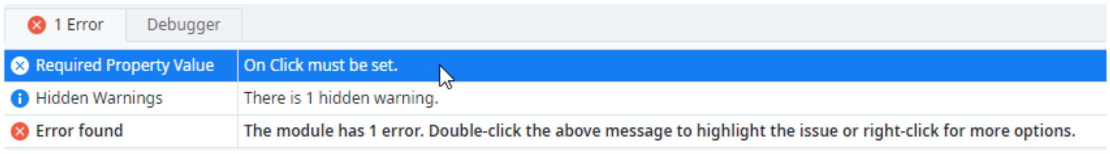
5) Select the link text that was automatically created and replace it by *Sign up here!*
.



You will have an error on your app. Don't worry, you will fix it in the next steps.

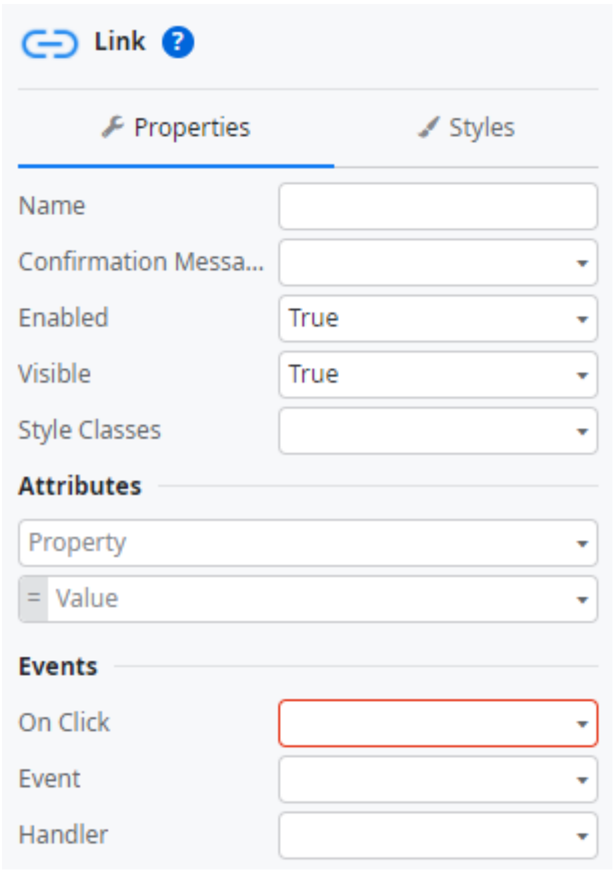


6) Click on the **1 Error** tab to view more details and double-click on the error to see more information about it.



Required Property Value	On Click must be set.
Hidden Warnings	There is 1 hidden warning.
Error found	The module has 1 error. Double-click the above message to highlight the issue or right-click for more options.

This will lead you right to the error. In this case, it will open the Link element properties on the right sidebar, where you can see that the OnClick property has a red border, indicating that is where the error exists.



Link

Properties | Styles

Name	
Confirmation Messa...	
Enabled	True
Visible	True
Style Classes	

Attributes

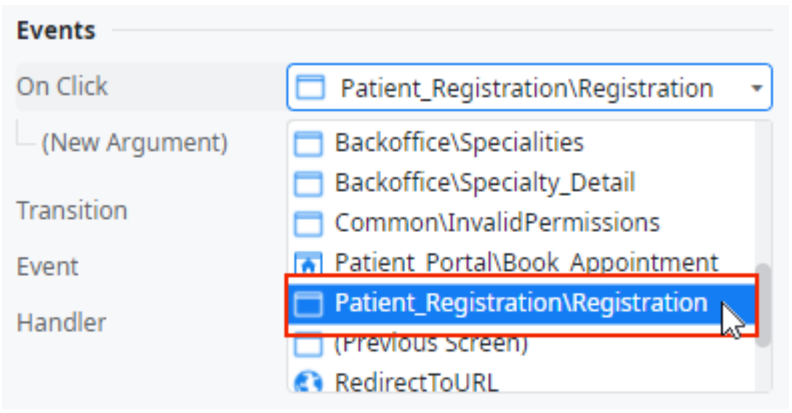
Property
= Value

Events

On Click	
Event	
Handler	

This error states that when we have a Link, we must make sure we define an action for when the user clicks on the link, which is defined in this property.

7) Click on the **OnClick** property's dropdown and select the **Registration** Screen.



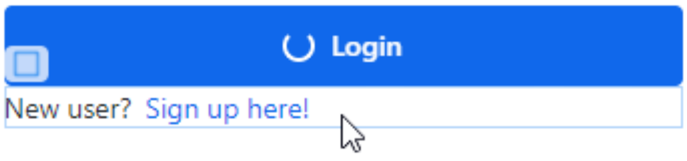
Events

On Click	Patient_Registration\Registration
(New Argument)	
Transition	
Event	
Handler	

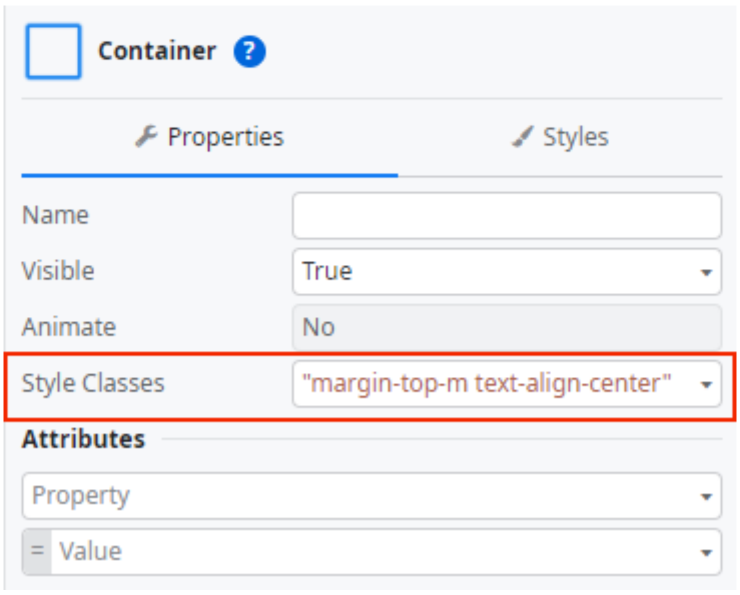
Backoffice\Specialities
Backoffice\Specialty_Detail
Common\InvalidPermissions
Patient_Portal\Book_Appointment
Patient_Registration\Registration
(Previous Screen)
RedirectToURL

This means that whenever a user clicks on the link, they will be redirected to the Registration Screen.

8) Click on the **Container** where the Text and Link are to see its properties. Make sure you select the Container and not the Text or Link. To be sure, just click in the area right next to the Link, as the image shows.



9) Set the **Style Classes** property to `"margin-top-m text-align-center"`.



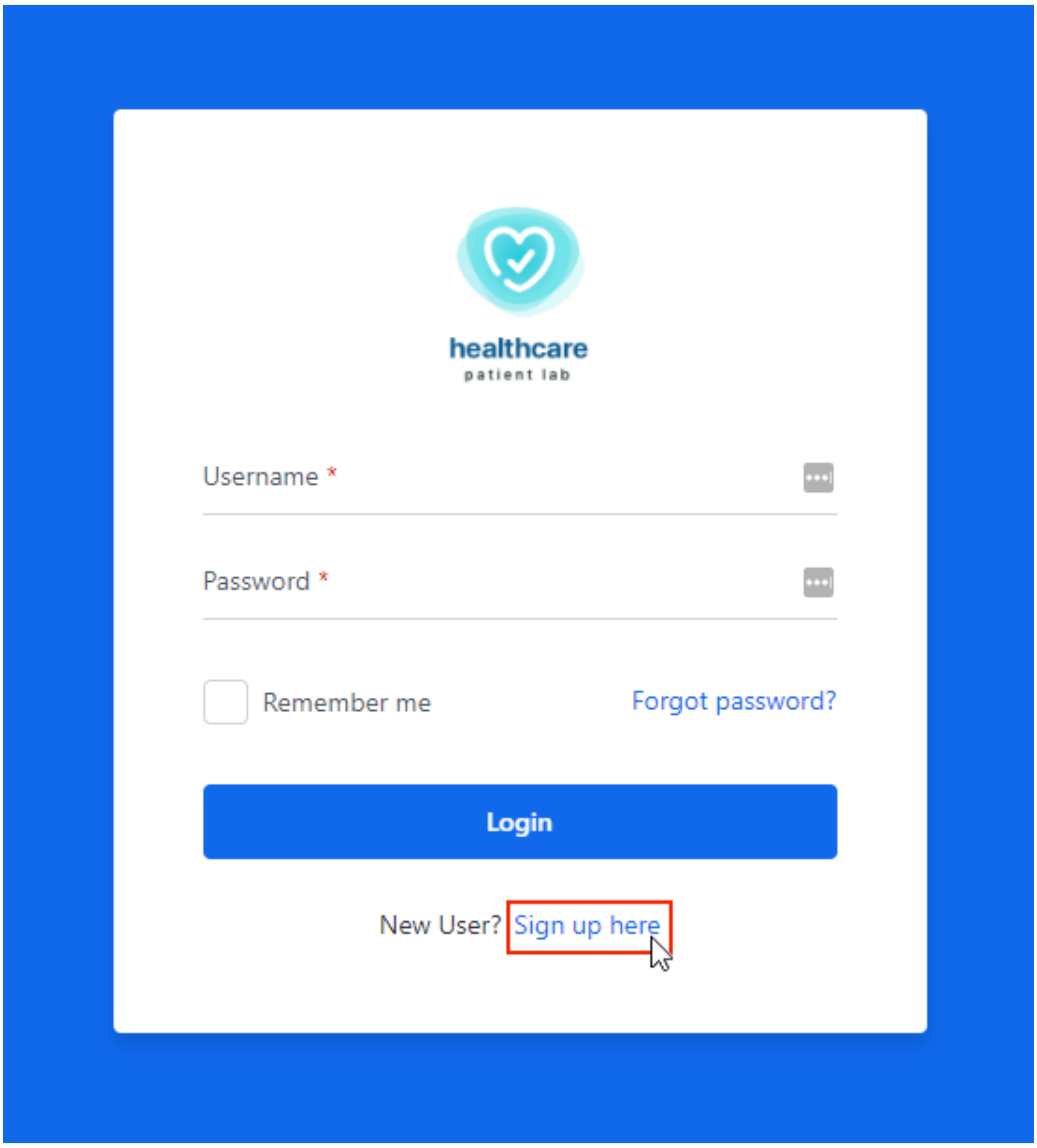
Name	
Visible	True
Animate	No
Style Classes	"margin-top-m text-align-center"

Style classes are used to change the look and feel of widgets in a reusable way. In this case, you're using some classes that already exist that will define a margin to the top (login button) and will align the text to center. Don't worry about it! There's no need to learn everything at once!

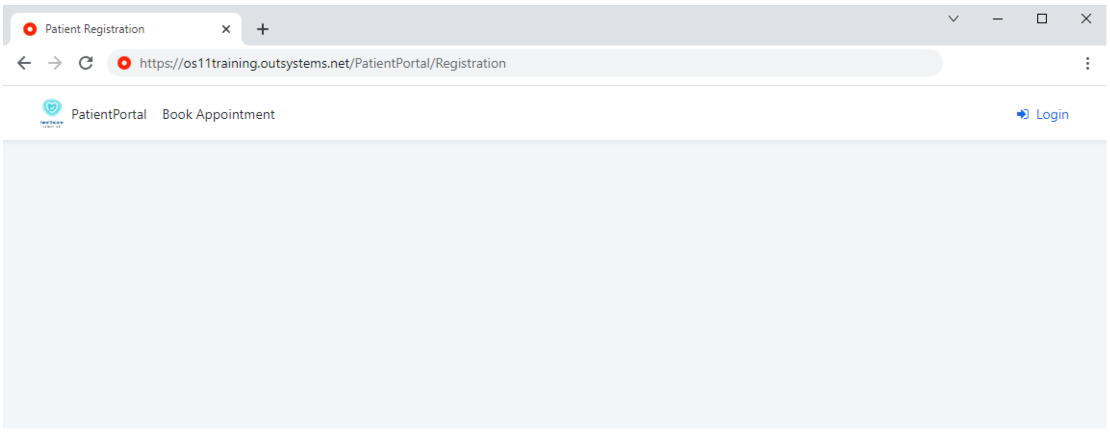
10) Publish the module.

11) When it is done publishing, click on the **Open in browser** blue button. This will open the app in the browser.

12) Click on the **Sign up** link to test what you just did.



So far the Registration Screen is empty and that's fine. You will have the opportunity to add some UI to it on the next lesson!


Patient Registration
https://os11training.outsystems.net/PatientPortal/Registration
PatientPortal
Book Appointment
Login

Wrapping up

Congratulations on finishing this tutorial. With this exercise, you had the chance to go through some essential aspects of OutSystems, and even 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) OutSystems Overview
2) Service Studio Overview
3) Intro to OutSystems Development
4) Modeling Data
5) Entity Relationships

See you in the next tutorial!