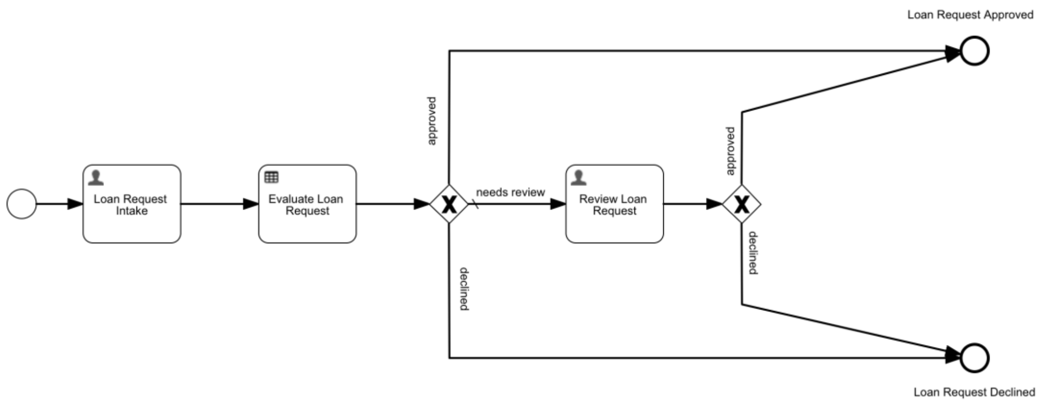
Decision Table

Introduction

In Flowable there is support to use decision tables directly in a case or process model, but it's also possible to model decision services to orchestrate the data of multiple decision tables.

Decision table: Model the Process

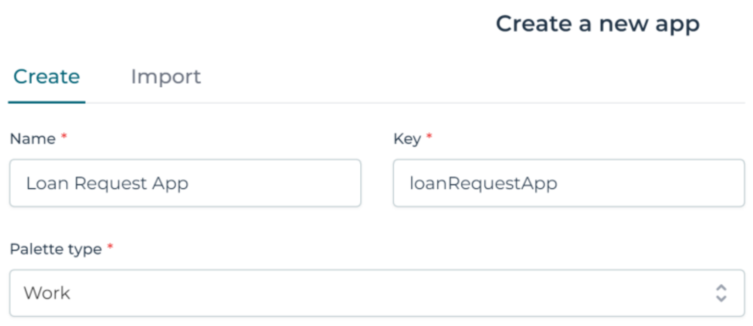


We start by creating the 'Loan Approval' app and a process called 'Loan Request Approval'. The process consists of a two-step happy path:

- Make the Loan Request: a user task form for providing the required data about the loan.
- Evaluate the Loan Request: a DMN decision table that evaluates the loan request.

The result of the decision execution is one of: "APPROVED", "DECLINED" or "REVIEW". In the last case, the process navigates to an additional review step.

First create the Loan Request Application:


Create a new app

Create | Import

Name * : Loan Request App

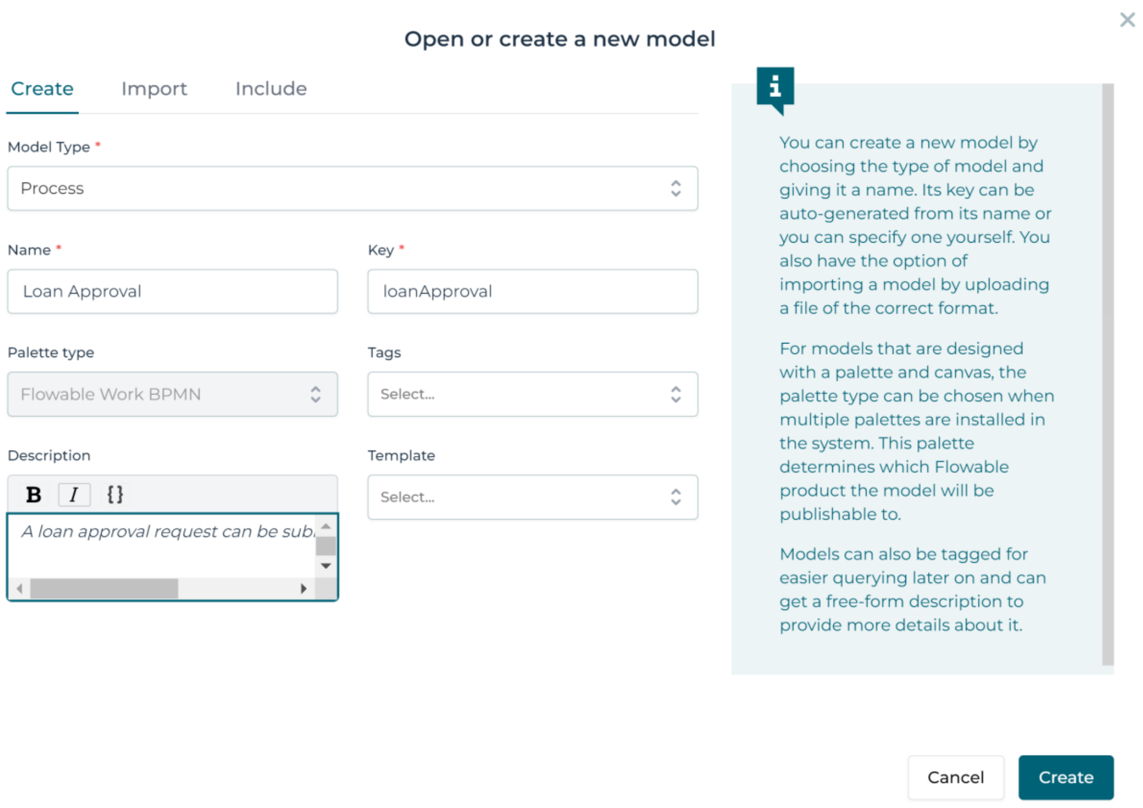
Key * : loanRequestApp

Palette type * : Work

Step 1: Create the Business Process Model

In Flowable Design, within the loan request application click the `Create` button and choose model type `Process`. Enter *Loan Approval* as the 'Model name', *loanApproval* as the 'Model key' and *A loan approval request can be submitted and will be approved.* as the 'Description'.

To create the model, click the `Create` button.



Model Type *: Process

Name *: Loan Approval

Key *: loanApproval

Palette type: Flowable Work BPMN

Tags: Select...

Description: A loan approval request can be sub

Template: Select...

You can create a new model by choosing the type of model and giving it a name. Its key can be auto-generated from its name or you can specify one yourself. You also have the option of importing a model by uploading a file of the correct format.

For models that are designed with a palette and canvas, the palette type can be chosen when multiple palettes are installed in the system. This palette determines which Flowable product the model will be publishable to.

Models can also be tagged for easier querying later on and can get a free-form description to provide more details about it.

Step 2: Create the 'Loan Request Intake' Step

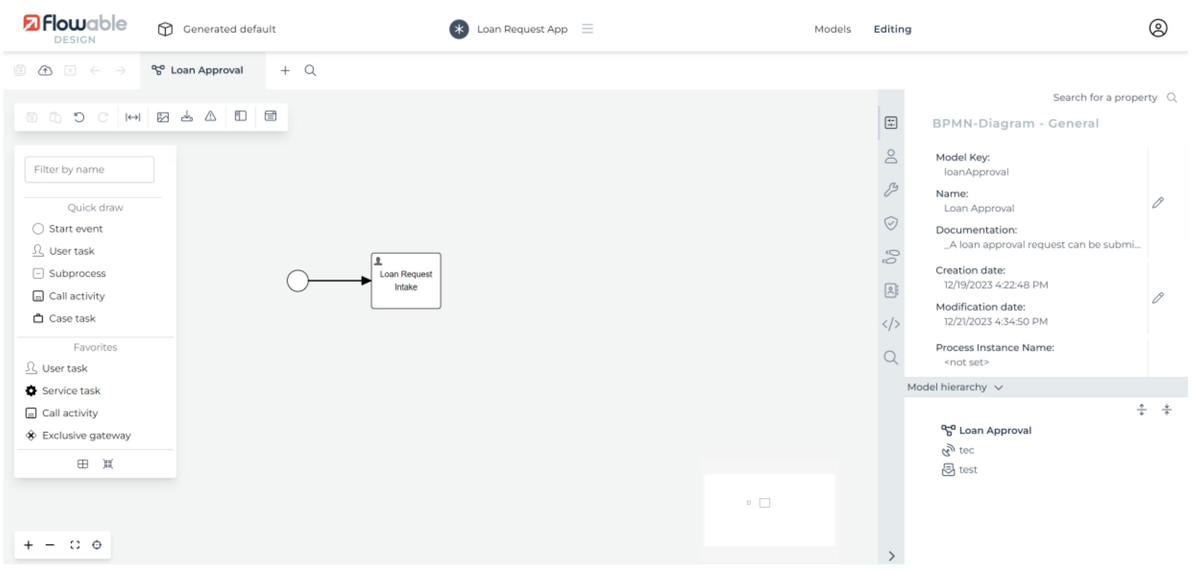
After creating the process model, you are directed to the BPMN designer. Here we can model our process.

We start by creating the `Intake Form`. For this, we create a BPMN `User task` with a form.

Step 2A: Create the 'Loan Request Intake User Task'

Select the `User task` from the palette on the left-hand side and drag it to the canvas (the middle section), and rename it to *Loan Request Intake*.

Connect the `Start event` with the newly created `User task` by selecting the `Start event` and selecting the `arrow` from the popover options and drag it to the *Loan Request Intake* User task.



BPMN-Diagram - General

Model Key:
loanApproval

Name:
Loan Approval

Documentation:
_A loan approval request can be submi...

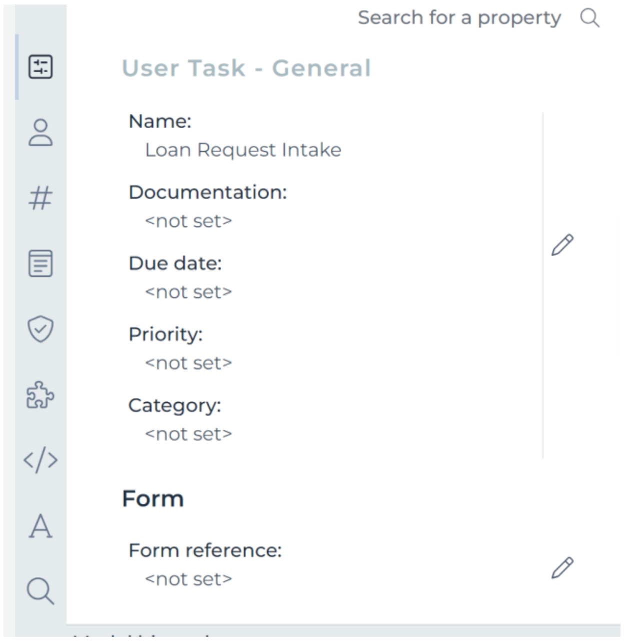
Creation date:
12/19/2023 4:22:48 PM

Modification date:
12/21/2023 4:34:50 PM

Process Instance Name:
<not set>

Next, we create a new form and attach it to the user task.

Select the `Loan Request Intake` task. Then in the panel on the right, select the `Form reference` property from the `User Task - General` section.



User Task - General

Name:
Loan Request Intake

Documentation:
<not set>

Due date:
<not set>

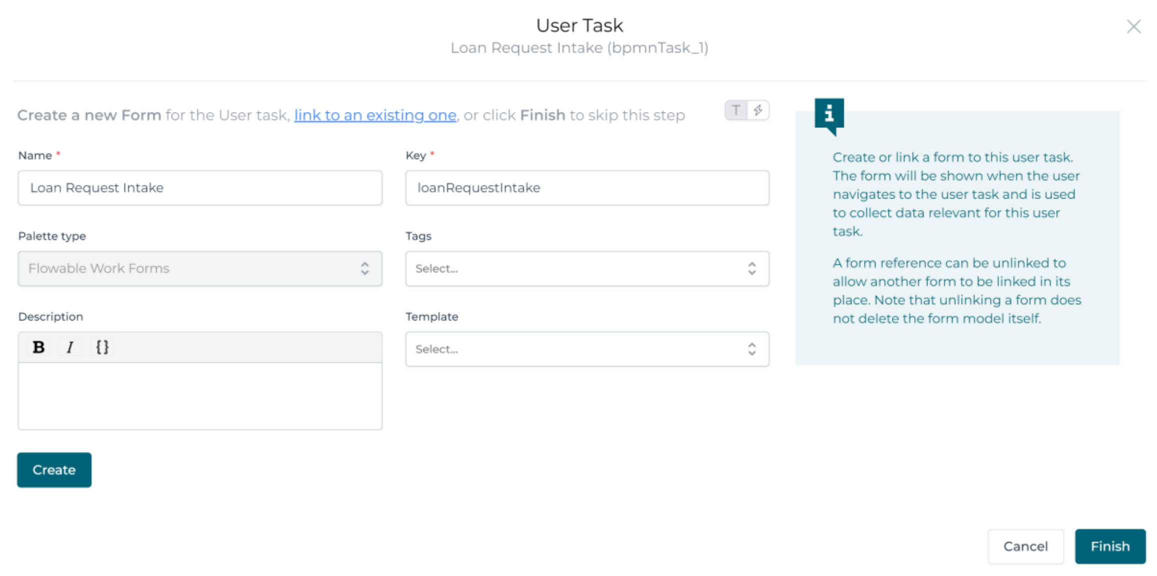
Priority:
<not set>

Category:
<not set>

Form

Form reference:
<not set>

In the resulting popup enter *Loan Request Intake* as 'Name' and `loanRequestIntake` as 'Key'. Then select `Create` followed by `Finish`.


User Task
Loan Request Intake (bpmnTask_1)

Create a new Form for the User task, link to an existing one, or click Finish to skip this step

Name *: Loan Request Intake
Key *: loanRequestIntake
Palette type: Flowable Work Forms
Tags: Select...
Description:
Template: Select...

Create or link a form to this user task. The form will be shown when the user navigates to the user task and is used to collect data relevant for this user task.

A form reference can be unlinked to allow another form to be linked in its place. Note that unlinking a form does not delete the form model itself.

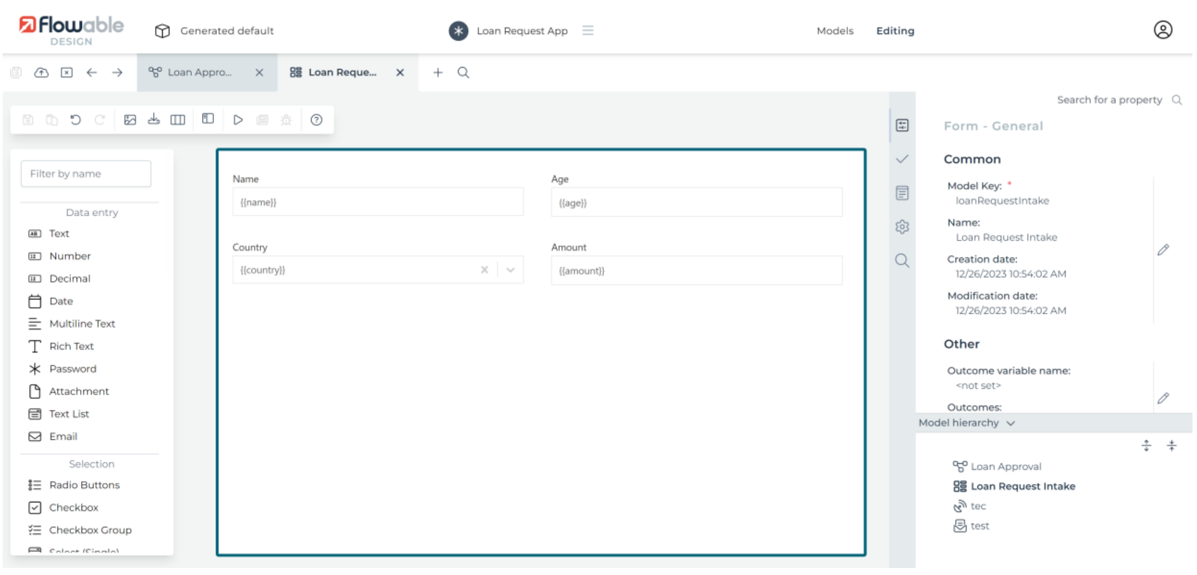
Step 2B: Create the 'Loan Request Intake Form'

In the Form Designer, we create the Intake form by dragging the required input fields onto the canvas.

Select the following data entry items from the palette on the left side and drag them to the canvas (the middle section).

When placing the fields on the canvas give them the following names and ids:

- type: 'Text', name: *Name*, id: *name*.
- type: 'Number', name: *Age*, id: *age*.
- type: 'Select (single)', name: *Country*, id: *country*.
- type: 'Number', name: *Amount*, id: *amount*.



Name: {{name}}
Age: {{age}}
Country: {{country}}
Amount: {{amount}}

Form - General

Common
Model Key: * loanRequestIntake
Name: Loan Request Intake
Creation date: 12/26/2023 10:54:02 AM
Modification date: 12/26/2023 10:54:02 AM

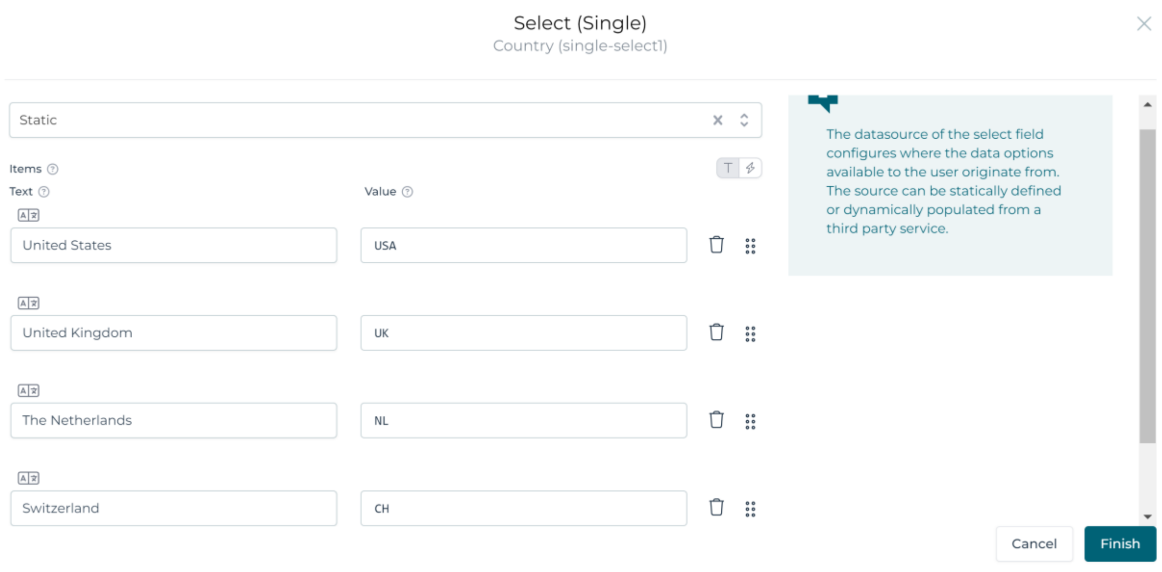
Other
Outcome variable name: <not set>
Outcomes:

The 'Single select' field *country* needs to have a list of values configured. This is done by selecting the field and then selecting the `Items` property in the `Select (Single) - Datasource` section in the panel on the right side of the screen.

Fill in the following country values:

- Text: *United States*, Value: *USA*
- Text: *United Kingdom*, Value: *UK*
- Text: *The Netherlands*, Value: *NL*
- Text: *Switzerland*, Value: *CH*

Finally, select `Finished`.


Select (Single)
Country (single-select1)

Static

Items
Text | Value

Text	Value
United States	USA
United Kingdom	UK
The Netherlands	NL
Switzerland	CH

The datasource of the select field configures where the data options available to the user originate from. The source can be statically defined or dynamically populated from a third party service.

The form is done and is saved by selecting the *Save* button in the toolbar. After the form is saved, navigate back to the process by selecting the 'Loan Approval Process' tab.

Step 3: Create the 'Evaluate Loan Request' step

In this step, a DMN decision table is used to evaluate the loan request. The decision about the loan is accomplished evaluating rules that utilize the data entered on the submitted form (intake).

The decision table can determine if a request is declined, approved, or if there is the need for a review.

The table consists of the following rules:

1. When the requester's age is under 21 the request is declined.
2. When the requester's age is 21 or over, lives in The Netherlands or Switzerland, and the requested amount is equal to or greater than 100000, a review is required.

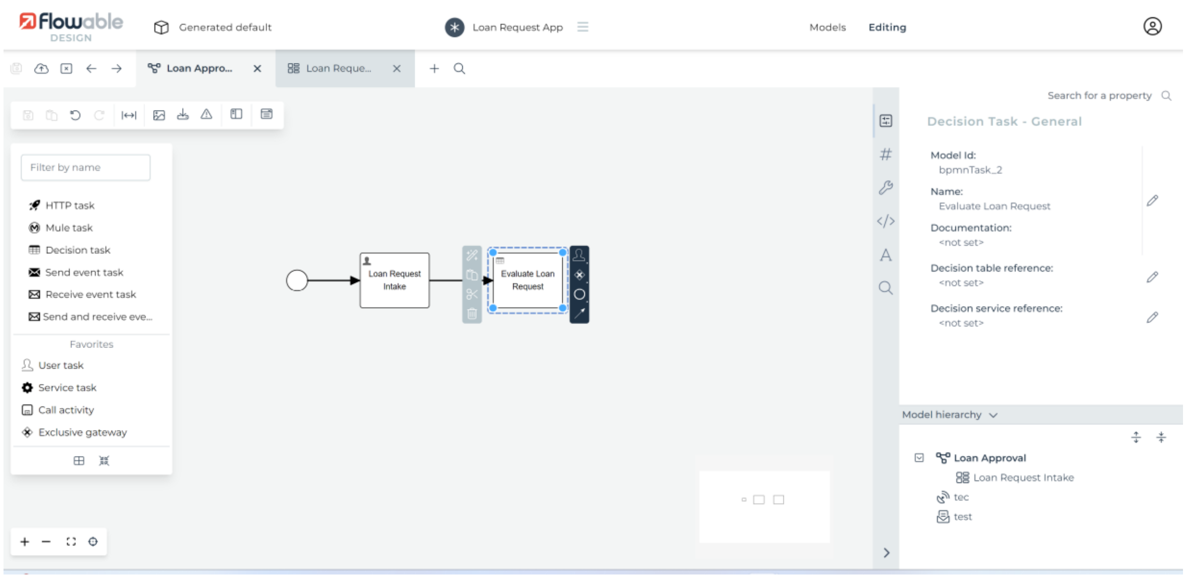
3. When the requester’s age is 21 or over, does not live in The Netherlands or Switzerland, and the requested amount is equal to or greater than 100000, the request is declined.
4. When the requester’s age is 21 or over, and the amount is less than 100000, the request is approved.

A decision table can have different 'hit policies'. The hit policy determines what the result of the decision table is when multiple rules are valid. In this case, we want that only one rule can be valid. For this, we chose the hit policy 'Unique'. If we design a table that, depending on the input data, has multiple valid rules, this would result in an invalid result.

Step 3A: Create the 'Evaluate Loan Request Decision Task'

Select the `Activites>Decision task` from the palette on the left-hand side and drag it to the canvas (the middle section). Double click the `Decision task` and rename it to *Evaluate Loan Request*.

Connect the *Loan Request Intake* task with the newly created `Decision task` by selecting the *Loan Request Intake* task and selecting the `arrow` from the popup options and drag it to the *Evaluate Loan Request* decision task.



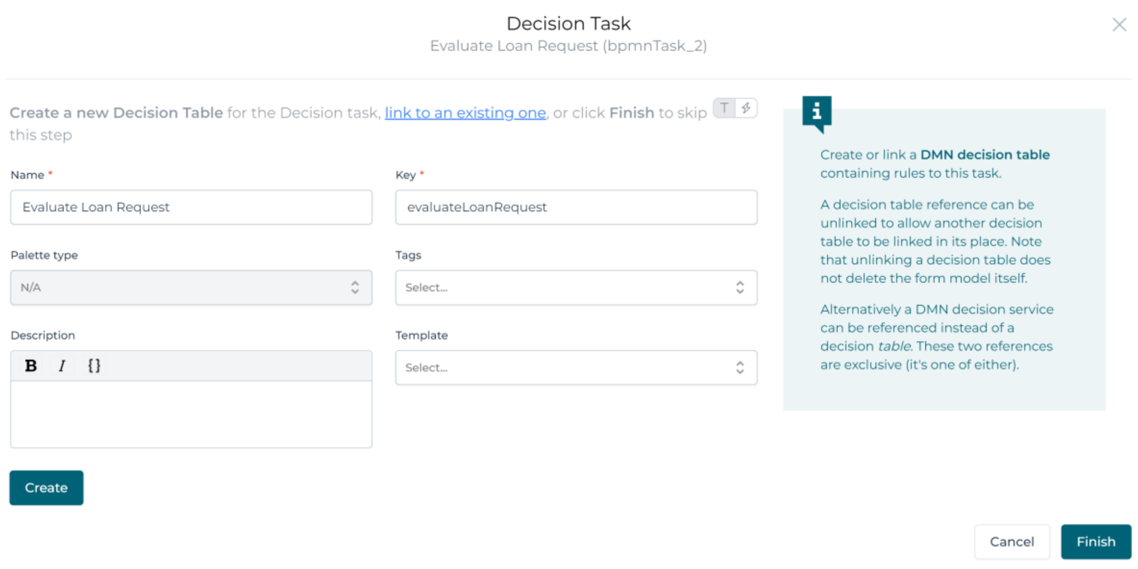
Decision Task - General

Model Id:
bpmnTask_2
Name:
Evaluate Loan Request
Documentation:
<not set>
Decision table reference:
<not set>
Decision service reference:
<not set>

We can now create the `Decision Table` and attach it to the Decision task by clicking the `Decision table reference` attribute in the `Decision Task - General` panel on the right side of the screen.

In the resulting popup set the 'Name' to *Evaluate Loan Request* and the 'Key' equal to *evaluateLoanRequest*.

Finally, select `Create` followed by `Finish`.


Decision Task
Evaluate Loan Request (bpmnTask_2)

Create a new **Decision Table** for the Decision task, link to an existing one, or click **Finish** to skip this step

Name *: Evaluate Loan Request
Key *: evaluateLoanRequest
Palette type: N/A
Tags: Select...
Description:
Template: Select...

Create

Create or link a **DMN decision table** containing rules to this task.

A decision table reference can be unlinked to allow another decision table to be linked in its place. Note that unlinking a decision table does not delete the form model itself.

Alternatively a DMN decision service can be referenced instead of a decision *table*. These two references are exclusive (it's one of either).

Cancel Finish

Step 3B: Create the 'Evaluate Loan Request Decision Table'

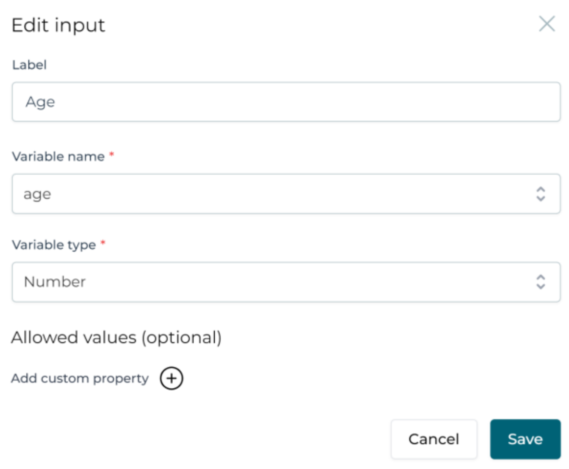
The DMN Decision Table Editor is automatically opened after creating the decision table.

A decision table consists of rules that have one or more expressions and one or more outcomes (outputs). The expressions are evaluated using the defined input data (inputs). In this case, some of the data from our intake form are used. This is done by referring to the variable names used in the definition of the intake form.

Inputs

Let us start by defining the inputs. Expand the left column and select `Edit column`, and enter the following values:

- Column label: *Age*, Variable name: *age*, Variable type: *number*.


Edit input

Label: Age
Variable name *: age
Variable type *: Number
Allowed values (optional)
Add custom property

Cancel Save

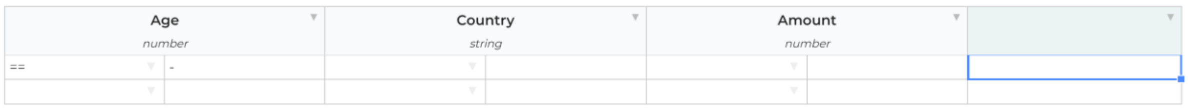
Next, select `Save` and then create the `country` input column by expanding the `Age` column and selecting `Add column`.

Specify the following information for the new `country` column:

- Column label: *Country*, Variable name: *country*, Variable type: *string*

Now create the `Amount` input column follow the same steps for adding the `Country` column.

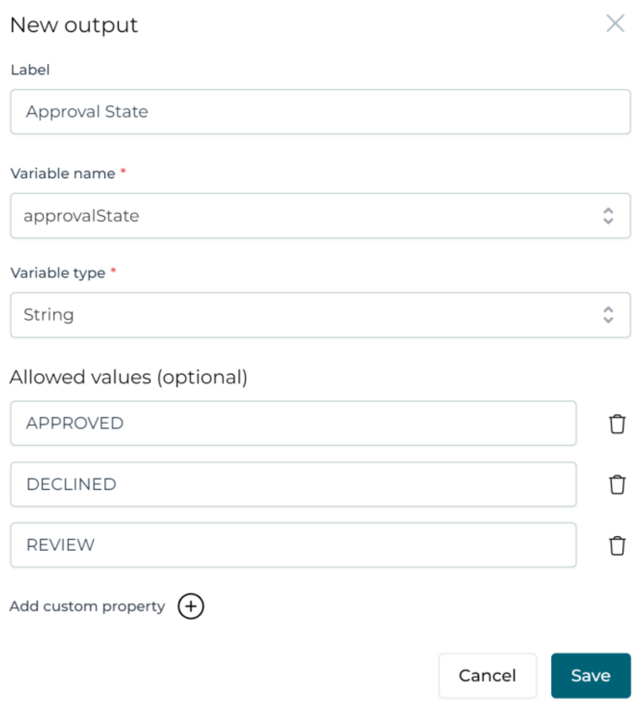
- Column label: *Amount*, Variable name: *amount*, Variable type: *number*



Age number		Country string		Amount number		
==	-					

Outputs

Now we define the output column. Expand the most right column header and select `Edit Column` and fill in the 'Column label' with *Approval State*, the 'Variable' with *approvalState*, and the 'Variable type' with *string*. For 'Output values', enter *APPROVED*, *DECLINED*, and *REVIEW* each into a row of their own.



New output

Label	Approval State
Variable name *	approvalState
Variable type *	String

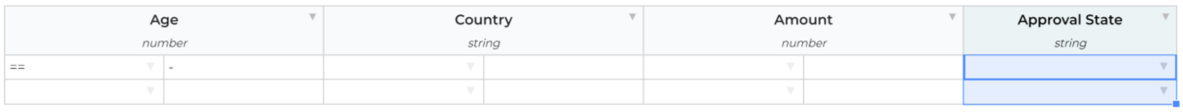
Allowed values (optional)

APPROVED

DECLINED

REVIEW

Again select `Save`. We now have the inputs and outputs defined:



Age number		Country string		Amount number		Approval State string
==	-					

And can proceed with constructing the rules.

Rules

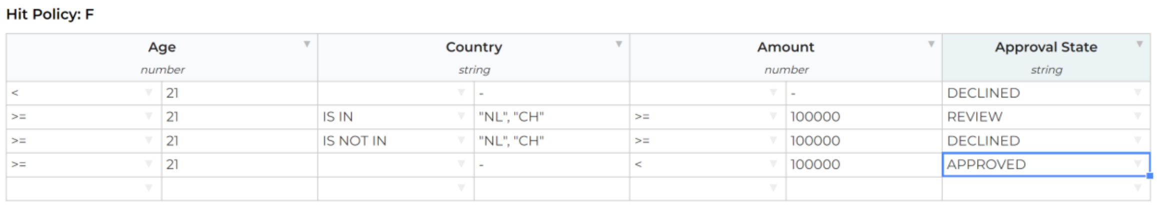
Each condition of a rule is constructed by specifying a value and an operator. The available operators depend on the variable type of the 'column'.

NOTE

There are two exceptions. One is when using the 'dash value'. The other is when using a custom Java Unified Expression Language (JUEL) expression. The dash operator ('-') means that the expression for that rule is ignored when evaluating the table. A custom JUEL expression is declared with surrounding '=' of '$' syntax.

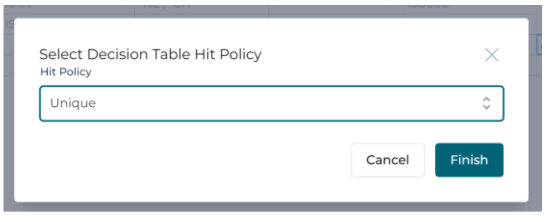
For this example, Create the following rules:

1. Age: < 21 / Country: - / Amount: - / Approval State: DECLINED
2. Age: >= 21 / Country: IS IN "NL", "CH" / Amount: >= 100000 / Approval State: REVIEW
3. Age: >= 21 / Country: IS NOT IN "NL", "CH" / Amount: >= 100000 / Approval State: DECLINED
4. Age: >= 21 / Country: - / Amount: < 100000 / Approval State: APPROVED


Hit Policy: F

Age (number)		Country (string)		Amount (number)		Approval State (string)
<	21		-		-	DECLINED
>=	21	IS IN	"NL", "CH"	>=	100000	REVIEW
>=	21	IS NOT IN	"NL", "CH"	>=	100000	DECLINED
>=	21		-	<	100000	APPROVED

Finally set the hit policy by selecting the hit policy indicator in the upper left-hand corner of the table. (By default there is an `F` for First.) Select `Unique` as the policy and then `Finish`.


Select Decision Table Hit Policy

Hit Policy: Unique

Finally, save the Decision Table and return to the process view.

Step 4: Create the Exclusive Gateway (and End Events)

The resulting output of the valid rule is set as the value of the defined output variable. In this case, there is a variable with the name `approvalState` with the value, 'APPROVED', 'DECLINED' or 'REVIEW' on the process execution scope. We can use that value to route our process to the next stage. Let us do that by creating an `exclusive gateway` and `conditional sequence flows`.

Step 4A: Exclusive Gateway

Select the `Exclusive gateway` from the `Gateways` section of the palette on the left and drag it to the canvas.

Connect the `Decision task` with the `Exclusive gateway` with a sequence flow (the arrow type line connector).

Step 4B: End Events

The result of the loan request is either approved or declined. We, therefore, want the process to have two different end states. We do that by defining two end events: `APPROVED` and `DECLINED`.

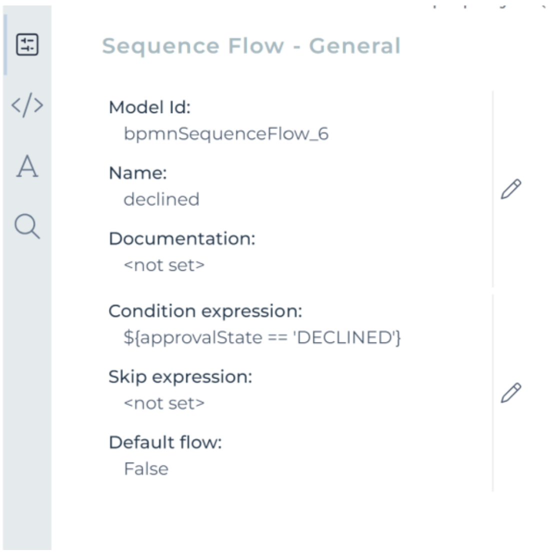
Select the `End event` in the palette and drag it to the canvas. Repeat the same drag action for another `End event`. Double click each `End event` and rename one to *APPROVED* and the other *DECLINED*.

Step 4C: Conditional Sequence Flows

To navigate the process to one of the two end events based on a value, we set a condition on the sequence flow.

Let us connect the 'Exclusive gateway' to both 'End events'. Name the one connecting the `APPROVED` end event *approved*, and the other one *declined*.

Now set the conditions by selecting the sequence flow and filling out the `Condition expression`. Set the field to `${approvalState == 'APPROVED'}` for the approved sequence flow and set `${approvalState == 'DECLINED'}` for the declined sequence flow.


Sequence Flow - General

Model Id:
bpmnSequenceFlow_6

Name:
declined

Documentation:
<not set>

Condition expression:
${approvalState == 'DECLINED'}

Skip expression:
<not set>

Default flow:
False

The process is defined to navigate to one of the two end events based on the value of the variable `approvalState`.

Step 5: Create the 'Review Loan Request' Step

In the previous step, we defined the routing of the process based on the decision table outcomes `APPROVED` and `DECLINED`. There was another outcome: `REVIEW`. In this step, we create a review task with which an appropriate user can manually approve or decline the request.

Step 5A: Create the 'Review Loan Request User Task'

Select the `User task` from the palette on the left and drag it to the canvas. Double click the `User task` and rename it to *Review Loan Request*.

Connect the `Exclusive gateway` with the newly created `User task` using a `Sequence flow` as we want to navigate to the `Review Loan Request User Task` in all cases other than `APPROVED` and `DECLINED` mark this sequence flow as the `Default flow`. The default is set by selecting the sequence flow and then `Default flow` attribute in the panel on the right and select the checkbox.

Step 5B: Create the 'Loan Request Intake Form'

To review the current loan request, we want to display the data from `Loan Request Intake Form`.

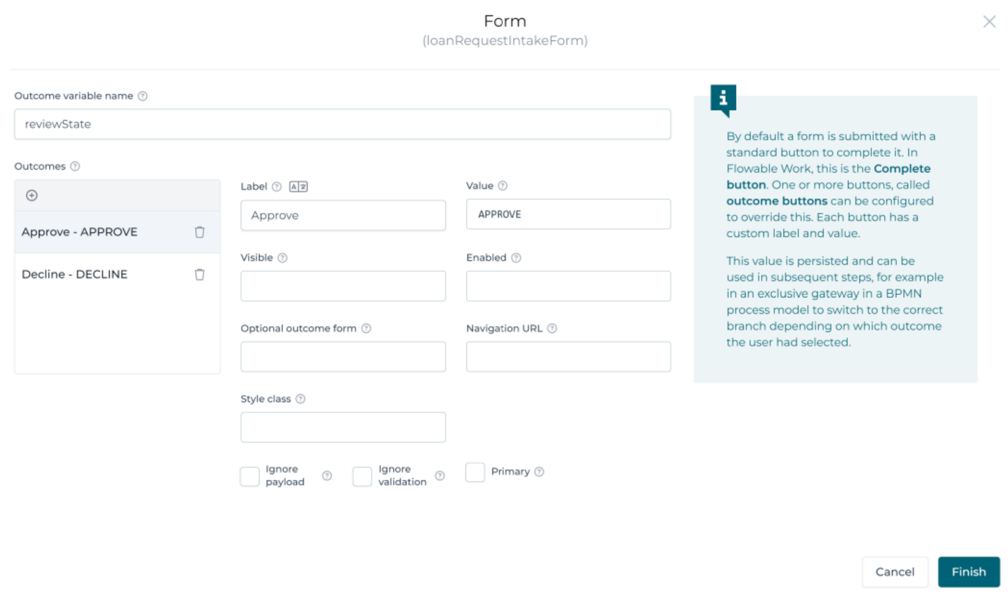
Let us create a form as done in Step 2 above. First, create the `Loan Request Intake Form` and in addition to the fields described in Step 2, add:

- type: 'Text', name: *Remarks*, id: *remarks*, value: `{{remarks}}`

For approving or declining the loan request, we configure two custom outcomes.

Select the form canvas; make sure you do not select a form field. Then from the `Form - General` panel in the `Other` section, select `Outcomes`, set *reviewState* as the variable name and enter the following values:

- Label: *Approve*, Value: *APPROVE*
- Label: *Decline*, Value: *DECLINE*


Form
(loanRequestIntakeForm)

Outcome variable name
reviewState

Outcomes

Label	Value
Approve	APPROVE

Approve - APPROVE
Decline - DECLINE

Visible
Enabled
Optional outcome form
Navigation URL
Style class
Ignore payload
Ignore validation
Primary

By default a form is submitted with a standard button to complete it. In Flowable Work, this is the **Complete button**. One or more buttons, called **outcome buttons** can be configured to override this. Each button has a custom label and value.

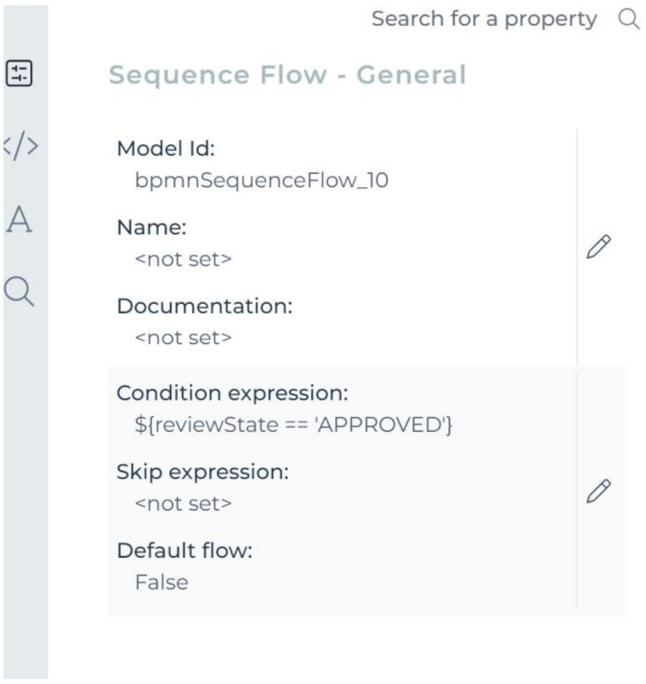
This value is persisted and can be used in subsequent steps, for example in an exclusive gateway in a BPMN process model to switch to the correct branch depending on which outcome the user had selected.

Next select `Finish` to save the information.

Step 5C: Exclusive Gateway

Add another `Exclusive gateway` and connect the `Review Loan Request` User task with a sequence flow. Connect the sequence flows from the `Exclusive gateway` to both the `APPROVED` and `DECLINED` End events.

Next set the condition expression: `${reviewState == 'APPROVED'}` for the sequence flow connecting to the `APPROVED'` End event. And enter the condition expression: `${reviewState == 'DECLINED'}` for the sequence flow connecting to the `DECLINED` End event.



Sequence Flow - General

Model Id:
bpmnSequenceFlow_10

Name:
<not set>

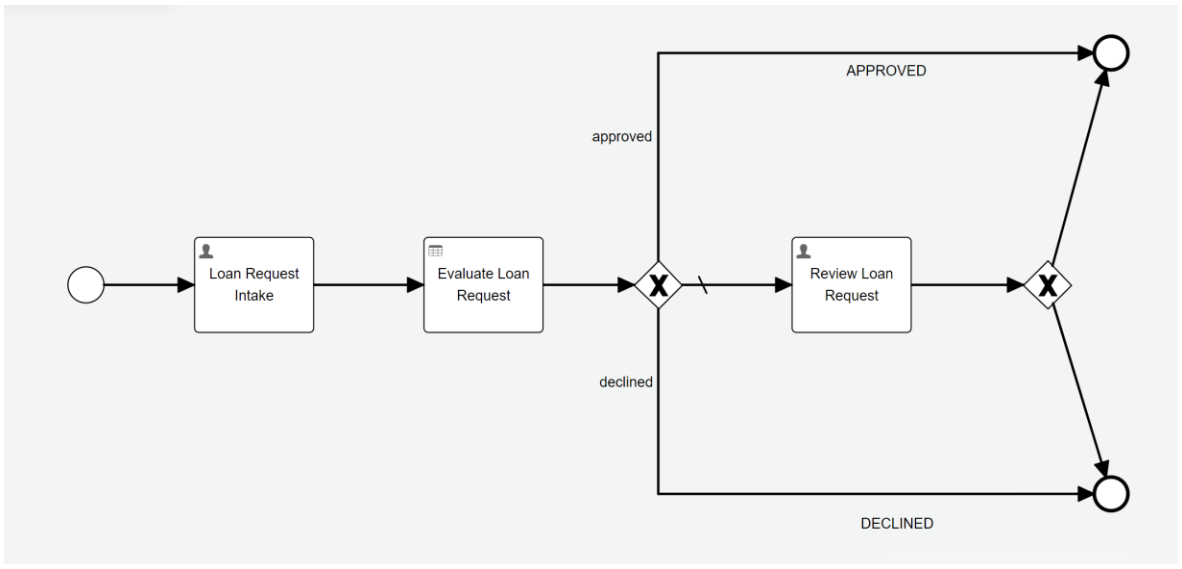
Documentation:
<not set>

Condition expression:
${reviewState == 'APPROVED'}

Skip expression:
<not set>

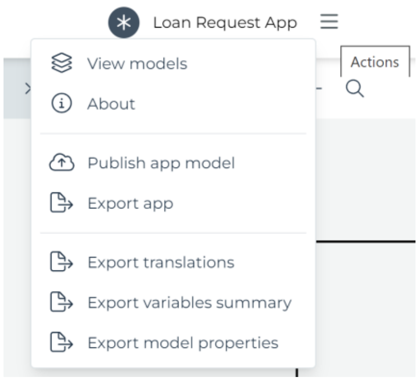
Default flow:
False

Save the process model, and you are done modeling the Loan Request Approval App.



Decision table: Deploy the Process

To execute the process in Flowable Work or Flowable Engage, we need to deploy the App



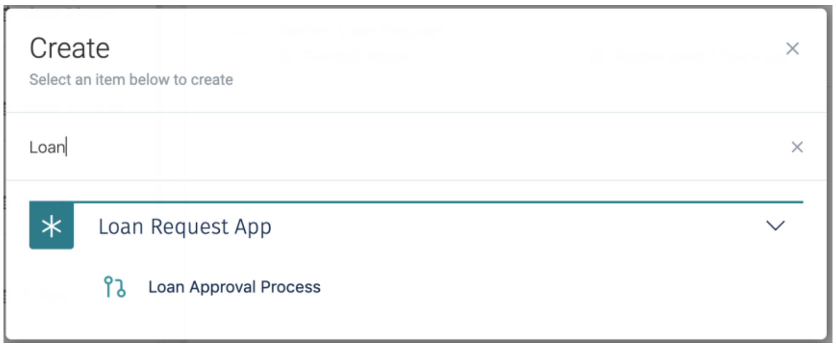
The App including the 'Loan Approval' process (and associated forms and the decision table) is now deployed.

Decision table: Execute the Process

Flowable Engage is used to demonstrate the process and two different scenarios are showing the various potential outcomes of the decision table.

Scenario 1

In Flowable Engage select `Work>Create new` in the left menu. Select the `Loan Request Approval Process` from the `Loan Request App`.



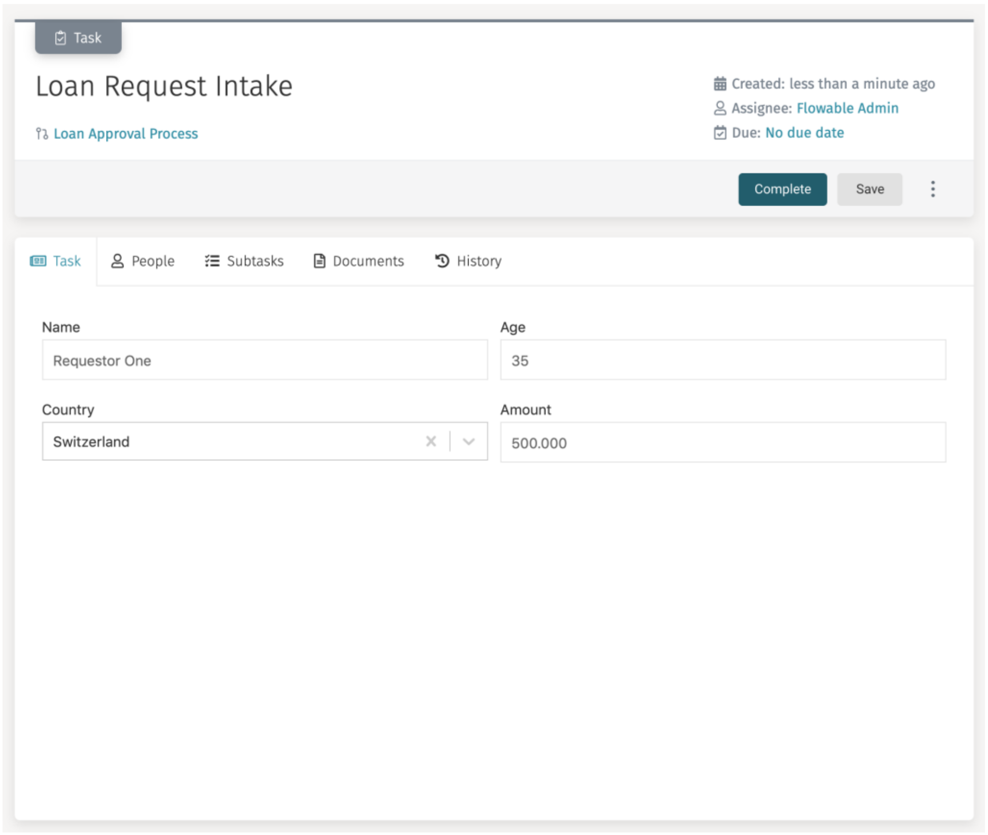
Select `Continue`.

A process instance is started and the process execution halts at the first step; the `Loan Request Intake` user task.

Select the task in the `Open tasks` list.

The form is presented. In this first scenario, enter the following data:

- name: *Requestor One*
- age: *35*
- country: *Switzerland*
- amount: *500.000*

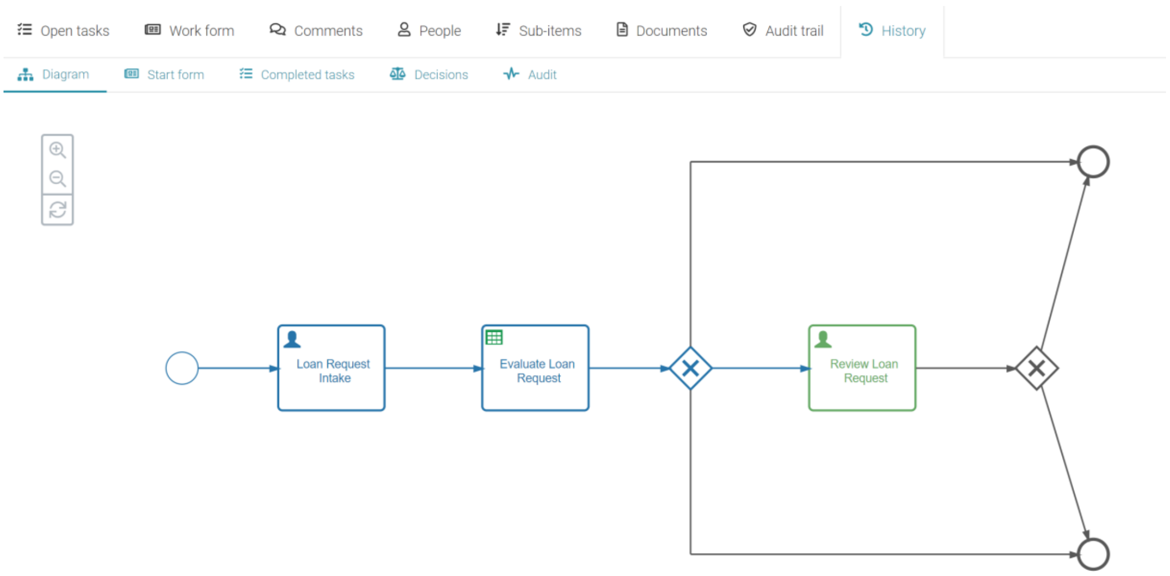


Name	Age
Requestor One	35
Country	Amount
Switzerland	500.000

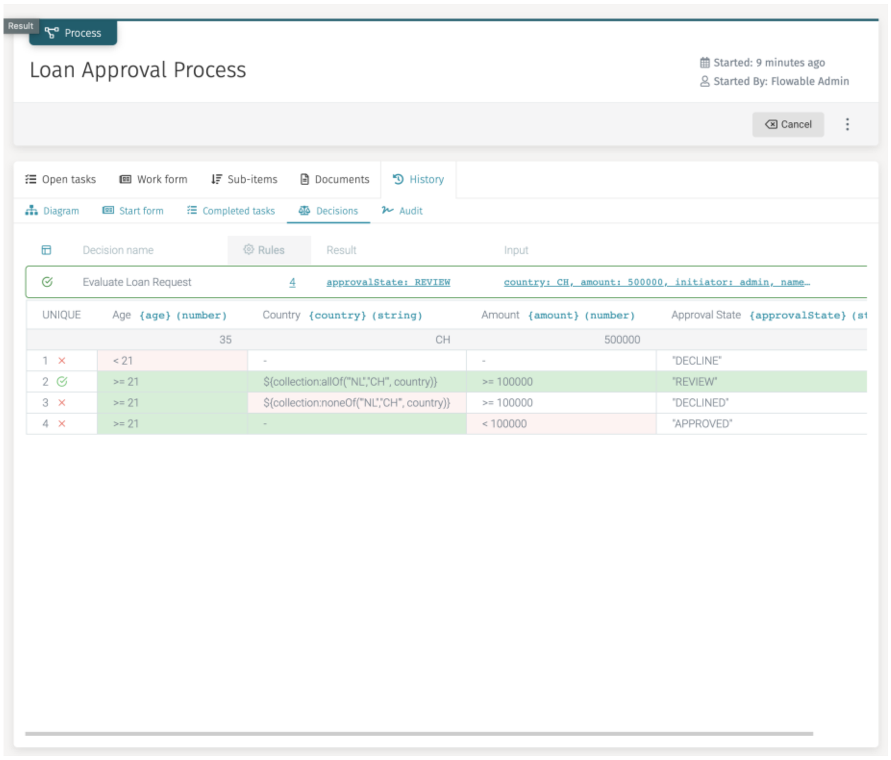
Next, select the *Complete* button.

There is now an open task named `Review Loan Request`. To find out why the process is in that state we look at the decision table audit.

Select `History` in the toolbar and see a process instance diagram depicting the current state of the process.



As you can see, the process navigated to the `Review Loan Request` user task. To find out what the result of the decision table was select `Decisions` in the submenu.



Loan Approval Process

Started: 9 minutes ago
Started By: Flowable Admin

	Decision name	Rules	Result	Input
✓	Evaluate Loan Request	4	`approvalState: REVIEW`	`country: CH, amount: 500000, initiator: admin, name…`

UNIQUE	Age `{age} (number)`	Country `{country} (string)`	Amount `{amount} (number)`	Approval State `{approvalState} (st`
	35	CH	500000	
1 ✗	< 21	-	-	"DECLINE"
2 ✓	>= 21	${collection:allOf("NL","CH", country)}	>= 100000	"REVIEW"
3 ✗	>= 21	${collection:noneOf("NL","CH", country)}	>= 100000	"DECLINED"
4 ✗	>= 21	-	< 100000	"APPROVED"

Here you can see that rule two was valid, and therefore, `REVIEW` is the outcome of the decision.

Now select `Open tasks` again and select the `Review Loan Task`. After reviewing the data, enter an optional remark and select `Approve`.

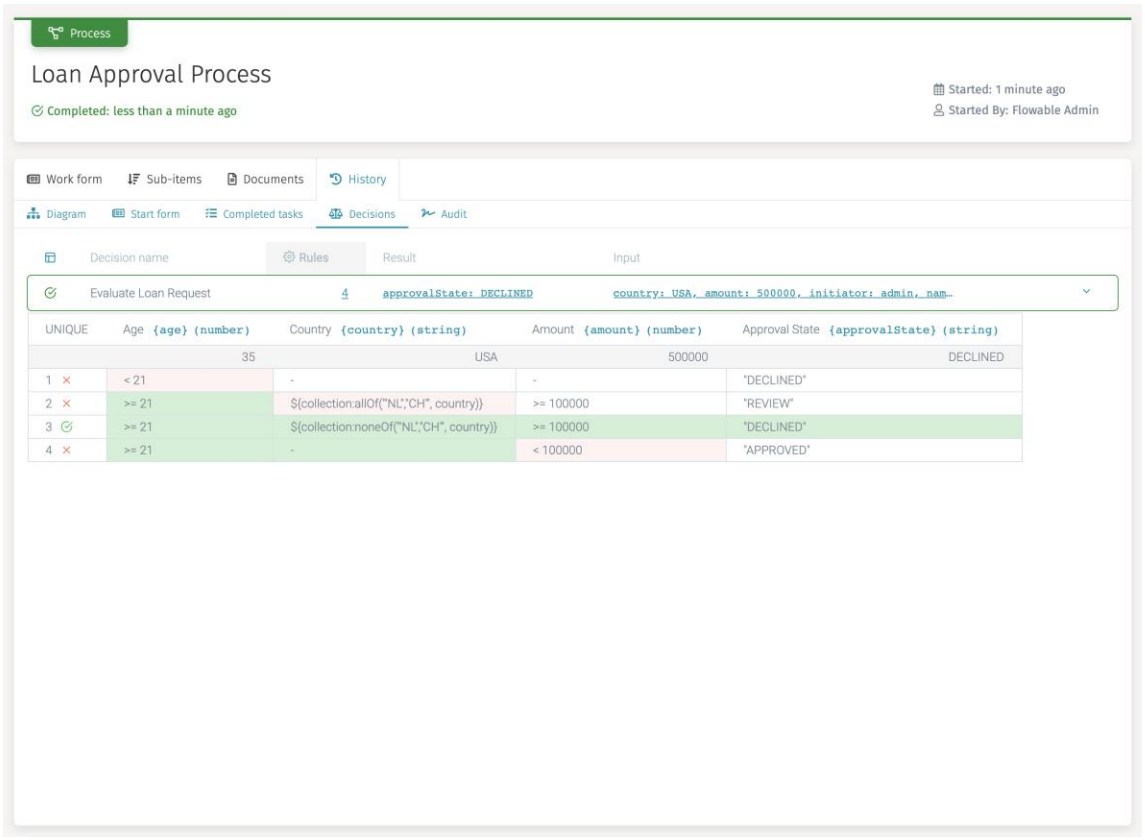
The process is now completed and the request is approved.

Scenario 2

Let us execute another Loan Approval process instance and this time provide the following data in the `Loan Request Intake` user task;

- name: *Requestor Two*
- age: *35*
- country: *United States*
- amount: *500.000*

This time the decision table evaluation results in `DECLINED`, and the process ends.



Loan Approval Process

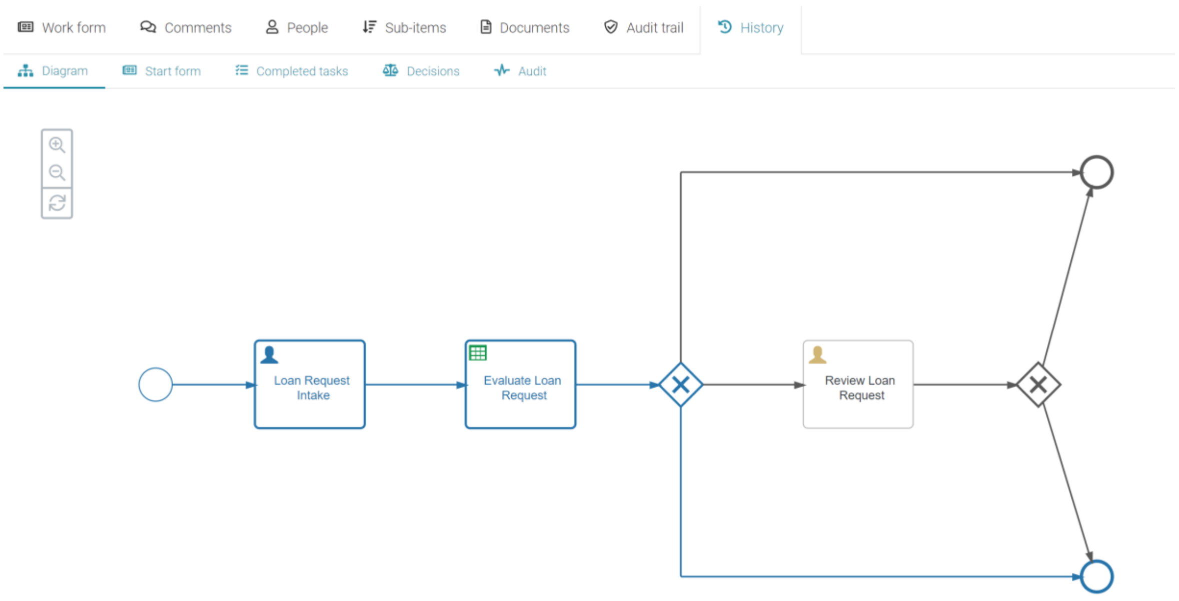
Completed: less than a minute ago

Started: 1 minute ago

Started By: Flowable Admin

	Decision name	Rules	Result	Input
✓	Evaluate Loan Request	4	approvalState: DECLINED	country: USA, amount: 500000, initiator: admin, nam...

UNIQUE	Age {age} (number)	Country {country} (string)	Amount {amount} (number)	Approval State {approvalState} (string)
	35	USA	500000	DECLINED
1 ✗	< 21	-	-	"DECLINED"
2 ✗	>= 21	${collection:allOf("NL","CH", country)}	>= 100000	"REVIEW"
3 ✓	>= 21	${collection:noneOf("NL","CH", country)}	>= 100000	"DECLINED"
4 ✗	>= 21	-	< 100000	"APPROVED"


Work form
Comments
People
Sub-items
Documents
Audit trail
History
Diagram
Start form
Completed tasks
Decisions
Audit
Loan Request Intake
Evaluate Loan Request
Review Loan Request