My First Flowable App

CMMN

What is CMMN?

What is CMMN? What is the purpose of CMMN? What are examples of case models? Let's get started and look at CMMN from a high-level point of view first.

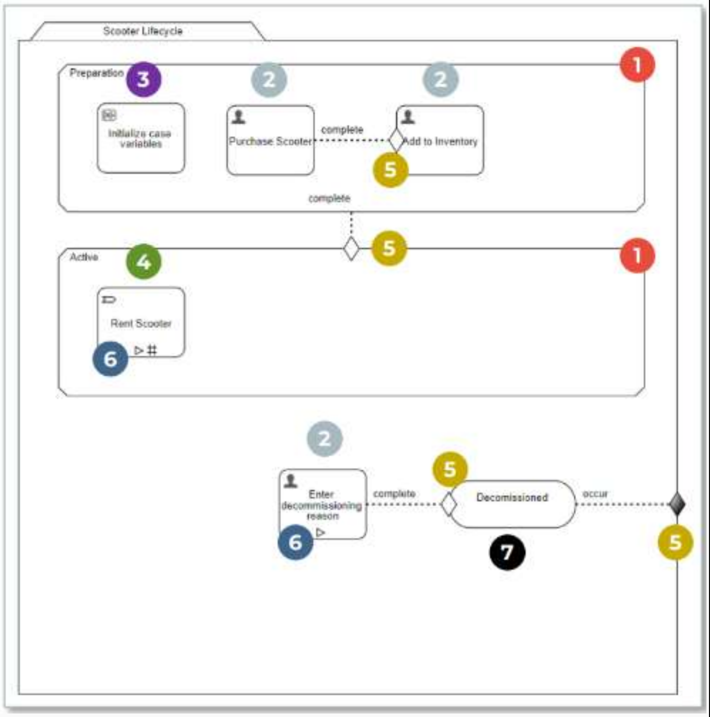
Case Management (CMMN) Explained

Case Management Model and Notation (CMMN) is a standard used within Flowable to handle cases with less structure compared to business processes (BPMN). While BPMN defines a straightforward flow, CMMN allows for greater flexibility, enabling users to decide the steps to be taken within the application's scope.

Key Features of Case Management:

1. **Flexibility**: Unlike rigid business processes, case management empowers users to choose the appropriate steps to take. Various options and possibilities can be presented to users, giving them the freedom to decide the best course of action for different scenarios.
2. **Collaboration**: Communication and clarification of information are crucial in case management. Instead of breaking the context by sending messages through external means like email or instant messaging, CMMN encourages communication within the case management tool itself. This ensures that all relevant parties are aware of discussions and decisions.
3. **Ad Hoc Tasks**: Case management allows for ad hoc tasks that may not have been pre-planned. In certain situations, a human decision is required to determine the appropriate action, especially when dealing with unstructured information or circumstances. Examples of this include handling customer complaints or onboarding new employees.
4. **Life Cycles**: Case management facilitates handling tasks with repetition, allowing different stages within the life cycle of a case. For instance, a customer may go through the onboarding stage, then become an active customer, and later transition to an inactive status. Different actions can be defined for each stage, making it easy to manage cases with changing requirements.

Elements of a Case in CMMN:



1. **Stage**: A container that groups multiple elements together. It represents a phase or milestone within the case.
2. **Human Task**: Requires user interaction. Actual users need to perform certain actions or decisions within these tasks.
3. **Service Task**: More technical in nature, where a computer or system performs specific computations or interacts with other systems.
4. **Process Task**: Used for communication with a business process. It allows reaching out to another process and executing structured steps one after the other.
5. **Entries and Exit Sentries**: Control the flow of the case. Entries decide when a task becomes available, while exit sentries signify when a task is no longer necessary and can be concluded.
6. **Decorators**: Specify the execution behavior, such as repeating a task multiple times or performing it only after certain user interactions.
7. **Milestones**: Mark significant accomplishments or completions of tasks within the case.

Starting Cases in CMMN: Cases in CMMN can be started in various ways:

- Manual initiation through the user interface.
- Programmatically starting cases from code.

- External events triggering the start of a case.
- Another process or case can also initiate a new case.

Case Elements: All the components discussed above, including stages, tasks, sentries, decorators, and milestones, are collectively referred to as "plan items" within a CMMN case.

Flow Control in CMMN: Unlike BPMN, CMMN does not have a direct flow. The case's flow is managed through sentries, which control when tasks can be performed or completed.

The First Task

In this module, we will explore the fundamental workings of Case Management Model and Notation (CMMN). We'll start by understanding the nature of a Case Plan Model, which serves as the central structure for case management scenarios. Throughout this course, we use the example of a "Scooter Lifecycle" modeled in CMMN.

The **Case Plan Model** is the heart of case management, representing the main item or case being worked on. It visually depicts the case's elements and their relationships. It can be named. A name such as "Scooter Lifecycle" would be appropriate in our use case.

In CMMN, each individual CMMN element directly placed inside a Case Plan Model will start simultaneously. There are many different CMMN elements, for the time being, we will concentrate on "**Human Tasks**". Each Human Task, when started, will create a Task in Flowable Work which can then be completed by a user. Optionally, we can attach a form to a Human Task, allowing our users to enter data that can be used later.

For instance, if we add two tasks, "Purchase Scooter" and "Add to Inventory," into the Case Plan Model, **they will both initiate when the case begins**. However, users have the freedom to complete "Purchase Scooter" before "Add to Inventory" or vice versa, as there is no strict order imposed.

When all tasks within the Case Plan Model have been **completed,** the entire case is marked as completed. This indicates that all the necessary actions or decisions have been executed, and the case is considered complete.

Sentries - Part 1

How can we control the flow in a case diagram? Sentries are the way to say what is coming next and give everything an order. Let's see how they look like and how they behave.

In this module, we'll explore how multiple elements can be connected using Sentries in Case Management Model and Notation (CMMN). Sentries play a crucial role in creating a sequence of different tasks within a case.

The Concept of Sentries:

Imagine a scenario where we have two tasks, "Purchase Scooter" and "Add to Inventory." Logically, these tasks must follow each other since we cannot add a scooter to the inventory before we've purchased it. To enforce this order, we use Sentries, which allow us to control the activation of another element based on a specified condition.

Representation of Sentries: Sentries are represented as a dotted line with a diamond shape. In reality, they consist of two elements: the "On-Part" (connector) and the "If-Part" (condition). The "On-Part" determines the event we are listening to, often "On Complete." However, more advanced use cases may involve other events. The "If-Part" enables us to control whether the activation occurs once the event from the "On-Part" is fired. If desired, it can contain a condition.

Simple Example: Let's revisit our previous example of the "Scooter Lifecycle." We want to connect "Add to Inventory" to "Purchase Scooter" to ensure the correct sequence. By doing so, we create a dependency between the two tasks. This means that in Flowable Work, once we start our case, we will only be presented with one task ("Purchase Scooter"). Once we have completed this task, the CMMN engine will understand that the next task ("Add to Inventory") needs to be created. The case will be completed when both tasks are completed.

Placing a Sentry: To add a sentry, click on the element you wish to connect with another. In our case, we want "Add to Inventory" to depend on "Purchase Scooter." Click on "Purchase Scooter" and then drag the small white diamond that appears on the top-right corner of the task to the edge of the task you want to connect it to. Once the task's edge lights up in green, drop the sentry to create the connection. The resulting sentry will simply establish a dependency between the two tasks without any conditions or special events.

Repetition Decorator

The true value of CMMN comes when you start to do things multiple times or optional. Now, first things first. In this module we are going to look at how you can repeat multiple things and do them more than once.

In this module, we will explore the concept of repetition in CMMN.

By default, all elements or tasks within a case can only be **executed once**. However, we can change this behavior by adding the "Repetition" decorator to an element.

Repetition Decorator: The "Repetition" decorator is a powerful feature that allows tasks to be executed multiple times. Let's consider our "Scooter Lifecycle" scenario as an example. We have two tasks, "Purchase Scooter" and "Add to Inventory." Both just have to be carried out once since we cannot purchase a scooter multiple times. Now, we want to introduce a third task, "Rent Scooter," which we want to be able to perform as long as the scooter still has its wheels attached.

How to make a Task Repeatable: To add a repeatable Human Task, follow these steps:

1. Add the "Rent Scooter" task to the Case Plan Model.
2. Mark the "Rent Scooter" task as repeatable by applying the "Repetition" decorator in the Attribute panel. A hash/number (#) symbol will appear under the name of the task.

Effect of Repetition: When we now start the case in Flowable Work, two tasks will be created immediately ("Purchase Scooter" and "Rent Scooter"). We retain the flexibility to execute all tasks in any order we prefer. However, once we complete the "Rent Scooter" task, it will automatically be recreated, leading to an infinite loop of repetitions. As a result, the case can never be closed, as there will always be open tasks.

Control and Customization: It is essential to note that, like most features in Flowable, repetition can be controlled and customized to meet specific requirements. For instance, we can add conditions to determine when a task is repeatable or limit the number of times it can be repeated.

Your first Case

1. Create a Human Task

The first step is to create a Human Task and add it to the Case Plan Model. Press save to update the validation list on the right.

1. Look out for the 'Human Task' component on the palette (the panel on the left hand side).
2. If you prefer, you can turn on the 'List View' by clicking on the 'List' symbol at the bottom left.
3. Drag and drop the Human Task into the Case Plan Model.
4. Save the model to validate the result.

2. Name the task 'Purchase Scooter'

Please rename the first task to 'Purchase Scooter', so that the user knows what to do.

1. Double-click on the task name.
2. Change the name of the task to 'Purchase Scooter'.
3. Save the model to validate the result.

3. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside. (left top corner)
2. Press this button and a popup window will appear.
3. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.
4. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

4. Start the case

In Flowable Work, try to start a new case and complete the 'Purchase Scooter' task. Click on 'New' and select 'Work'

1. On the left hand menu, click on 'New'
2. Select 'Work' since you would like to start a new work item, which can be a case or a process.
3. Press 'Continue' to confirm that you would like to start the case.
4. You will now see a new case with the open task 'Purchase Scooter'
5. Click on the 'Purchase Scooter' task to open the task.
6. Press the 'Complete' button in the upper right corner to finish the task.
7. You will be automatically redirected back to Flowable Work upon completion.

5. Create a second Human Task

Next, we need to add a second Human Task.

1. Look for the 'Human Task' (person symbol) component in the palette on the left side.
2. Drag and drop the Human Task into the Case Plan Model.
3. Save the model to validate the result.

6. Name the second task 'Add to Inventory'

Please rename the second task to 'Add to Inventory' so that the user knows what needs to be done.

1. Double-click on the task name.
2. Change the name of the task to 'Add to Inventory'.
3. Save the model to validate the result.

7. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside. (left top corner)
2. Press this button and a popup window will appear.

3. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.
4. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

8. Start the case

In Flowable Work, execute everything what you have done so far.

1. On the left hand menu, click on 'New'
2. Select 'Work' since you would like to start a new work item, which can be a case or a process.
3. Press continue on the screen to confirm that you would like to start the case.
4. You see now a new case with the open task 'Purchase Scooter' and 'Add to Inventory'
5. Click on the 'Purchase Scooter' task to open the task.
6. Press the 'Complete' button in the upper right corner to finish the task.
7. Click on the 'Add to Inventory' task to open the task.
8. Press the 'Complete' button in the upper right corner to finish the task.
9. You will be automatically redirected back to Flowable Work upon completion.

9. Create a Sentry between the tasks

Add a Sentry to ensure that the 'Add to Inventory' task is opened only after the completion of the 'Purchase Scooter' task.

1. Click on the 'Purchase Scooter' task to see the quick draw menu.
2. Grab the 'empty' diamond (top one) from the right menu and drag it.
3. Place the cursor on the edge of the 'Add to Inventory' task and release it when the task's border lights up in green.
4. Let the diamond go and it will draw the Sentry.
5. Save the model to validate the result.

10. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside. (left top corner)
2. Press this button and a popup window will appear.
3. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.

11. Start the case

In Flowable Work, execute everything what you have done so far.

1. On the left hand menu, click on 'New'
2. Select 'Work' since you would like to start a new work item, which can be a case or a process.
3. Press continue on the screen to confirm that you would like to start the case.
4. You see now a new case with the open task 'Purchase Scooter'
5. Click on the 'Purchase Scooter' task to open the task.
6. Press the 'Complete' button in the upper right corner to finish the task.
7. You are back on the task list and see now the 'Add to Inventory' task.
8. Click on the 'Add to Inventory' task to open the task.
9. Press the 'Complete' button in the upper right corner to finish the task.
10. You will be automatically redirected back to Flowable Work upon completion.

12. Create a third Human Task

Let's now add one more Human Task to actually rent the scooter.

1. Look for the 'Human Task' (person symbol) component in the palette on the left side.
2. Drag and drop the Human Task into the Case Plan Model.
3. Save the model to validate the result.

13. Name the last task 'Rent Scooter'

Rename the final task to 'Rent Scooter' to clearly communicate the required action to the user.

1. Double-click on the task name.
2. Change the name of the task to 'Rent Scooter'.
3. Save the model to validate the result.

14. Make rental repeatable

To ensure that we can rent the scooter more than once, we would like to make the rental repeatable.

1. Select the 'Rent Scooter' task in the canvas.
2. Search the properties on the right side for 'Repetition' (it's below the # menu, but you can also use the search).
3. Check the 'Repetition' property to make the selected item repeatable.
4. Save the model to validate the result.

15. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the top menu bar which above the canvas.
2. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside.

3. Press this button and a popup window will appear.
4. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.
5. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

16. Start the case
In Flowable Work, execute everything what you have done so far.

1. On the menu on the left side, click on 'New'
2. Select 'Work' since you would like to start a new work item, which can be a case or a process.
3. Press continue on the screen to confirm that you would like to start the case.
4. You see now a new case with the open tasks 'Purchase Scooter' and 'Rent Scooter'
5. Click on the 'Purchase Scooter' task to open the task.
6. Press the 'Complete' button in the upper right corner to finish the task.
7. You are back on the task list and see now the 'Rent Scooter' and 'Add to Inventory' tasks.
8. Click on the 'Add to Inventory' task to open the task.
9. Press the 'Complete' button in the upper right corner to finish the task.
10. You are back on the task list and see now the 'Rent Scooter' task.
11. Click on the 'Rent Scooter' task to open the task.
12. Press the 'Complete' button in the upper right corner to finish the task.
13. You are back on the task list and see now the 'Rent Scooter' task.
14. Click on the 'Rent Scooter' task to open the task.
15. Press the 'Complete' button in the upper right corner to finish the task.
16. You will be automatically redirected back to Flowable Work upon completion.

Stages

Do you need a bit more structure? Let's introduce stages which allow us to bundle elements into multiple sections. With that it's clearer when we can do something and what the current status is.

In this module, we will delve into the concept of "stages". Stages are pivotal elements that provide a structured way to organize and group tasks within a case.

Organizing Tasks with Stages: In our "Scooter Lifecycle" case, we now have three tasks: "Purchase Scooter," "Add to Inventory," and "Rent Scooter." As the case becomes more complex, it's essential to maintain clarity and organization. Stages help achieve this by allowing us to group related tasks together.

Unlike a Human Task, there is no way to just "complete" a task with the press of a button. A stage acts almost like a "Mini Case" in a way: By default, all elements inside a stage need to be completed/terminated before the stage is considered complete. This then triggers the "complete" event. As always in the world of CMMN, there are ways to customize this behavior.

Benefits of Using Stages:

1. **Structuring the Case:** Stages provide a clear and logical structure to the case, ensuring tasks are organized in a meaningful manner.
2. **Enhanced Understanding:** By grouping tasks into stages, it becomes easier for stakeholders to understand the case's progression and the relationships between different tasks. Remember that one of the advantages of using CMMN and BPMN is that you can actually show them to anyone who's interested!
3. **Flexibility in Execution:** Stages allow us to determine the order in which tasks can be executed, adding an additional layer of control to the case. For instance, if you have two distinct stages "Preparation" and "Active", you can make sure that all tasks in the "Preparation" stage must be completed before moving on to the "Active" stage.

Implementing Stages:

To implement stages in our "Scooter Lifecycle" case:

1. Identify tasks that naturally belong together. In our case "Purchase Scooter" and "Add to Inventory" are good candidates.
2. Create a stage and add these related tasks to it. Simply drag a "Stage" element from the palette into your Case Plan Model. You can name stages either by double-clicking and typing a name in the canvas directly or through the attribute panel.
3. While it is possible to have multiple stages running at the same time, in our case, each stage is representing a distinct "phase" of our case. Thus, it makes sense to connect the two stages through a sentry.

Visual Representation: Stages are visually represented as containers that hold the associated tasks. When viewing the Case Plan Model, you will see how tasks are grouped within their respective stages.

Stages are Plan Items: Stages are the second type of *Plan Item* you learned about, the first being Tasks. In many ways, they act very similar to tasks:

- They both have states, e.g. "active", "available", or "completed
- They can be connected with Sentries. You can even connect a task and a stage that way!
- They can be configured to be repeatable, manually activated (more on that in the next lesson) etc.

Manual Activation

Optionality is really important to give the user the flexibility to decide what they would like to do next. Especially when you are in the context of a case in which you would like to have that flexibility. Let's see how we can do that!

In this module, we will explore the concept of "Manual Activation" in CMMN.

Default Automatic Activation: In the default behavior of a case, all elements, such as tasks, are automatically activated as soon as the case starts. For example, when we initiate the "Scooter Lifecycle" case, tasks like "Purchase Scooter" and "Rent Scooter" are automatically activated and ready for execution.

In the last lesson, we learned how to use stages to deter the execution of the "Rent Scooter" task. However, a rental process will still be started as soon as the "Active" stage is triggered. This does not really make sense, does it? We want to be in control on when a new scooter is rented out.

Introducing Manual Activation: Now, let's make the "Rent Scooter" task manually activatable in our "Scooter Lifecycle" case. We can do this through the "Manual Activation" attribute that is available on all Plan Items. By doing this, we make the task optional and give users control over when to activate it. Obviously, we want our scooters to be rented out, so it's not really "optional" from the business perspective.

When the stage "Active" is started, "Rent Scooter" will now no longer be automatically initiated like the other tasks. Instead, we can activate it manually by clicking the "Rent Scooter" button in Flowable Work which now appears in the menu bar when we open our case.

Benefits of Manual Activation:

1. **Selective Task Execution:** With Manual Activation, we can rent one scooter at a time by activating the "Rent Scooter" task when needed. While the task is open, we cannot rent another scooter, ensuring that each rental is managed independently.
2. **Flexibility and User Control:** Manual Activation provides greater flexibility, allowing users to decide when to start specific tasks based on real-time needs and priorities. In combination with the Repetition Decorator, we can achieve some really interesting results.

Every Plan Item Can Be Manually Activatable: It's important to note that the concept of Manual Activation is not limited to tasks alone. Every plan item, including milestones and stages can be made manually activatable. This means users have the autonomy to trigger various elements as per the context of the case.

Manual Activation really is what sets CMMN apart from more rigid or "flow-oriented" notations.

Sentries - Part 2

In this module, we will dive deeper into the world of sentries.

Problem with Repeatable Tasks: With the introduction of the repeatable "Rent Scooter" task, we encountered a challenge in closing our case. As the task can always be activated, the case remains perpetually open, preventing us from closing the case. But what if there is a new generation of scooters or our scooter is so badly damaged that we need to replace it?

Introducing "Enter Decommissioning Reason" Task: To resolve this issue, we add a new manually activatable task called "Enter Decommissioning Reason." Unlike the repeatable "Rent Scooter" task, this task is not repeatable, ensuring that it is a one-time occurrence.

The purpose of the "Enter Decommissioning Reason" task is to collect a reason for decommissioning the scooter, such as loss or damage. But how can we make this task complete the case?

Entry Sentries vs. Exit Sentries: As we have learned, Entry Sentries are used to control the activation of tasks or elements in a case based on specific conditions or events. They determine when an element should become active. On the other hand, Exit Sentries are employed to manage the case's closure by triggering specific actions when certain conditions are met.

Exit Sentries look like filled Entry Sentries but act in the opposite way: Instead of activating a Plan Item, they terminate them. Termination is almost the same as completion, both will "end" an element.

If you place an Exit Sentry on a task, the task will be terminated, meaning the task will be marked as "Complete". An Exit Sentry on the case will complete the case. This is what we will do next.

Enhanced Case Closure: By connecting the "Enter Decomissioning Reason" task and the Case Plan Model with an Exit Sentry, we can now close the case even if there are other open tasks. The "Enter Decommissioning Reason" task serves as the final step, allowing us to provide a reason for case closure, thus fulfilling the case's lifecycle. Only when that task is completed, will the case be completed.

Adding Conditions on Sentries: It's worth noting that we can add conditions on both Entry and Exit Sentries, providing us with further control over when these sentries are executed. Conditions are based on "Expressions" which usually evaluate the value of variables in our case. More on that later!

Optional "On-Part" for Conditions: Another thing to note is that the "On-Part" (dashed line) in sentries is optional. This is especially useful when working with conditions. With conditions in place, they will continually evaluate whenever there are changes in the case, such as task completions or variable updates. This flexibility allows us to create dynamic behavior.

Important Tasks

In this chapter, we will explore the diverse task types available in Case Management Model and Notation (CMMN) within Flowable. While we have mainly focused on user-driven Human Tasks so far, we will now uncover essential tasks that add complexity and functionality to our case management processes.

Human Tasks and Data Input: In real-life applications, Human Tasks are usually connected to forms. These forms enable users to input data into the case using "variables." These variables capture critical information that drives decision-making and process automation within the case management process, making it more sophisticated and data-driven.

Expanding Task Types for Greater Impact: To create truly dynamic and impactful case management scenarios, we need to incorporate various task types. Let's explore some of the most important tasks available in CMMN within Flowable:

1. **Email Task:** The Email Task allows us to send out emails to one or more recipients, complete with text and attachments. This task is essential for communicating critical information and updates during the case lifecycle. For instance, we could imagine a Email Task to be used to inform the staff if the scooter hasn't been returned after 30 days.
2. **Decision Task:** Decision Tasks execute Decision Tables modeled in the Decision Model and Notation in Flowable. Decision Tables enable us to define complex rules based on input, resulting in outputs that are stored as variables within our case. These rules-driven tasks add intelligence to case management processes. A possible example could be the decision whether or not a specific scooter type requires the customer to upload their driver's license.
3. **Process Task:** The Process Task lets us initiate a well-structured BPMN Process, which is suitable for straightforward logic and well-defined workflows, such as a "Rental Process." We will delve deeper into BPMN notation in subsequent chapters.
4. **Service Task:** Service Tasks act as a simple way to call Java Logic in Flowable Work, optionally storing results as variables. They are useful for making calculations, performing actions, and invoking other operations. More advanced Java Logic calls are also possible. A good example for our use case would be a Service Task that actually remotely unlocks our Scooter in the real world.
5. **Case Task:** The Case Task initiates a new CMMN Case, allowing us to create nested case management structures and handle complex scenarios within a broader context. An example could be the creation of a "Retrofitting Case" when a new model of a scooter comes out.

Dynamic Cases

1. Create a Stage

Let's start structuring the case in stages. Create a new stage that is outside of the current tasks.

1. Look for the 'Stage' component in the palette on the left side. This is the rectangle with cut corners on the top right
2. Drag and drop the Stage into the empty space in the Case Plan Model.
3. Save the model to validate the result.

2. Name the Stage 'Preparation'

Please rename the Stage to 'Preparation'.

1. Double-click on the Stage.
2. Change the name of the Stage to 'Preparation'.
3. Save the model to validate the result.

3. Move the tasks into the Stage

Expected relation Add to Inventory to be inside Preparation but did not found

Move the 'Purchase Scooter' and 'Add to Inventory' tasks into the 'Preparation' Stage.

1. Click on the 'Purchase Scooter' task.
2. Press and hold the shift key and click on 'Add to Inventory' task to select the second task as well.
3. Drag and drop the selections into the new Stage.

4. Create a second Stage and name it 'Active'

In the next step we need to do the same again, only this time for the 'Rent Scooter' task. Create a new Stage and name it 'Active'.

1. Look for the 'Stage' component in the palette on the left side.
2. Drag and drop the Stage into the empty space in the Case Plan Model.
3. Double-click on the Stage.
4. Change the name of the Stage to 'Active'.
5. Save the model to validate the result.

5. Move the tasks into the Stage

Expected relation Rent Scooter to be inside Active but did not found

Now you need to move the 'Rent Scooter' tasks into the 'Active' Stage.

1. Click on the 'Rent Scooter' task.
2. Drag and drop the 'Rent Scooter' task into the new Stage.

6. Create a Sentry between the Stages

Add a Sentry so that the 'Rent Scooter' task is only active after the 'Preparation' Stage has finished.

1. Click on the 'Preparation' Stage to see the quick draw menu.
2. Grab the 'empty' diamond (top one) from the right menu and drag it onto the 'Active' Stage. You can see a green border around the Stage when you are in the right position.
3. Let the diamond go and it will draw the Sentry.
4. Save the model to validate the result.

7. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the top menu bar which above the canvas.
2. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside.
3. Press this button and a popup window will appear.
4. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.
5. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

8. Start the case

In Flowable Work, execute everything what you have done so far. You need to rent the scooter at least twice, otherwise it won't count!

1. On the left hand menu, click on 'New'
2. Select 'Work' since you would like to start a new work item, which can be a case or a process.
3. Press continue on the screen to confirm that you would like to start the case.
4. You see now a new case with the open task 'Purchase Scooter'
5. Click on the 'Purchase Scooter' task to open the task.
6. Press the 'Complete' button in the upper right corner to finish the task.
7. You are back on the task list and see now the 'Add to Inventory' task.
8. Click on the 'Add to Inventory' task to open the task.
9. Press the 'Complete' button in the upper right corner to finish the task.
10. You are back on the task list and see now the 'Rent Scooter' task.
11. Click on the 'Rent Scooter' task to open the task.
12. Press the 'Complete' button in the upper right corner to finish the task.

13. You are back on the task list and see now the 'Rent Scooter' task.
14. Click on the 'Rent Scooter' task to open the task.
15. Press the 'Complete' button in the upper right corner to finish the task.
16. You will be automatically redirected back to Flowable Work upon completion.

9. Make rental manually activated

Let's make sure that we are in control when a new rental is started. We want to mark the 'Rent Scooter' task with the 'Manual Activation' decorator.

1. Select the 'Rent Scooter' task in the canvas.
2. Search the properties on the right side for 'Manual Activation'. It's located in the 'Control' attribute group which is denoted by a 'Play' symbol.
3. Check 'Manual Activation' for this task.
4. Save the model to validate the result.

10. Create a Human Task to dispose of the scooter

The last step is going to be a Human Task that will close the case. This task should be named 'Enter decommissioning reason'.

1. Look for the 'Human Task' (person symbol) component in the palette on the left side.
2. Drag and drop the Human Task into the Case Plan Model.
3. Double-click on the title and change it to 'Enter decommissioning reason'.
4. Save the model to validate the result.

11. Make enter decommissioning manually activated

This task should be optional. We want to have the enter decommissioning task is marked with the 'Manual Activation' decorator.

1. Select the 'Enter decommissioning reason' task in the canvas.
2. Search the properties on the right side for 'Manual Activation'. It's located in the 'Control' attribute group which is denoted by a 'Play' symbol.
3. Check 'Manual Activation' for this task.
4. Save the model to validate the result.

12. Create an Exit Sentry to end the case once the 'Enter decommissioning reason' is done

Add a Exit Sentry so that the 'Enter decommissioning reason' task is ending the case once done.

1. Click on the 'Enter decommissioning reason' task to see the quick draw menu.
2. Grab the filled diamond, the second one from top to bottom and drag it.

3. Place the cursor on the edge of the 'Case Plan Model' and release it when the border of the case lights up in green.
4. Let the diamond go and it will draw the Sentry.
5. Save the model to validate the result.

13. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the top menu bar which above the canvas.
2. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside.
3. Press this button and a popup window will appear.
4. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.
5. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

14. Start the case

In Flowable Work, execute everything what you have done so far. You need to rent the scooter at least twice, otherwise it won't count! At the end, decomission the scooter.

1. On the left hand menu, click on 'New'
2. Select 'Work' since you would like to start a new work item, which can be a case or a process.
3. Press continue on the screen to confirm that you would like to start the case.
4. You see now a new case with the open task 'Purchase Scooter'
5. Click on the 'Purchase Scooter' task to open the task.
6. Press the 'Complete' button in the upper right corner to finish the task.
7. You are back on the task list and see now the 'Add to Inventory' task.
8. Click on the 'Add to Inventory' task to open the task.
9. Press the 'Complete' button in the upper right corner to finish the task.
10. You are back on the task list and see no new task.
11. Press the 'Rent Scooter' button at the top right.
12. You should now see the 'Rent Scooter' task.
13. Click on the 'Rent Scooter' task to open the task.
14. Press the 'Complete' button in the upper right corner to finish the task.
15. You are back on the task list and see no new task.
16. Press the 'Rent Scooter' button at the top right.
17. You should now see the 'Rent Scooter' task.
18. Click on the 'Rent Scooter' task to open the task.
19. Press the 'Complete' button in the upper right corner to finish the task.

20. The task list again empty, but this time you need to press the 'Enter decommissioning reason' button.
21. Click on the 'Enter decommissioning reason' to open the task.
22. Press the 'Complete' button in the upper right corner to finish the task.
23. You will be automatically redirected back to Flowable Work upon completion.

Forms and Pages

Using Forms

Let's look at Flowable Forms and how we can use them.

In our journey, we've encountered a key concept known as "variables." Simply put, variables are like labeled containers where we store specific pieces of information that are important for our processes.

Understanding Variables:

In the real world, we like to categorize things. We intuitively know whether a scooter is red, green, or black. Computers, however, need a bit more guidance. Variables provide that guidance and help us define and organize data for our processes.

How Variables Work: Let's consider the example of "Scooter Type." In our scooter rental case, the "Scooter Type" variable could hold information about the different scooter models available, such as "AX-123" or "RUN-Y-RUN." Just like you would label different boxes to keep things organized, we use variables to keep essential data organized and accessible during our processes.

Each variable consists of the following parts:

1. **Name:** The name of the variable, like "scooterType." The name follows the "camel case" notation, where spaces are removed, and the following letter is capitalized.
2. **Type:** The type of the variable, such as "String" (text), "Boolean" (true/false values), "Integer" (whole numbers), "Date," or "DateTime."
3. **Scope:** Variables are scoped to a specific "thing," often called an "Entity." For our scooter rental case, the variables would be associated with a single Scooter Case. Variables can also be scoped to processes, tasks, and even apps.

Understanding Case Data and Its Importance:

When we talk about "Case Data," we refer to the information we need to collect and work with during a case or process. It encompasses all variables stored on a case. For instance, besides "Scooter Type," we may also want to store data about the scooter's color, customer information, or rental duration. This data is crucial because it allows us to make informed decisions, generate reports, and carry out specific actions within the process.

Introduction to Flowable Forms: Flowable Forms provide a practical way to capture and store data. Each form consists of different components, representing specific fields like the "Scooter Type" mentioned earlier. Components can take various forms, such as text fields, checkboxes, and dropdown menus, making it easy for users to input the necessary data.

Usage of Forms in Flowable Work: Forms are widely used throughout Flowable Work to efficiently collect essential inputs. Whether it's a task that needs completion, the start of a new case or process, or an ongoing workflow, forms allow users to capture and manage data at the right stage.

Start and Work Forms:

Start Forms are used to gather data before a case or process begins, ensuring all necessary information is available from the very start. For instance, we can collect the type of scooter right from the beginning, making it convenient for users to enter it on the Start Form.

On the other hand, Work Forms act as overviews during the ongoing case or process, allowing users to monitor and access critical data as needed.

Adding Forms and Leveraging Referencing: To include Start Forms or Work Forms, you can easily create a new form or refer to an existing one. Referencing forms promotes reusability, saving time and effort across different processes and tasks. A new form can be added by clicking on the Case Plan Model and searching for the "Start Form" or "Work Form" attribute.

Structuring Forms

Let's deep dive into the structure of a form and the components which we have available to further understand how we can leverage it. Those details will help to create the first form.

In this module, we will have a closer look at the structure of a Flowable Form.

Components in a Form: Each Flowable Form is composed of components that capture and display data. We have the flexibility to add as many components as needed to fulfill our data requirements. Each component represents a single field, such as "Scooter Type," "Maximum Speed," or "Color." Depending on the type of data we want to collect, we can choose from various component types, such as text fields, checkboxes, or date pickers.

Types of Components:

1. **Data Entry Components:** These components, like Text, Number, Decimal, Date, and Attachments, enable users to input specific data directly into the form.
2. **Selection Components:** Radio Buttons, Checkboxes, and Selection (Dropdowns) allow users to choose data from a pre-defined list of values, providing a selection-based data input method.
3. **Display Components:** Components like Text Display, Image, Data Table, and HTML Display are used to showcase information to users, offering a read-only view of data.

4. **Container Components:** Containers, such as Panel, Modal, Subform, and Tabs, aid in organizing and structuring the form's layout, making it more user-friendly.
5. **Flowable Work Components:** These are specialized components like Task List, Chart, or Action Buttons, designed specifically for Flowable Work features.

Form Layout: Rows and Columns with a 12-Grid System:

Flowable Forms are structured in rows and columns, following a responsive 12-grid layout system. This grid system is widely used in modern design because 12 is divisible by many numbers (2, 3, 4, and 6), making it highly versatile for organizing and aligning components. The 12-grid layout provides a consistent structure that adapts well to different screen sizes and devices, ensuring a seamless user experience across various platforms.

Grouping Components with Containers: In addition to the grid layout, we can use certain containers to group related components together, enhancing form organization and visual clarity. Panels, Tabs or Subforms act as containers, allowing us to organize data entry fields or display components in a logical and cohesive manner.

For example, if our form contains multiple sections, such as personal information and rental details, we can place each section within a separate tab. This way, users can easily distinguish between different sections and focus on the specific information they need to interact with.

Customizing Form Outcomes: In addition to defining data input components, we can specify outcomes for the form. By default, forms usually have pre-defined outcomes based on the context, such as "Start" or "Complete." However, we have the power to tailor the user experience by setting our own outcomes.

Practical Example: For instance, imagine a form that requires approval for a scooter rental. Instead of using the standard "Complete" outcome, we could customize it with two outcomes: "Accept" and "Reject." By doing so, the user completing the form can explicitly choose between these options, making the approval process more intuitive and transparent.

Create a Form

1. Create a Start Form with the key 'FLW-F001'

Create a form and reference it as the Start Form of your case. Simply click on the Case Plan Model and select Start Form. The form must have the key 'FLW-F001'. Don't forget to save your case model.

1. Click on the outermost Stage which is called Case Plan Model.
2. In the property list, search for 'Start Form'.
3. Enter 'FLW-F001 Scooter Start Form' as name.

4. Enter as key 'FLW-F001' as key.
5. Press the 'Create' button to create the form.
6. Press 'Finish' afterwards to close the dialog.
7. Save the model to validate the result.

2. Create a Single Select

The first step is to add a Single Select to your form. Press save to update the validation list on the right.

1. Look for the 'Select (Single)' component in the palette on the left side.
2. Drag and drop the 'Select (Single)' component into the form model.
3. Save the model to validate the result.

3. Name the Single Select 'Scooter Type'

One of the following labels is missing: Scooter Type

Please rename the first field to 'Scooter Type', so that the user knows what to do.

1. Double-click on the field.
2. Change the name of the field to 'Scooter Type'.
3. Save the model to validate the result.

4. Provide the list of items

As a next step, we need to add the items to the select. Therefore you can search for 'Items' and add 'AX-123', 'SUPER-FLY X1' and 'RUN-Y-RUN'. The text and the value should be the same.

1. Select the 'Scooter Type' field.
2. Search the properties on the right side for 'Items'.
3. Press the 'Add item' button.
4. Write 'AX-123' in the fields 'Value' and 'Text'.
5. Press 'Add item' again and repeat for 'SUPER-FLY X1'.
6. Press 'Add item' again and repeat for 'RUN-Y-RUN'.
7. Press the 'OK' button.
8. Save the model to validate the result.

5. Make the field 'Scooter Type' mandatory

The last step for the scooter type field is to make it mandatory.

1. Select the 'Scooter Type' field.
2. Search for the 'Required' attribute located in the 'Validation' section on the right side.
3. Save the model to validate the result.

6. Create a number field

Next we would like to have a number field for the maximum speed.

1. Look for the 'Number' component in the palette on the left side.
2. Drag and drop the number into the form model.
3. Save the model to validate the result.

7. Name the field 'Maximum Speed (km/h)'

One of the following labels is missing: Maximum Speed (km/h)

Please rename the number field to 'Maximum Speed (km/h)', that the user knows what to enter.

1. Double-click on the field.
2. Change the name of the field to 'Maximum Speed (km/h)'.
3. Save the model to validate the result.

8. Set the value to '{{maxSpeed}}'

We want to manually change the label to '{{maxSpeed}}'. Form values are typically in two curly braces and written in camelCase.

1. Select the 'Maximum Speed (km/h)' field.
2. Search the properties on the right side for 'Value'.
3. Enter '{{maxSpeed}}'.
4. Save the model to validate the result.
5. Consider activating the 'Show grid' icon at the top to help you with the alignment.

9. Resize 'Maximum Speed (km/h)' to only takes the left third of the canvas

Let's make the 'Maximum Speed (km/h)' field smaller so that it only takes the left third of the canvas. With that, we can place the next fields next to it.

1. Select the 'Maximum Speed (km/h)' field.
2. Ensure that it is all the way to the left and directly below the Scooter Type field in the second row.
3. Grab the three dots on the right of the field and move them towards the left.
4. Let it go as soon as the component fills four columns used.
5. You can activate a grid view to help you with the alignment by clicking on the 'Show grid' icon at the top.
6. Save the model to validate the result.

10. Make maximum speed mandatory

Again, also the maximum speed should be mandatory. Search for 'Required' below validations on the right side.

1. Select the 'Maximum Speed (km/h)' field.
2. Search for the 'Required' attribute in the validation category.
3. Save the model to validate the result.

11. Create a second number field

Next we would like to have a number field for the maximum distance.

1. Look for the 'Number' component in the palette on the left side.
2. Drag and drop the number into the form model.
3. Save the model to validate the result.

12. Name the field 'Maximum Distance (km)'

One of the following labels is missing: Maximum Distance (km)

Please rename the number field to 'Maximum Distance (km)', that the user knows what to enter.

1. Double-click on the field.
2. Change the name of the field to 'Maximum Distance (km)'.
3. Save the model to validate the result.

13. Set the value to '{{maximumDistance}}'

We want to manually change the label to '{{maximumDistance}}'. Form values are typically in two curly braces and written in camelCase.

1. Select the 'Maximum Distance (km)' field.
2. Search the properties on the right side for 'Value'.
3. Enter '{{maximumDistance}}'.
4. Save the model to validate the result.

14. Resize 'Maximum Distance (km)' field and place it next to the 'Maximum Speed (km/h)' field

Let's make the 'Maximum Distance (km)' field smaller so that it only takes the middle third of the canvas. Place it right of the 'Maximum Speed (km/h)' field.

1. Select the 'Maximum Distance (km)' field.
2. Move it directly next to the 'Maximum Speed (km/h)' field.
3. Let it go as soon as the component fills four columns used.
4. You can activate a grid view to help you with the alignment by clicking on the 'Show grid' icon at the top.
5. Save the model to validate the result.

15. Create a decimal field

Next we would like to have a decimal field for the charging time.

1. Look for the 'Decimal' component in the palette on the left side.
2. Drag and drop the number into the form model.
3. Save the model to validate the result.

16. Name the field 'Charging Time'

One of the following labels is missing: Charging Time

Please rename the first field to 'Charging Time', that the user knows how to use the field.

1. Double-click on the field.
2. Change the name of the field to 'Charging Time'.
3. Save the model to validate the result.

17. Resize 'Charging Time' field and place it next to the 'Maximum Distance (km)' field

Let's make the 'Charging Time' field smaller so that it only takes the right third of the canvas. Place it right of the 'Maximum Distance' field.

1. Select the 'Charging Time' field.
2. Move it directly next to the 'Maximum Distance (km)' field.
3. Save the model to validate the result.

18. Create another single select

The next step is to create a second single select and add it to the form model. Press save to update the validation list on the right.

1. Look for the 'Select (Single)' component in the palette on the left side.

2. Drag and drop the Select (Single) into the form model.
3. Save the model to validate the result.

19. Name the field 'Color'

One of the following labels is missing: Color

Please rename the first field to 'Color', that the user knows what to do.

1. Double-click on the field.
2. Change the name of the field to 'Color'.
3. Save the model to validate the result.

20. Provide the list of items

As a next step, we need to add the items to the select. Search for 'Items' and add the following items: 'Black', 'Blue', 'Green', and 'Red'. The value should be the same as the text in lowercase.

1. Select the 'Color' field.
2. Search the properties on the right side for 'Items'.
3. Press the 'Add item' button.
4. Write 'black' in the field 'Value' and 'Black' in the field 'Text'.
5. Press 'Add item' again and repeat for 'blue' / 'Blue.
6. Press 'Add item' again and repeat for 'green' / 'Green'.
7. Press 'Add item' again and repeat for 'red' / 'Red'.
8. Press the 'OK' button.
9. Save the model to validate the result.

21. Create a checkbox group component

The last element to add is a checkbox group. Search the checkbox group and place it on the canvas.

1. Look for the 'Checkbox group' component in the palette on the left side.
2. Drag and drop the Checkbox group into the form model.
3. Save the model to validate the result.

22. Name the field 'Scooter Features'

One of the following labels is missing: Scooter Features

Please rename the checkbox group field to 'Scooter Features', that the user knows what they are about.

1. Double-click on the field.
2. Change the name of the field to 'Scooter Features'.
3. Save the model to validate the result.

23. Provide the list of items

As a next step, we need to add the items to the select. Search for 'Items' and add 'Foldable', 'Stand', 'Suspension', and 'Display'. The text should be the name as provided and the value the same in lowercase.

1. Select the 'Scooter Features' field.
2. Search the properties on the right side for 'Items'.
3. Press the 'Add item' button.
4. Write 'Foldable' in the field 'Value' and 'foldable' in the field 'Text'.
5. Press 'Add item' again and repeat for 'Stand' / 'stand.
6. Press 'Add item' again and repeat for 'Suspension' / 'suspension'.
7. Press 'Add item' again and repeat for 'Display' / 'display'.
8. Press the 'OK' button.
9. Save the model to validate the result.

BPMN

What is BPMN?

BPMN (Business Process Model and Notation) and CMMN (Case Management Model and Notation) are two widely used modeling notations in the Flowable platform, each serving different purposes. While both BPMN and CMMN are used to model business processes, they have distinct characteristics and are suited for different scenarios.

BPMN - Linear Structure: BPMN is a standardized notation for modeling business processes in a visual way. It is widely used for modeling structured, predefined processes that follow a linear flow of actions. In BPMN, processes are represented as a series of interconnected activities and events that define the flow of work from start to finish. It allows you to represent complex processes with various tasks, gateways, and decisions in a clear and understandable manner.

Advantages of BPMN:

1. **Linear Flow**: BPMN is well-suited for processes where a straight line of actions is desired, and the order of execution is known in advance. It is particularly effective for well-defined, repeatable processes that have a predictable sequence of steps.
2. **Visual Clarity**: The graphical representation of BPMN diagrams makes it easy to understand the process flow, even for non-technical stakeholders. This helps in effective communication and collaboration among teams.

3. **Standardization**: BPMN is an industry-standard notation, which means that models created using BPMN are easily understandable and transferable across different organizations and BPMN-compliant tools.

BPMN Examples:

1. **Well-Structured Approval Process:** BPMN is well-suited for modeling processes with a predefined sequence of actions, making it ideal for approval workflows. For instance, you can model an approval process where certain decisions lead to different outcomes. If an item is approved, the process proceeds as planned. However, if it's not approved, it may go back for reevaluation or follow a different path. While this process could be modeled in CMMN, BPMN's linear structure provides better clarity and manageability for such scenarios.
2. **Document Generation Process:** Suppose you need to generate a document based on specific inputs collected through a form. This process can be efficiently modeled using BPMN, as it involves a clear sequence of tasks leading to the final document generation. The predefined steps ensure a structured and efficient workflow, making BPMN a suitable choice.
3. **Service Orchestration and Integration:** BPMN is particularly useful in Straight Through Processing scenarios, where data needs to be processed efficiently. It facilitates the integration of services and systems, allowing you to model the flow of data through various tasks and service calls. BPMN's visual nature, ease of understanding, and ability to handle technical tasks with ease make it advantageous in service-oriented scenarios.

Using BPMN for Sequential Steps: In our use case, the "Rent Scooter" activity would indeed be a better fit for a BPMN process instead of a human task if you anticipate multiple sequential steps (e.g., asking for the driver's license, unlocking the scooter, handling repairs, etc.). With BPMN, you can define these steps as individual activities, decision points, and gateways, providing a structured and linear representation of the process flow. This approach ensures that each step is completed before moving to the next, offering better control and predictability.

In conclusion, while BPMN is ideal for linear processes with well-defined sequences, CMMN shines in complex scenarios where flexibility and adaptability are required. Selecting the appropriate notation depends on the nature of the process and the desired level of control and collaboration during its execution.

Key Elements in BPMN:

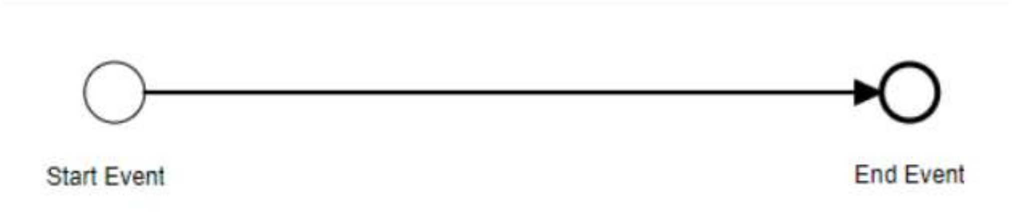
1. **Start Events:** Represents the initial point of a process. It triggers the start of the process flow. Examples include "None Start Event" and "Timer Start Event."
2. **Sequence Flows:** Represents the directional flow of work from one activity to another. It defines the order in which activities should be executed.
3. **User Task:** Represents a task performed by a user or a group of users. It involves human interaction and can be assigned to specific users or groups. In Flowable, it works exactly the same way as a Human Task in CMMN.

4. **Gateways:** Represents decision points in the process where the flow diverges or converges. There are various types of gateways, such as "Exclusive Gateway" (XOR) for making exclusive decisions and "Parallel Gateway" (AND) for parallel processing.
5. **Service Tasks:** Represents a technical task where a machine or automated system performs a specific action. It allows for the integration of external services, such as calling APIs, executing business rules, etc. Service Tasks are executed by the Flowable BPMN and CMMN Engine and are the most basic way to call the backend.
6. **End Events:** Represents the final point of a process. It marks the completion of the process flow and can have different outcomes, depending on the scenario.

Simple Process

How does a really simple process look like? Let's look at the first model and see how that could work with Flowable.

In this chapter, we will explore the fundamental concepts of BPMN through a visual demonstration. Watch the video above to understand how the Start Event, Sequence Flow, and End Event work together in the simplest process possible:



1. **Start Event:** The Start Event marks the initiation point of a process. When a process instance is started, it begins its execution at the Start Event. In the simplest process, the Start Event acts as the entry point, and as soon as a new process instance is created, it immediately moves to the next step. Clicking on the User Task will reveal some interesting options. For instance, we could add a Start Form to the event to spice things up.
2. **Sequence Flow:** A Sequence Flow represents the path of execution from one element to another in a process. In our simple process, the Sequence Flow connects the Start Event to the End Event, creating a direct path. This signifies that there are no intermediate steps or tasks, and the process will complete instantly.
3. **End Event:** The End Event defines the termination point of a process. When the process flow reaches the End Event, the process instance is considered complete, and its execution is finished.

Explanation:

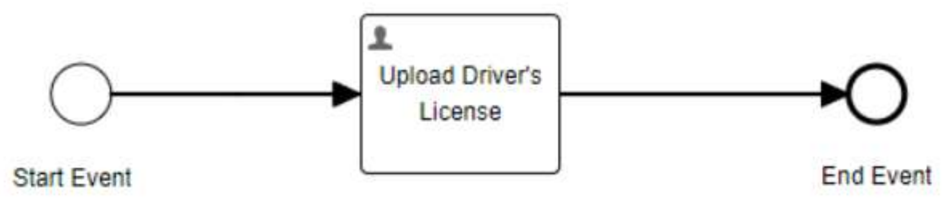
In the simplest process, we have only a Start Event and an End Event. This minimal process structure allows us to start a new process instance in Flowable Work. Unlike a case instance, which requires further actions or tasks, the process instance is started and immediately ends because there are no additional tasks to be performed.

When you start a new process instance in Flowable Work, it follows the Sequence Flow directly from the Start Event to the End Event, completing the process instantly.

In the next lesson, we will delve deeper into BPMN and introduce User Tasks to add logic and more complexity to our processes.

User Task

In this lesson, we will enhance our simple process by introducing a User Task called "Upload Driver's License." This addition will create a clear and straightforward flow from the start to the task and finally to the end of the process:



1. **Start Event:** We still begin with the Start Event, which marks the initiation point of our process.
2. **Sequence Flow:** The Sequence Flow connects the Start Event to the User Task "Upload Driver's License." This means that when a new process instance is started, it will move from the Start Event to the User Task.
3. **User Task "Upload Driver's License":** The User Task "Upload Driver's License" represents a task that requires human interaction.
4. **Sequence Flow (cont.):** After the User Task is completed, the Sequence Flow connects the User Task to the End Event. This signifies that once the user completes the "Upload Driver's License" task, the process will move to the End Event and be considered complete.
5. **End Event:** Finally, the process will be ended. We will no longer be able to make any changes to this process.

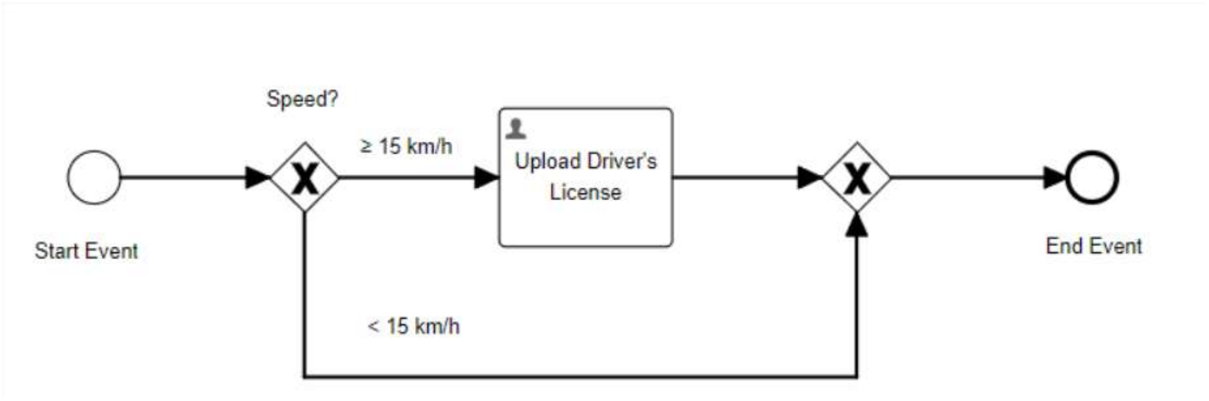
User Task Customization: By default, the User Task "Upload Driver's License" is assigned to the same process who initiated the process. However, it's important to note that this assignment can be customized based on specific requirements. For instance, you can assign the task to a specific user or group, depending on your business needs.

Furthermore, the task that will be created will not allow us to actually enter any data. To do that, we would need to attach a task form. That way the assignee of the task can actually upload an attachment, we would need to add a form to this task.

Gateways

To add more flexibility to your process and split the sequence flow into multiple different paths, you can use gateways. Let's look at how we can use them and the different kind of gateways which are available.

In this lesson, we will explore gateways in BPMN and understand their role in routing process flows based on certain conditions. As part of our scooter rental use case, we will implement an exclusive gateway to decide whether the user needs to upload the driver's license based on the scooter's speed.



Understanding Gateways: Gateways in BPMN are decision points that control the flow of a process based on specific conditions. They allow you to define branching paths, making it possible to create more complex and flexible process flows.

Scenario: Deciding on Driver's License Upload In our scooter rental use case, we want to route the process flow in a way that the user is only required to upload the driver's license if the scooter's speed exceeds 15 km/h.

Adding Exclusive Gateway: To implement this decision point, we add an exclusive gateway to our process model. The exclusive gateway is denoted by a diamond shape.

Adding Conditions with the "Condition Builder": Next, we use the "Condition Builder" to define the condition for each outgoing sequence flow from the gateway. The "Condition Builder" allows us to create expressions that evaluate to "true" or "false". For example, the condition expression could look like this: ${vars:greaterThanOrEquals(vehicleSpeed, 15)}. Here, "vehicleSpeed" is a variable that should be provided, e.g. through a Start or Task Form.

Evaluating Conditions: When the process execution reaches the exclusive gateway, it will evaluate each condition one by one, starting from the top. If one of the conditions evaluates to true, the process will take the corresponding route.

Important Considerations:

- Ensure Distinct Conditions: It is essential to make sure that all conditions in an exclusive gateway are distinct and mutually exclusive. Each condition should evaluate to either true or false, and only one path should be taken, otherwise, you will end up with unexpected situations and your may get stuck.
- Name your gateways and sequence flows properly: To enhance the clarity of the process model, it is recommended to name the gateways and sequence flows. For instance, the exclusive gateway could be named "Vehicle Speed?" and have two outgoing flows named "≥ 15 km/h" and "15 km/h".

Types of Gateways:

In Flowable, you will mainly interact with the following types of gateways:

1. **Exclusive Gateway (XOR):** This gateway allows one and only one path to be taken based on the conditions. It also has a default flow that will be taken if none of the conditions evaluate to true.
2. **Parallel Gateway (AND):** Unlike the exclusive gateway, the parallel gateway allows all paths to be taken simultaneously, and conditions are ignored.
3. **Inclusive Gateway (OR):** The inclusive gateway requires at least one outgoing sequence flow, and it will take more if no condition is specified or if the conditions on multiple flows evaluate to true. You can look at it as a hybrid between an exclusiv and parallel gateway.

Create a Process

1. Link the form to the Start Event

To gather some data we now would like to link the already existing Start Form to the process. With that the same data is acquired than we already have for the case.

1. Select the Start Event.
2. Look out on the properties on the right side for the 'Start Form reference'
3. Click on 'link to an existing one' to select a reference.
4. Select the 'FLW-F001 Scooter Start Form' form.

2. Create an Exclusive Gateway

The next step is to create an Exclusive Gateway.

1. Look for the 'Exclusive Gateway' component in the palette on the left side.
2. Drag and drop Exclusive Gateway to the process.
3. Save the model to validate the result.

3. Add a Sequence Flow

Draw a Sequence Flow between Start Event and the Exclusive Gateway to make the Gateway follow the Start Event.

1. Click on the start none event.

2. Locate the quick-draw items on the right side of the symbol. In case they are not there, click outside and inside again.
3. Drag the arrow, which is the bottom rigth quick draw item.
4. Let it go once you hover the Gateway.

4. Set the label for the Gateway to 'Vehicle Speed?'

Double-click the Exclusive Gateway and type the text 'Vehicle Speed?'

1. Select the Exclusive Gateway.
2. Search the property 'Name' on the right hand side.
3. Enter the text 'Vehicle Speed?'.

5. Create a User Task

The next step is to create a User Task with a Sequence Flow after the Gateway.

1. Click on the Exclusive Gateway.
2. Locate the quick draw items on the right side directly next to the Exclusive Gateway.
3. Click on the user symbol to add a User Task.
4. Save the model to validate the result.

6. Name the Sequence Flow '>= 15 km/h'

The User Task should only be available when the speed of the scooter is at least 15 km/h an hour. Therefore, we first add a descriptive label.

1. Click on the Sequence Flow.
2. Locate in the properties on the right side the option for the 'Name'.
3. Change the 'Name' to '>= 15 km/h'.
4. Save the model to validate the result.

7. Add a condition to verify the maximum speed is at least 15 km/h

Select the Sequence Flow and search on the right hand side for the 'Condition expression'. Click on 'Condition expression' and then click on the the 'Condition Builder' icon on the top right.

1. Click on the Sequence Flow.
2. Locate in the properties on the right side the option for the 'Condition expression'.
3. Click on 'Condition expression' and then click on the the 'Condition Builder' icon on the top right.
4. Search for the property max speed in the 'Select variable...' field.
5. Select 'Greater than or equal' for the operator.
6. Enter the value 15 into the 'Enter number value...' field.
7. Press 'Ok' to add the condition.

8. Save the model to validate the result.

8. Name the User Task 'Upload Driver's License'

Double-click the task name and name it 'Upload Driver's License'.

1. Double-click the task name.
2. Change it to 'Upload Driver's License'.
3. Save the model to validate the result.

9. Create an Exclusive Gateway

The next step is to create another Exclusive Gateway after the User Task. It must be linked with a Sequence Flow.

1. Click on the User Task.
2. Locate the quick draw items on the right side directly next to the User Task.
3. Click on the Gateway symbol to add an Exclusive Gateway.
4. Save the model to validate the result.

10. Add an alternative path between the Gateways

Add an additional Sequence Flow between the two Exclusive Gateways. Mark it as a default flow and add the label '< 15 km/h'.

1. Click on the first Exclusive Gateway.
2. Drag and drop the Sequence Flow into the second Exclusive Gateway.
3. Pull the Sequence Flow down so that it's not above the User Task.
4. Search the properties on the right side for 'Default flow' and check it.
5. Change the name of the Sequence Flow to '< 15 km/h'.
6. Save the model to validate the result.

11. Create an unlock scooter task

The next step is to create another User Task after the last Exclusive Gateway. It must be linked with a Sequence Flow and named 'Unlock Scooter'.

1. Click on the merging Exclusive Gateway.
2. Locate the quick draw items on the right side directly next to the Exclusive Gateway.
3. Click on the user symbol to add an User Task.
4. Name the User Task 'Unlock Scooter'.
5. Save the model to validate the result.

12. Create a return scooter task

The next step is to create another User Task after the unlock scooter task. It must be linked with a Sequence Flow and named 'Return Scooter'.

1. Click on the unlock scooter task.
2. Locate the quick draw items on the right side directly next to the task.
3. Click on the user symbol to add a User Task.
4. Name the User Task 'Return Scooter'.
5. Save the model to validate the result.

13. Create an End Event

Last, you need to create an End Event. This can be done by clicking on the Return Scooter task and select the round circle without content.

1. Click on the return scooter task.
2. Locate the quick draw items on the right side directly next to the task.
3. Click on the round circle without content.
4. Save the model to validate the result.

14. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the top menu bar which above the canvas.
2. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside.
3. Press this button and a popup window will appear.
4. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.
5. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

15. Rent a scooter which goes 15 km/h

In Flowable Work, execute everything what you have done so far. Create a scooter which goes 15 km/h.

1. Select 'New' below 'Work' or 'Work' below 'New'
2. Select the process 'FLW-P001 Rent Scooter'.
3. In the Start Form, select at least a scooter type of your choice and put in for the speed 15.
4. Press 'Submit' and run through the tasks. Was the 'Upload Driver's License' Task executed?
5. Once you return the scooter, this task will be completed.

16. Rent a scooter which goes 14 km/h

In Flowable Work, execute everything what you have done so far. Create a scooter which goes 14 km/h.

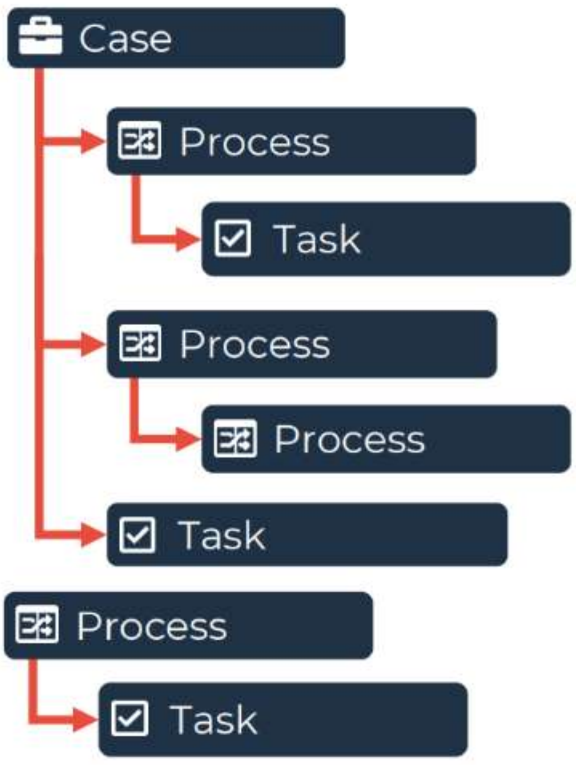
1. Select 'New' below 'Work' or 'Work' below 'New
2. Select the process 'FLW-P001 Rent Scooter'.
3. In the Start Form, select at least a scooter type of your choice and put in for the speed 14.
4. Press 'Submit' and run through the tasks. Was the 'Upload Driver's License' Task executed? This time it shouldn't.
5. Once you return the scooter, this task will be completed.

Build Hierarchies

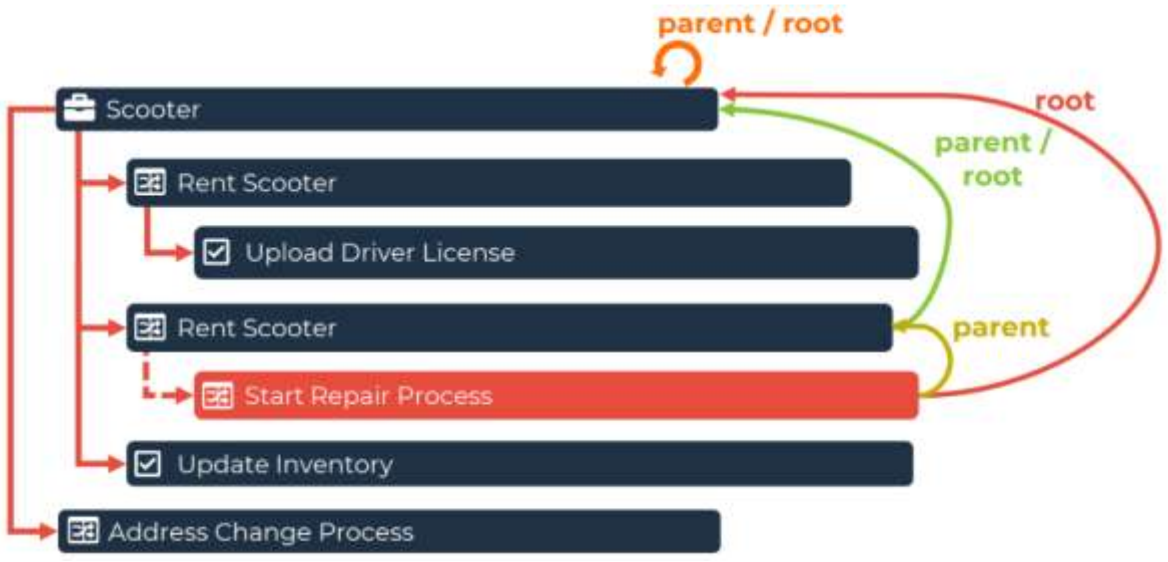
To enable the real power of CMMN and BPMN, we need to use both together. Let's look at how we can do that.

In this chapter, we will explore how to combine BPMN and CMMN in Flowable to create hierarchies of processes and cases. This integration allows us to design more complex and sophisticated workflows, enabling us to build advanced solutions for our business scenarios.

Understanding Hierarchies: In Flowable, we can create hierarchies of cases and processes, allowing us to establish relationships between them. For example, we can have a case at the top level, followed by multiple processes below it. Within each process, we can have tasks and subprocesses. This hierarchical approach provides a structured and organized way to manage and execute various activities within a broader context.



Example: Scooter Rental Management Let's consider an example where we have a "Scooter Rental" case at the top level. Below it, we can have multiple "Rent Scooter" processes, each responsible for handling individual scooter rentals. Each "Rent Scooter" process can have tasks such as "Upload Driver's License" (required only if the scooter is fast) and optionally a "Repair" process. Furthermore, we could have additional processes initiated from the scooter, like "Scooter Disposal," etc.



Entity Links: The connections between cases, processes, and tasks are referred to as "Entity Links". In most cases, these links are automatically created by Flowable, enabling seamless communication and data exchange between different levels of the hierarchy.

Referring to Higher Levels: To store important case-wide information at the top level and refer to it on the process level, we can use the "parent" and "root" keywords. For instance, to access the scooter type within the process, we would use ${root.scooterType}, ensuring that Flowable looks for the variable on the case level rather than the process level.

Connecting CMMN and BPMN: We can establish connections between CMMN and BPMN elements through specific tasks:

1. **Process and Case Tasks in CMMN:** These tasks can be used to trigger BPMN processes or cases from within a CMMN diagram. When a Process or Case Task is reached, a new instance of the BPMN process or case is created, and each instance will have its own set of variables.
2. **Call Activities and Case Tasks in BPMN:** In BPMN, we can use Call Activities to invoke other BPMN processes or Case Tasks to trigger CMMN cases. This enables seamless integration between the two notations, allowing for flexible and dynamic process orchestration.

Process Tasks

There is a layer of abstraction in process management as well. By simply using the Process Task, you can call another process and wait for the result.

In this chapter, we will explore Process Tasks and learn how to use them to replace existing user tasks in our BPMN models. As previously teased, it's now time to retire our old "Rent Scooter" Human Task.

Replacing User Task with Process Task: To replace the existing "Start Rental" Human Task, we will use a Process Task instead. In the Process Task configuration, we need to set the "Process reference" attribute. We can either create a new process or reference an existing one. Since we've already created the "Rent Scooter" process, we can simply reference it. Remember to set the process task as manual activation and repeatable if you create it anew.

Impact on Data Storage: With the introduction of Process Tasks, our case structure will change. Data that needs to persist throughout the entire case's lifetime, such as "scooterType", "maxSpeed", "color", etc., will now stay on the case level. On the other hand, data that changes for every rental process, like "customerName" and the "driversLicense" variable, will be stored on the process level. This separation allows for more efficient data management and ensures that relevant information is kept in the appropriate context. It also makes sure that only the people who need access get access.

Adjusting Conditions: Since our Process Tasks now start new process instances, we need to update the conditions that were previously present after the gateway (Speed?). Instead of accessing the vehicleSpeed variable directly, we'll now use the root keyword to access the case. The conditions will look like this:

${vars:greaterThanOrEquals(root.vehicleSpeed, 15)}

${vars:lessThan(root.vehicleSpeed, 15)}

Alternatively, we could use the "parent" keyword since we only have one hierarchy level. In case there's no higher hierarchy level, both root and parent will refer to the current entity. For instance, if we stored the scooter type on {{root.scooterType}} instead of {{scooterType}} in our start form, it would have the same effect.

Process Tasks in BPMN: To achieve the same effect in BPMN, we use "Call Activities." These work similarly to Process Tasks in CMMN, allowing us to create new process instances. It's essential to be cautious and not confuse Call Activities with "Sub-processes," as they serve different purposes. For example, in our rental process, a "Start Repair Process" could be implemented as a sub-process, which is only initiated if major repairs are necessary. When we create such a process, we can refer to the Rental Process with "parent" and the Scooter Lifecycle case with "root."

CMMN vs. BPMN

Should you use CMMN or BPMN? Long story short, use both. Let's discuss when you use which.

In our journey through Flowable, we have explored both CMMN and BPMN, two powerful modeling notations with distinct characteristics. Let's dive deeper into the comparison between these notations and understand when each one is best suited for our process modeling needs.

CMMN: Cases for High-Level Overview CMMN provides a high-level overview of complex business processes. Cases are designed to be reactive and flexible, ideal for scenarios where the exact flow is uncertain or subject to frequent changes. Business data is typically stored at the case level, providing a centralized view of relevant information. CMMN allows for repetition and optional steps, supporting a wide range of ad-hoc interactions.

BPMN: Clarity and Structured Flow On the other hand, BPMN excels in modeling structured and predefined processes with clear, linear flows. Processes in BPMN follow a predictable sequence of activities and events, making it easier to visualize and understand the workflow. BPMN is designed for processes with well-defined steps and is less agile in handling changes during process execution.

Combining CMMN and BPMN: Choosing between CMMN and BPMN is not about exclusivity but about combining them effectively. In more complex, human-driven scenarios, it is often beneficial to start with a CMMN case that provides the overarching framework and then model individual steps using BPMN. CMMN acts as the skeleton, providing flexibility, while BPMN adds the meat with its structured and clear flow.

Leveraging Advanced Concepts: Though CMMN is often associated with more ad-hoc interactions, BPMN also incorporates advanced concepts to handle complex scenarios. For instance, BPMN includes Event Sub-processes that allow for more fine-grained control and event-based decision making. This means that there may be cases where you do not need CMMN at all!

No One-Size-Fits-All Solution: In the world of process modeling, there is no one-size-fits-all solution. A seemingly simple case may become quite complicated when modeled in BPMN, while a straightforward BPMN process could reveal complexities when represented as a CMMN case. It is essential to carefully analyze the nature of the process and the desired level of control and flexibility before deciding on the appropriate notation.

Hierarchies

1. Replace the Rent Scooter with a 'Process Task'

Currently the rental of our scooter task is a simple Human Task. But now we have the process which is taking care about the rental. Let's replace first the rent scooter task with a process task.

1. Click on the 'Rent Scooter' Human Task.
2. Locate the wrench icon directly below the task (hover text is 'Transform')
3. Select Tasks and then Process Task
4. Save to validate the result.

2. Link the existing process to the case

In the last exercise we created a process. Now let's use the process inside the case. Let's add the 'Process reference' and select a reference.

1. Click on the Process Task in the diagram.
2. In the property list, search for 'Process reference' which is located in the 'General' section.
3. Click on 'link to an existing one'.
4. Search 'FLW-P001 Rent Scooter' in the list.
5. Click on 'FLW-P001 Rent Scooter' to select the item.
6. Save the model to validate the result.

3. Adapt the max speed condition to use the parent variable

The scope of the variables has changed and the process itself no longer has a variable 'maxSpeed'. Change the condition to read the variable from the parent or root instead.

1. Click on the Sequence Flow with the condition (after the gateway).
2. Locate in the properties on the right side the option for the 'Condition expression'.
3. Click on 'Condition expression' and then click on the the 'Condition Builder' icon on the top right.
4. Search for the property root.maxSpeed in the 'Select variable...' field.
5. Select 'Greater than or equal' for the operator.
6. Enter the value 15 into the 'Enter number value...' field.
7. Press 'Ok' to add the condition.
8. Save the model to validate the result.

4. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the top menu bar which above the canvas.
2. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside.
3. Press this button and a popup window will appear.
4. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.

5. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

5. Create a scooter and rent it twice

In Flowable Work, create a new case and rent the scooter twice. Once you have done that, decommission the scooter to complete the exercise.

1. Select 'New' below 'Work' or 'Work' below 'New'
2. Select the case 'FLW-C001 Scooter'.
3. In the Start Form, enter any data you want.
4. Press 'Start' and run through the tasks.
5. Once ready, press the rent scooter button.
6. Complete the rent scooter process.
7. Once done, execute the rent scooter one more time.
8. Complete the rent scooter process.
9. Once you then decommission your scooter, the exercise will be done.

6. Create a Form for the 'Return Scooter' task

Create a form and reference it as the task form of the 'Return Scooter' task within the process. Simply click on the task and select 'Form reference'. The form must have the key 'FLW-F013'.

1. Click on the process task and open the process.
2. Click on the task 'Return Scooter'.
3. In the property list, search for 'Form reference' which is located in the 'General' section.
4. Enter 'FLW-F013 Return Scooter' as name.
5. Enter as key 'FLW-F013' as key.
6. Press the 'Create' button to create the form.
7. Save the model to validate the result.

7. Create a single select and name it 'Damages'

The next step is to create a single select and name it 'Damages'. Press save to update the validation list on the right.

1. Look for the 'Select (Single)' component in the palette on the left side.
2. Drag and drop the Select (Single) into the form model.
3. Enter the name 'Damages'.
4. Save the model to validate the result.

8. Provide the list of items

As a next step, we need to add the items to the select. Therefore you can search for 'Items' and add 'None', 'Small', and 'Large'. The text should be the name as provided and the value the same in lowercase.

1. Select the 'Damages' field.
2. Search the properties on the right side for 'Items'.
3. Press the 'Add item' button.
4. Write 'none' in the field 'Value' and 'None' in the field 'Text'.
5. Press 'Add item' again and repeat for 'small' / 'Small'.
6. Press 'Add item' again and repeat for 'large' / 'Large'.
7. Press the 'OK' button.
8. Save the model to validate the result.

9. Create an exclusive gateway after the scooter return and name it 'Requires Repair?'

Go back to our process. Then the next step is to create another Exclusive Gateway after the Return Scooter task. It must be linked with a Sequence Flow.

1. Increase the space between the User Task and the end event.
2. Locate the exclusive gateway in the palette on the left side.
3. Drag and drop the exclusive gateway from the palette onto the sequence flow.
4. Let it go once the sequence flow is green.
5. Save the model to validate the result.

10. Create a merging exclusive gateway after the 'Requires Repair?' and don't give it a name

The next step is to create another Exclusive Gateway after the 'Requires Repair?' gateway. It must be linked with a Sequence Flow. The purpose is to have a merging gateway after the conditions.

1. Increase the space between the last gateway and the end event.
2. Locate the exclusive gateway in the palette on the left side.
3. Drag and drop the exclusive gateway from the palette onto the sequence flow.
4. Let it go once the sequence flow is green.
5. Save the model to validate the result.

11. Change the sequence flow between the gateways to the default flow and label it 'No'

Let's use the sequence flow between the two gateways as a default Flowable in case of no repairs are required. Therefore, we are going to mark it as a default flow and add the label 'No'.

1. Locate the new sequence flow between the two gateways.

2. Click on the sequence flow to select it.
3. Search the properties on the right side for 'Default flow' and check it.
4. Change the name of the Sequence Flow to 'No'.
5. Save the model to validate the result.

12. Add an alternative path for small repairs

Create an alternative path between the exclusive gateways for small repairs. Name the sequence flow outgoing from the gateway 'Small damages' and create a User Task 'Conduct Small Repair'. The sequence flow following that task should then merge back into the final exclusive gateway. Don't forget to set the condition on the sequence flow after the splitting gateway

1. Select the exclusive splitting gateway.
2. Locate the quick draw right of the gateway.
3. Take the User Task and drag and drop it above the existing sequence flow.
4. Change the name of the User Task to 'Conduct Small Repair'
5. Double click on the newly created sequence flow and give it the name 'Small damages'
6. Locate the attribute 'Condition expression' in the attribute panel
7. Click on 'Condition expression' and then click on the the 'Condition Builder' icon on the top right.
8. Select the variable 'damages'
9. Choose the operator 'Equals'
10. Select the value 'Small' from the list.
11. Click on the User Task and locate the quick draw menu.
12. Take the sequence flow and drop it on the merging gateway.
13. Press save to validate the result.

13. Add an alternative path for large repairs

Create an alternative path between the exclusive gateways for large repairs. Name the sequence flow outgoing from the gateway 'Large damages' and create a call activity 'Start Repair Process'. The sequence flow following that task should then merge back into the final exclusive gateway. Again, don't forget the condition.

1. Select the exclusive splitting gateway.
2. Locate the quick draw right of the gateway.
3. Select the 'Task' symbol on the top and select 'Call activity'.
4. Double click on the newly created sequence flow and give it the name 'Large damages'
5. Locate in the properties on the right side the 'Condition expression'
6. Click on 'Condition expression' and then click on the the 'Condition Builder' icon on the top right.
7. Select the variable 'damages'

8. Choose the operator 'Equals'
9. Select the value 'Large' from the list.
10. Click on the call activity and locate the quick draw menu.
11. Take the sequence flow and drop it on the merging gateway.
12. Press save to validate the result.

14. Create the Repair Scooter Process

Now we need to create a new process for the repair. Therefore select the call activity and create a new process with the name 'FLW-P002 Repair Scooter Process' and key 'FLW-P002'. Save all models to validate the result.

1. Click on the Call Activity model.
2. In the property list, search for 'Process reference' which is located in the 'General' section.
3. Enter 'FLW-P002 Repair Scooter Process' as name.
4. Enter 'FLW-P002' as key.
5. Click on 'Create' to create a new process.
6. Save all models to validate the result.

15. Create a User Task 'Write assessment'

As a first step in the new process we should write an assessment. Let's create a User Task with the associated sequence flow.

1. Click on the start event in the FLW-P002.
2. Locate the user symbol in the quick draw next to the start symbol.
3. Drag and drop the user symbol.
4. Rename the User Task to 'Write assessment'.

16. Change the assignee to '${root.initiator}'

Now the default assignee is ${initiator} and we can't use that inside the sub-process since the variable is not set. Let's change that variable to the initiator of our case instance which is root.initiator.

1. Click on the User Task 'Write assessment'.
2. In the properties on the right side, locate the field 'Assignee' which is located in the attribute group 'Assignment'.
3. Change it to '${root.initiator}'
4. Click the save button to save the process.

17. Create a splitting and a merging parallel gateway

Create two parallel gateways directly after the User Task. The first parallel gateway should split the sequence flow while the second one is merging the sequence flow again. Please also make sure that you have an end event after the last gateway.

1. Click on the User Task and locate the quick draw.
2. Take the end event and drag and drop it all the way to the right that you have some space in the middle.
3. Locate the parallel gateway in the palette on the left hand side.
4. Drag and drop two parallel gateways to the sequence flow.
5. Save the model to validate the result.

18. Create a parallel task 'Check Tires'

Between the parallel gateways there should be a task 'Check Tires' which is assigned to the case initiator.

1. Ensure you have enough space between the two parallel gateways.
2. Drag and drop a User Task on the sequence flow between the gateways.
3. Rename the User Task to 'Check Tires'.
4. Change the assignee to '${root.initiator}'
5. Save the model to validate.

19. Create a parallel task 'Check Electronics'

Between the parallel gateways there should be a task 'Check Electronics' which is assigned to the case initiator.

1. Move the existing User Task to the top.
2. Create a new sequence flow between the parallel gateways
3. Drag and drop a User Task on the sequence flow between the gateways.
4. Rename the User Task to 'Check Electronics'.
5. Change the assignee to '${root.initiator}'
6. Save the model to validate.

20. Create a parallel task 'Check Frame'

Between the parallel gateways there should be a task 'Check Frame' which is assigned to the case initiator.

1. Create a new sequence flow between the parallel gateways
2. Drag the sequence flow below the task.
3. Drag and drop a User Task on the sequence flow between the gateways.
4. Rename the User Task to 'Check Frame'.
5. Change the assignee to '${root.initiator}'
6. Save the model to validate.

21. Publish the changes

To publish your app, click on the cloud icon featuring a small arrow to the top.

1. Locate the top menu bar which above the canvas.

2. Locate the publish button in the toolbar. This is the cloud symbol with a little arrow inside.
3. Press this button and a popup window will appear.
4. In the popup window, confirm the deployment target and the app. In this environment you do not really have a choice and you can simply say 'Publish'.
5. Once you completed this exercise you will be redirected automatically to Flowable Work. In a real application you would now need to switch to 'Flowable Work' manually.

22. Create a scooter and rent it twice

In Flowable Work, create a new case and rent the scooter twice. For the first time select small repair, for the second time select large repair. Once you have done that, decommission the scooter to complete the exercise.

1. Select 'New' below 'Work' or 'Work' below 'New'
2. Select the case 'FLW-C001 Scooter'.
3. In the Start Form, enter any data you want.
4. Press 'Start' and run through the tasks.
5. Once ready, press the rent scooter button.
6. Ensure that for the return you select small damages
7. Complete the rent scooter process.
8. Once done, execute the rent scooter one more time.
9. Ensure that for the return you select large damages
10. Complete the rent scooter process.
11. Once you then decommission your scooter, the exercise will be done.