



noKodr User Guide

By Enzigma Solutions LLP





Contents

I Introduction	2
I.I Orektic	2
I.II Enzigma LLC	2
II Product Introduction	3
III Product Features	4
IV Product Versions	5
V Configuration Guide	6
V.I Installation Steps	6
V.II Pre-requisites	7
V.III Step by Step Walkthrough	8
V.III.1 Setup	8
V.III.1.1 Types of Layout and Layouts Creation	8
V.III.1.2 Layouts and its Row Actions	10
V.III.1.3 Components and its Config	13
V.III.1.4 noKodr Components and it's Attributes	15
V.III.1.5 Models in Layout	71
V.III.1.6 Charts	74
V.III.1.7 Variables	77
V.III.1.8 Workflow	80
V.III.1.9 Listview Creation and its Config	86
V.III.1.10 Publish Layout and List Views	96
V.III.1.11 Add Publish Layouts & List Views to Flexi Pages	97
V.III.1.12 REST API Configuration	103
VI Contact Us	125



I Introduction

I.I Orektic

Orektic is known for state-of-the-art cloud-based, SAAS solutions that help our clients to transform the way they do business.

We exist to solve the critical issues facing our clients, both large and small. Our unique approach is not only what differentiates us, but also what makes us successful. We provide a broad range of services and solutions to help organizations facilitate change, achieve their vision, and optimize performance and productivity.

From implementing new business strategies to ultra-efficient work processes, Orektic is ready to tackle any challenge and put you on the path to success. With state-of-the-art cloud-based, SAAS solutions transform the way you do business.

Orektic is one of the software companies from the Enzigma group.



I.II Enzigma LLC

Enzigma LLC is the exclusive reseller of Orektic products.





II ▶ Product Introduction

With our innovative thoughts, we have come up with a new AppExchange Package that will help the Salesforce admin, and developers which provides drag and drop-able Lightning Web Components that can be used on the home page, or the Record details page to ease the use of records according to the user's custom requirement.

Using noKodr you can design a UI as you want using different components, which are easy to integrate and configure with lots of features. There are various components that enhance your UI and give you a better experience and also you have the power of client (browser) side user flows and validations using workflows that are simple and easy to design as per your business logic and workflows provide various actions that are most suitable and usable actions that provide you with strong control over user experience.





III Product Features

- UI Workflows with finer control for business automation
- Event handling and workflow mechanism to communicate between components
- Conditional visibility/validation/disabling of fields and components
- Amazing repeater to interact with multiple records
- Built for admins, not for developers
- Interactive UI with powerful interdependent components
- More than 15 readily available components to create custom UI
- User-friendly configuration
- Empowering the admins
- Low to no dependency on developers and development process
- Enables the organization to go to production faster at a low cost
- Dedicated support team





IV Product Versions

Version	Release Date	Description
noKodr 2.1/2.1.0	02/20/2024	<ul style="list-style-type: none">• Added a new feature that is Master-Detail Re-parenting• Introduced our Layouts and List View on the Digital Experience Community Site• Enhanced the Filter Designer Functionality and layout optimization• Addressed some minor bugs to enhance the performance of the Application• Changes for Branding from PWR Forms to noKodr Enhanced Time Field Functionality





V Configuration Guide

V.I Installation Steps

To install noKodr managed package do follow the mentioned steps here:

1. To install **noKodr** [click here](#)
2. The link will direct you to the PWR Commons application page, it is a prerequisite package for noKodr
3. Click on Get it now from the apps information page
4. Enter the credentials for the org in which you want to install and log in
5. Choose to install the package in either the production org or Sandbox by clicking "Install in Production" or "Install in Sandbox"
6. Tick the checkbox indicating your agreement with the terms and conditions, then proceed by clicking on the "Confirm and Install" button
7. Select the desired installation option (Install for All Users, Install for Admins Only, or Install for Specific Profiles), and click "Install"
8. Click "Done" once the installation is complete
9. It may take some time to complete the installation package. You will be notified through Email once the installation is completed
10. Now open the PWR Commons application from the application manager
11. Click "Install Now" in front of the **noKodr** application in the list of products
12. Enter the credentials of the logged-in organization and log in
13. Select the desired installation option, and click "Install"
14. Click on Done once the installation is completed
15. It may take some time to complete the installation package. You will receive an email notification when the installation is finished

**** For reference and more details about package installation, please [click here](#)**

Any potential customer with a package link for noKodr from the support or sales team, then to install the noKodr managed package do follow the mentioned steps here:

1. Open the installation URL received from our sales or support team in the browser.
2. Enter your credentials for the Salesforce organization in which you want to install the package and then click on **Log In**.
3. Repeat the steps as mentioned above from step 13



V.II Pre-requisites

To get started, you need to make sure you are all good with the list of prerequisites/checklists mentioned below.

As a prerequisite, the Salesforce admin needs to make sure that the Salesforce org is already set up with their domain name.

In case your org is not set up with the Domain Name then the contents of the Package will not be available for use. To enable the domain name open Setup from the Quick Find box, find “My Domain” and set up a Domain Name for your org. For more details and information, please [click here](#).





V.III Step by Step Walkthrough

Using noKodr you can design a UI as you want using different components, which are easy to integrate and configure with lots of features. There are various components that enhance your UI and give you a better experience and also you have the power of client (browser) side user flows and validations using workflows that are simple and easy to design as per your business logic and workflows provide various actions which are most suitable and usable actions which provide you a strong control on user experience.

noKodr is a bundle of the following lightning components, which helps the Salesforce Admin/Developers achieve their requirements and goals with easy drag-and-drop-able lightning components. The below components help in overcoming the limitation of Salesforce configuration.

- Layouts
- List Views

V.III.1 Setup

V.III.1.1 Types of Layout and Layouts Creation : There are two types of Layout in noKodr that a user can configure which are as follows:

- **Application Layout:** In Application Layout, users can design a Layout according to their requirements with the help of Components, Models, Variables, Events, etc.
- **Record Layout:** Record Layout is an Object Specific Layout, In Record Layout, there are some predefined data like Models, Workflows, and Actions. In Layout one default 'Form' component is used through which the user can create or update object-specific records.

Creation of Application Layout: Here are the steps required to configure the Application Layout.

1. To create an application layout, click on setup then navigate to the layouts page

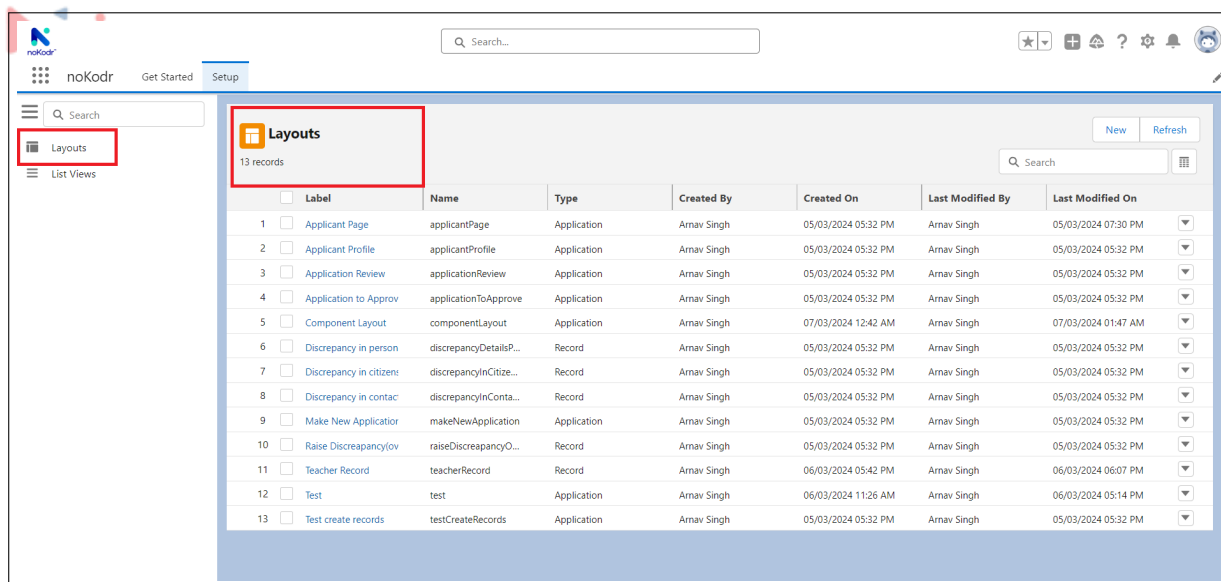


Figure 5.3.1: Layouts

2. Click on 'New'
3. Enter all the mandatory fields
4. Select type as Application
5. Click on Save

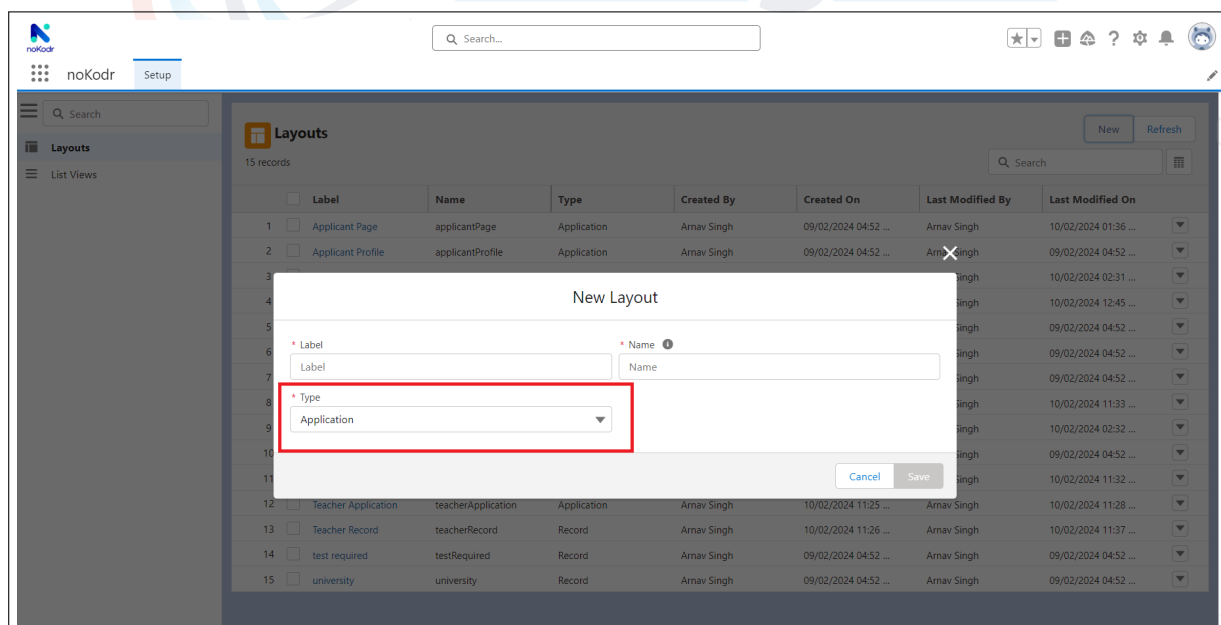


Figure 5.3.2: New Application layout

Creation of Record Layout: Here are the steps required to configure the record Layout.

1. To create a Record layout, click on setup then navigate to the layouts page



2. Enter all the mandatory fields and select the Object for which you want to create the layout and type as Record

The screenshot shows the 'New Layout' dialog box in the noKodr application. The dialog is open over a table of existing layouts. The fields are as follows:

Field	Value
Label	Account Layout
Name	accountLayout
Type	Record
Object	Account

Buttons: Cancel, Save

Figure 5.3.3: New Record Layout

This will create a record layout and it will navigate you to the Layout Designer page

V.III.1.2 Layouts and its Row Actions : On the Layout, we can perform five actions present as a drop-down value from the row action provided on the Layouts.

The screenshot shows the 'Layouts' table in the noKodr application. The table has 13 records. The first record is highlighted, and its row actions are shown in a dropdown menu.

	Label	Name	Type	Created By	Created On	Last Modified By	Last Modified On	Row Actions
1	Applicant Page	applicantPage	Application	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 07:30 PM	Design
2	Applicant Profile	applicantProfile	Application	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	Preview
3	Application Review	applicationReview	Application	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	Edit
4	Application to Approv	applicationToApprove	Application	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	Delete
5	Component Layout	componentLayout	Application	Arnav Singh	07/03/2024 12:42 AM	Arnav Singh	07/03/2024 01:42 AM	Publish
6	Discrepancy in person	discrepancyDetailsP...	Record	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	
7	Discrepancy in citizen	discrepancyInCitize...	Record	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	
8	Discrepancy in contac	discrepancyInConta...	Record	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	
9	Make New Applicator	makeNewApplication	Application	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	
10	Raise Discreapancy(ov	raiseDiscreapancyO...	Record	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	
11	Teacher Record	teacherRecord	Record	Arnav Singh	06/03/2024 05:42 PM	Arnav Singh	06/03/2024 06:07 PM	
12	Test	test	Application	Arnav Singh	06/03/2024 11:26 AM	Arnav Singh	07/03/2024 12:57 PM	
13	Test create records	testCreateRecords	Application	Arnav Singh	05/03/2024 05:32 PM	Arnav Singh	05/03/2024 05:32 PM	

Figure 5.3.4: Layouts and operations



1. **Design:** With the help of design action you will be able to design the Layout as per business requirements. Layout Designer allows to Drag and Drop of various Components in the desired section of the Record Layout

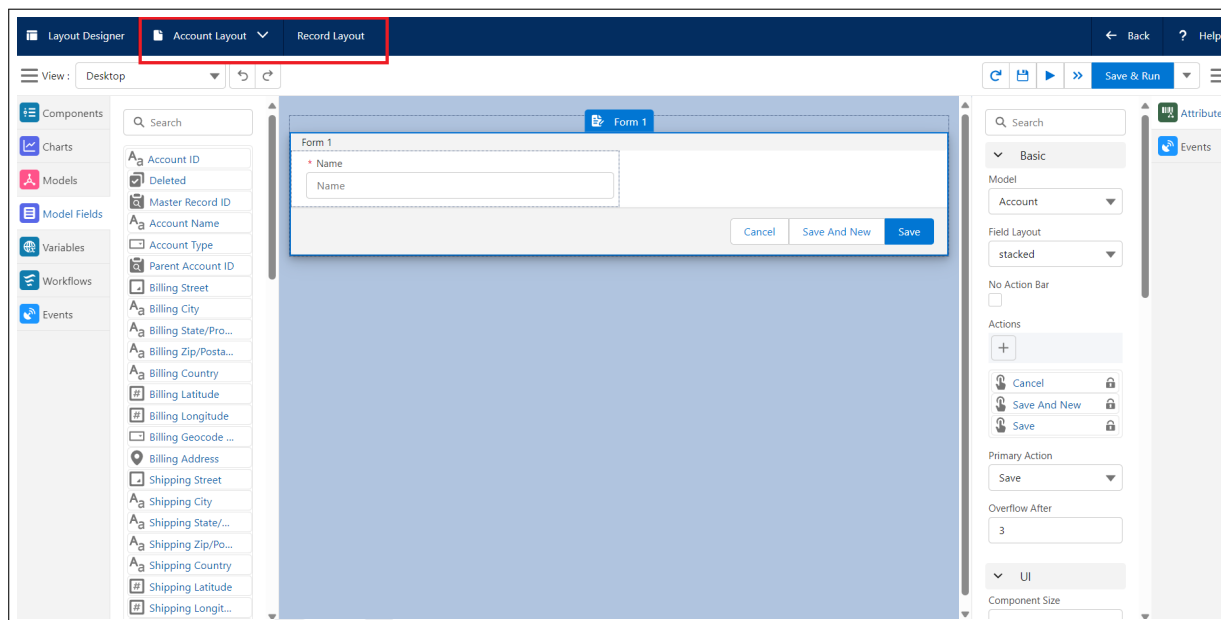


Figure 5.3.5: Layout Designer

2. **Preview:** Previews allow you to see how the layout will appear without having to run the entire Layout. Previews can help identify layout issues, such as overlapping components, incorrect spacing, or alignment problems, before saving the entire Layout
3. **Edit:** With the help of the edit action the user will be able to edit/update the Layout

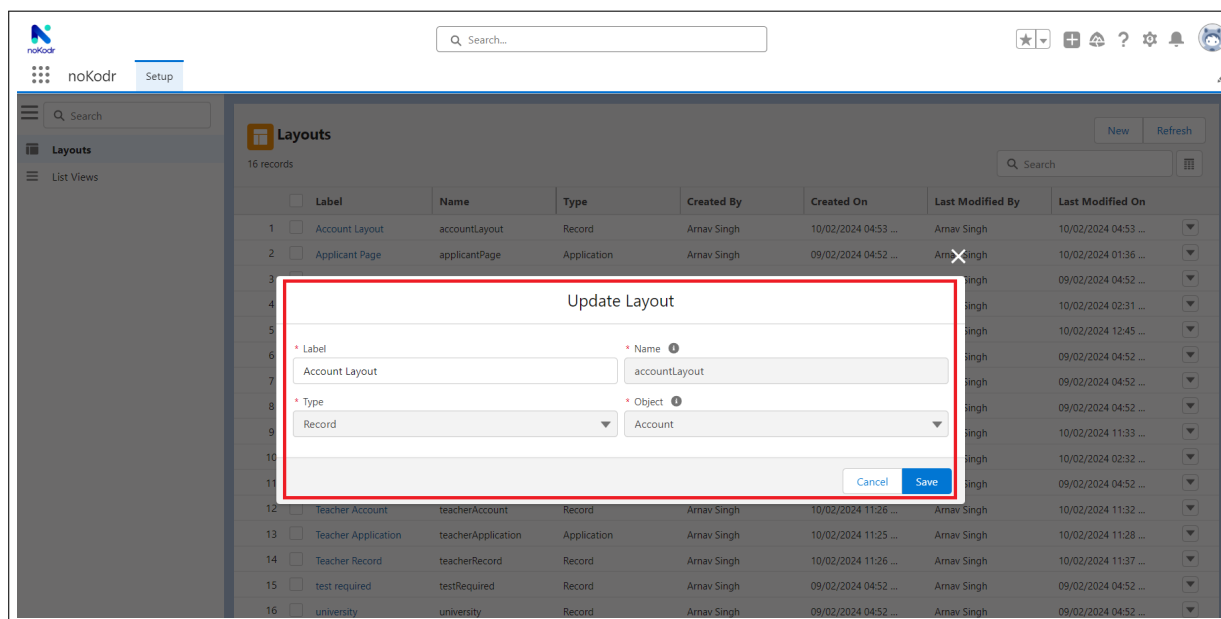


Figure 5.3.6: Edit Layout

4. **Delete:** With the help of the Delete action you can delete that specific layout out

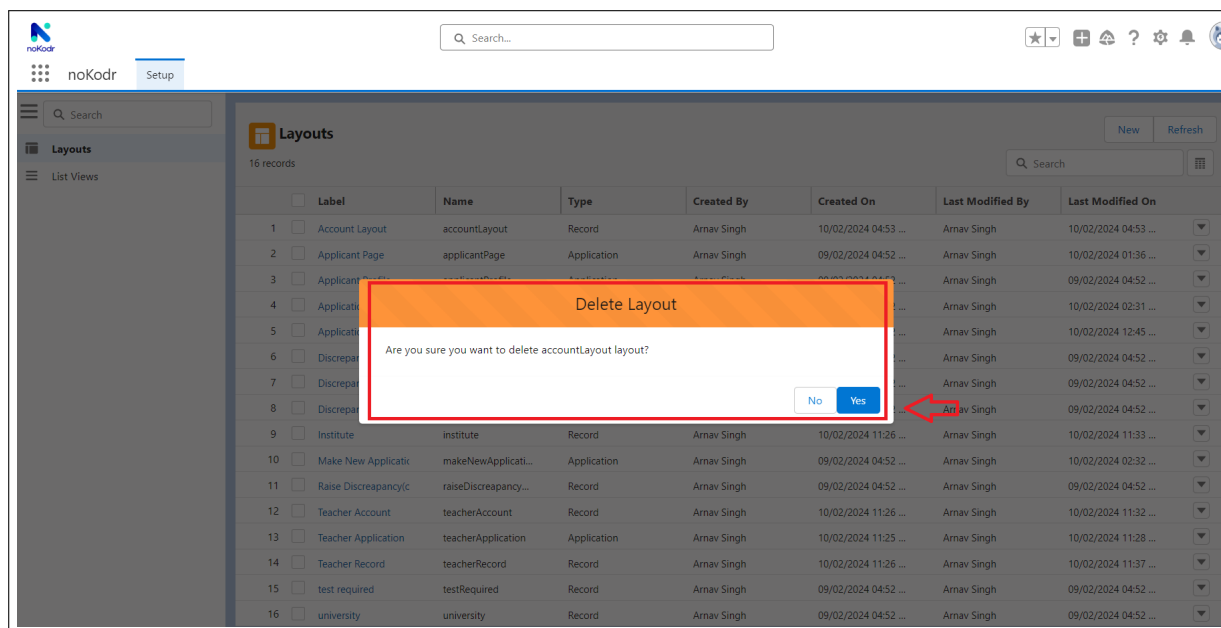


Figure 5.3.7: Delete Layout

5. **Publish:** noKodr gives the option to Publish Layout and List Views. Publishing helps the users to use Layouts and List Views on the Home Page or Record Page

Note: Without publishing, you cannot use Layouts and List Views on the home page.

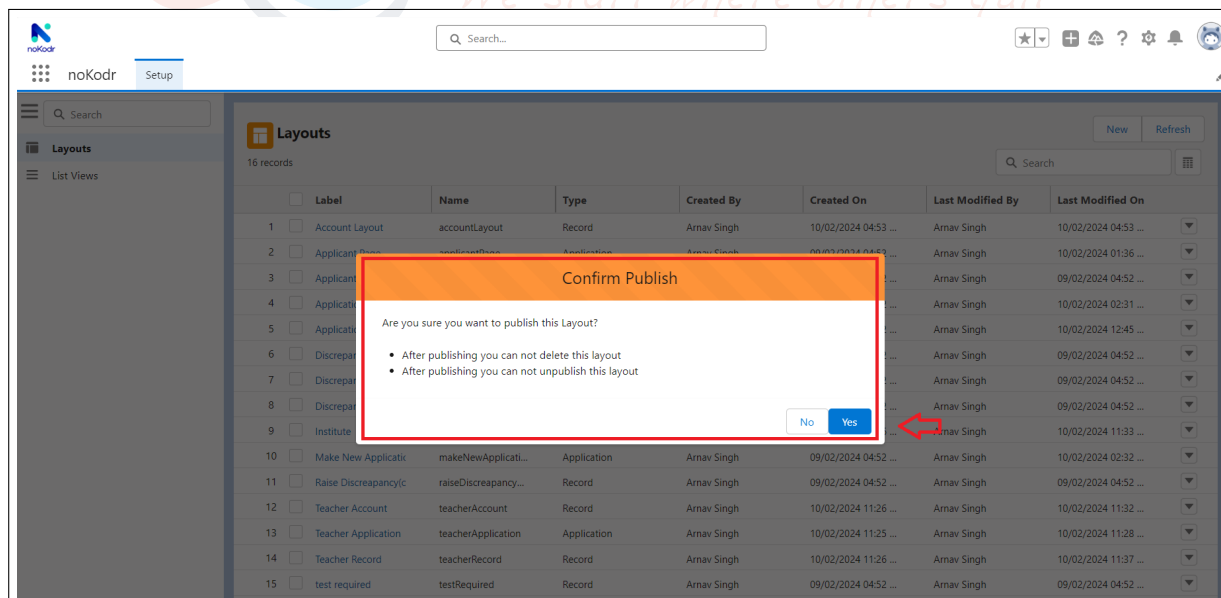


Figure 5.3.8: Publish Layout

V.III.1.3 Components and its Config :

- **Components:** Components is an HTML UI Where the user can Drag and Drop the Component in the Layout. Here are some common attributes of various components

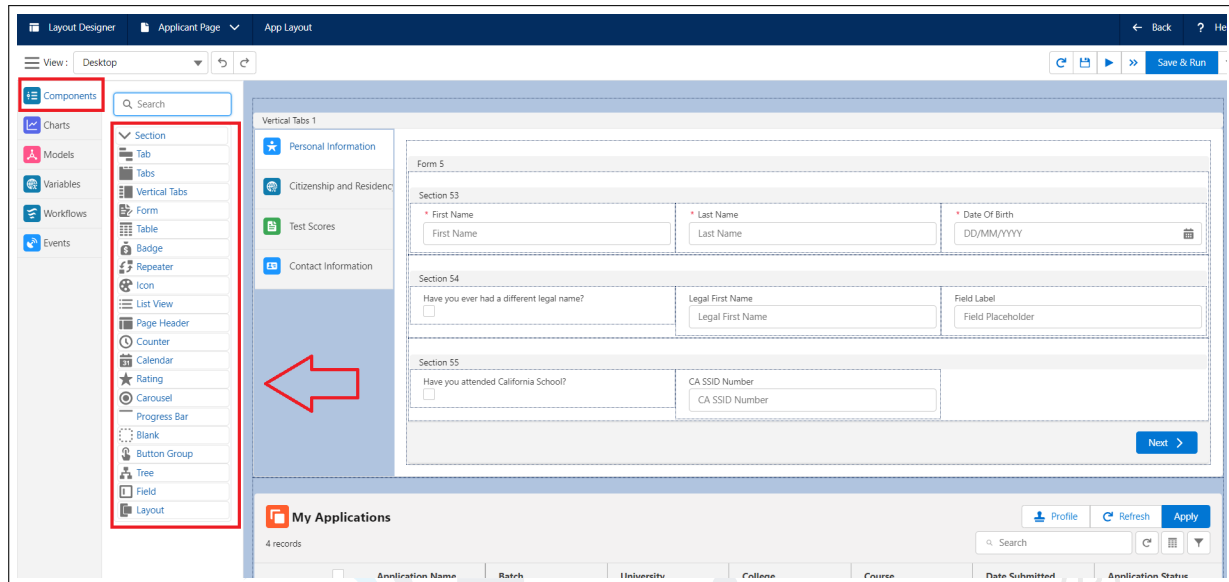


Figure 5.3.9: Components

- **How to Configure any Component:**

1. Go to Layout Designer

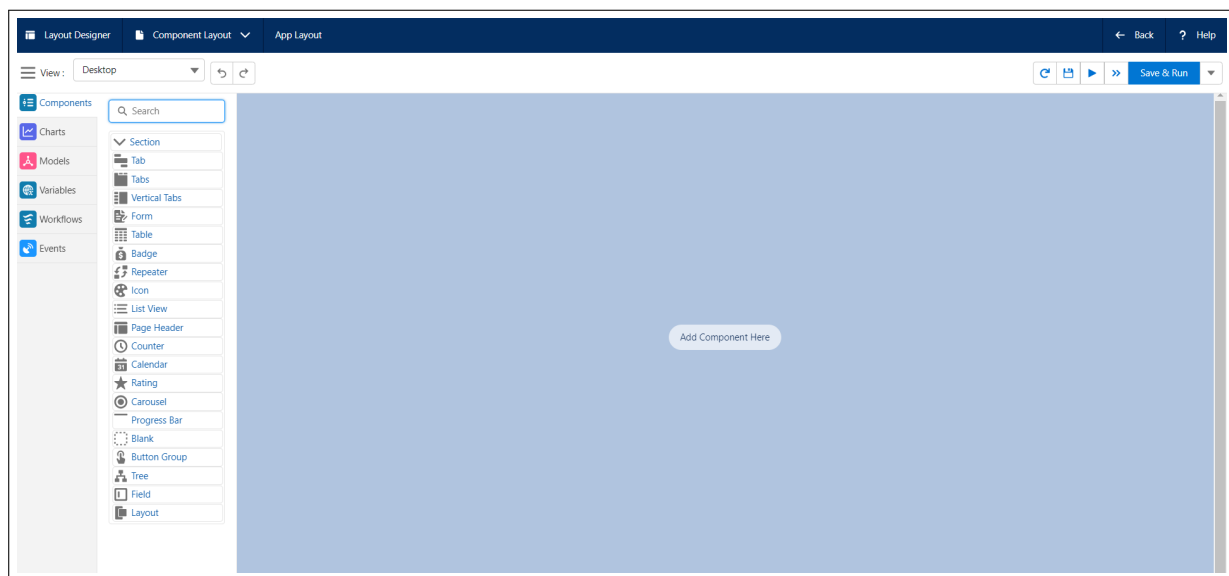


Figure 5.3.10: Layout Designer

2. Search and select the Component

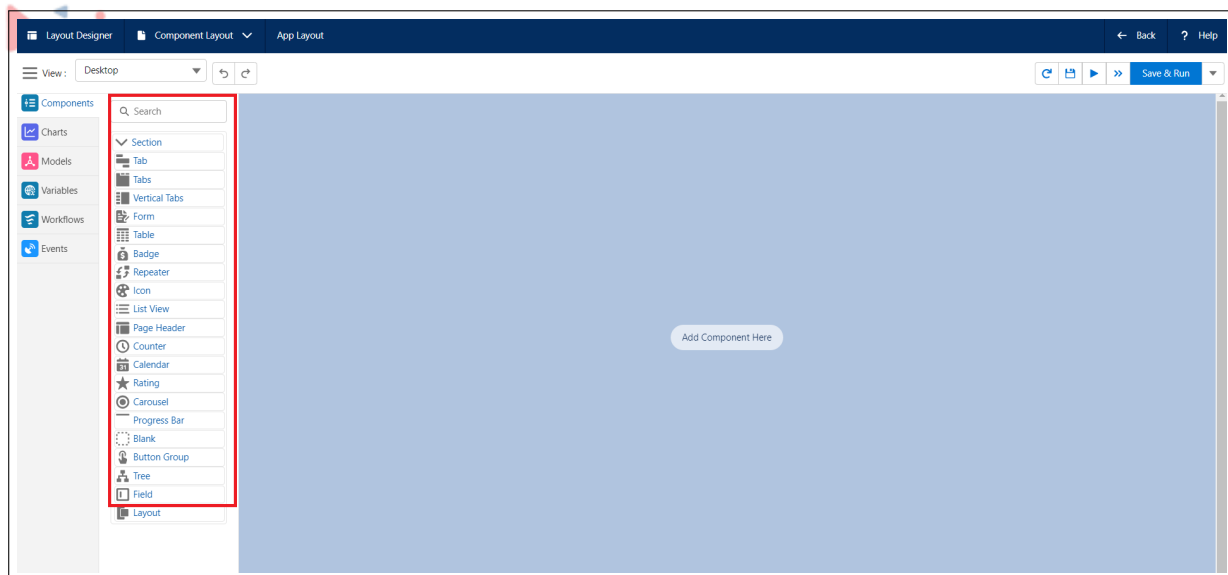


Figure 5.3.11: components

3. Drag and drop the Component in the layout. You can also select the drop zone in a layout and then click on the component it will add in place of the drop zone.

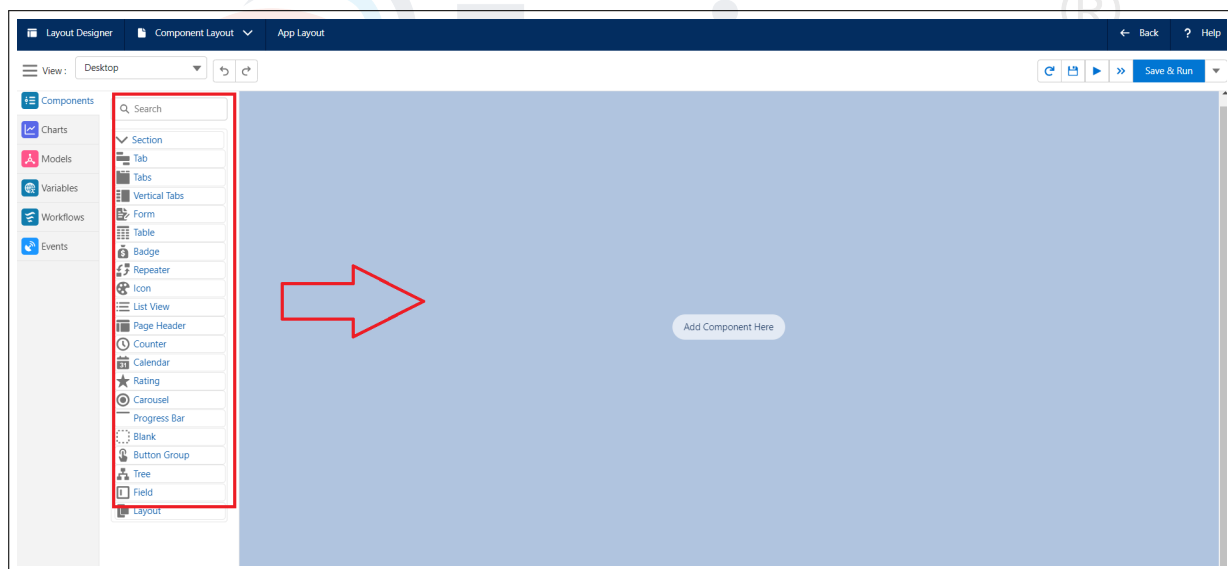


Figure 5.3.12: Components

4. Configure the attributes of each component as per your requirement which is available, once you drag and drop the component on the layout designer, on the right-hand side attributes vary according to the components.

V.III.1.4 noKodr Components and it's Attributes :

1. **Section:** Section is nothing but a Container, which contains multiple components like Form, Table, Badge, etc. The Section is a container with a header that is used to group related content. Users can collapse the Section by clicking on the header or on the drop-down arrow also users can add actions to the Section header.

Attributes of Section:

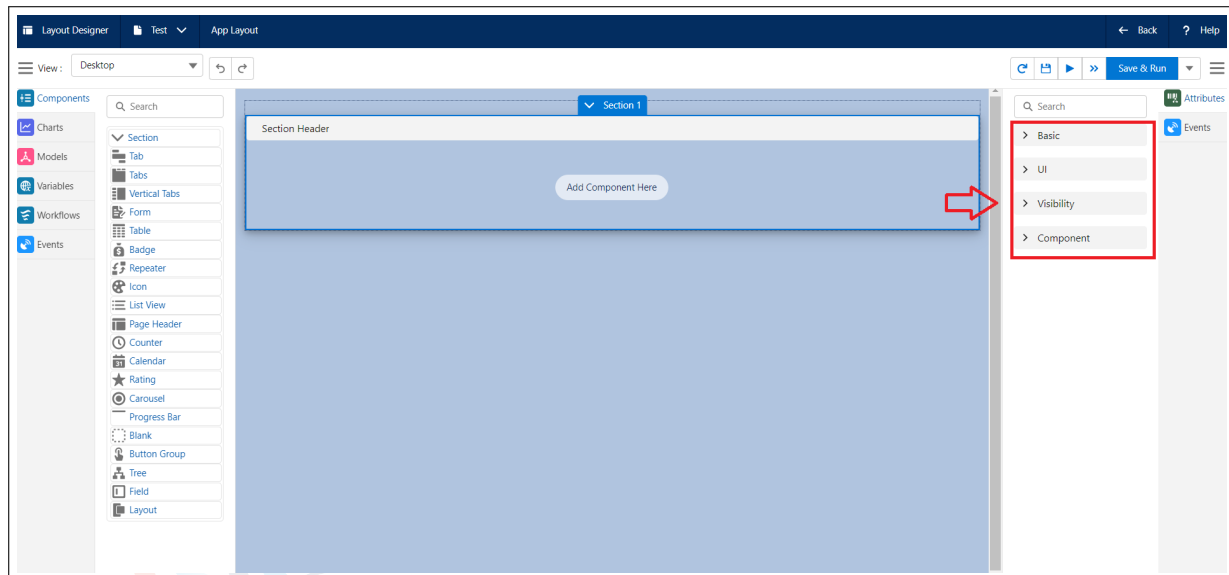


Figure 5.3.13: Attributes of Section

• Basic:

- Header: The header is a short description to define the Section. It is displayed at the top left of the Section. e.g. in the below screenshot, client details are used to specify the header of the Section
- No Header: If the No Header checkbox is checked then the header will not be visible on the Section

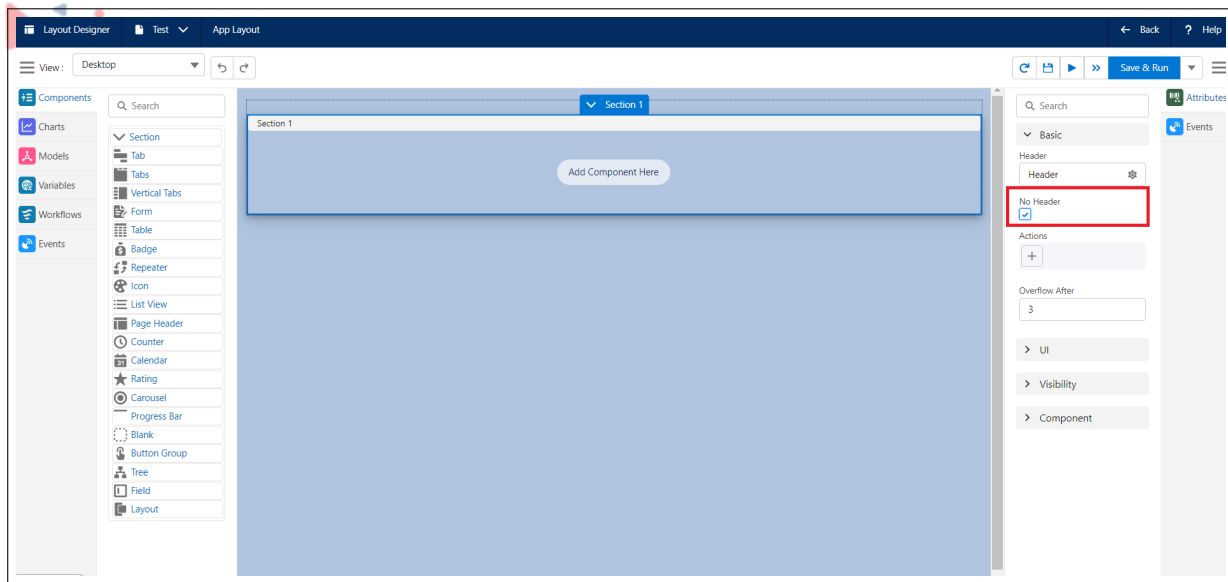


Figure 5.3.14: No Header

- Is Collapsible: You can make the Section collapsible by checking the 'Is Collapsible' checkbox. 'Is Collapsible' is used in the Section for a better User Experience. On clicking the down arrow or on the Section header the Section will collapse or expand.

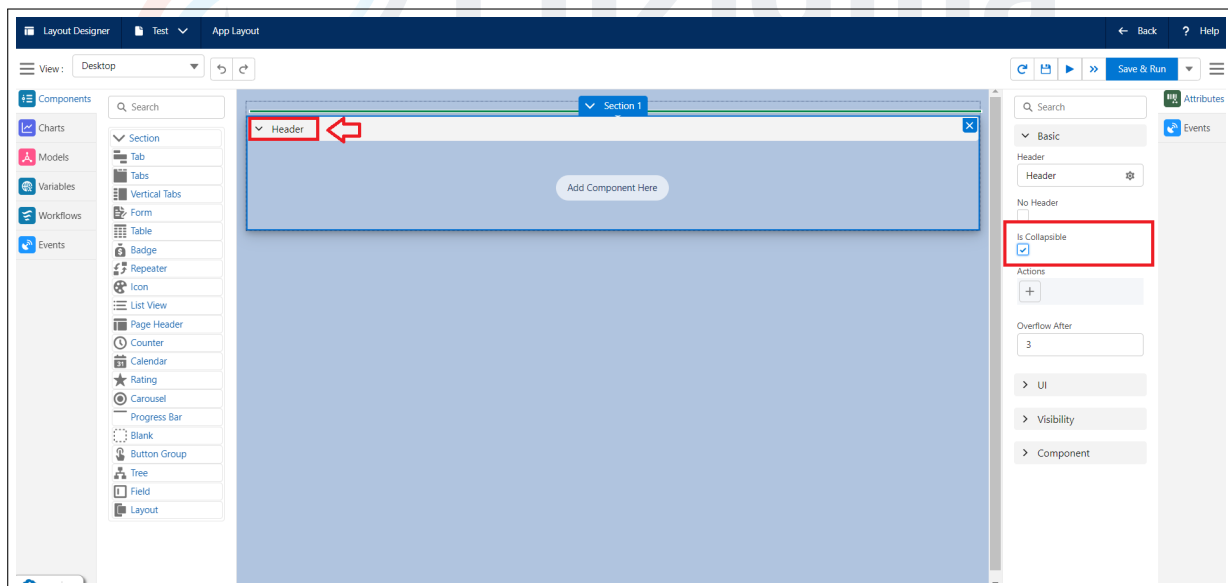


Figure 5.3.15: Is Collapsible

- Actions: You can add multiple actions and each action is displayed as a button. Only two actions will be displayed on the Section header and the rest of the actions will be displayed in a dropdown. By clicking on the button it performs a particular action e.g. invoking a specific action or workflow.

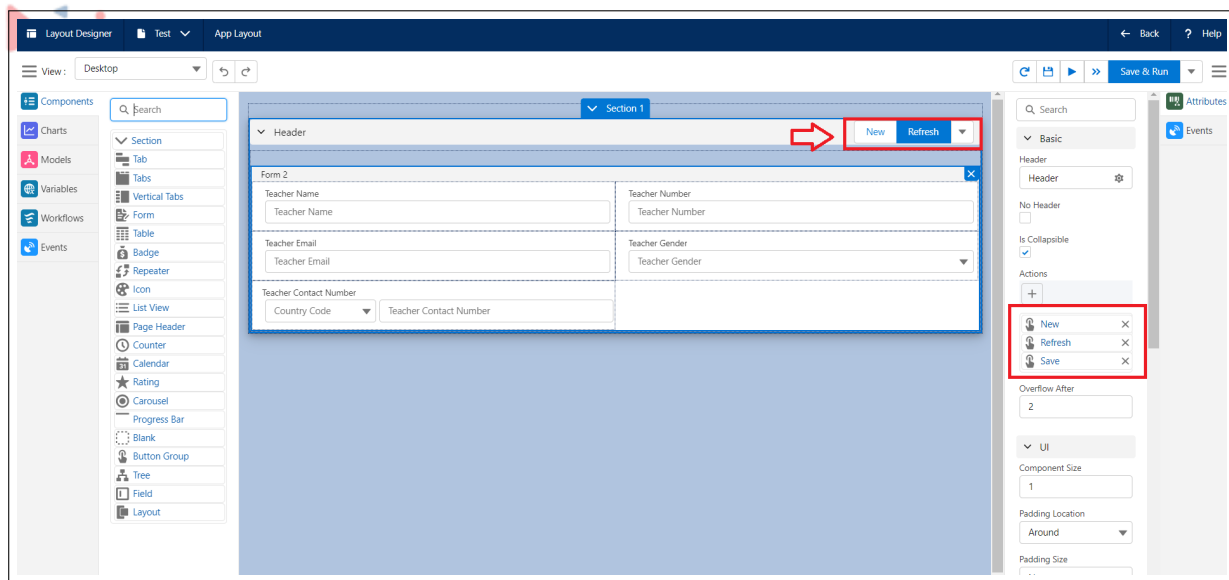


Figure 5.3.16: Action

- Overflow After: Overflow After an attribute is used to display the actions in list format after reaching its entered limit. By default, the value is 3 which means the three actions will displayed on a section header. If you added the new action despite having 3 actions then the new action will appear in the drop-down list section.
- UI:
 - Component Size: The user can modify the size of the component in the layout as per grid size.

In the following example, the section component has a size of 3 and the inside section page header has a component size of 2

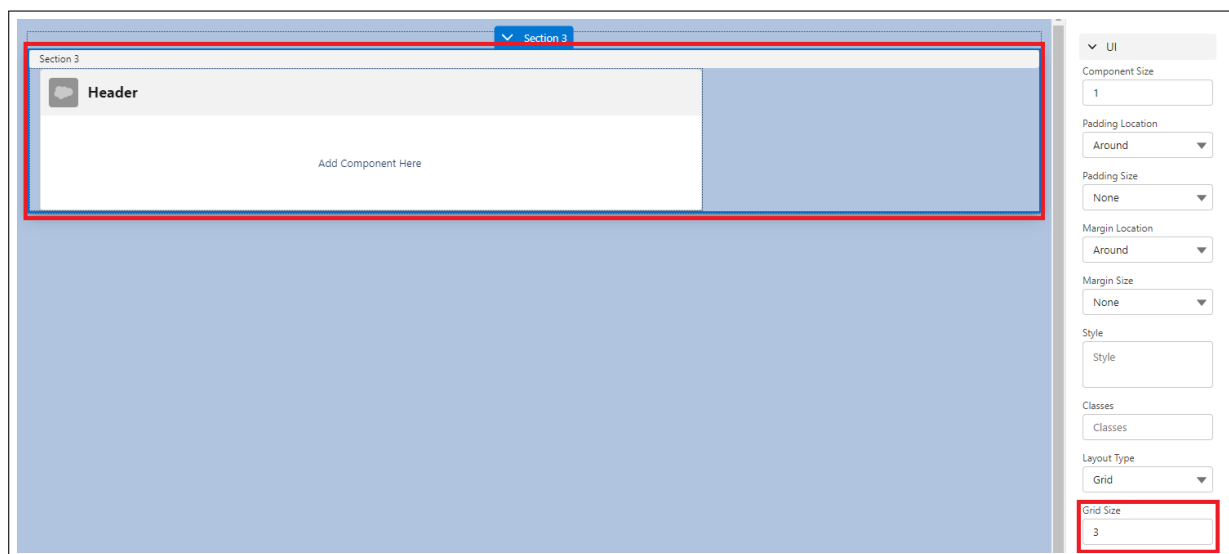


Figure 5.3.17: component size

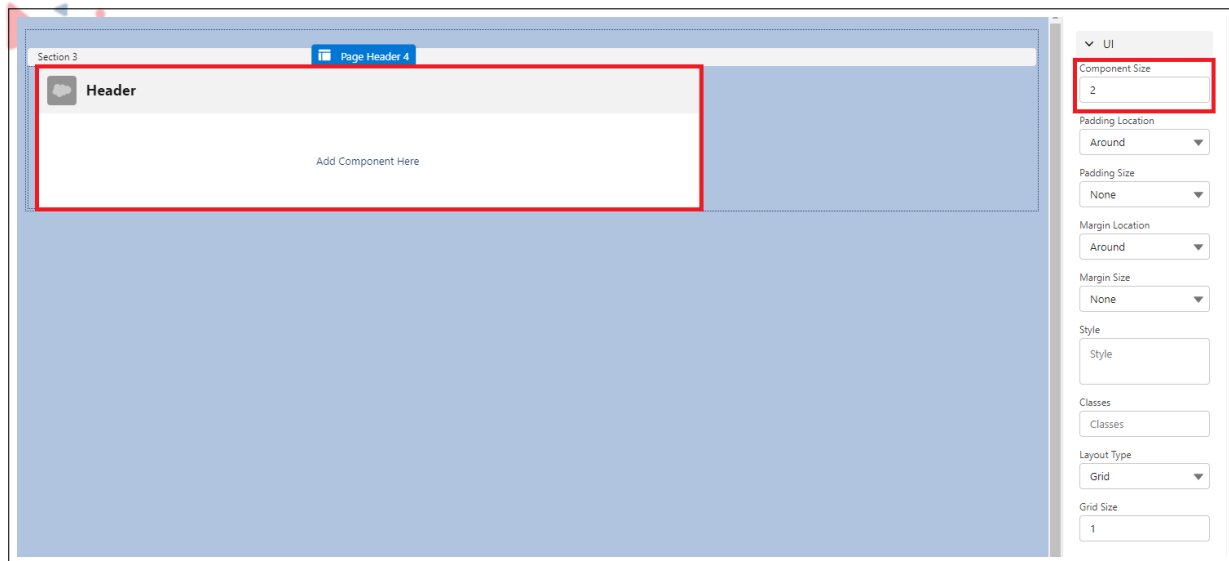


Figure 5.3.18: component size

- **Padding Location:** Defines the position of the padding for a component. The padding creates extra space within a component.

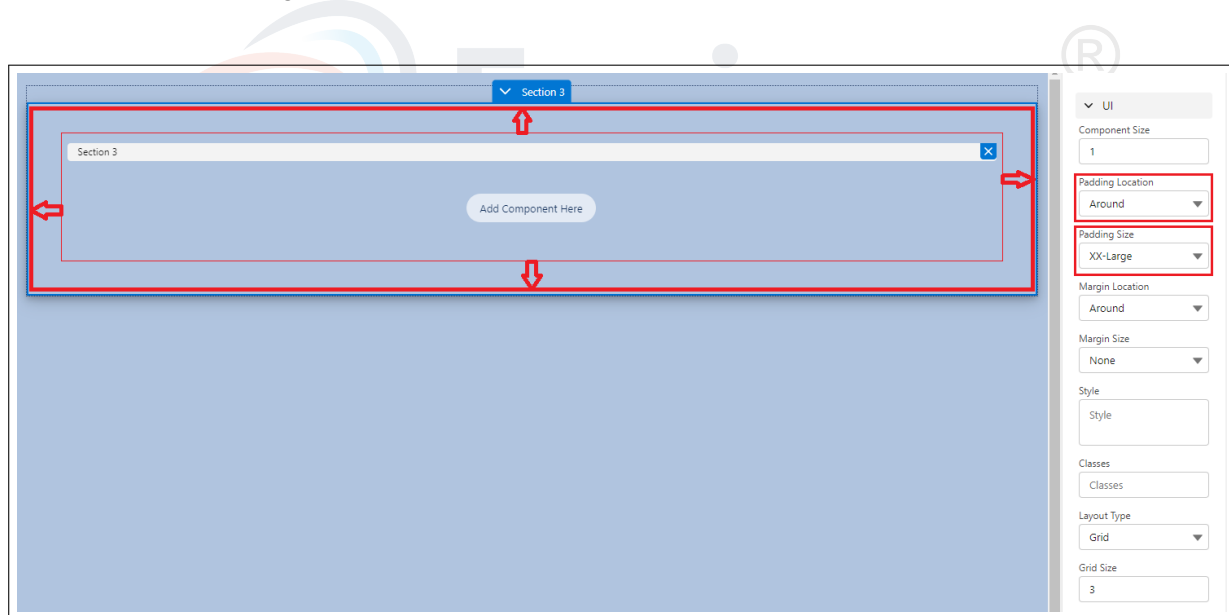


Figure 5.3.19: Padding location

Types of Padding Locations:

- Around:** Creates padding around the component
- Top:** Creates padding at the top of the component
- Left:** Creates padding at the left side of the component
- Bottom:** Creates padding at the bottom of the component
- Right:** Creates padding at the right side of the component
- Horizontal:** Creates padding horizontally

(g) Vertical: Creates padding vertically

- Padding Size: The padding size of the component can be set to None, XXX-Small, XX-Small, X-Small, Small, Medium, Large, X-Large, XX-Large
- Margin Location: Defines the position of the margin for a component. Margin creates extra space around a component.

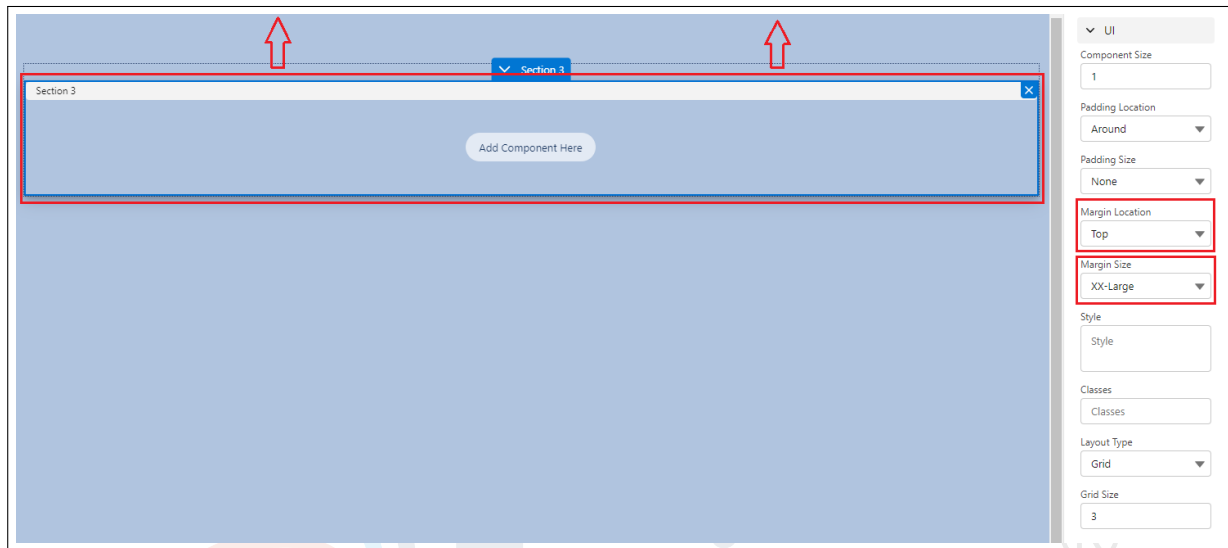


Figure 5.3.20: Margin

Type of Margin Locations :

- * Around: Margin gets added around the component
 - * Top: Margin gets added at the top of the component
 - * Left: Margin gets added at the left side of the component
 - * Bottom: Margin gets added at the bottom of the component
 - * Right: Margin gets added at the right side of the component
 - * Horizontal: Margin gets added horizontally
 - * Vertical: Margin gets added vertically
- Margin Size: The margin size of the view can be set to None, XXX-Small, XX-Small, X-Small, Small, Medium, Large, X-Large, XX-Large.
 - Style: The style attribute is used to add styles to a Component, such as color, font, size, and more. Setting the style of a Component can be done with the style attribute. The style attribute has the following syntax: “property: value” The property is a CSS property. The value is a CSS value.

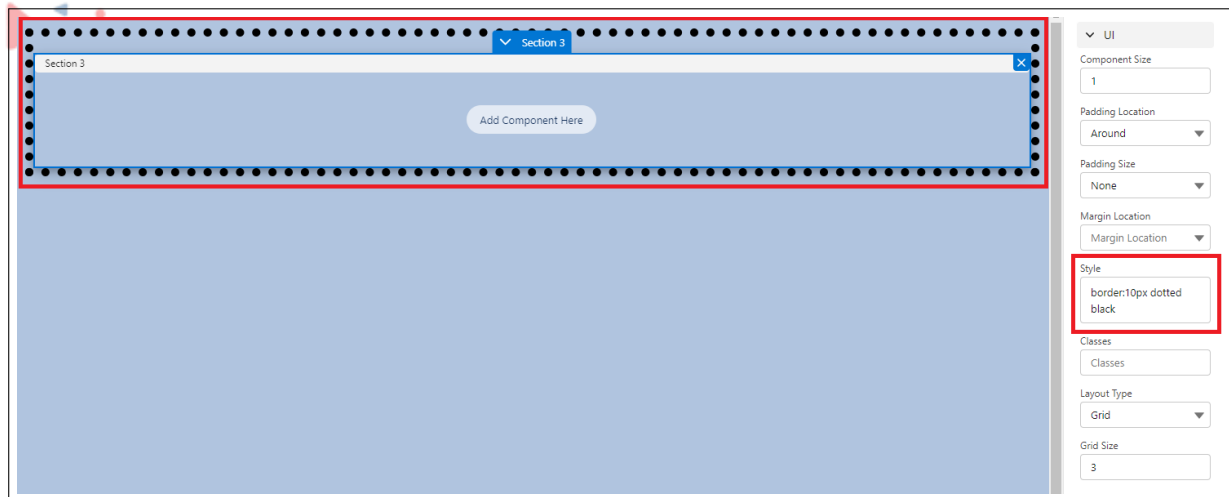


Figure 5.3.21: Style

- Layout Type: We can set the layout type in two ways as given below:
- Grid: The Grid Layout type offers a grid-based layout system, with rows and columns, making it easier to design a Layout.

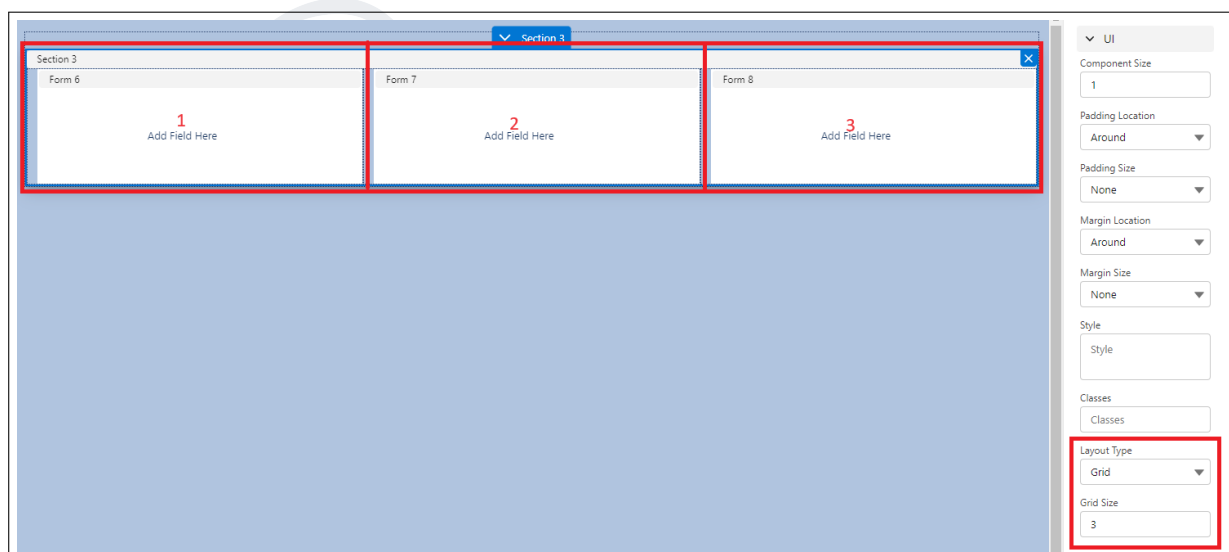


Figure 5.3.22: Grid

- Float: The Float layout type is used for positioning and formatting components. The Float property can have one of the following values:
 - * Fit to Content:

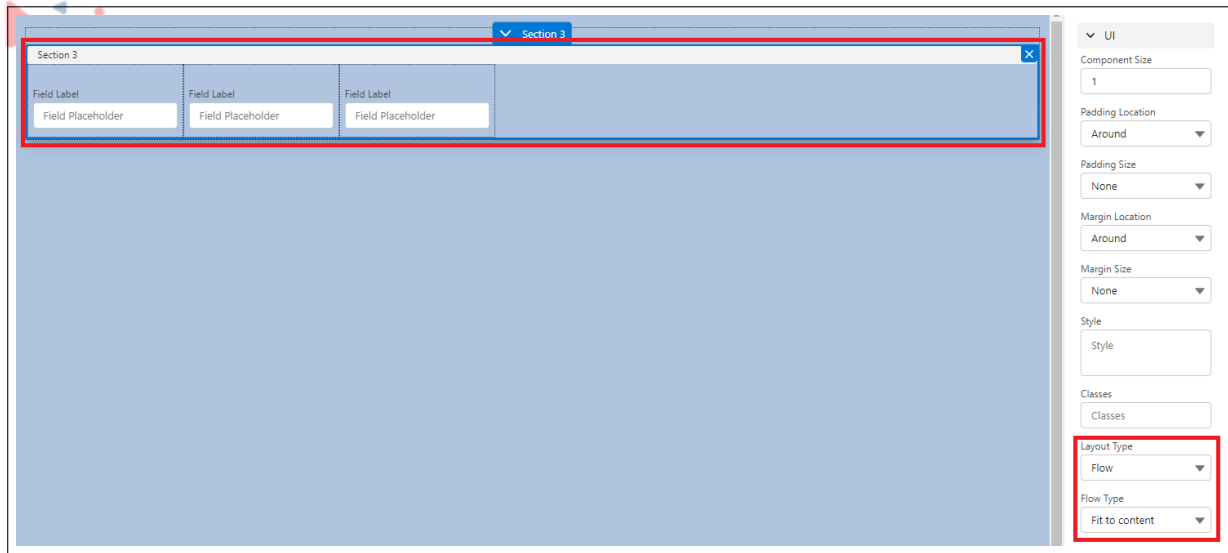


Figure 5.3.23: Fit to content

* Equally Distributed:

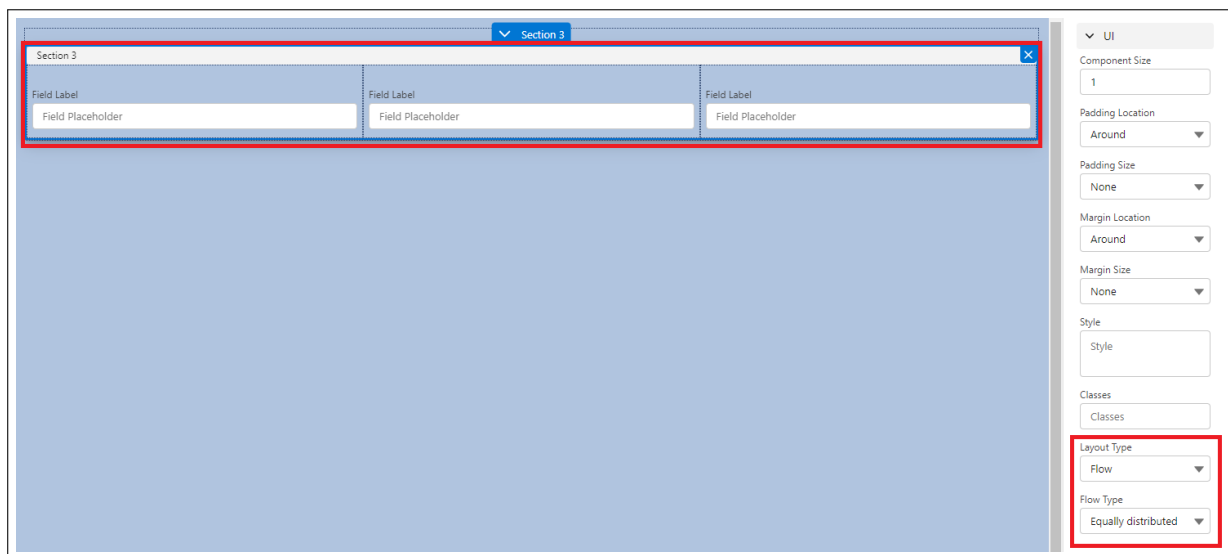


Figure 5.3.24: Equally Distributed

- Visibility: The visibility property specifies whether or not a component is visible on the layout or not. The following are the visibility types:
 - * Never: The field will not be visible at all
 - * Always: The field will be always visible
 - * Conditional: Depending on the visibility criteria, the component can be set as visible or not visible.
- Component: This field shows the name of the components with the count of its usage. e.g. if you are adding the section for the third time in a layout then it will display labeled as Section 3



2. **Tab:** Tab component is used as vertical tabs or horizontal tabs. When you add multiple Tabs, the Tab label is displayed horizontally on the top of it if it's a Tabs component and if it's vertical then the label is on the left-hand side. If you click on the label of the Tab, related tab content is displayed, If it's horizontal then on the bottom, and if vertical then on the right-hand side.

- **Attributes of Tab:**

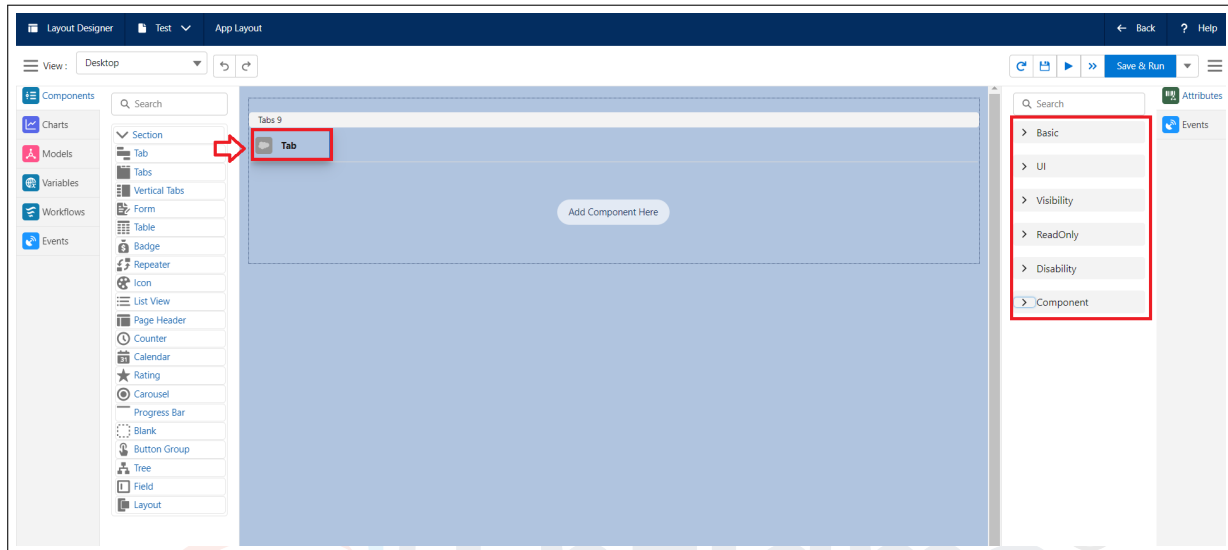


Figure 5.3.25: Attributes of Tab

- **Config:**

- **Label:** The label is a short description given to the Tab. Generally, the label is displayed in the component at the top left corner. e.g. For the below screenshot Personal Details is used to specify the first tab label and Educational Summary is used to specify the second tab label.

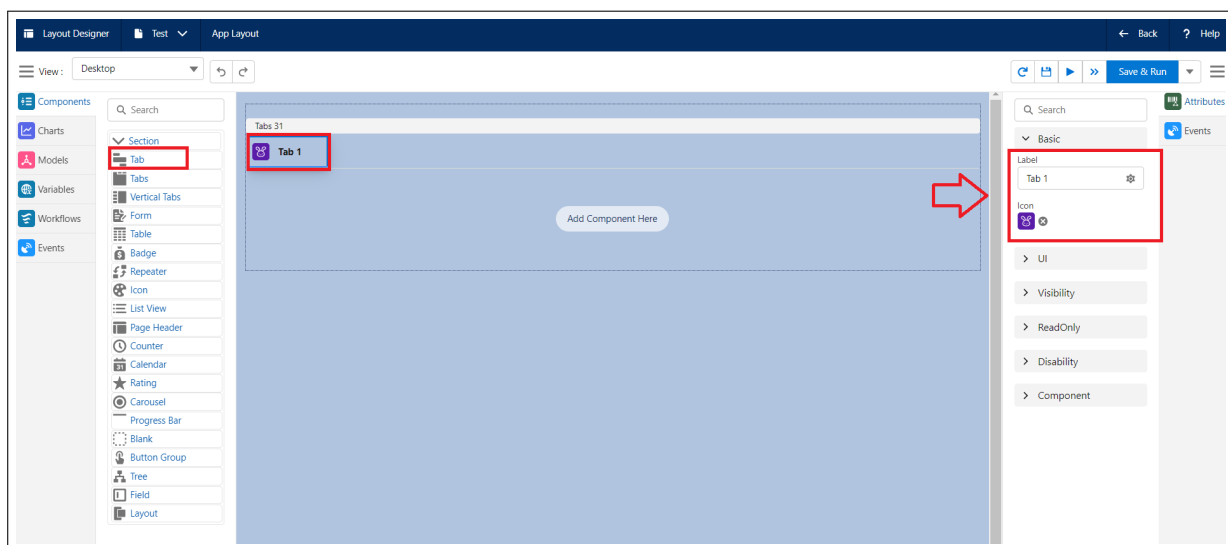


Figure 5.3.26: Tab Label



- Icon: The icon is a visual element that gives the idea of the context. Users can search and use icons manually through the icon box that they will display at the beginning of the label.
- **UI:** Same as defined for Section Component.
 - Read Only: This property specifies whether or not a component is ReadOnly and the following are the ReadOnly conditions:
 - * Never: The field will never be in Read Only I.e always editable
 - * Always: The field will be always read-only
 - * Conditional: Depending on the read-only criteria, the field can be set as Read-only or not
 - Disability: This property specifies whether or not a component is Disable and the following are the disability conditions:
 - * Never: The field will never be disabled
 - * Always: The field will be always disabled
 - * Conditional: Depending on the disability criteria, the field can be set as disabled or not
 - Component: Same as defined for the section component

3. **Tabs:** Tabs are nothing but the Set of Tab. Tabs keep related content in a single container that is shown and hidden through navigation. It is a container and represents the list of a Tab used for grouping the related content in a single container. When you add multiple tabs, the tab label is displayed horizontally on the top of the Tabs component. If you click on the label of the Tab, related tab content is displayed on the bottom side. All components except the Navigation Node can be added to the Tabs.

• **Attributes of Tabs:**

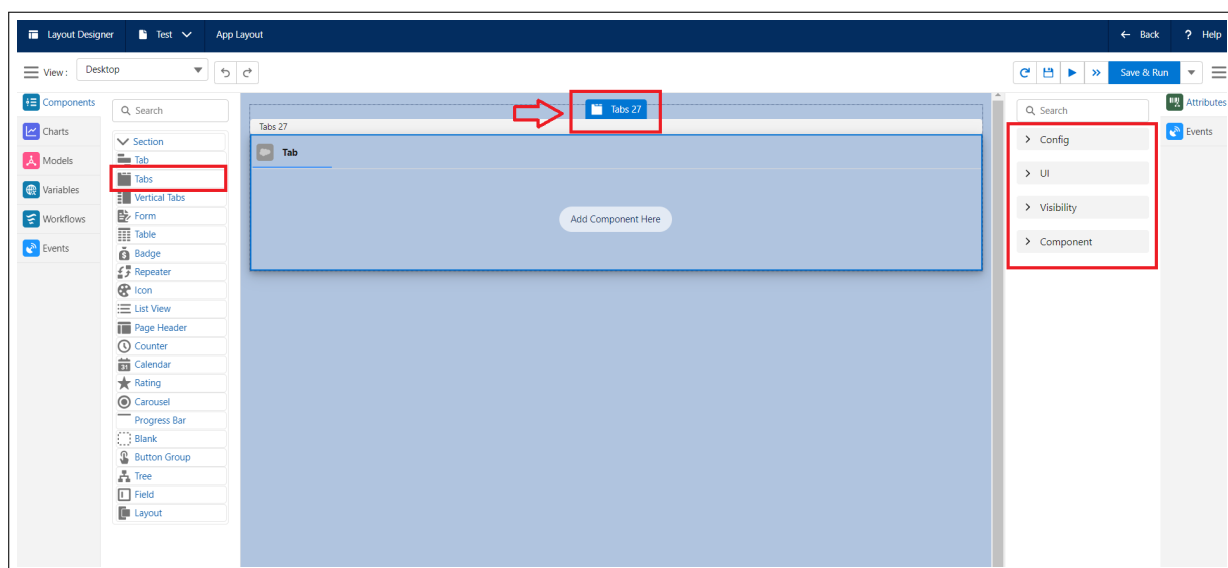


Figure 5.3.27: Attributes of Tabs

• **Config:**

- **Default Tab:** You can select a Tab that will be displayed as the Default Tab. When the layout is rendered the Default Tab will be displayed on the Tabs. e.g. You have three tabs Personal Details, Educational Details, and Address Details and you set a default tab 1 then you can land on the Educational Details tab when you open the form.
- **Overflow Limit:** Limit lets you decide the number of tabs you want to display. If the Tab limit exceeds the entered value, then the remaining tabs will appear in drop-down list format. If you enter the value for example 4 then it will display four tabs and if you added the above four tabs, then it will display the fifth tab in the drop-down list format.

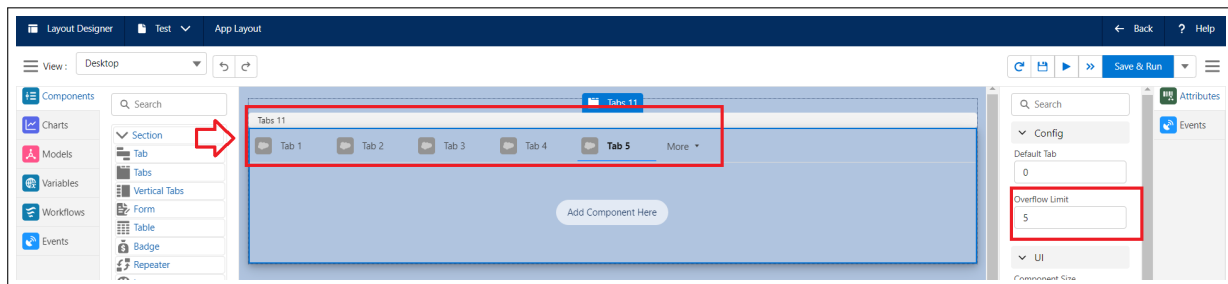


Figure 5.3.28: Tabs Overflow Limit

- **UI:**
Same as defined for the Section component.
- **Visibility:**
Same as defined for the Section component.
- **Component:**
Same as defined for the Section component.



4. **Verticals Tabs:** The Vertical Tabs is a container. The Vertical Tabs component is used for grouping the related content in a single container. Tabs can be added vertically and generally label displays on the left-hand side. If you click on the label of the tab, the related tab's contents are displayed on the right-hand side. All components except the Navigation Node can be added to the Vertical Tab.

• **Attributes of Vertical Tabs:**

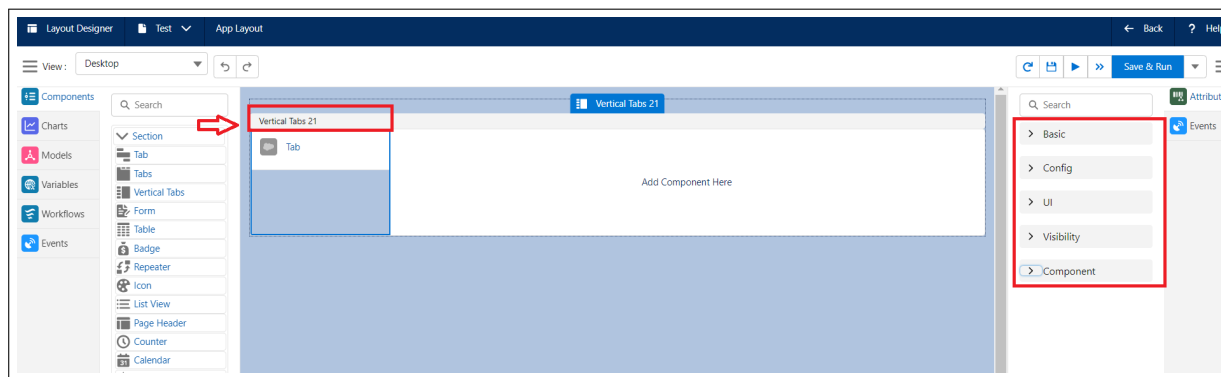


Figure 5.3.29: Attributes of Vertical Tabs

• **Basic:**

- **Show Tabs On Right:** Vertical Tabs are shown by default on the left side of the layout. By checking the checkbox, the tabs will be displayed on the right side.

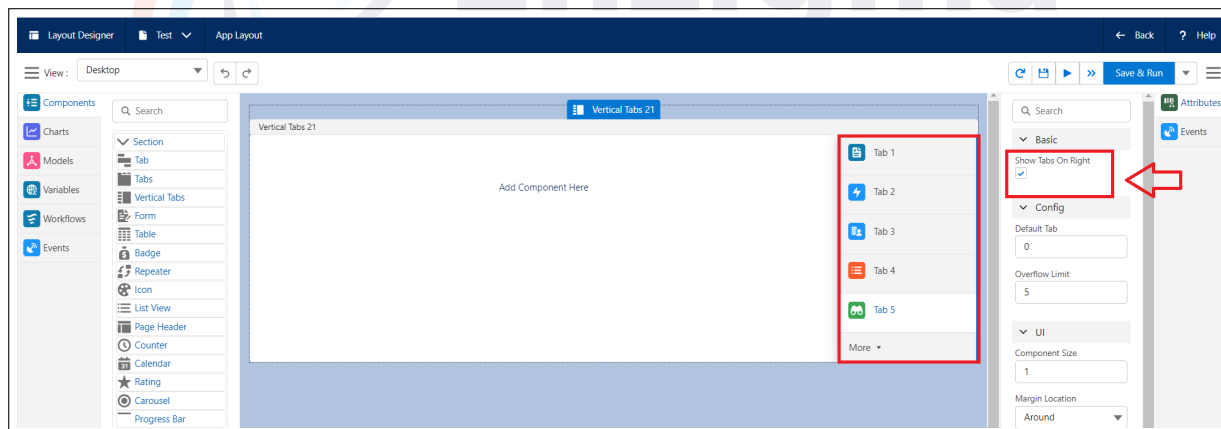


Figure 5.3.30: Show Tabs on Right

• **Config:**

- **Default Tab:** You can select a Tab that will be displayed as the Default Tab. When the layout is rendered the Default Tab will be displayed on the Tabs. e.g. If You have three tabs Personal Details, Contact Details, and Educational Details and you set Contact Details as the default tab then you will land on the Contact Details tab when you open the form.

• **UI:**

Same as defined for the Section component earlier.

- **Visibility:**
Same as defined for the Section component earlier.
- **Component:**
Same as defined for the Section component earlier

5. **Form:** The Form serves as a container for capturing user input. To enable the Form to function, you must associate the model with it. Once the model is linked, the fields corresponding to the selected object during model creation will appear on the left side. You have the flexibility to drag and drop these fields into the Form. All the fields within it will be visible on the left side. Multiple actions can be added to the Form, presented as buttons. You have the option to designate one action as the primary action.

- **Attributes of Form:**

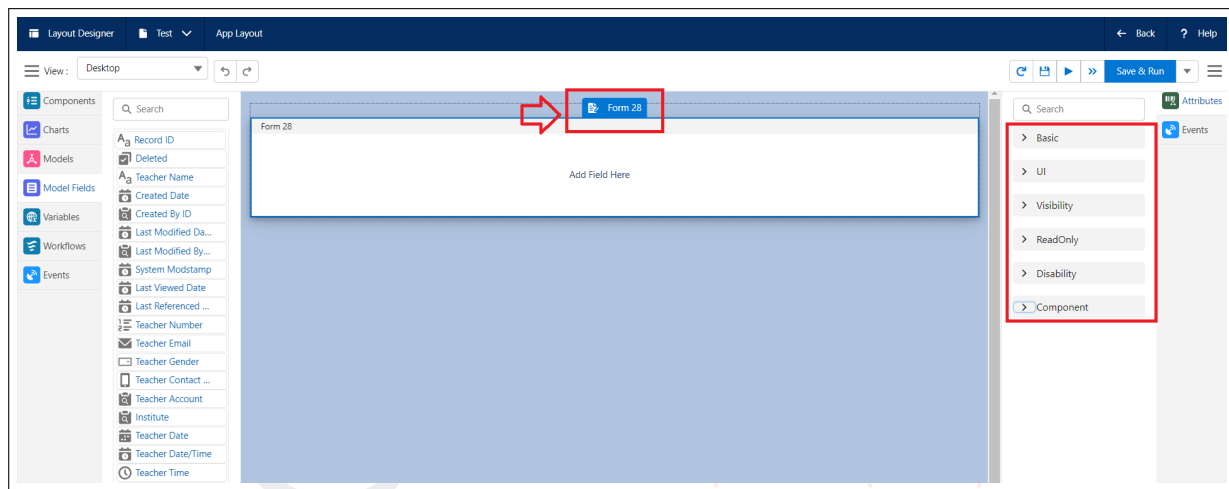


Figure 5.3.31: Attributes of Form

- **Basic:**
 - **Model:** All models you have created will appear in the list. You can select a specific model from the list to save and display the field value. Only one model(Record Type - Single) can be bound at a time.

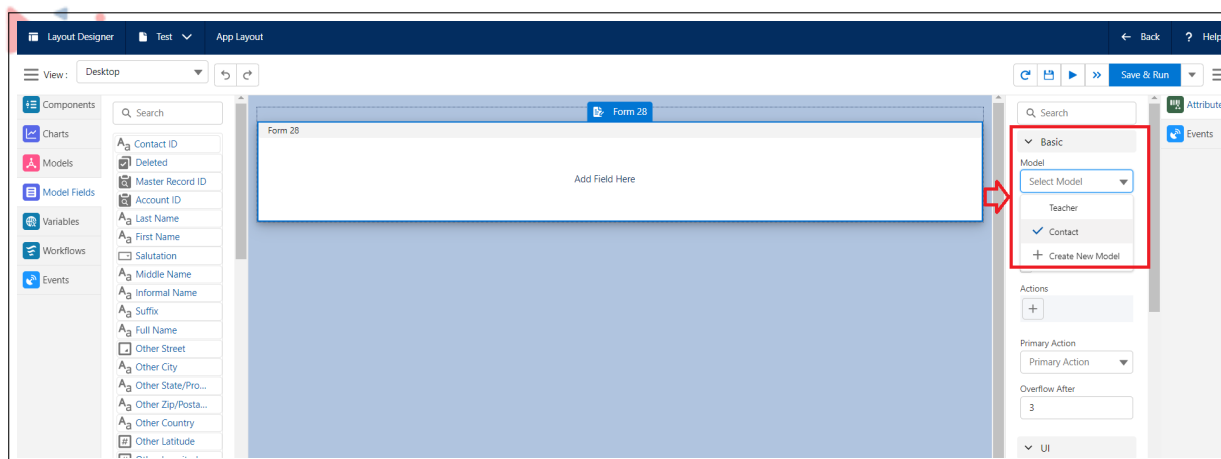


Figure 5.3.32: Model

- Field Layout: Field Layout enables you to arrange the field on the form.

Types of layout:

- * Stacked: In a stacked layout, the input/output field is positioned beneath the field label with a slight margin around the label.
- * Horizontal: In a horizontal arrangement, the input/output field is positioned ahead of the field label with a slight margin surrounding the label.

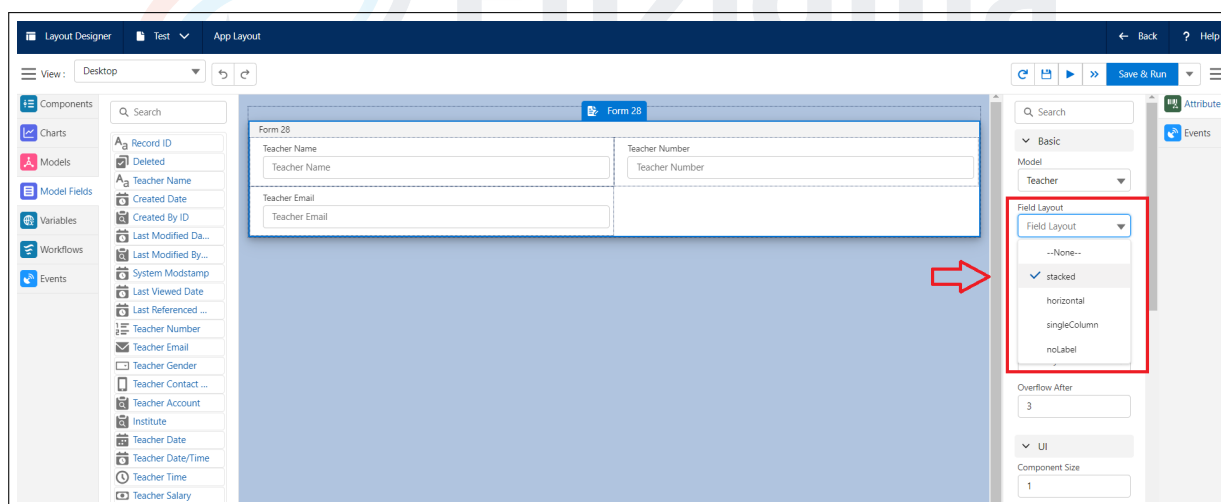


Figure 5.3.33: Field Layout

- * Single Column: In a Single Column layout, the input/output field size is larger than the field label, distinguishing it from a horizontal layout.
- * No Label: In the No Label layout, only the input/output field is visible without displaying the label name.
- No Action Bar: If the "No Action Bar" checkbox is selected, the action bar will not appear on the form.

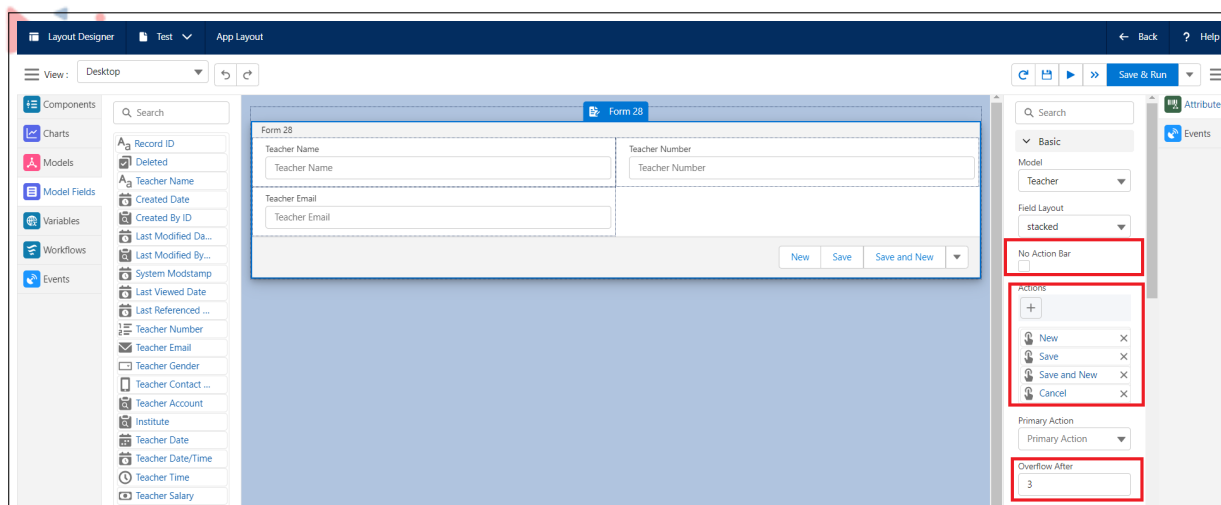


Figure 5.3.34: Basic

- **Overflow After:** Same as defined for the Section component earlier
- **Actions:** You can include multiple actions, with each action appearing as a button. When you click on a button, it initiates a specific action or workflow. The form will show only three actions, while the remaining actions will be accessible from a drop-down menu.
- ★ **Primary Action:** You can choose a primary action for the form from the list of actions available. Pressing the Enter key will execute the Primary Action.

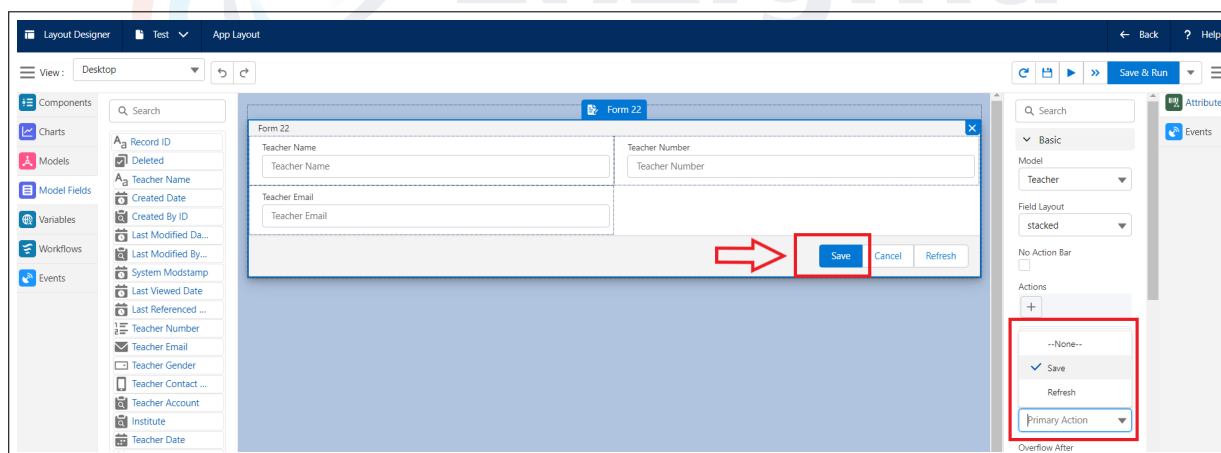


Figure 5.3.35: Primary Action

- **UI:**
Same as defined for the Section component earlier.
- **Visibility:**
Same as defined for the Section component earlier.
- **ReadOnly:**
This property specifies whether or not a component is ReadOnly and the following are the ReadOnly conditions:

- Never: The field cannot be read-only
- Always: The field will be always read-only
- Conditional: Depending on the read-only criteria, the field can be set as read-only or not
- **Disability:** This property specifies whether or not a component is Disable and the following are the disability conditions:
 - Never: The field cannot be disabled
 - Always: The field will be always disabled
 - Conditional: Depending on the disability criteria, the field can be set as disabled or not
- **Component:**
Same as defined for the Section component earlier.

6. **Table:** Table is used to show multiple records of objects to the user and is also able to perform multiple operations on that. For the Table component, we bind the Multiple Record Count Model. Below are the table attributes.

- **Attributes of Table:**

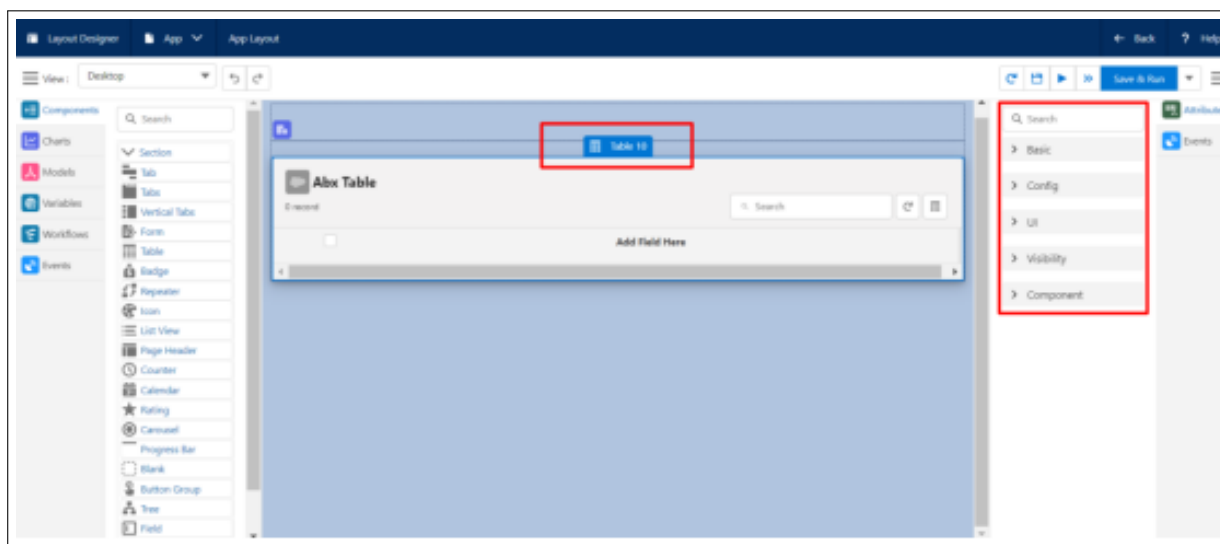


Figure 5.3.36: Attributes of Table

- **Basic:**
 - Model: Models are used to integrate the Objects of Salesforce with noKodr, it works as a mediator between the Salesforce and noKodr app. All models you have created will appear in the list. You can select a specific model from the list to save and display the field value. Only one model(Record Type - Multiple) can be bounded at a time.
 - Hide Selection: By enabling the checkbox we can hide the selection box in front of records in the table.
 - Show Index: By enabling the checkbox we can show the index value in front of records in the table.

- Hide Sorting: By enabling the checkbox we can hide the sorting of records in ascending or descending order.
- Hide Re-sizeable: By enabling the checkbox we can hide re-sizing the width of columns in the table.
- Hide Search: By enabling the checkbox we can hide the search box on the table.
- Hide Page Size: By enabling the checkbox we can hide the page size drop-down.
- No Header: By enabling the checkbox we can hide the header part of the table.
- No Footer: By enabling the checkbox we can hide the footer part of the table.
- Page Size Options: By using page size options we can set multiple page size options for the use.
- Table Actions: Table Actions are the actions that are given at the top right corner of the Table that perform table-specific actions like
 - * New: With the help of New action you can create a New Record.
 - * Refresh: With the help of Refresh action we can refresh the table data.

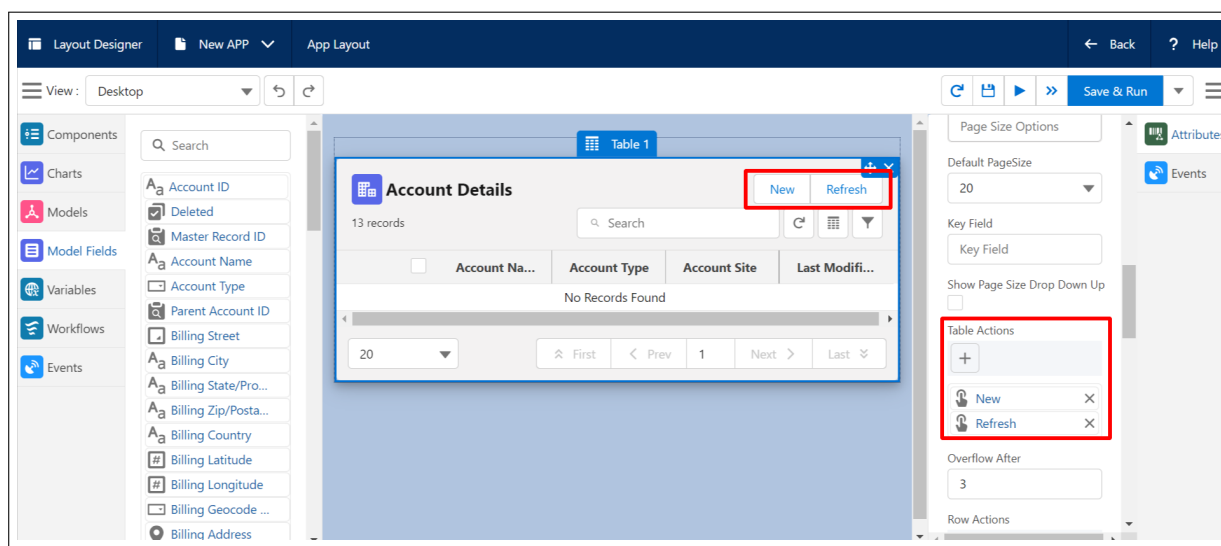


Figure 5.3.37: Table Actions

- Row Actions: With the help of Row Actions, you can perform row-specific actions like
 - * Edit: With the help of the edit action, you can edit the record.
 - * Delete: With the help of the delete action you can delete that specific record.

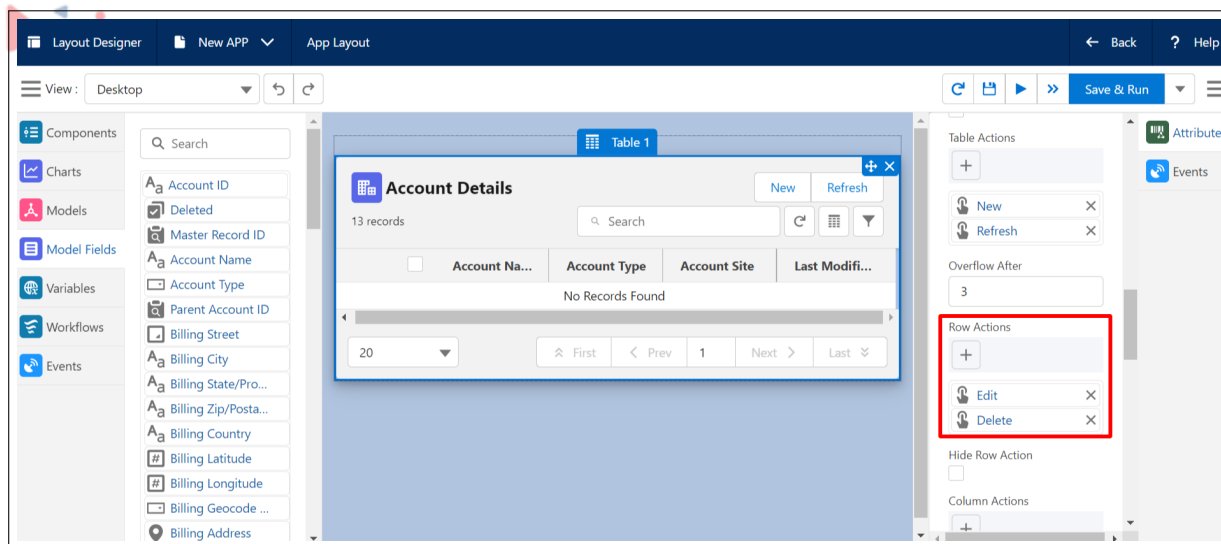


Figure 5.3.38: Row Actions

- Column Actions: With the help of Column Actions, you can perform column-specific actions like
 - * Push Modal: With the help of the Push modal, on clicking the specified column record, we can push a different modal.
 - * Toaster: With the help of toaster action, we can display a toaster by clicking the specified column record.

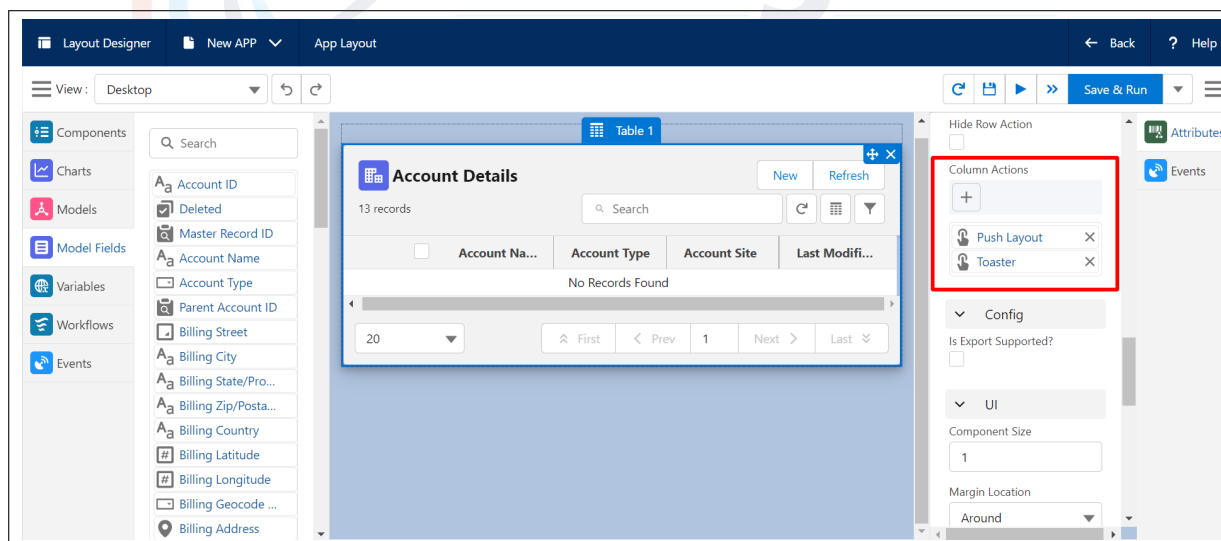


Figure 5.3.39: Column Actions

- **Config:**
 - Is Export Supported?: By enabling the checkbox we can export the records from the table in CSV, Excel, or PDF format.
- **UI:**

Same as defined for the Section component earlier.

- **Visibility:**

The visibility property specifies whether or not a component is visible and the following are the visibility types:

- Never: The field cannot be visible
- Always: The field will be always visible
- Conditional: Depending on the visibility criteria, the component can be set as visible or not

- **Component:**

This field shows the name of the components with the count of their usage. e.g.: if you are adding the section for the third time in a layout then it will display with the label Section 3

7. **Badge:** Bits of information are contained in colorful text components called badges. It's employed to emphasize information and classify stuff. Unread notifications or the labeling of a text block can both be done using a badge. Because a badge cannot contain a hyperlink, it is ineffective for navigation.

- **Attributes of Badge:**

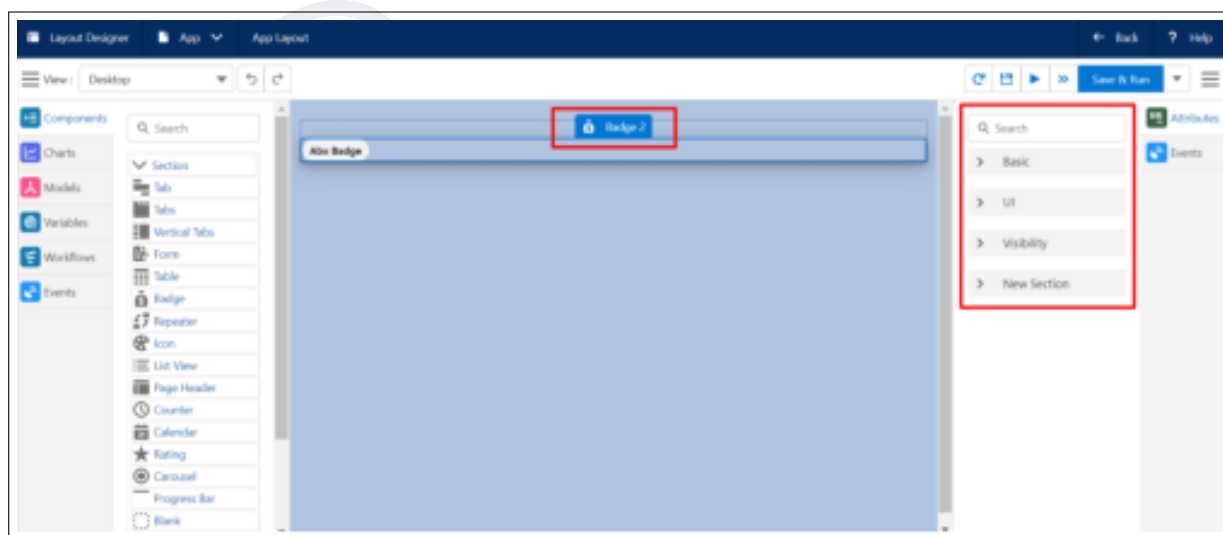


Figure 5.3.40: Attributes of Badge

- **Basic:**

- Variant: Used to display the badge in different colors. Below are the different types of variants available:
 - * Default: Default is the option used to display the default color i.e. white for the badge.
 - * Inverse: Inverse is a variant of Badge, which is displayed in Grey color.
 - * Lightest: Lightest is a variant of Badge, which is displayed in White color.
 - * Success: Success is a variant of Badge, which is displayed in Green color.
 - * Warning: Warning is a variant of Badge, which is displayed in Orange color.

- * Error: Error is a variant of Badge, which is displayed in Red color.
- Left Icon: The user can select the icon from the icon list and place that icon on the left side of the Badge.
- Right Icon: The user can select the icon from the icon list and place it on the right side of the Badge.

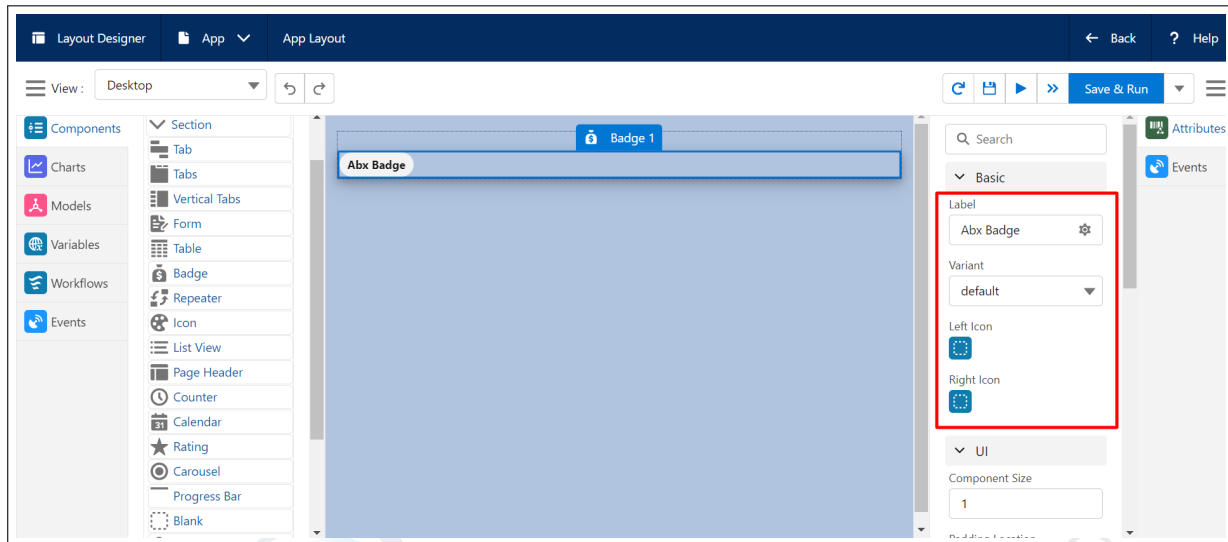


Figure 5.3.41: Basic

- **UI:**
Same as the Section component described earlier.
- **Visibility:**
The visibility property specifies whether or not a component is visible and the following are the visibility types:
 - Never: The field cannot be visible
 - Always: The field will be always visible
 - Conditional: Depending on the visibility criteria, the component can be set as visible or not
- **Component:** This field shows the name of the components with the count of their usage. e.g.: if you are adding the section for the third time in a layout then it will display with the label Section 3

8. **Repeater:** The Repeater is a container where you can add various components. You have the flexibility to determine the appearance of the user interface (UI). The Repeater will replicate the components multiple times based on the number of records in the model.

- **Attributes of Repeater:**

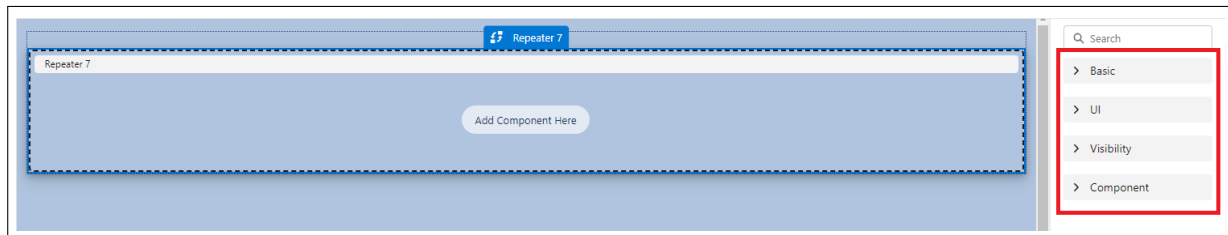


Figure 5.3.42: Attributes of Repeater

- **Basic:**

- Model: The Repeater needed multiple records, so it needed a multi-record model.
 - * Creation of a New Model: You can select a model from the list or you can create a new model by clicking on the "Create New Model" option for your Repeater.

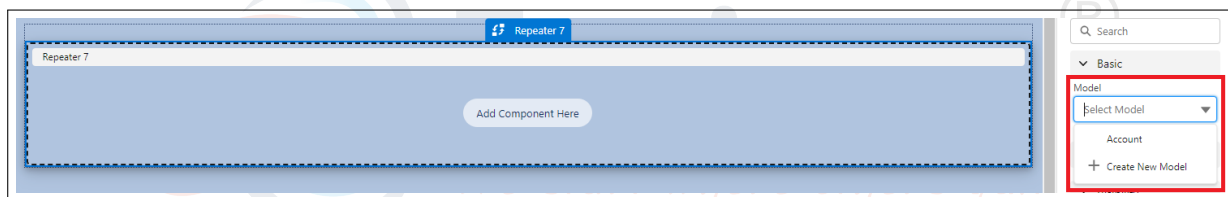


Figure 5.3.43: Model

- * Context Models: An auto-generated Context Model is created using a multi-record model.

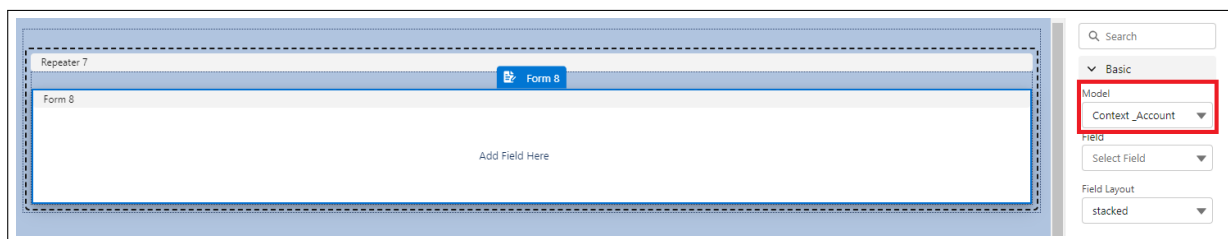


Figure 5.3.44: Context Model

- **UI:**
Same as the section component explained earlier.
- **Visibility:**
The visibility property specifies whether or not a component is visible and the following are the visibility types:
 - Never: The field cannot be visible

- Always: The field will be always visible
- Conditional: Depending on the visibility criteria, the component can be set as visible or not
- **Component:** This field shows the name of the components with the count of their usage e.g.: if you are adding the section for a third time in a layout then it will display with the label of Section 3.

9. **Icon:** Represents a visual element that provides context and enhances usability. Users can easily find different categories of icons in the category section and can set only one Icon at a time.

• **Attributes of Icon:**

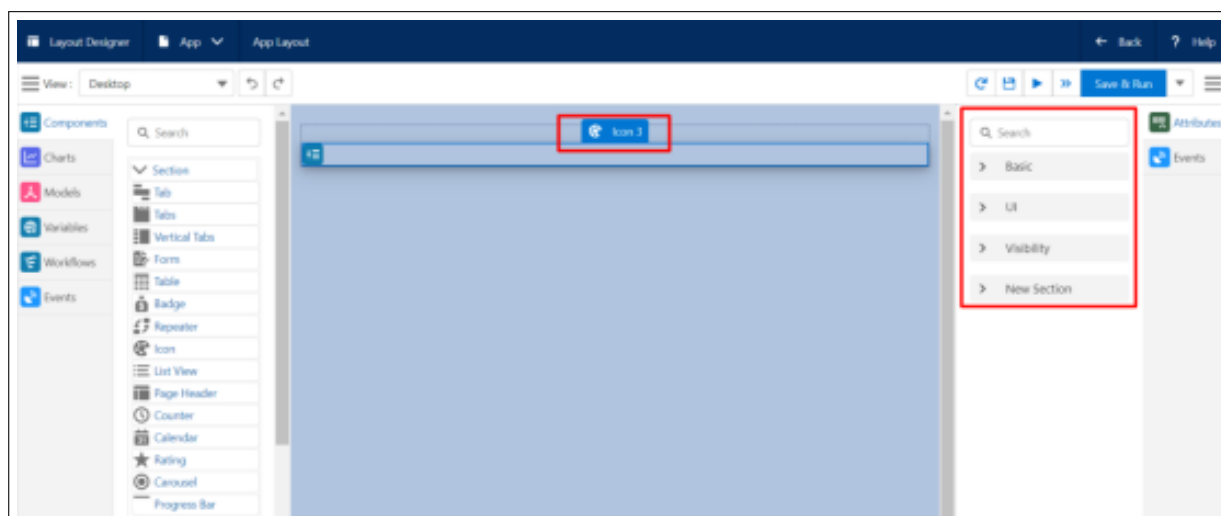


Figure 5.3.45: Attributes of Icon

- **Basic:**
 - Icon Picker: Used to pick the icons from different categories like utility, action, custom, doctype, and standard and also can set the size of the icon.
- **UI:**

Same as the Section component described earlier.
- **Visibility:**

The visibility property specifies whether or not a component is visible and the following are the visibility types:

 - Never: The field cannot be visible
 - Always: The field will be always visible
 - Conditional: Depending on the visibility criteria, the component can be set as visible or not
- **Component:** This field shows the name of the components with the count of its usage e.g.: if you are adding the section for the third time in a layout then it will display with the label Section 3

10. **List View:** The List View component displays records from various objects, enabling users to perform multiple operations. It presents records in a tabular format with rows and columns, offering features like pagination, search, sort, and data export. Each object can have multiple List Views, but each List View is associated with a single object.

- **Attributes of ListView:**

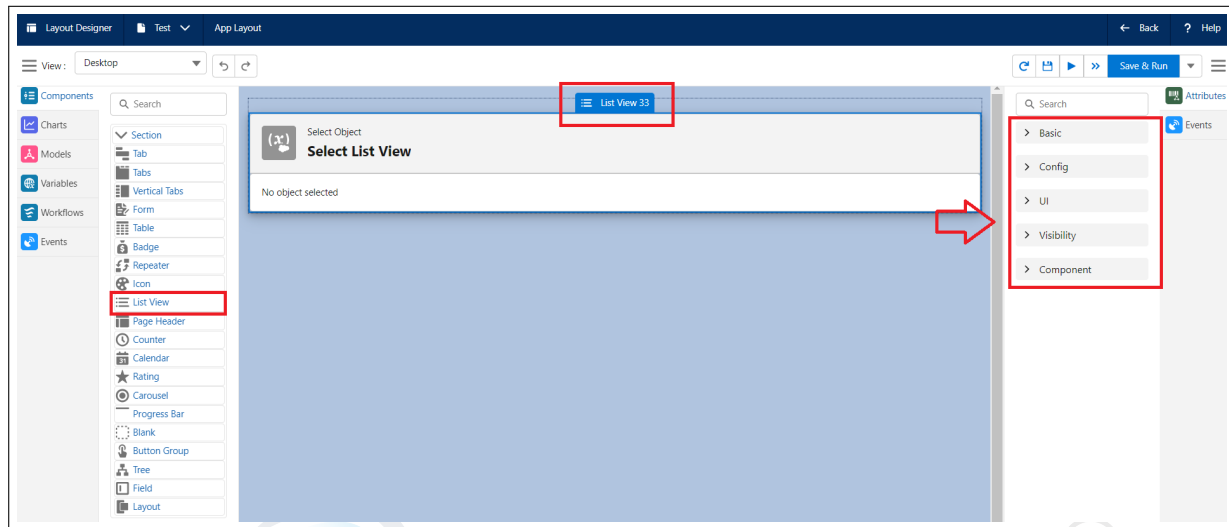


Figure 5.3.46: Attributes of ListView

- **Basic:**

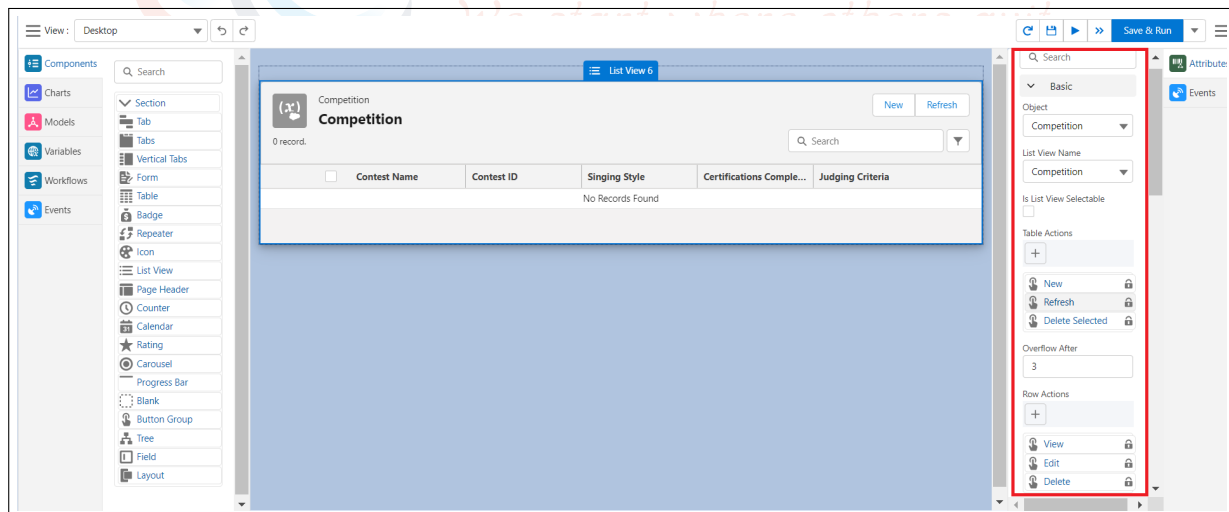


Figure 5.3.47: Basic

- **Object:** You are required to choose the object for displaying the List View. Only standard and custom objects will be available for selection here; metadata and system objects are excluded from the list.
- **List View Name:**
 - * One object can have multiple List Views

- * You can select only one List View at a time
- * You can select the List View name from the list
- * List View will be visible once you select an object and List View
- Is List View Selectable:
 - * This allows the end user to change the List View at runtime
 - * If selected it shows a drop-down icon next to the object selector
- Is Object Selectable:
 - * This allows the end user to change the object at runtime
 - * If selected it shows a drop-down icon next to the List View selector
- Table Actions:
 - * You can create multiple table actions on the List View
 - * You can only show two actions, which are visible in the top right corner, and the rest of the actions are displayed under a drop-down at the right
 - * These actions are mainly used to perform operations such as create, refresh, etc. on the selected records or unselected records
 - * Each action has a workflow that actually performs an operation
- Steps to Create Table Action
 - * You can add the table actions by clicking on the '+' icon
 - * Fill in all the details in the Create Table Action model and click the Save button

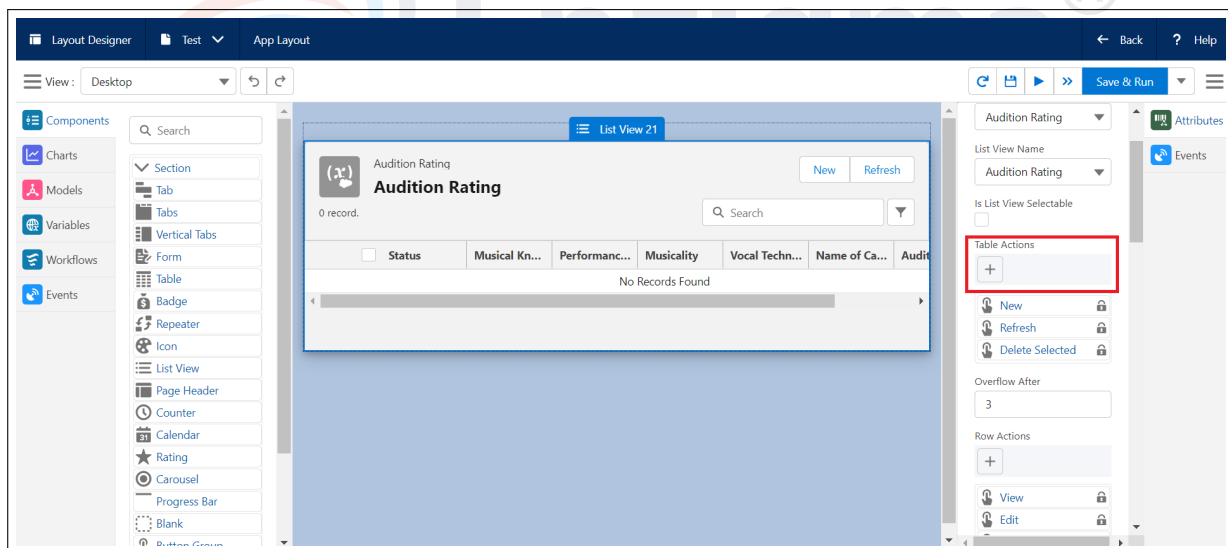


Figure 5.3.48: Table Actions

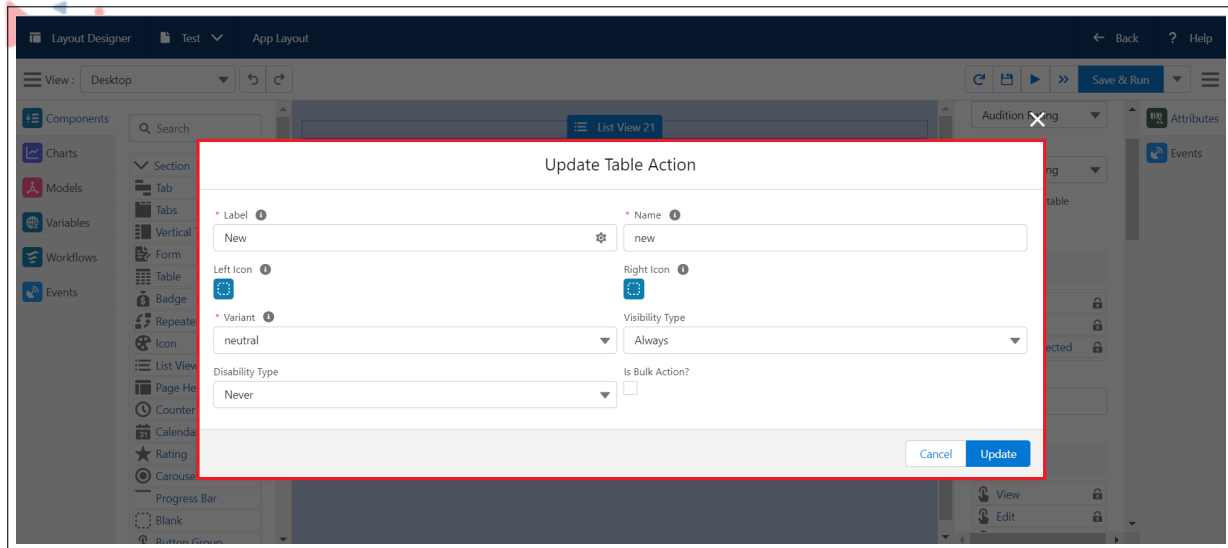


Figure 5.3.49: Update Table Actions

– Overflow After:

- * Overflow After an attribute is used to display the actions in list format after reaching its entered limit
- * By Default the value is 3 which means the three actions will displayed on a section header
- * If you added the new action despite having 3 actions then the new action will appear in the drop-down list section

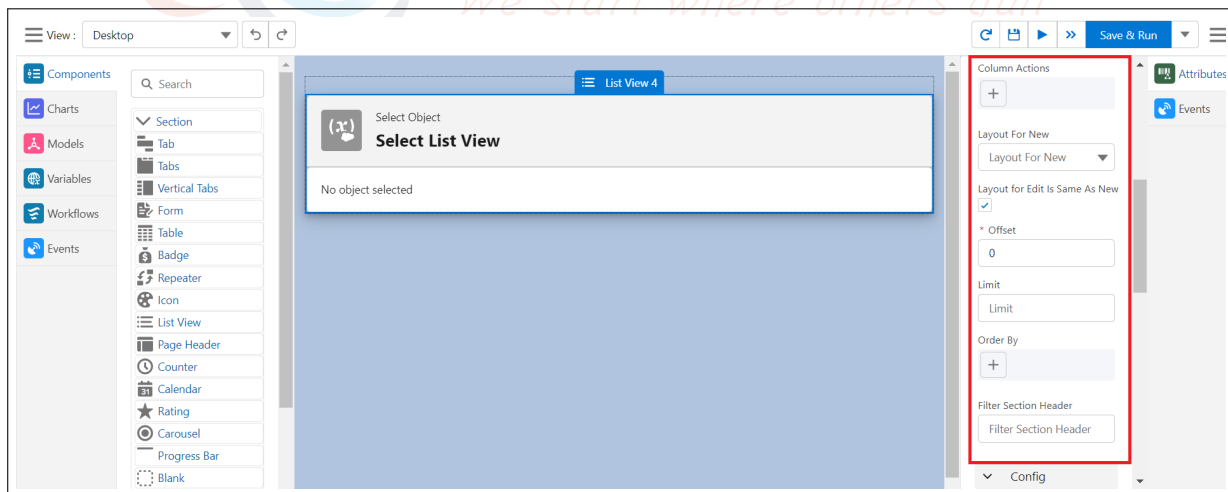


Figure 5.3.50: Basic

– Row Action:

- * Actions to be performed at the record level for the respective record
- * These actions are mainly used to read, update, or delete a single record
- * Each action has a workflow that actually performs an operation



- * Default row actions are edit, delete, and view, but you can create more actions as per your needs as shown below
- * Steps to create Row Action
- * You can add the row actions by clicking on the '+' icon

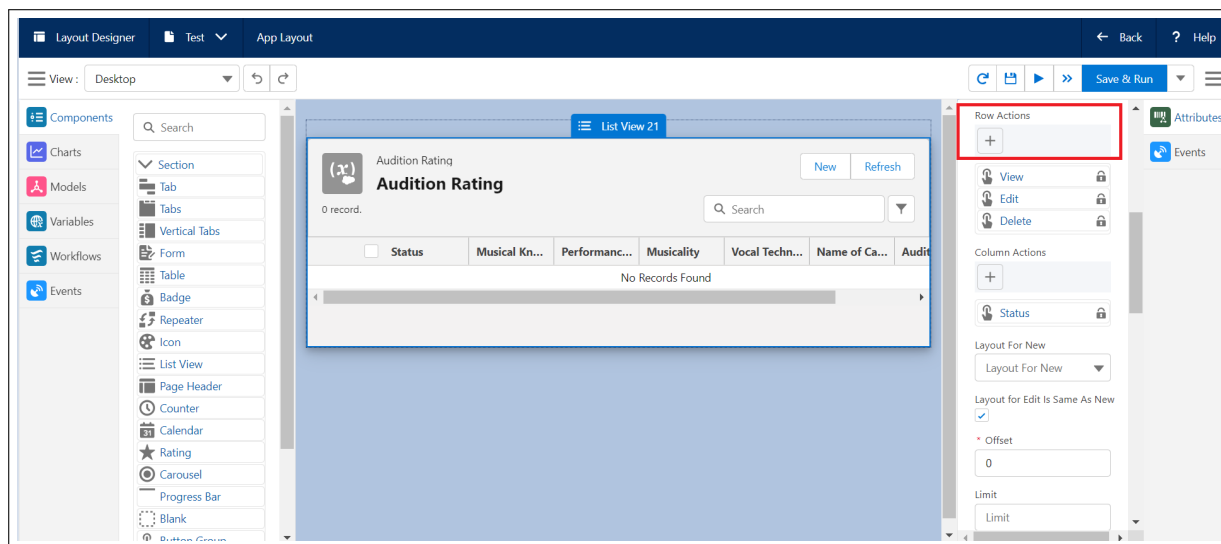


Figure 5.3.51: Row Actions

- * Fill in all the details in the Create Row Action model and click the Save button

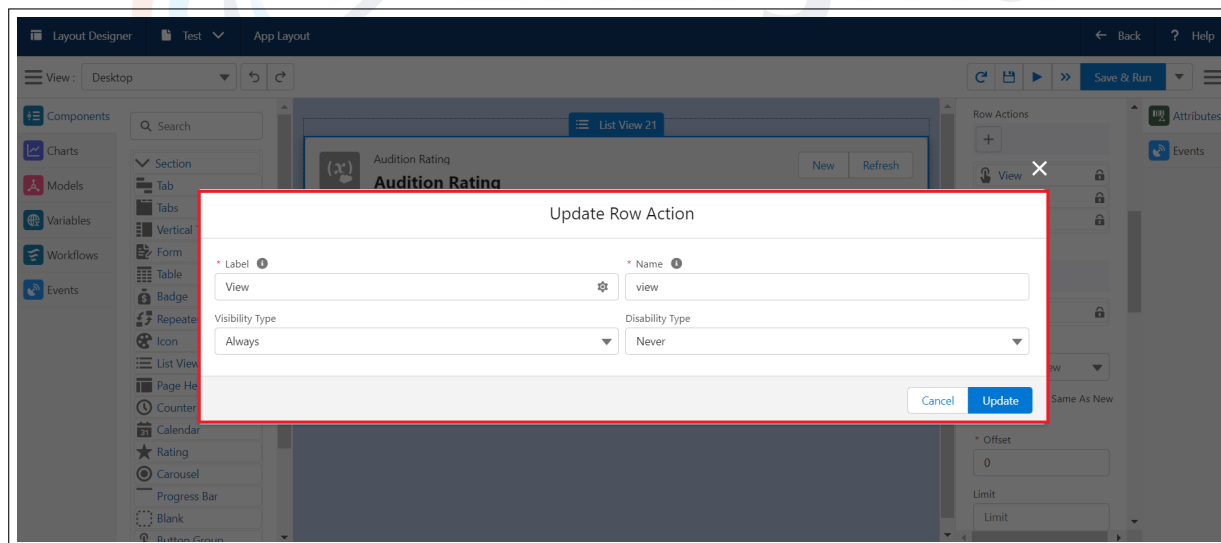


Figure 5.3.52: Update Row Action

– Column Action:

- * You can set the actions at the column level on the List View
- * You can assign only one action to any column
- * When you click record in the column action gets executed
- * Each action has a workflow that actually performs an operation

- * When you set column action, that record in the cell becomes a link
- * When the user clicks such a record, the action bound to that column gets executed
- * Steps to create Column Action
- * You can add the column actions by clicking on the '+' icon

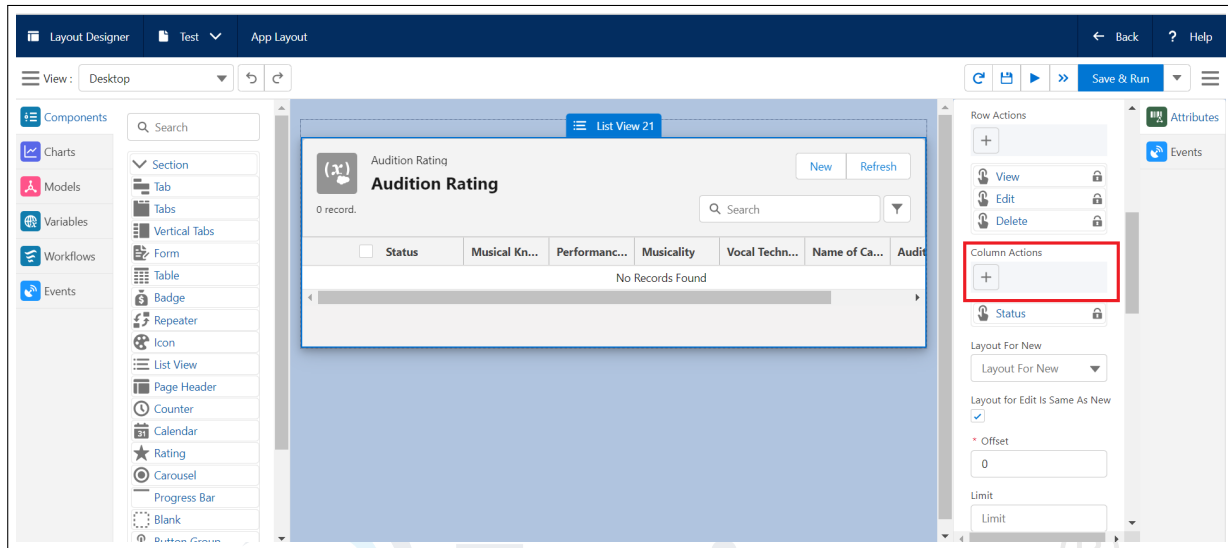


Figure 5.3.53: Column Actions

Fill in all the details in the Create Column Action model and click the Save button

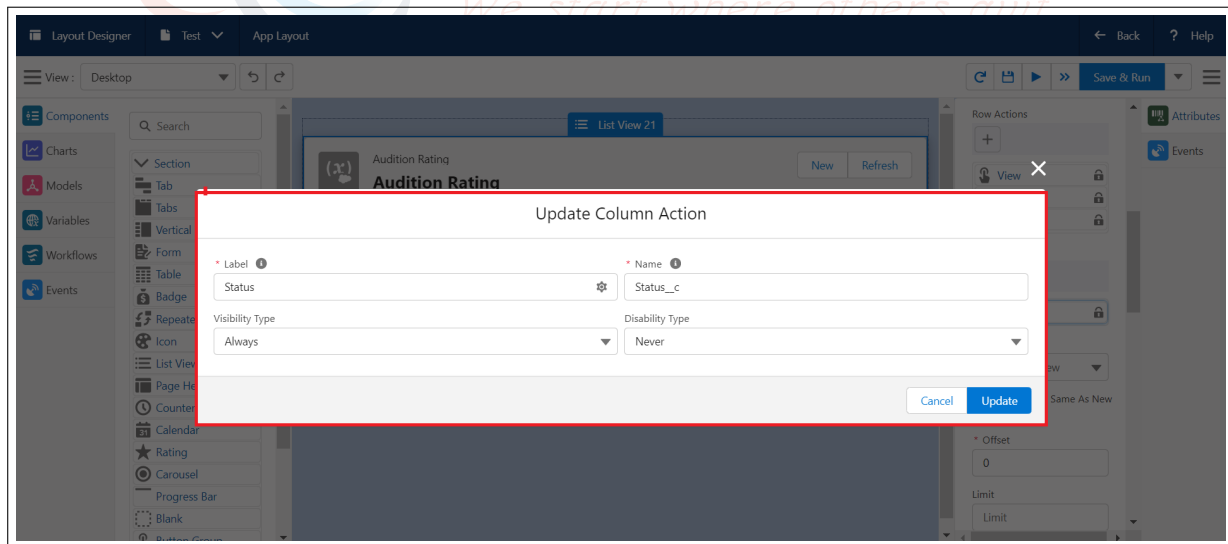


Figure 5.3.54: Update Column Action

- Layout for New :
 - * You can set the layout for table action New
 - * When you click the New button layout that you set will open



- Layout for Edit Is Same As New: If the "Layout for Edit Is Same As New" checkbox is enabled, the layout chosen from the "Layout For New" pick list will be used for both New and Edit operations. If disabled, users can select any layout available in the system for editing purposes
- Layout for Edit: You can select any layout from the drop-down for editing purposes
- Offset:
 - * You can set the offset for the query on the object
 - * If the offset is "n" then the query will take the records "n+1" onwards
 - * For e.g. if there are 200 records and you set offset 100 then it will show records from the 101st record on the List View
- Limit:
 - * Number of the records to be queried at once
 - * e.g. if there are 200 records of an object and you set a limit of 100 then it will query the first 100 records and show them on the List View
- Order By:
 - * You can set the order by on the fields to records to be queried and displayed in the List View
 - * You can order records in an ascending or descending manner

• **Config:**

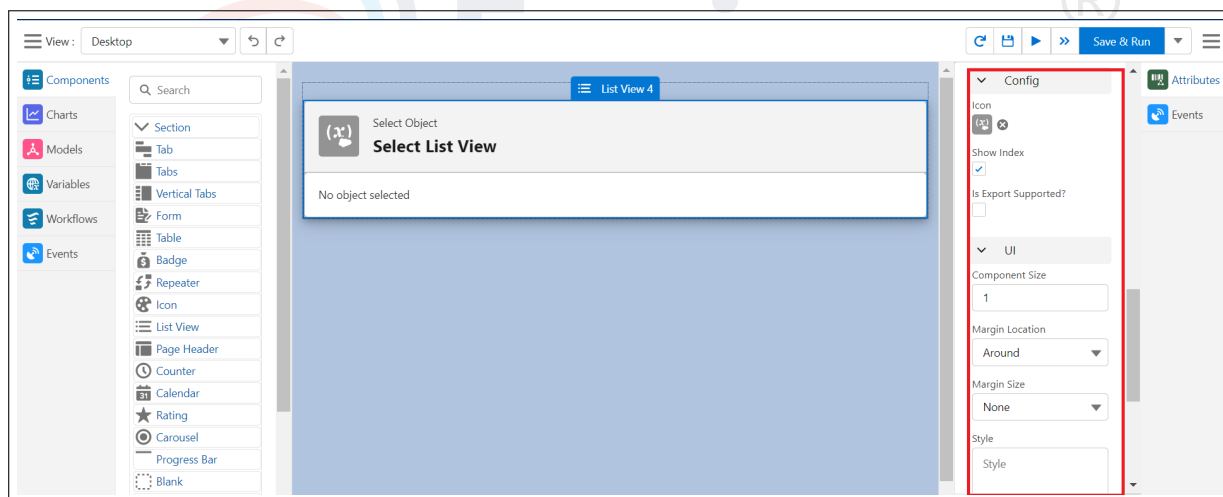


Figure 5.3.55: Config

- Icon Picker:
 - * You can select the icon to be displayed on the List View
 - * Generally icon displays on the left side of the header and subheader
 - * You can also remove the icon by clicking on "x" which appears on the icon
- Show Index: To show the index of records on the List View
- Is Export Supported?:
 - * If you check that checkbox then the download icon will appear on the right side of the search box

- * Upon clicking the download icon on the List View, all the records can be exported in a CSV file

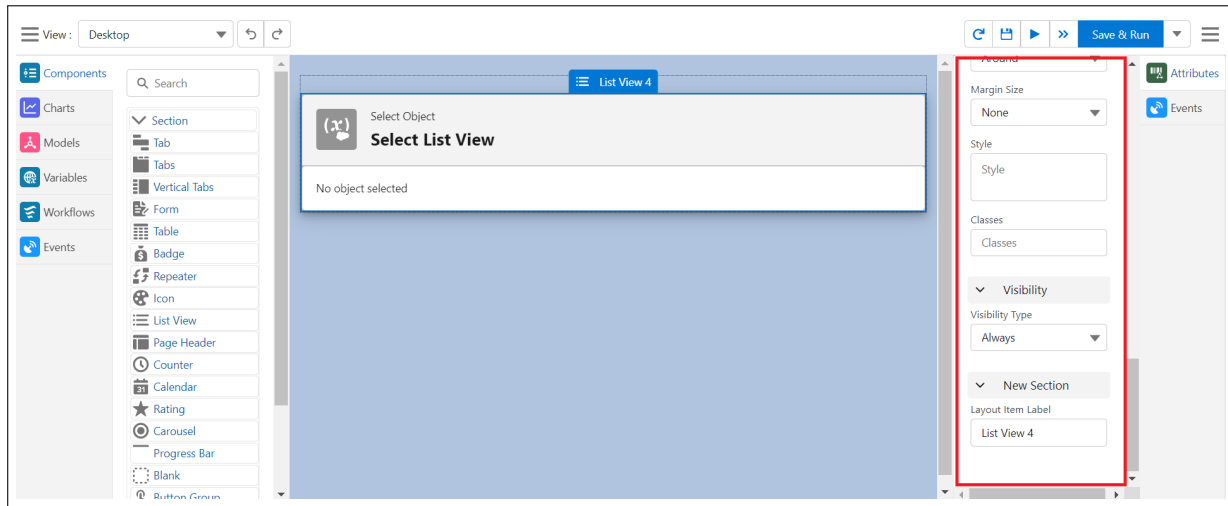


Figure 5.3.56: UI

- **UI:**
Same as the Section component described earlier.
- **Visibility:**
The visibility property specifies whether or not a component is visible and the following are the visibility types:
 - Never: The field cannot be visible
 - Always: The field will be always visible
 - Conditional: Depending on the visibility criteria, the component can be set as visible or not
- **Component:**
This field shows the name of the components with the count of their usage e.g.: if you are adding the List View for the third time in a layout then it will display with the label List View 3

11. **Page Header:** The Page Header defines the title of the component which will act as a container where you can drag and drop the other components in it. The Page Header name will be displayed in the top left corner of the layout.

• **Attributes of Page Header Component:**

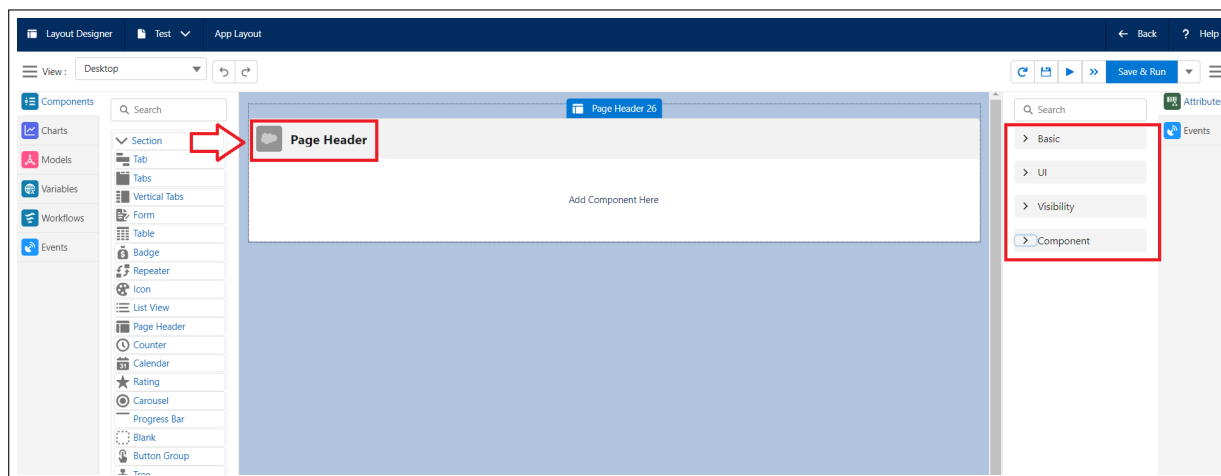


Figure 5.3.57: Attributes of Page Header

• **Basic:**

- Header: Name of the header which is displayed on the top left corner of the component
- Sub Header: Sub Header is displayed above the Header in the layout. The size of the Sub Header is smaller than the Header
- No Body: If you select this checkbox then there is no space for adding another component under the Page Header

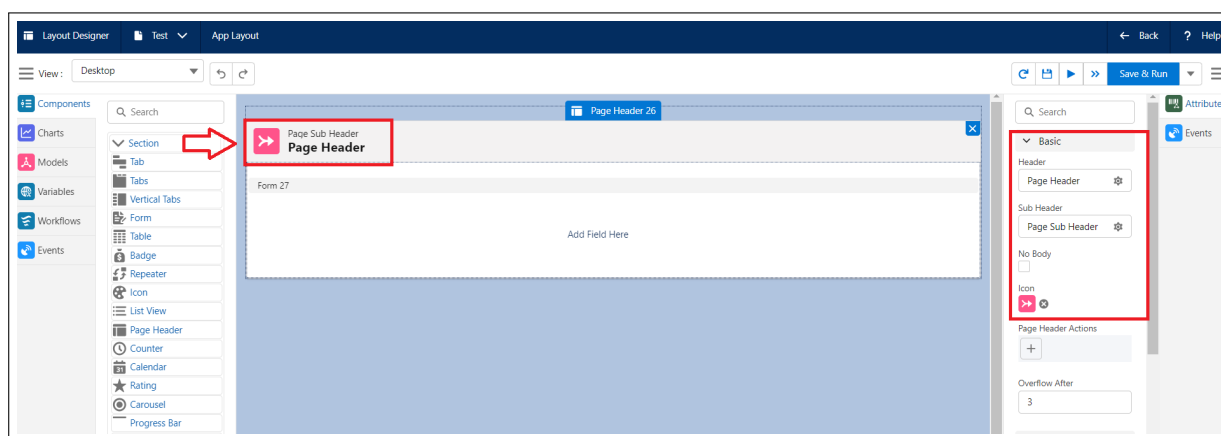


Figure 5.3.58: Basic

- Page Header Actions: Action is a set of operations that can be performed by clicking on an action button, e.g. Save, Cancel, etc. You can add as many actions as you want. each action has a workflow that actually performs an operation.

- **Overflow After:** The Overflow After attribute shows actions in list format once the limit is reached. By default, this limit is set to 3, indicating that three actions will be shown on a Page Header Component. If a new action is added when the limit of 3 actions has already been reached, the new action will be displayed in the drop-down list section.
- **UI:**
Same as the Section component explained earlier.
- **Visibility:**
The visibility property specifies whether or not a component is visible and the following are the visibility types:
 - **Never:** The field cannot be visible
 - **Always:** The field will be always visible
 - **Conditional:** Depending on the visibility criteria, the component can be set as visible or not
- **Component:** This field shows the name of the components with the count of their usage. e.g.: if you are adding the section for the third time in a layout then it will display with the label Section 3

12. **Counter:** Counter is a component used to count the time/duration. Counter displays duration in HH: MM: SS format. You can monitor events using a counter. counter allows users to increase or decrease a numerical value. In simple terms, you can set it as a stopwatch or a countdown timer. e.g. A stopwatch to record how fast one can type in a 100-word paragraph and A count-down timer to finish an exam in the stipulated time.

- **Attributes of Counter:** *We start where others quit*

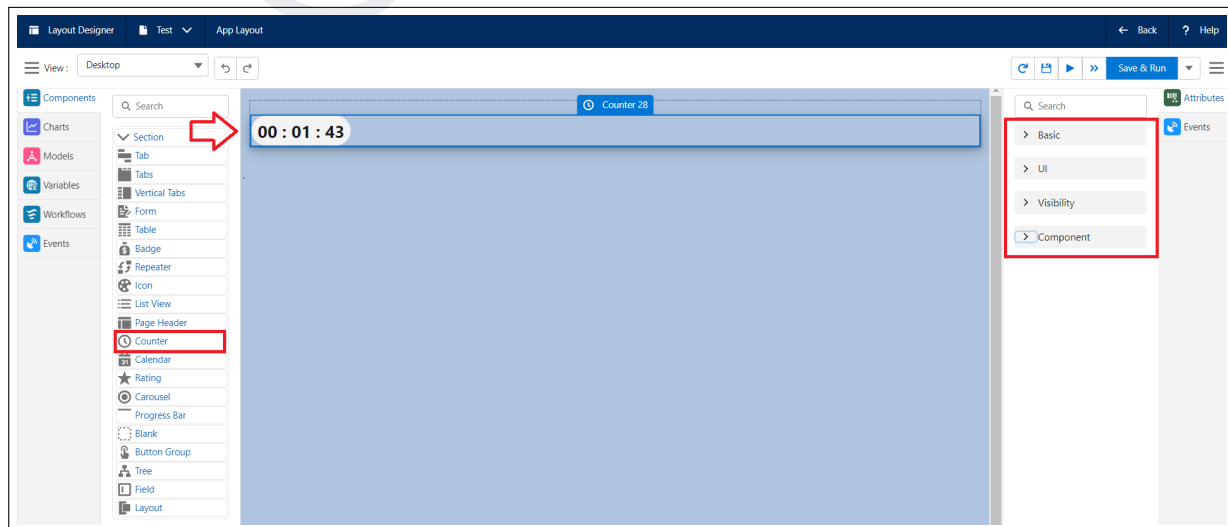


Figure 5.3.59: Attributes of Counter

- **Basic:**
 - **Time in Milliseconds:** You can give the total time in Milliseconds. Once that time is met the Counter will stop counting the time ahead

- Counter Order: You can set Counter orders i.e incrementing or decrementing
- Increasing: Counter will start with a 00 value and go on increasing up to the maximum time set by you
- Decreasing: Counter will start with the time value given by you in milliseconds in layout and goes on decreasing up to 00 value
- **UI:**
Same as the Section component described earlier
- **Visibility:** The visibility property specifies whether or not a component is visible and the following are the visibility types:
 - Never: The field cannot be visible
 - Always: The field will be always visible
 - Conditional: Depending on the visibility criteria, the component can be set as visible or not
- **Component:** This field shows the name of the components with the count of their usage. e.g.If you are adding the section for the third time in a layout then it will display with the label Section 3

13. **Calendar:** The Calendar component is visible in three views, i.e., Monthly, Weekly, and Daily, and by default, today's date is selected in the component. Users can save the records in the Calendar component.

- (a) The Calendar component is used for viewing all the events in the calendar format
- (b) This component can display data in Calendar format if the object bounded using the model to it has a date and date time field in it
- (c) The Calendar displays data in 3 formats i.e. monthly, weekly, and daily

• **Attributes of Calender:**

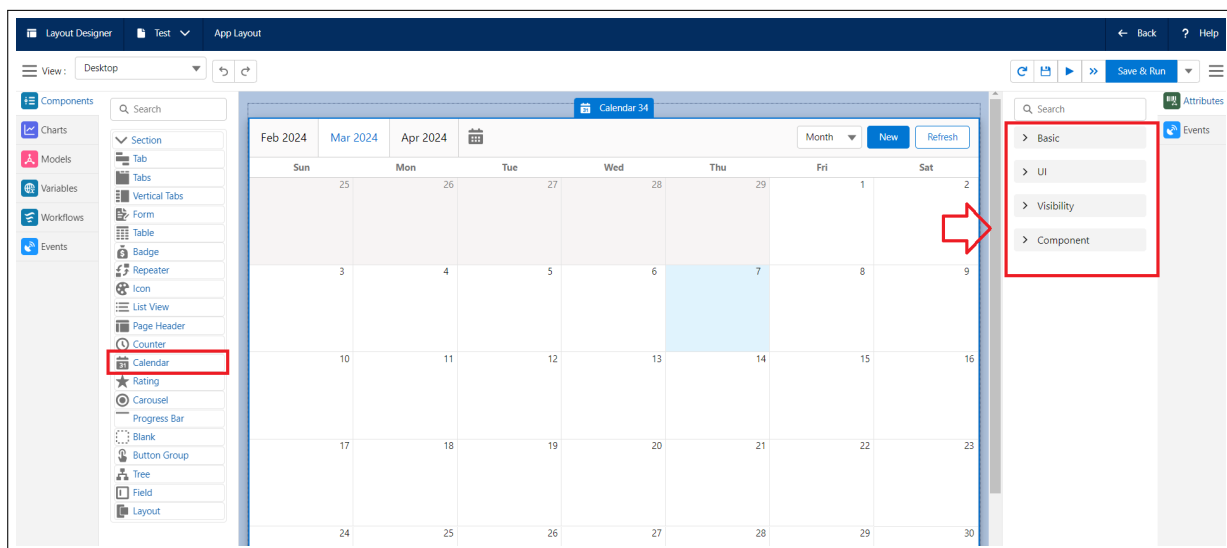


Figure 5.3.60: Attributes of Calender

- **Basic:**

- Models: Models are used to display content (fields) in the layout. mainly used for the creation or editing of a record. A model needs to be added when you perform any action in an event. For the Calendar component, you need to create a Multi Record Model, You can get help from Models for creating the models.

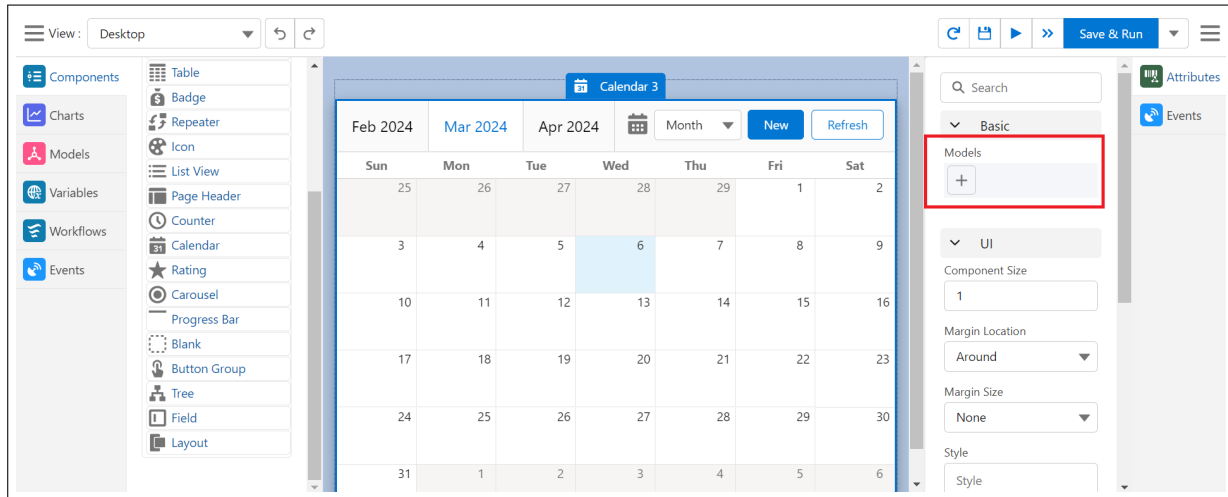


Figure 5.3.61: Models

- * Click the Calendar icon
- * To display the Date, Month, and Year in Calendar format
- * Click the Month drop-down
- * You can select Month, Week, and Day

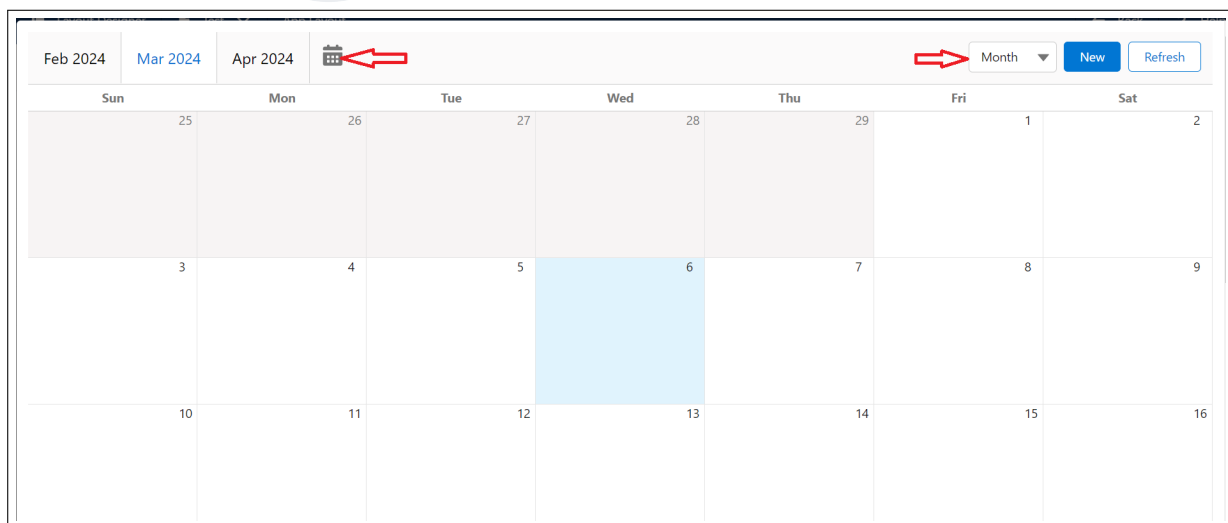


Figure 5.3.62: Calendar

- Workflows:
 - * You need to create workflows for the New and Refresh buttons of the calendar component

- * Workflow is basically a container that automates specific actions based on particular criteria
- * If criteria are met, actions execute; otherwise, no action occurs
- New:
 - * You can create a new record by using New Action and for the record creation, click on the New button

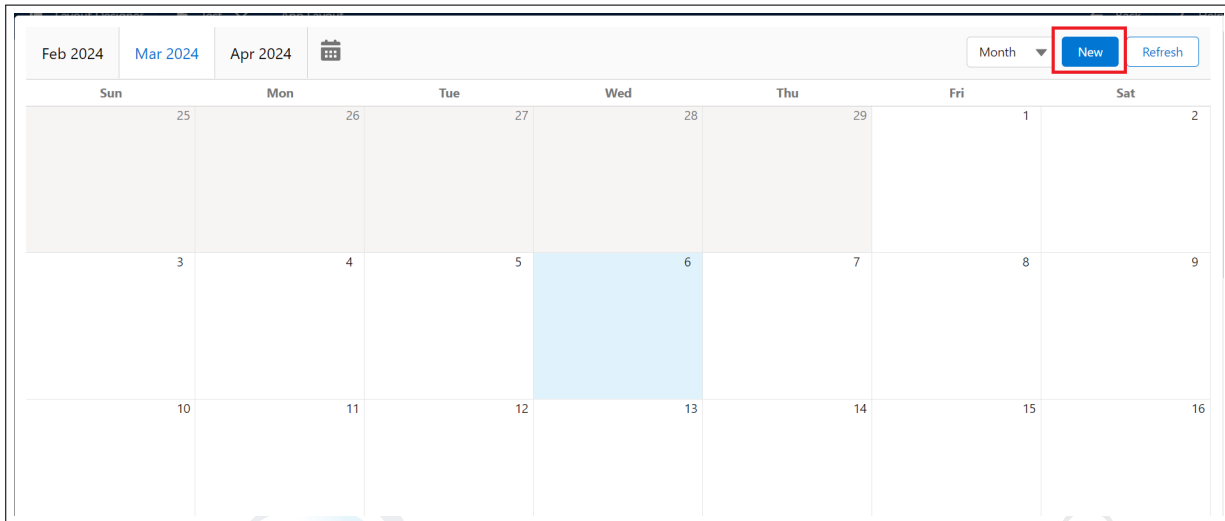


Figure 5.3.63: Calender New

- Refresh:
 - * To reload any previously saved information, click on the Refresh button

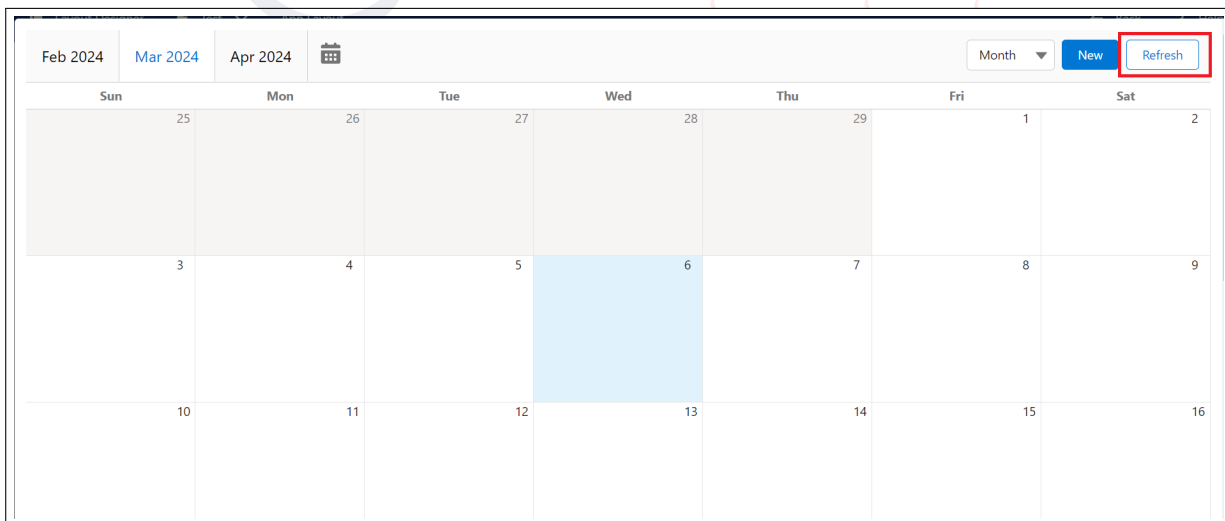


Figure 5.3.64: Calender Refresh

- **UI:**
Same as the Section component described earlier
- **Visibility:** The visibility property specifies whether or not a component is visible and the following are the visibility types:

- Never: The field cannot be visible
- Always: The field will be always visible
- Conditional: Depending on the visibility criteria, the component can be set as visible or not
- **Component:** This field shows the name of the components with the count of their usage. e.g. if you are adding the section for the third time in a layout then it will display with the label Section 3

14. **Rating:** Rating displays valuation or rank on a scale. e.g. Rating can be given to a movie from zero to five stars. Rating is used for giving appreciation to any application like, you give ratings to applications on the Play Store or App Store.

• **Attributes of Rating:**

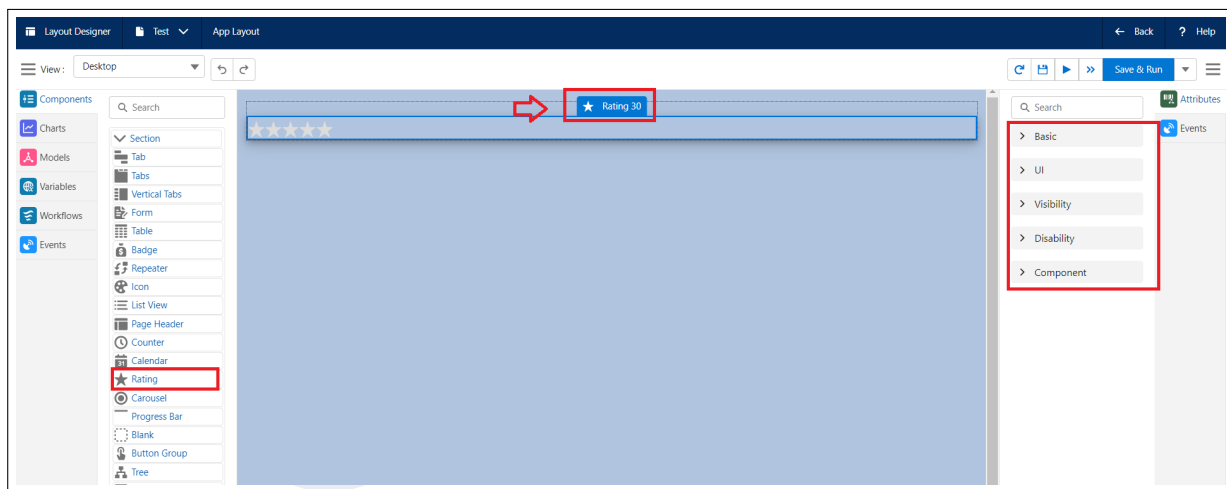


Figure 5.3.65: Attributes of Rating

• **Basic:**

- Value Destination Type: This attribute decides if the user wants to save a value in a variable or model.
- Variable: From the value destination variable, you can choose the variable in which you want to save the value of the field. You can bind the value of the variable to the component by creating a variable.

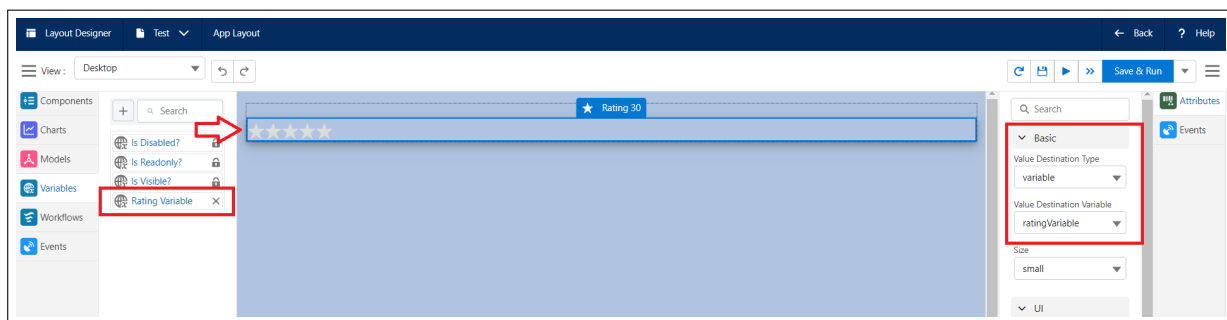


Figure 5.3.66: Basic

- **Model:** From the value destination model, you can choose the specific model in which you can save the value of the field. You can bind a value to the component using a model, value comes from the object that we use while model creation. You can either choose the model from the list or you can create a new model by clicking on the “Create New Model” Option for your table.

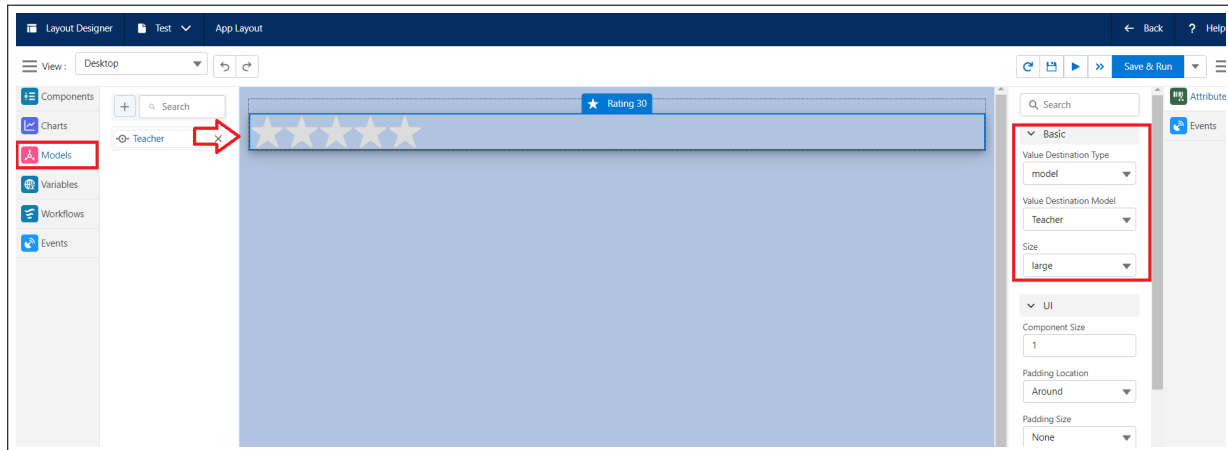


Figure 5.3.67: Rating Basic

- **Size:** The rating can be displayed in five different sizes e.g. xx-small, x-small, small, medium, large.
- **UI:** Same as the Section component described earlier.
- **Visibility:** Same as the components described earlier.
- **Disability:** Same as the components described earlier
- **Component:** Same as the components described earlier



15. **Carousel:** Carousel is a collection of images that gets displayed one at a time. The Carousel component displays a series of images in a single container, with image indicators and a control button below the image panel. Carousel displays one image at a time, and you can select particular images by clicking the indicators. Carousel provides a header and descriptive text that displays below the image.

- **Attributes of Carousel:**

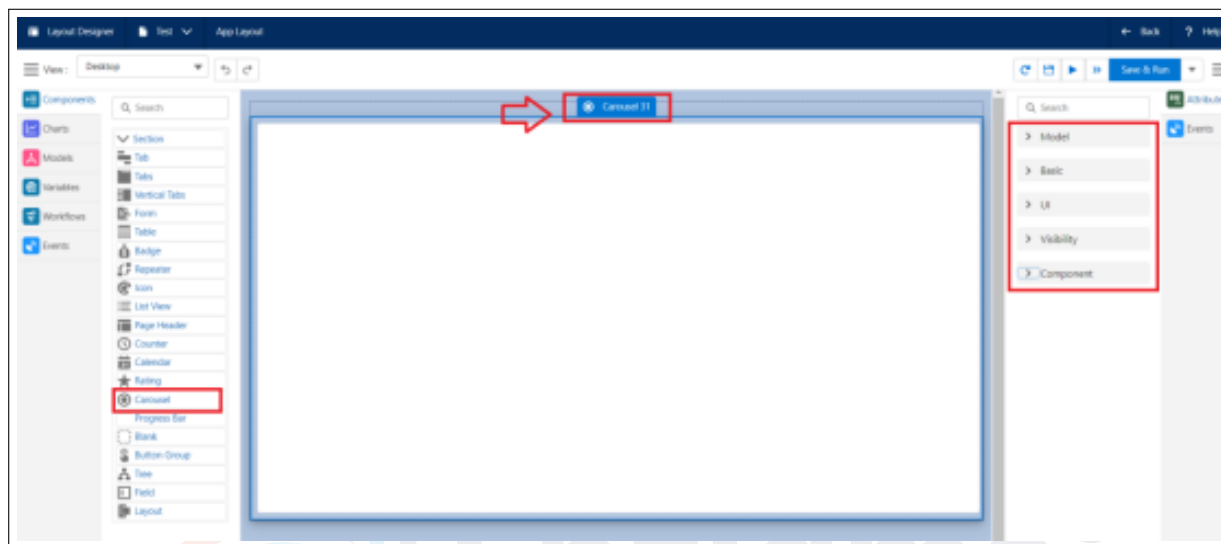


Figure 5.3.68: Attributes of Carousel

- **Models:** In the carousel, you can bind the multi-record model to display the data and images. You can either choose the model from the list or you can create a new model by clicking on the “Create New Model” Option for your table.

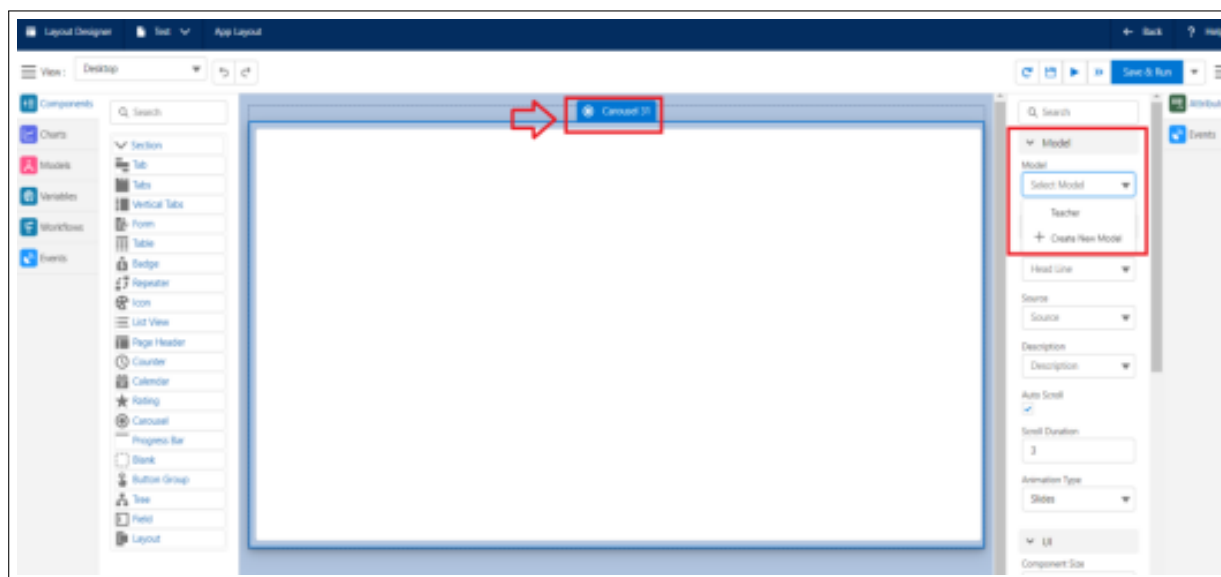


Figure 5.3.69: Models



• **Basic:**

- Head Line: The fields you created are displayed in the Head Line attribute and the data that you saved in that field is displayed below the carousel image. Click on Head Line and Select the field from the drop-down menu.
- Source: In the source, attribute you save the image URL which is displayed as carousel images in the carousel component layout. Click on Source and Select the Source from the drop-down menu.
- Description: In the Description attribute, you can give a description of the carousel images which are displayed below the Head Line attribute. Click on Description and Select the Description from the drop-down menu.
- Auto Scroll: If the checkbox is true then in preview mode it automatically scrolls images

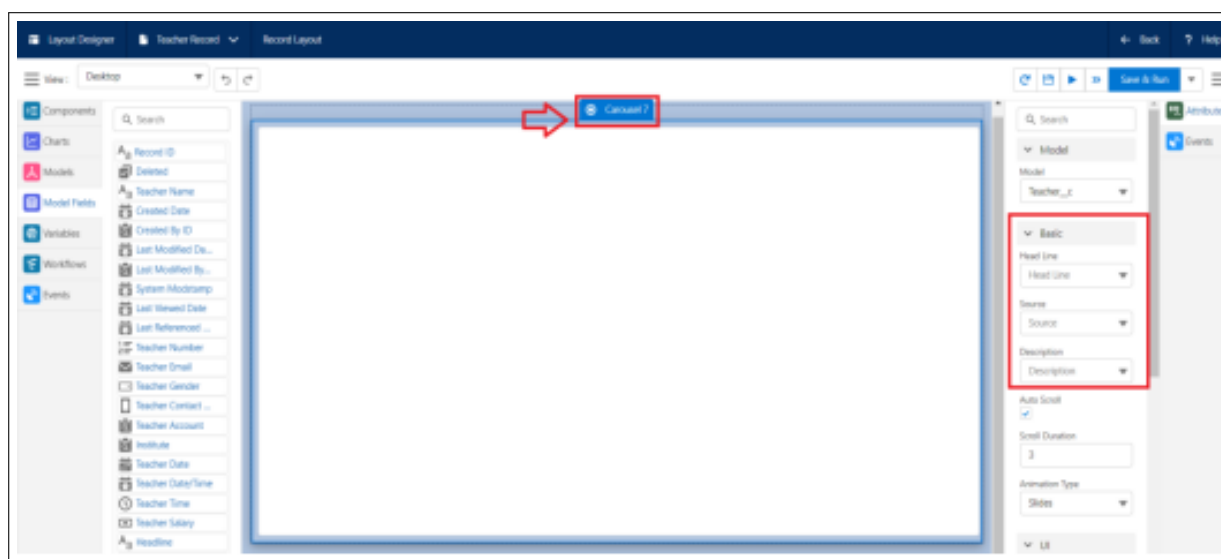


Figure 5.3.70: Carousel Basic

- Scroll Duration: You can set the duration for scrolling images in the carousel. e.g. if you set 3 second duration then in preview mode the next image scrolls after 3 seconds.
- Animation Type: In Carousel, you can use customized animation effects for slide transitions using the animation types. Animated Carousels are way more engaging than still carousels. There are five types of animation which are as follows:
 - * Slides: A slide transition is how one slide is removed from the screen and the next slide is displayed during a presentation
 - * Scale: An animation that controls the scale of an object. You can specify the point to use for the center of scaling
 - * Fade: Fade animation is used to change the appearance and behavior of objects over a particular interval of time. It will provide a better look and feel for our applications
 - * Flip: Flip animation is viewed by viewing successive images in a quick motion so that they seem to form a sequence. A flip animation is a simple type of



animation created by viewing successive images so quickly that they seem to form a sequence.

- * **Jack In The Box:** A Jack In The Box, Creates a transition effect by imitating a popular joke toy movement. It starts with the central fading in and continues with shaking from one side to another.

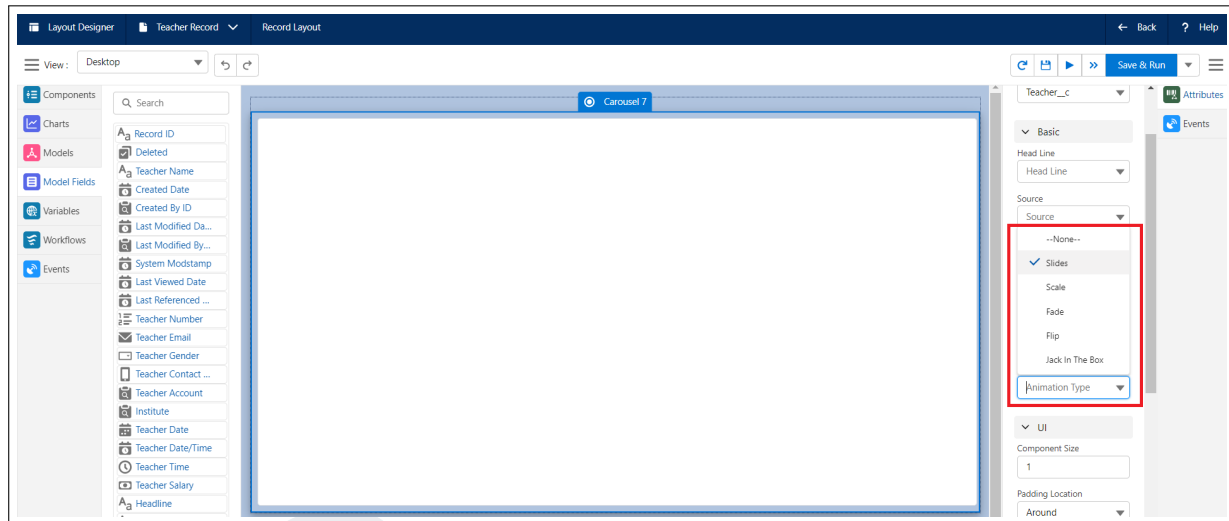


Figure 5.3.71: Animation Type

- **UI:**
Same as the Section components described earlier.
- **Visibility:**
Same as for the components described earlier.
- **Component:**
Same as for the components described earlier.

16. **Progress Bar:** noKodr Progress Bar component helps Salesforce users to get a graphical representation to show the progress of the work. It can be configured to show/hide the Percentage of the progress, on hovering over the component it shows the current percentage. Progress Bar displays the progress of an operation, such as a file download or upload. Generally, the Progress Bar displays horizontally from left to right indicating the progress of an operation.

- **Attributes of Progress Bar:**

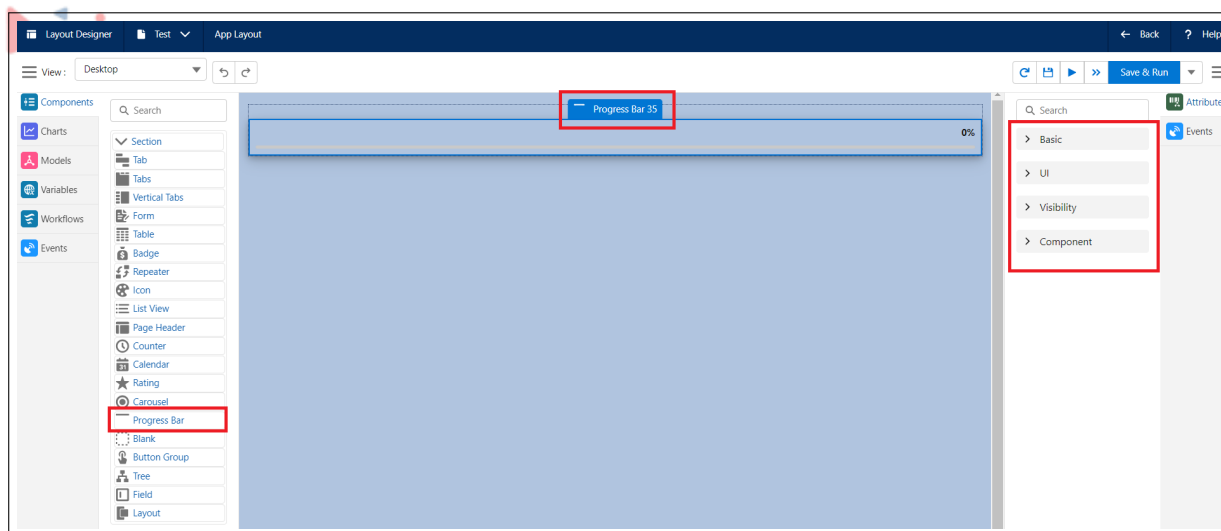


Figure 5.3.72: Attributes of Progress Bar

- **Basic:**

- Label:

- * The label is a short description given to the Progress Bar
 - * Generally, the label is displayed at the top left corner

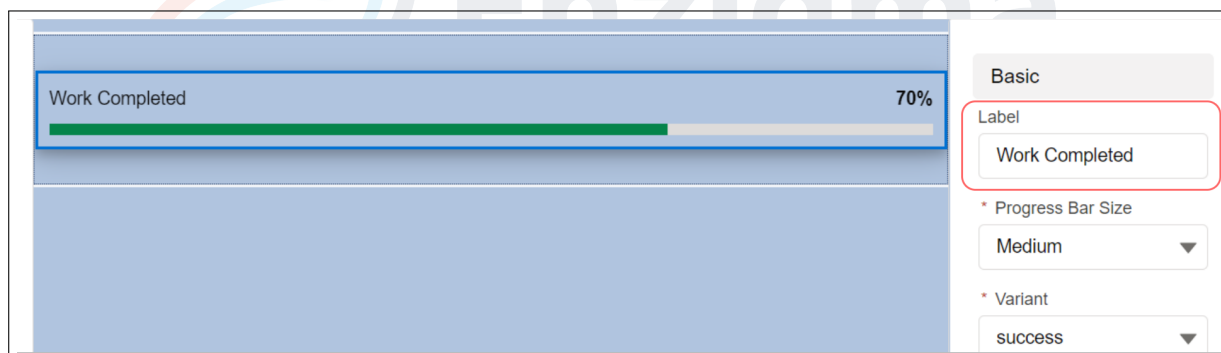


Figure 5.3.73: Progress Bar Basic

- Progress Bar Size: Progress Bars are available in four sizes i.e. x-small, small, medium & large.

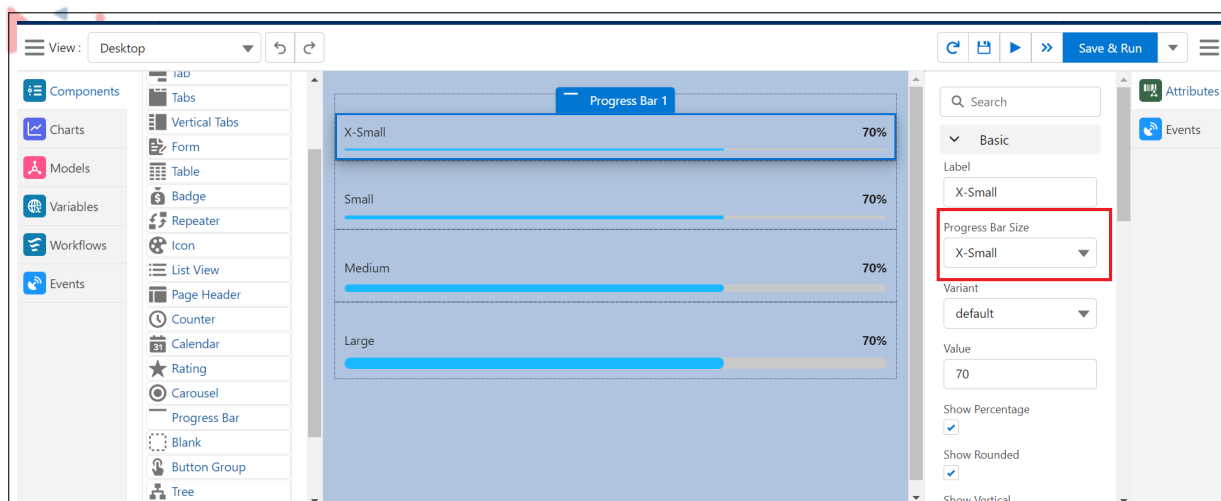


Figure 5.3.74: Progress bar Size

- Variant: Variants display the Progress Bar with different colors to convey different meanings.

Progress bar comes in five variants:

- * Default: Shows Progress Bar in blue color
- * Inverse: Shows Progress Bar in gray color
- * Success: Shows Progress Bar in green color
- * Warning: Shows Progress Bar in yellow color
- * Error: Shows Progress Bar in red color

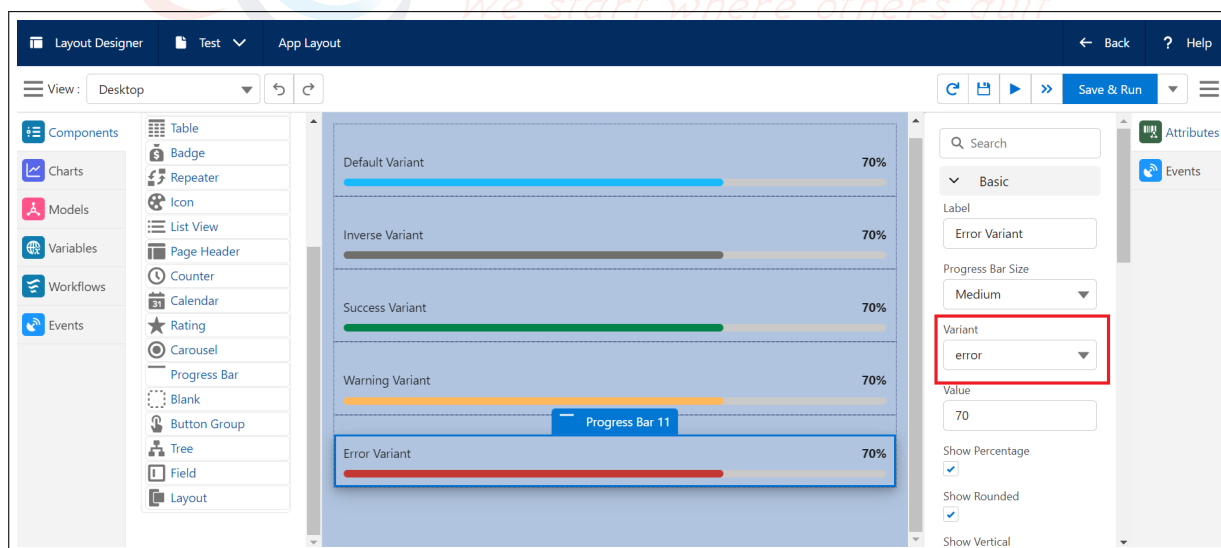


Figure 5.3.75: Variant

- Value: The percentage value of the Progress Bar, value is an indication of progress. e.g. the Progress Bar denotes completion when the value is 100.

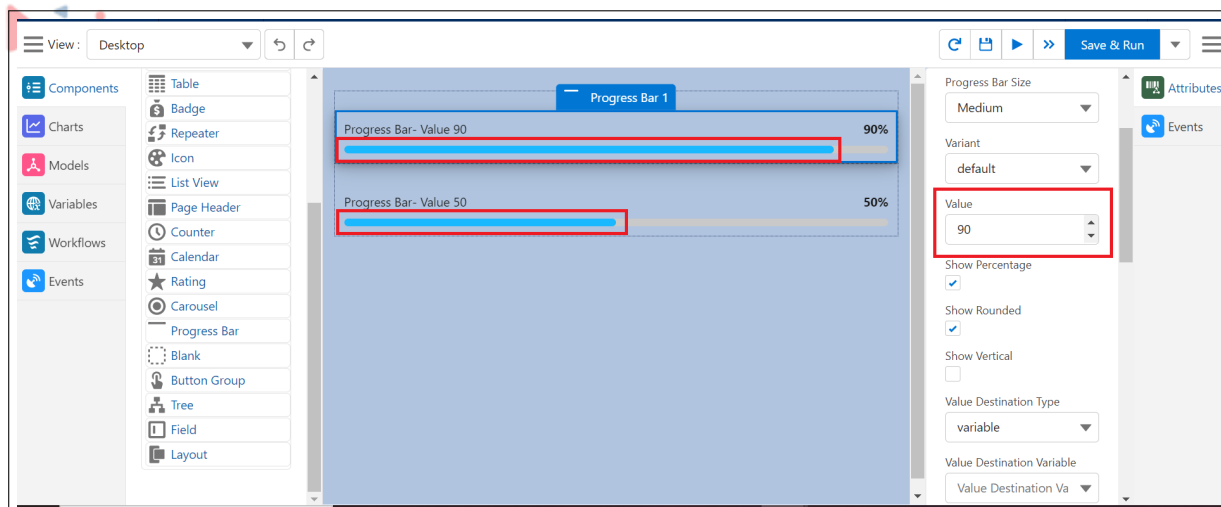


Figure 5.3.76: Value

- Show Percentage: To indicate the progress of an operation in terms of percentage.

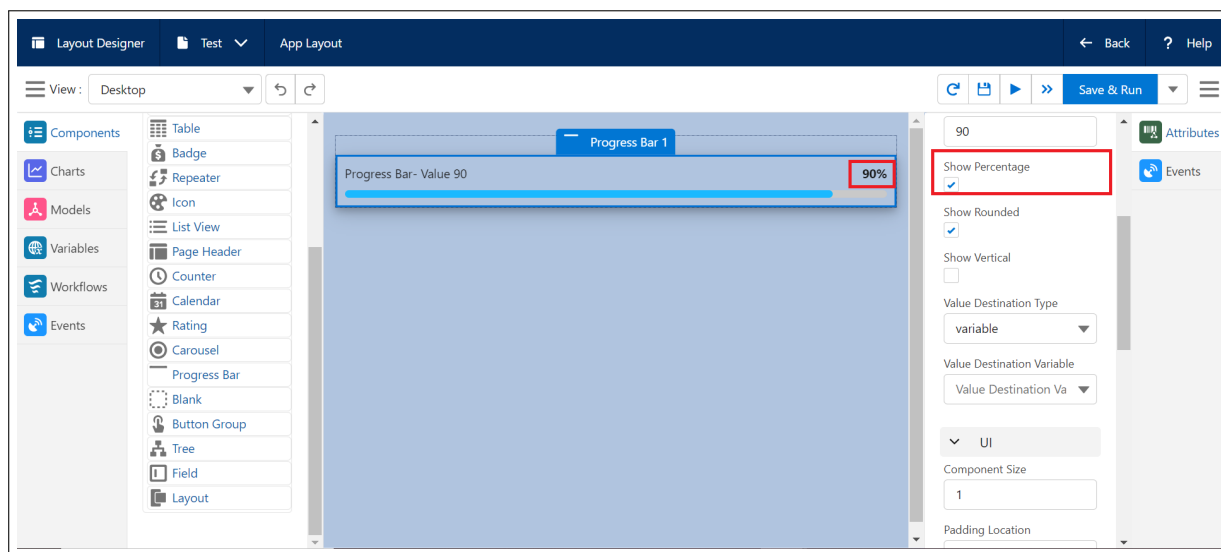


Figure 5.3.77: Show Percentage

- Show Rounded: It adds a border radius to the Progress Bar to give it a rounded look

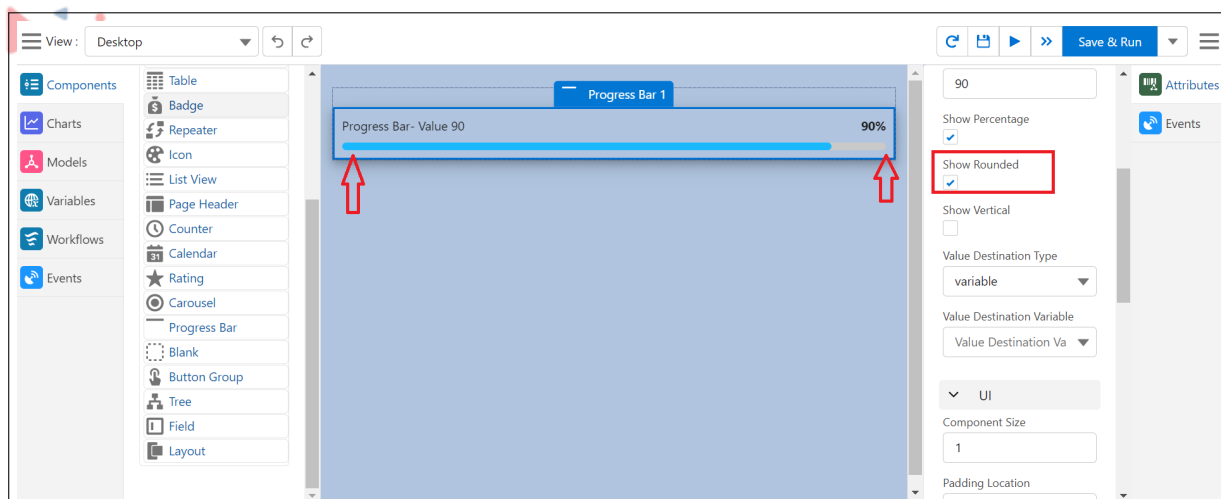


Figure 5.3.78: Show Rounded

- Show Vertical: Displays a vertical Progress Bar from top to bottom to indicate the progress of an operation.

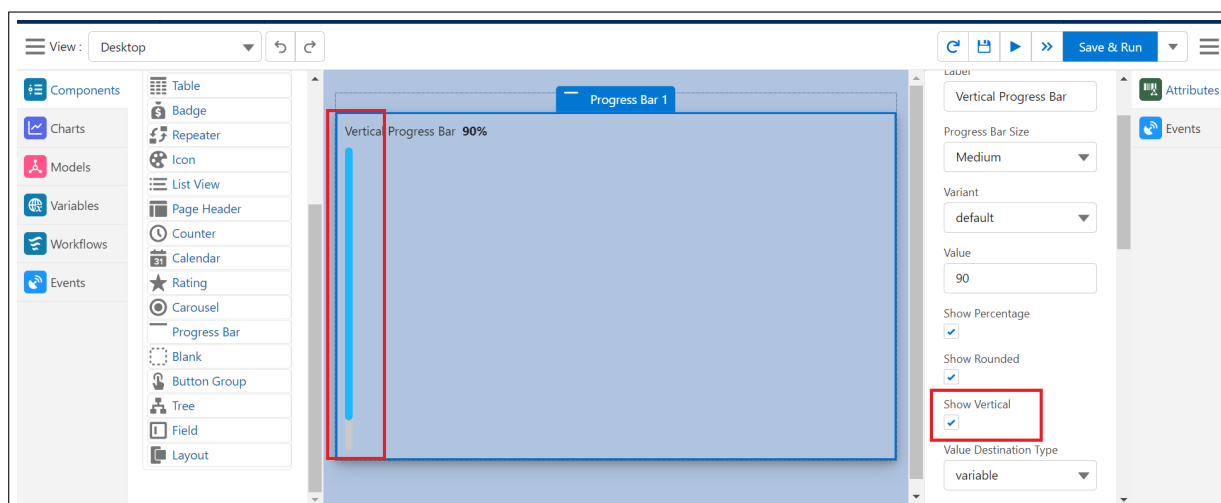


Figure 5.3.79: Show Vertical

- Value Destination Type: There are two types of Destination are available
 - * Variable Destination Type: This type allows you to store the values in the variable, if you select Destination Type as a variable then it will show you the variable that you have to store
 - * Model Destination Type: This type allows you to store the values in the Models, if you select destination Type as the model then it will show you the model that you have to store

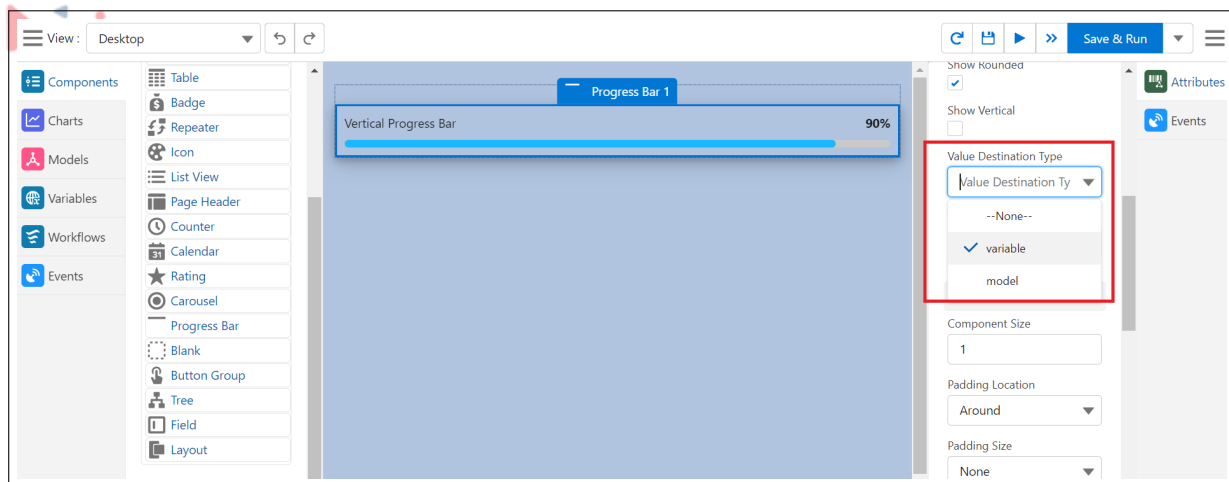


Figure 5.3.80: Value Destination Type

- **UI:**
Same as other components described earlier.
- **Visibility:**
Same as other components described earlier.
- **Component:**
Same as other components described earlier.

17. **Blank:** The blank component serves as an empty space between two components/fields. The blank component is essential in the Layout Designer to enhance the user experience by separating components/fields without visible dividers.

• **Attributes of Blank:**

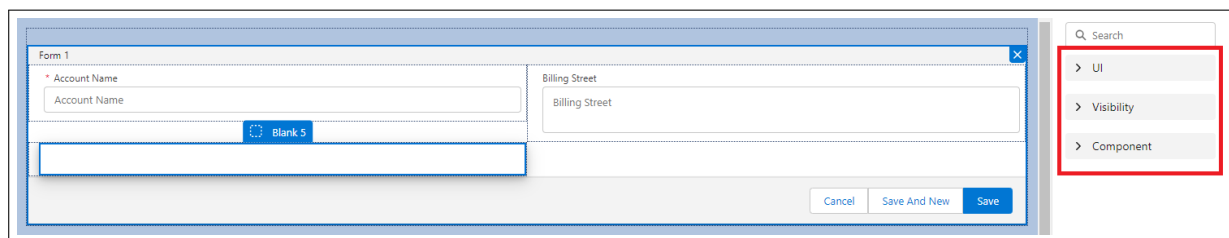


Figure 5.3.81: Attributes of Blank

A blank component in preview mode:

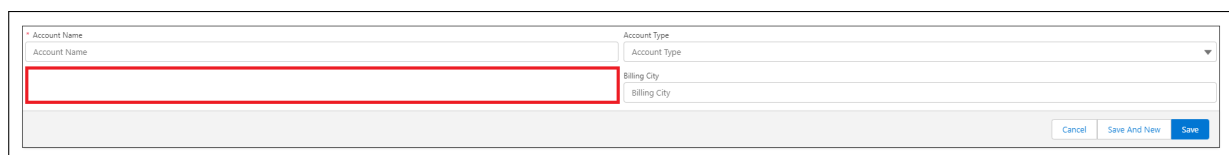


Figure 5.3.82: Blank Component

- **UI:**
Same as other components described earlier
- **Visibility:**
Same as other components described earlier
- **Component:**
Same as other components described earlier

18. **Button Group:** Buttons are interactive elements designed to trigger a specific action. Button Groups showcase basic default buttons, buttons with icons, and different versions grouped together in a single unit, allowing for the creation of a horizontal series of buttons without gaps.

- **Attributes of Button group:**

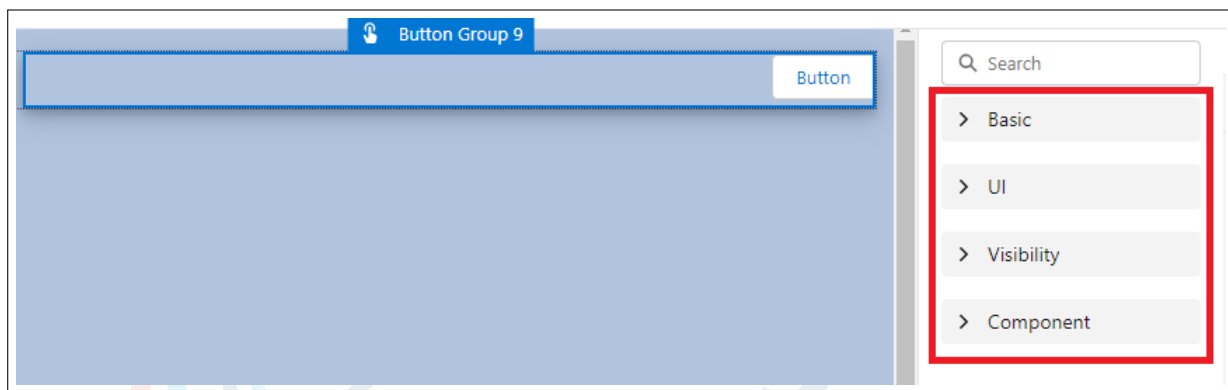


Figure 5.3.83: Attributes of Button group

- **Basic:**
 - **Actions:** Perform an action by clicking on the button. When you click or hover over it, the selected action will take place. You have the option to add multiple actions. To create a new action, simply click on the + icon.

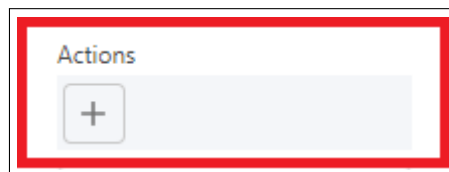


Figure 5.3.84: Show Percentage

- **Label:** The button label is shown on the button.
- **Name:** The name attribute defines the button's name.
- **Left Icon:** You have the option to include an icon on the left side of the button
- **Right Icon:** You have the option to include an icon on the right side of the button
- **Variant:** The variant feature enables buttons to appear in various color schemes
- **Neutral:** Neutral is the default button style displayed in white



- Brand: Brand represents a blue button designed to emphasize the primary action on a page
 - Outline-Brand: Outline-brand is similar to the brand style but uses color for the label and border only
 - Destructive: Destructive is a red button indicating a negative effect
 - Success: Success is a green button denoting a successful action
- **UI:**
Same as for the other components described earlier.
 - **Visibility Type:**
Same as for the other components described earlier.
 - **Component:**
Same as for the other components described earlier.

19. **Tree:** A tree component is a user interface element used to display hierarchical data in a tree-like structure. It organizes data into nodes connected by edges, allowing users to expand or collapse nodes, select items, and interact with the data.

- **Attributes of Tree:**

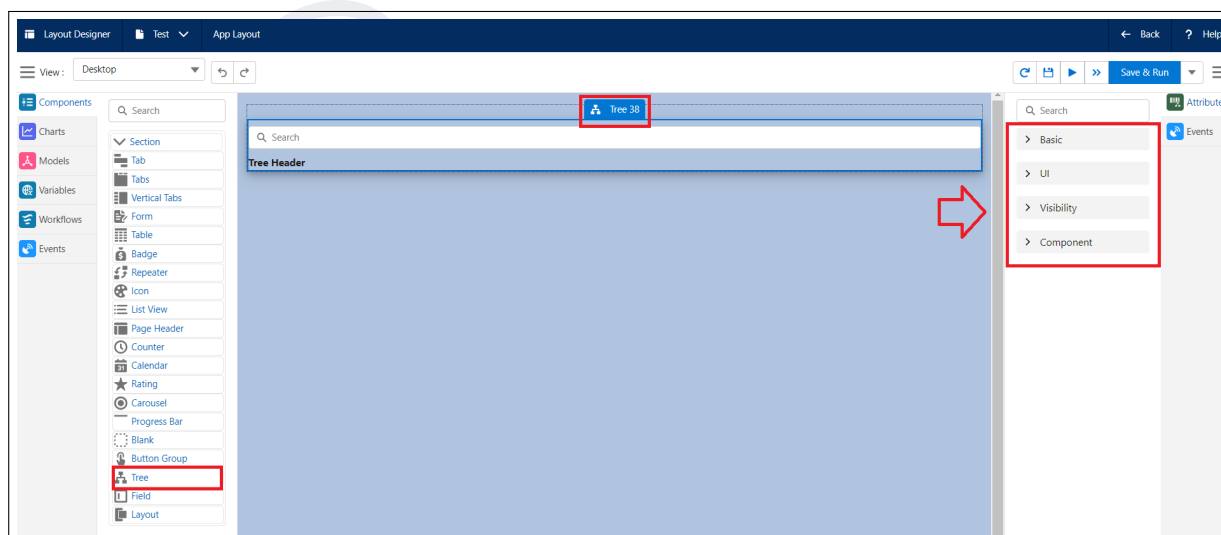


Figure 5.3.85: Attributes of Tree

- **Basic:**
 - Model: All the models will be displayed in the list that you have created using an object, you can choose the specific model. Models are used to display content in the layout. You need to create a multi-record model for a Tree component, refer to this Model.
 - Create New Model: You can either choose the model from the list or you can create the new model by clicking on the “Create New Model” Option for your tree

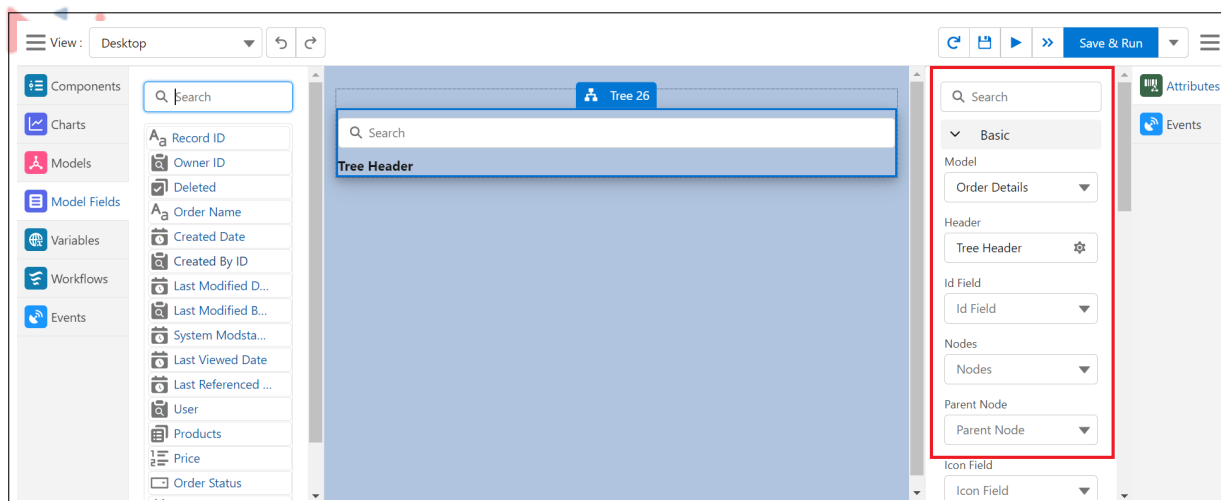


Figure 5.3.86: Basic

- Header: Name of the header which is displayed on the bottom of the component
- Id Field: Each tree node must possess a unique identifier for identification purposes. All fields are visible with the exception of image, address, array, and object type fields.
- Nodes: The label field is designated for displaying the label of the tree node. All fields are visible except for image, address, array, and object type fields.
- Parent Node: The lookup field type should be utilized as a self-lookup to generate a hierarchy. Only lookup fields are visible in this context.

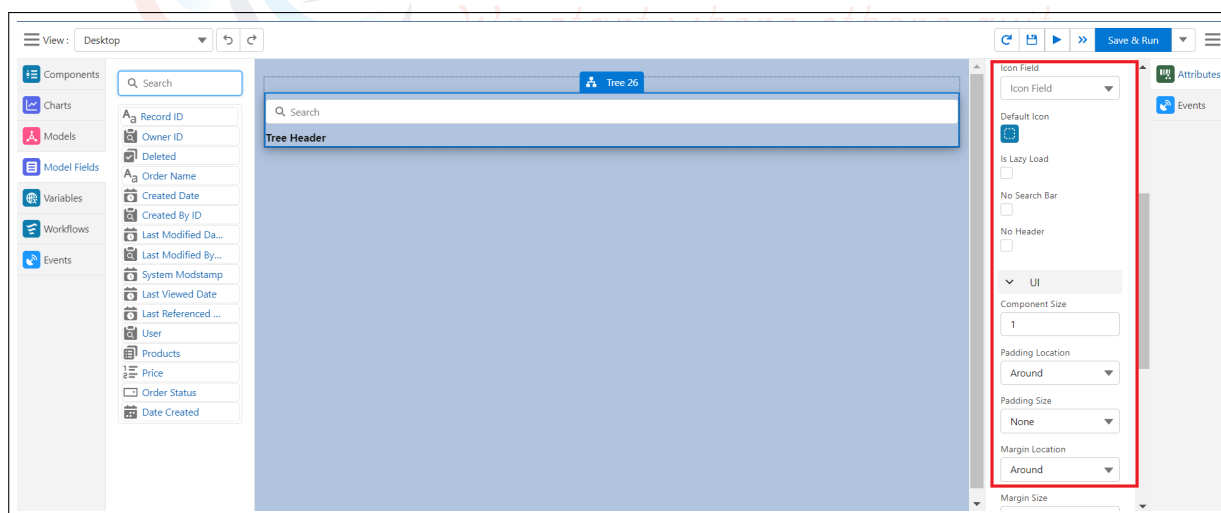


Figure 5.3.87: Basic

- Icon Field: This is used to display icons for individual nodes, utilizing a field of type "text" for alternate text.
 - * Format for icon field data: iconCteory:iconName example => 'utility:user'
- Default Icon: You can set the default icon for the tree
- Is lazy Load: This checkbox type field is utilized during hierarchy creation. When



set to true, it generates only parent-level hierarchy, with children loaded as the user expands the tree node. There is no visible difference in the UI.

- No Search Bar: Checkbox used to hide the search box of a Tree
- No Header: Checkbox used to hide the header part of a Tree

- **UI:**

Other attributes same as for the other components described earlier.

- Style: The style attribute is used to add styles to a Component, such as color, font, size, and more. Setting the style of a Component can be done with the style attribute. The style attribute has the following syntax: “property:value” The property is a CSS property. The value is a CSS value.
- Classes: Classes are used to apply unique styling and formatting to the fields in preview mode. Users need to create one cdn link from the CSS tab on the layout designer page for using classes

- **Visibility:**

Other attributes same as for the other components described earlier.

- **Component:**

Other attributes same as for the other components described earlier.

20. **Field:** Fields in noKodr represent what the columns represent in relational databases. It can store data values that are required for a particular object in a record. Fields can store different types of data.

- The Field is a generic component using which you can accept values in different formats and data types
- e.g. number, phone, email, text, etc
- It can be bound with the model Field and variable
- If you want to accept/display data to the user without an object you can use the Field component

- **Attributes of Field:**

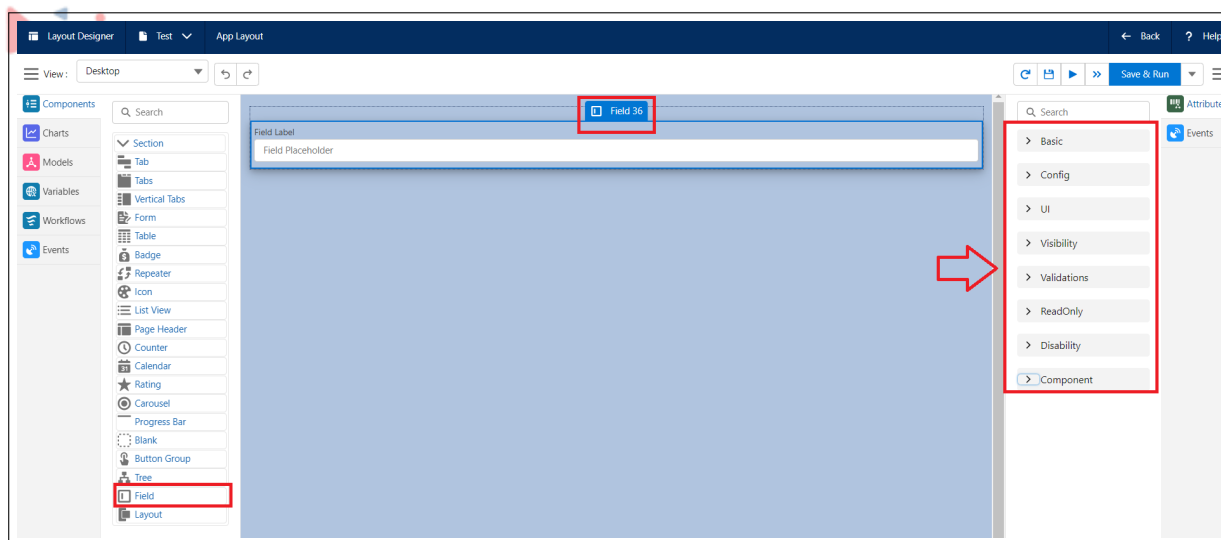


Figure 5.3.88: Attributes of Field

- **Basic:**

- Name: The name attribute specifies the name of Field
- Label: The label is a short description of the field that will be displayed before the data input/output
- Placeholder: A placeholder is a short text, located inside the input data field
- Field Type: You can see all the fields in a list and change their types as needed.
- Layout: It is a design using which the user can arrange the fields. The Layout comes in 5 different forms:
 - * Stacked: In a stacked layout, the input/output field is under the field label with a small margin around the label, and the input/output field
 - * Horizontal: In a horizontal layout, the input/output field is in front of the field label with a small margin around the label, and the input/output field
 - * Single Column: A Single Column layout is the same as a horizontal layout the only difference is the input/output field size is greater than the field label in a layout
 - * No Label: In the No Label layout, the label name is not visible only the input/output field is present in the layout

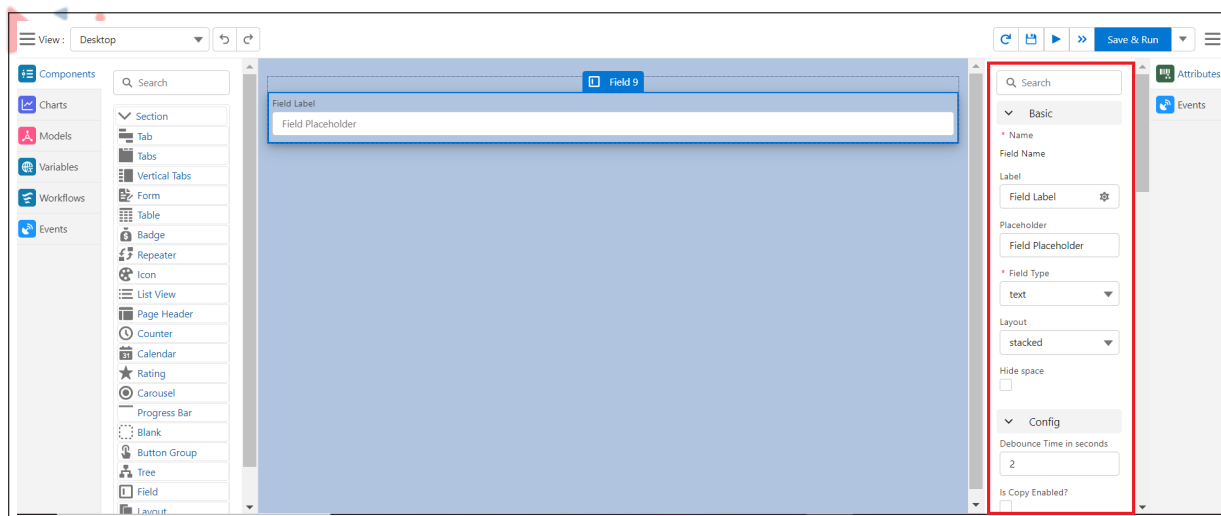


Figure 5.3.89: Basic

- Hide Space: If the 'Hide Space' checkbox is enabled, the padding around the field will be removed
- Debounce Time in seconds: It is the time taken to complete the two actions e.g. When you click on the save button, the time it takes to complete the save action is debouncing time
- Is Copy Enabled?: Checkbox is enabled, and users can copy the text that has been added to the placeholder

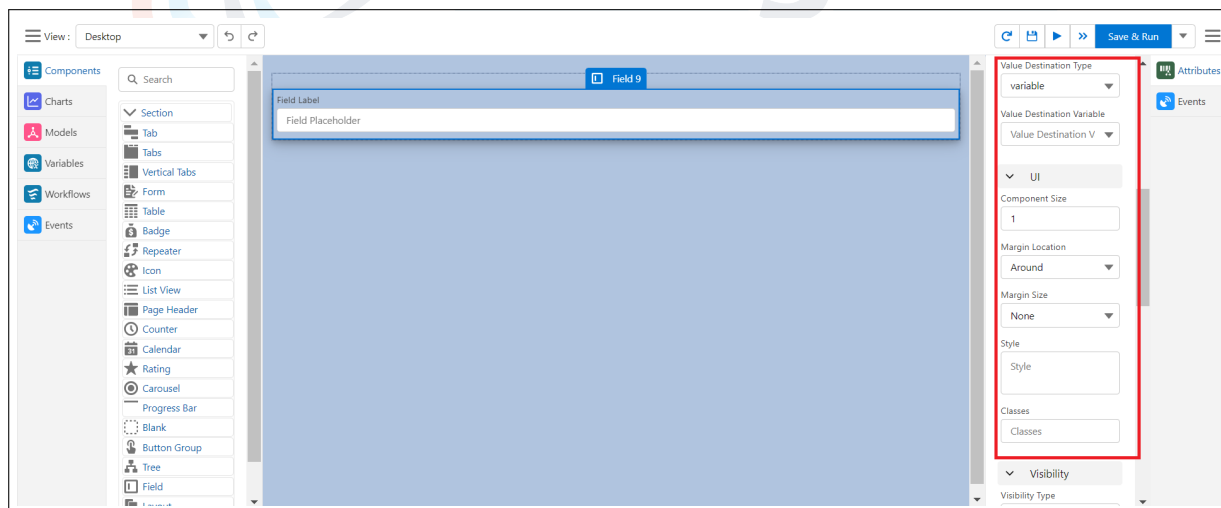


Figure 5.3.90: UI

- Value Destination Type: There are two types of Destination are available
 - * Variable Destination: This type allows you to store the values in the variable, if you select Destination Type as a variable then it will show you the variable that you have to store
 - * Model: This type allows you to store the values in the Models, if you select Destination Type as the model then it will show you the model that you have to



store

- **Config:**

- Value Destination Type: We can select a particular destination type to store the value.i.e.
- Variable: Used to store the value in a variable.
Value Destination Variable: Depending on the selected Destination Type, Value Destination Variable will be displayed.
- Model: Used to store the value in the field of the model.
Value Destination Model: Depending on the selected Destination Type, the Value Destination Model will be displayed.

- **UI:**

Same as for the other components explained earlier.

- **Visibility:**

Same as for the other components explained earlier.

- **Validations:**

- Required Type: Ensures that a specific type of input is required.
- Required Error Message: Specifies the error message to be displayed when the required type is not provided.
- Minimum Length: Specifies the minimum length of input required.
- Min Length Error Message: Specifies the error message to be displayed when the input length is less than the minimum required length.
- Maximum Length: Specifies the maximum length of input allowed.
- Max Length Error Message: Specifies the error message to be displayed when the input length exceeds the maximum allowed length.
- Pattern: Specifies a pattern or format that the input must adhere to.
- Pattern Error Message: Specifies the error message to be displayed when the input does not match the specified pattern.

- **Read-only:**

- Never: The field cannot be read-only
- Always: The field will be always read-only
- Conditional: Depending on the read-only criteria, the field can be set as read-only or not

- **Disability:**

- Never: The field cannot be disabled
- Always: The field will be always disabled
- Conditional: Depending on the disability criteria, the field can be set as disabled or not

- **Component:**

Same as for the other components described earlier

21. **Layout:** Represents a responsive grid system for arranging containers on a page. Control the layout and organization of various components like buttons, fields, tabs, icons, and tables on record pages. The layout allows designing various layout configurable things like workflows, events, variables, models, CSS, etc. It also helps to determine which fields are visible, read-only, and required to use page layouts to customize the content of record pages for your users.

• **Attributes of Layout:**

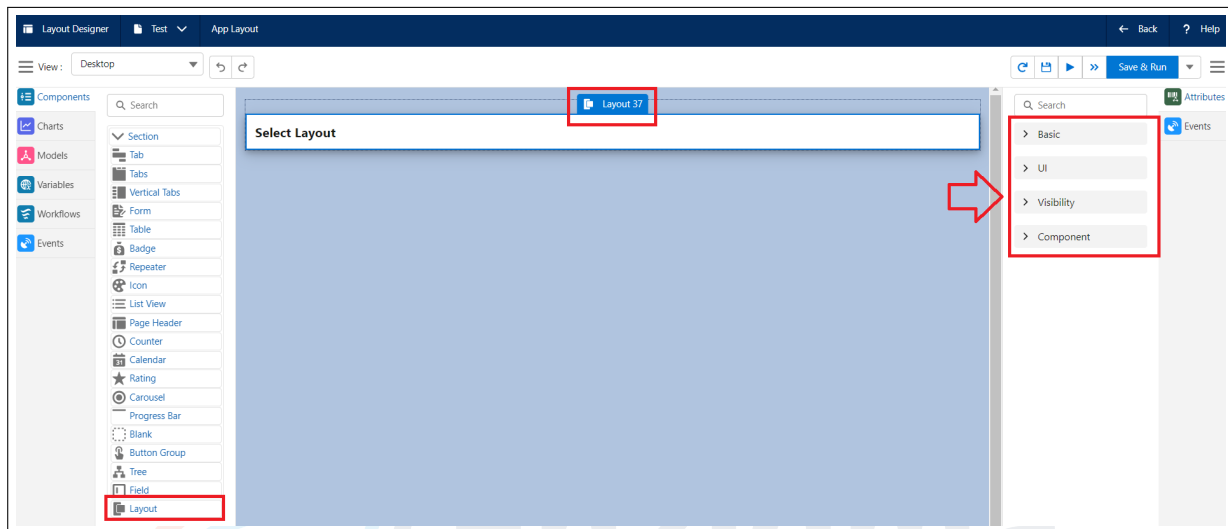


Figure 5.3.91: Attributes of Layout

• **Basic:**

- Layout: The Layout attribute displays the available list of existing Layout

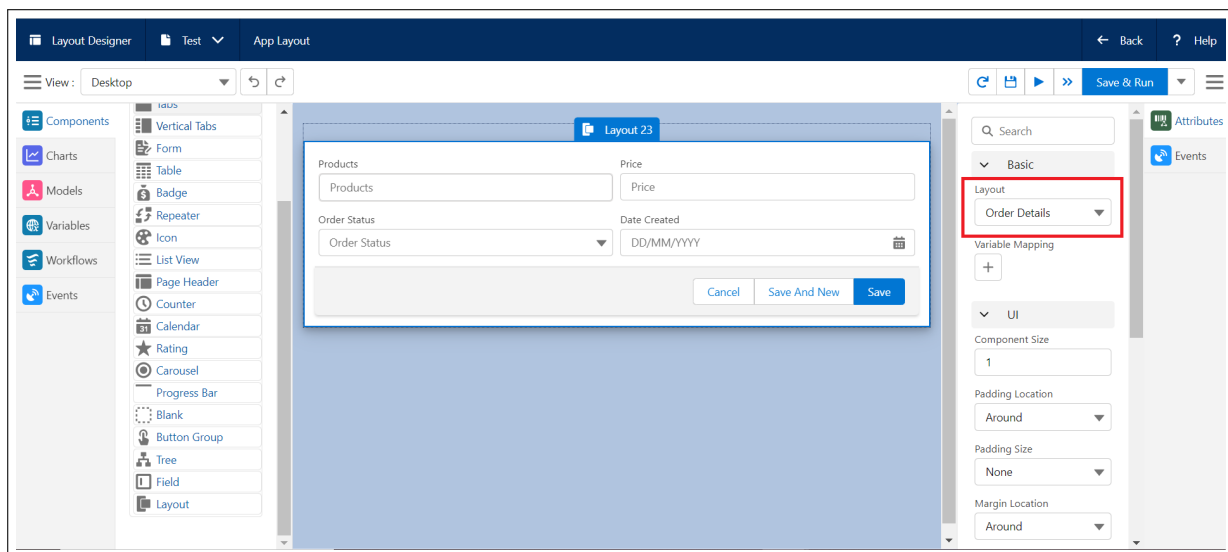


Figure 5.3.92: Layout

- Variable Mapping: The layout chosen from the Layout drop-down will display

associated variables in the Variable Mapping - Input Variable list drop-down. These variables enable assigning values to the layout using multiple source types, including Static, Variable, Model, Null, User, and Blank. You can add the variables by clicking on the '+' icon, e.g. Record Id, Is Visible, etc.

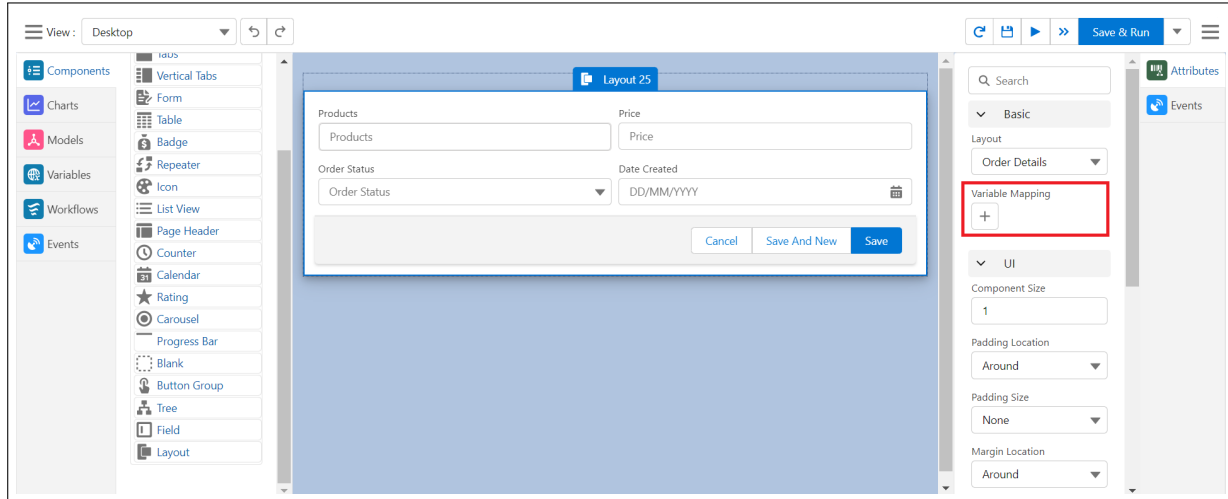


Figure 5.3.93: Variable Mapping

Click on save, and the variable mapping will be visible

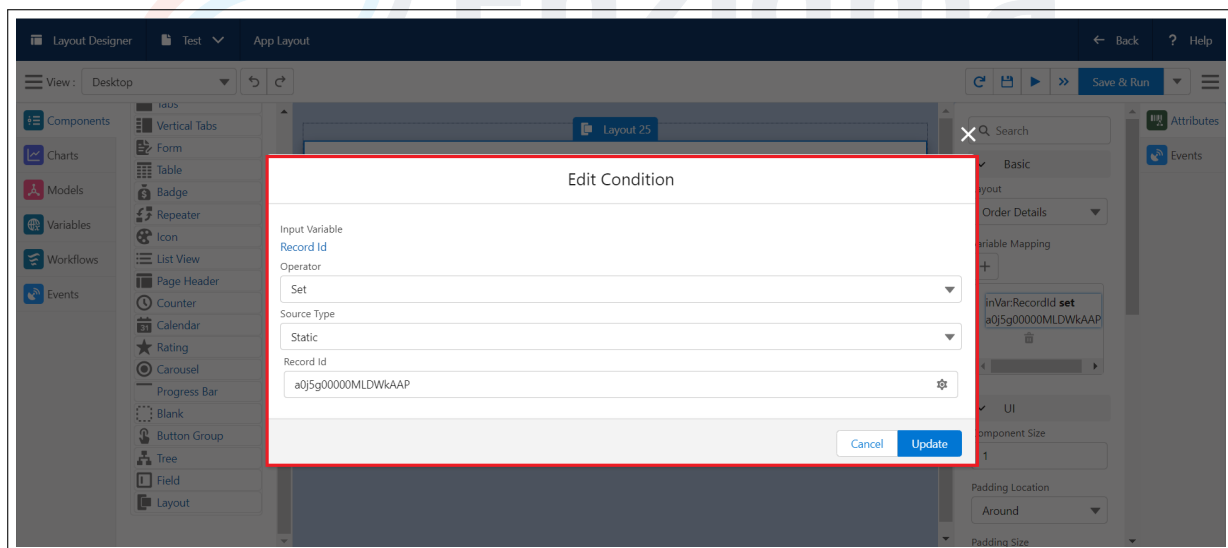


Figure 5.3.94: Edit Condition

- **UI:**
Same as for the other components described earlier.
- **Visibility:**
Same as for the other components described earlier.
- **Component:**
Same as for the other components described earlier.

22. **HTML:** The HTML component empowers users to input and modify content in a format like HTML, encompassing text, images, links, videos, and other elements, as well as merge text (enabling users to create merged text using multiple values).

- **Attributes of HTML:**

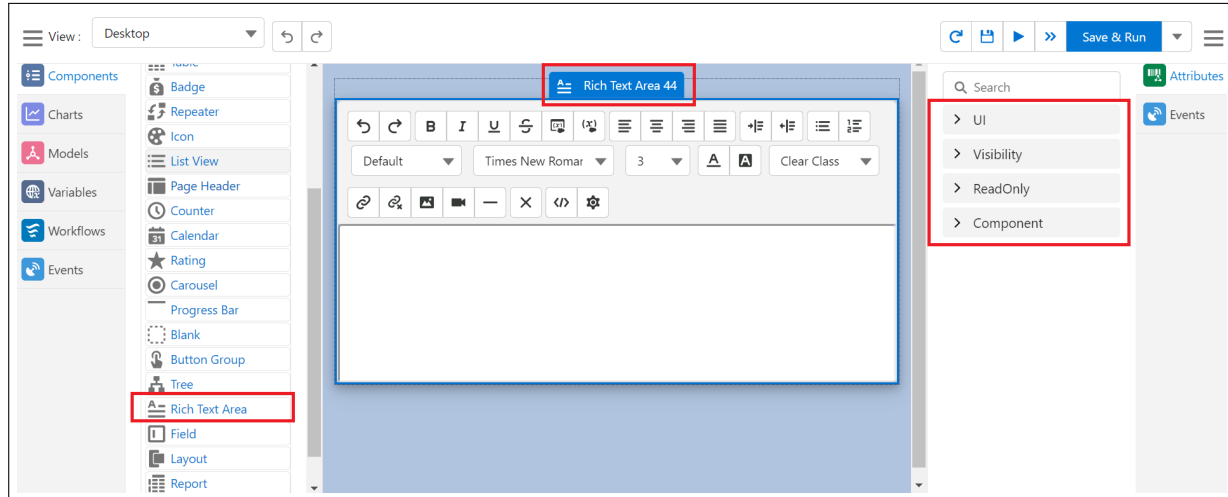


Figure 5.3.95: Attributes of HTML

- **UI:**
 - **Component Size:** The user can modify the size of the component in the layout as per grid size
 - **Padding Location:** Defines the position of the padding for a component. The padding creates extra space within a component.

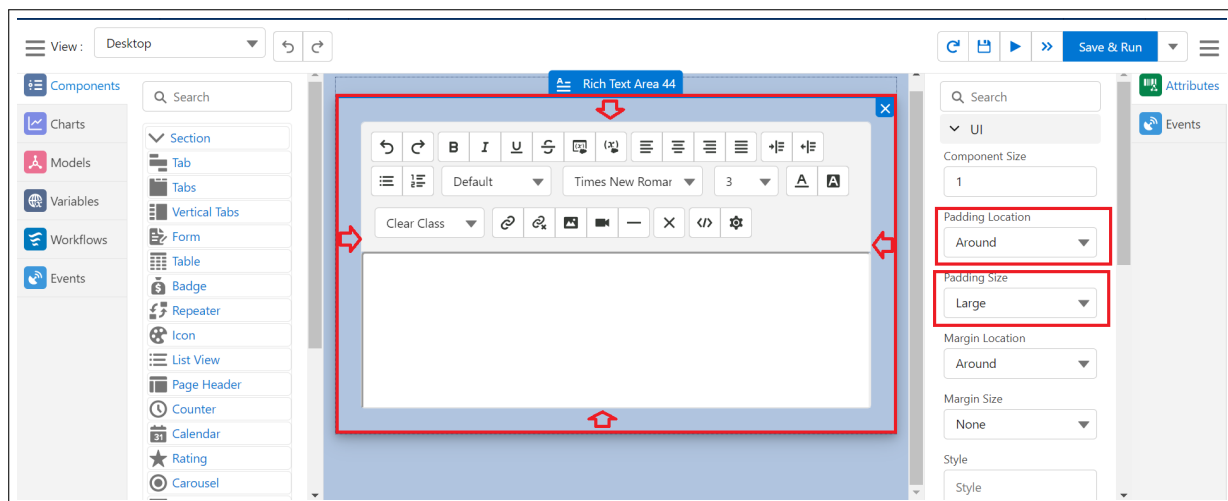


Figure 5.3.96: Attributes of HTML

Type of Padding Locations :

- * **Around:** Creates padding around the component
- * **Top:** Creates padding at the top of the component

- * Left: Creates padding at the left side of the component
- * Bottom: Creates padding at the bottom of the component
- * Right: Creates padding at the right side of the component
- * Horizontal: Creates padding horizontally
- * Vertical: Creates padding vertically
- Padding Size: The padding size of the component can be set to None, XXX-Small, XX-Small, X-Small, Small, Medium, Large, X-Large, XX-Large
- Margin Location: Defines the position of the margin for a component. Margin creates extra space around a component.

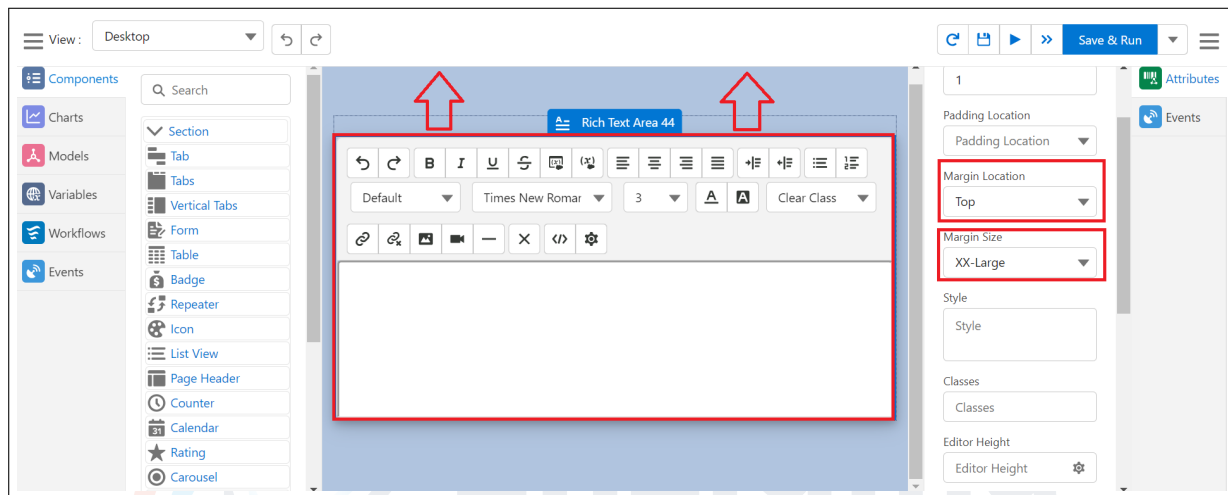


Figure 5.3.97: Attributes of HTML

Types of Margin Locations:

- * Around: Margin gets added around the component
- * Top: Margin gets added at the top of the component
- * Left: Margin gets added at the left side of the component
- * Bottom: Margin gets added at the bottom of the component
- * Right: Margin gets added at the right side of the component
- * Horizontal: Margin gets added horizontally
- * Vertical: Margin gets added vertically
- Margin Size: The margin size of the view can be set to None, XXX-Small, XX-Small, X-Small, Small, Medium, Large, X-Large, XX-Large.
- Style: The style attribute is used to add styles to a Component, such as color, font, size, and more. Setting the style of a Component can be done with the style attribute. The style attribute has the following syntax: “property:value” The property is a CSS property. The value is a CSS value.

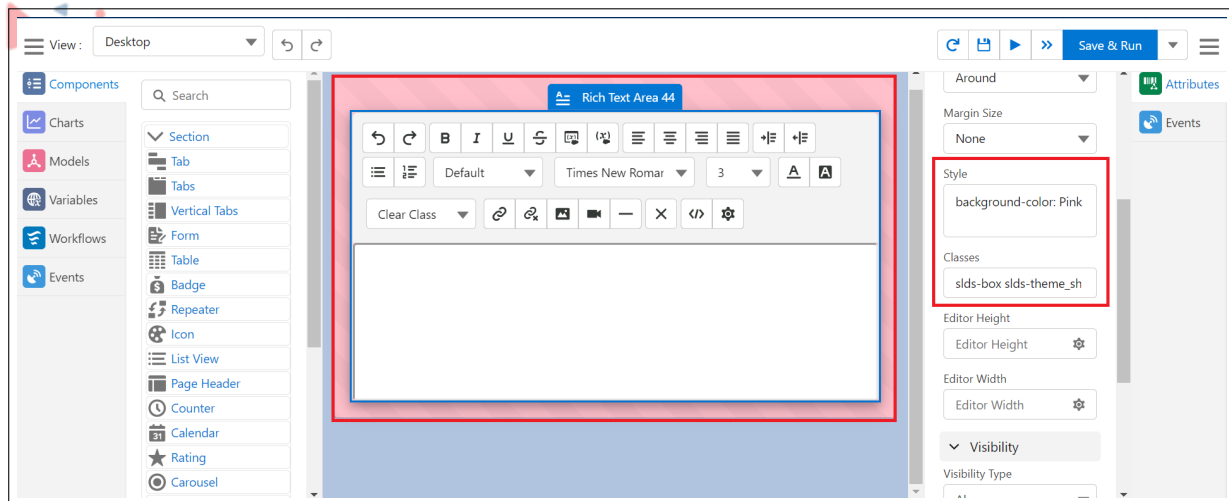


Figure 5.3.98: Attributes of HTML

- Editor Height: The "editor height" refers to the vertical size or dimension of an editor, typically measured in pixels, em, %, rem, cm, vh, vw, in, indicating how tall the editor appears on the screen
- Editor Width: The editor width" denotes the horizontal size or dimension of an editor, representing its width across the screen, also measured in pixels, em, %, rem, cm, vh, vw, in.

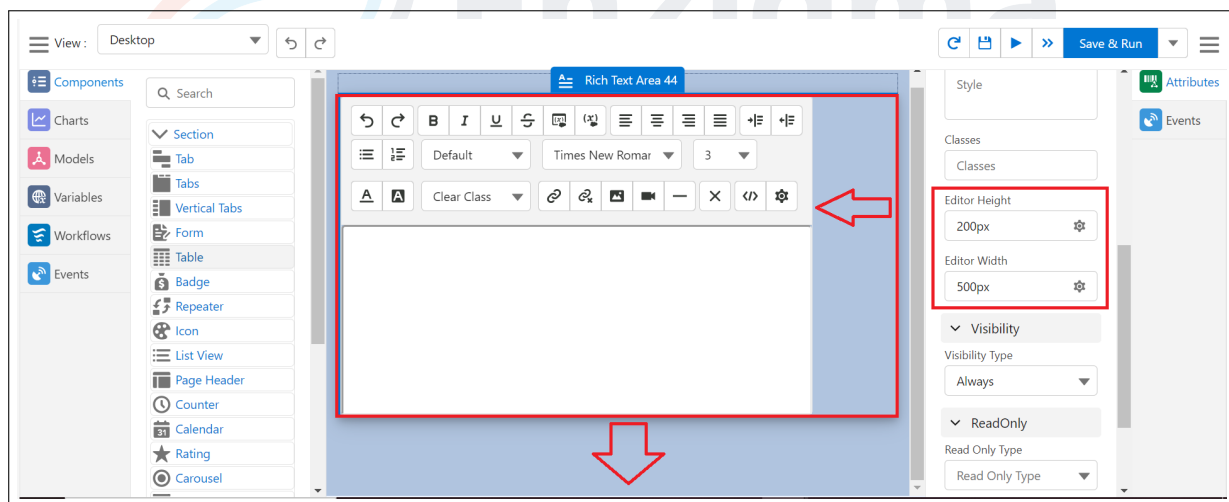


Figure 5.3.99: Attributes of HTML

- Visibility: The visibility property specifies whether or not a component is visible on the layout or not. The following are the visibility types:
 - Never: The field will not be visible at all
 - Always: The field will be always visible
 - Conditional: Depending on the visibility criteria, the component can be set as visible or not visible.
- Read Only: This property specifies whether or not a component is ReadOnly and the following are the ReadOnly conditions:



- Never: The field will never be in Read Only I.e always editable
 - Always: The field will be always read-only
 - Conditional: Depending on the read-only criteria, the field can be set as Read-only or not
- Component: This field shows the name of the components with the count of its usage.



V.III.1.5 Models in Layout :

- **Models** : Models are used to display content in the layout and are typically used for record creation or editing, as well as various types of messaging. The model needs to be added when you perform any action in an event. Users need to create Models as per their requirements for creating workflows or binding data on various components.

- How to create Model: To create a Multi record or Single record model
 1. Go to the Model tab in the Layout designer
 2. Click on the + icon
 3. Enter information, click Save

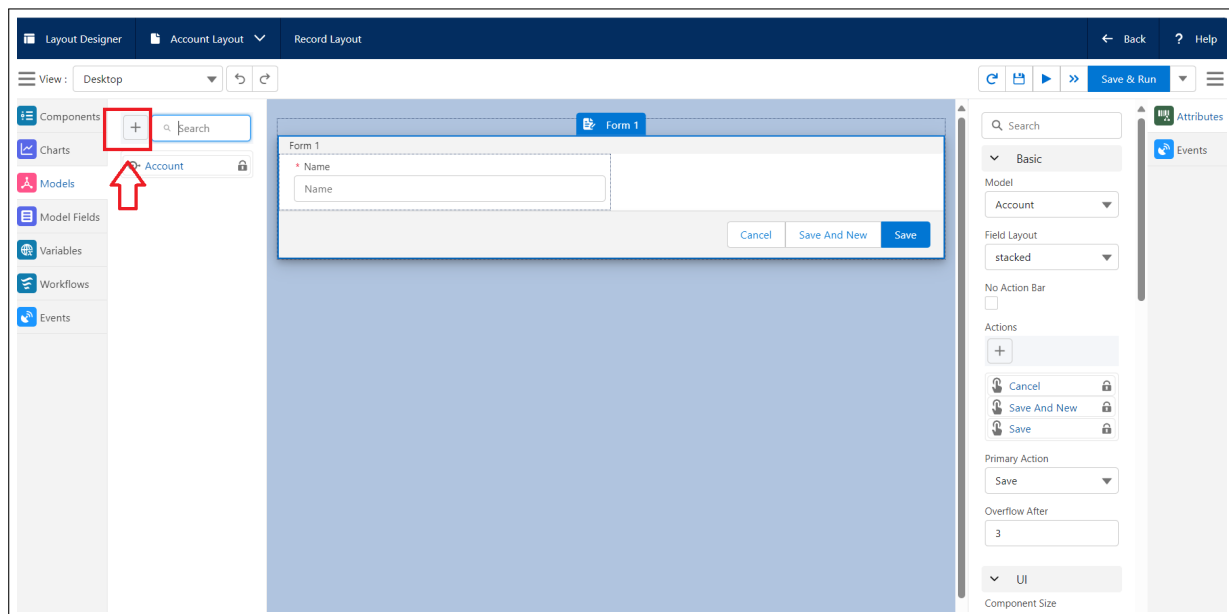


Figure 5.3.100: Model Creation

- **Fields of Model:**

1. Label: The label is used to identify the Model at the UI.
2. Name: Name is the unique identifier of the Model.
3. Type: Display the Model's type, there are two types that are Record, and API. Select the type from the drop-down menu
 - (a) Record Type: Record type model stores the data of the Salesforce objects.
 - (b) API Type: API type models are used to integrate different web applications.

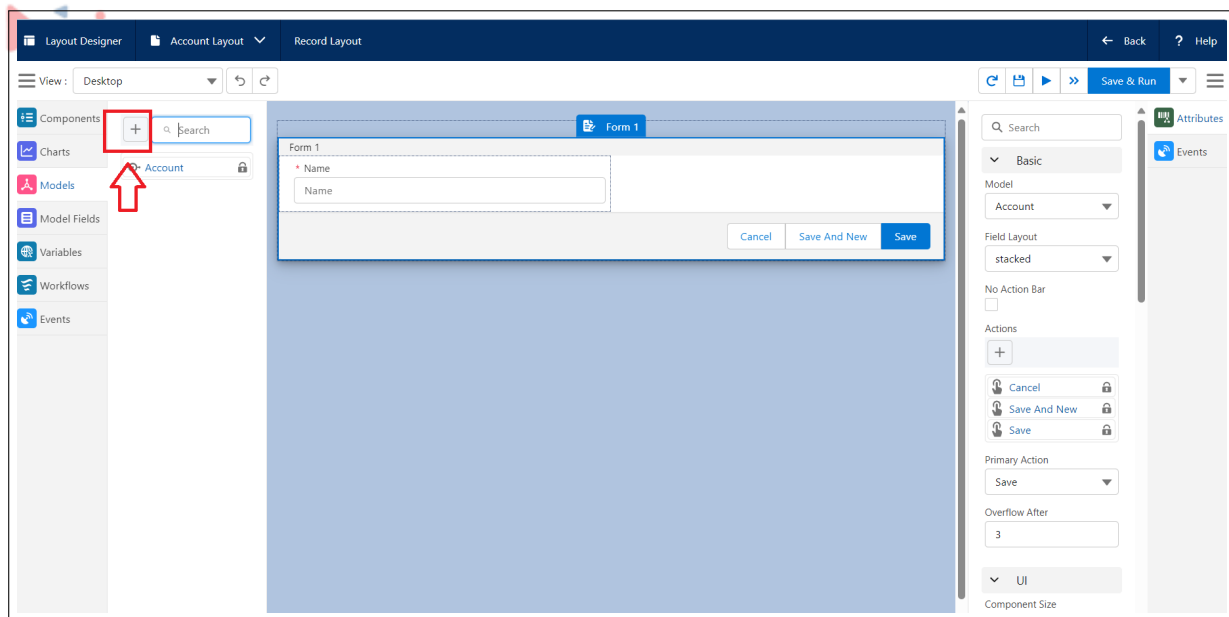


Figure 5.3.101: Create Model

4. Record Count: In Models, there are two types of Record counts i.e. Single record and Multi record
 - (a) single: We use single type record count, where you want to query a single record eg. For Form components, we use single types.
 - (b) multirecord: We use multirecord type record count, where you want to query multiple records like Ex. For Table components, we use multiple types

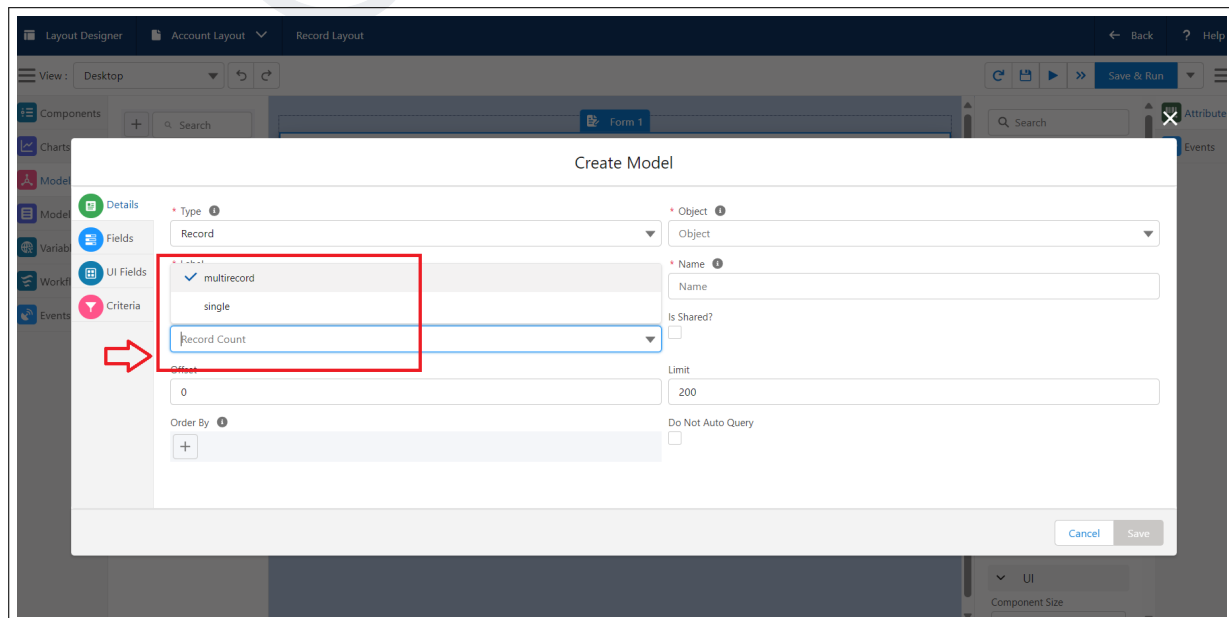


Figure 5.3.102: Record Count

5. Object Def: You can bind the object with the model



- (a) The complete list of objects in your instance is displayed
 - (b) Select the object from the drop-down menu
- 6. Offset: You can set the offset for the query on the object. If the offset is “n” then the query will take the records “n+1” onwards. e.g. Set Offset: 0, then the query will take the record 1 onwards.
- 7. Limit: Set the limit for how many records you want to display at a time.
- 8. Order By: You can set the order by on the fields to records to be queried and displayed in the Model.





V.III.1.6 Charts : A chart is a visual representation of data designed to make information easily understandable at a glance. Charts use graphical elements such as lines, bars, points, and slices to convey numerical or categorical information. They play a crucial role in data visualization, making complex datasets more accessible and aiding in the interpretation of patterns, trends, and relationships within the data.

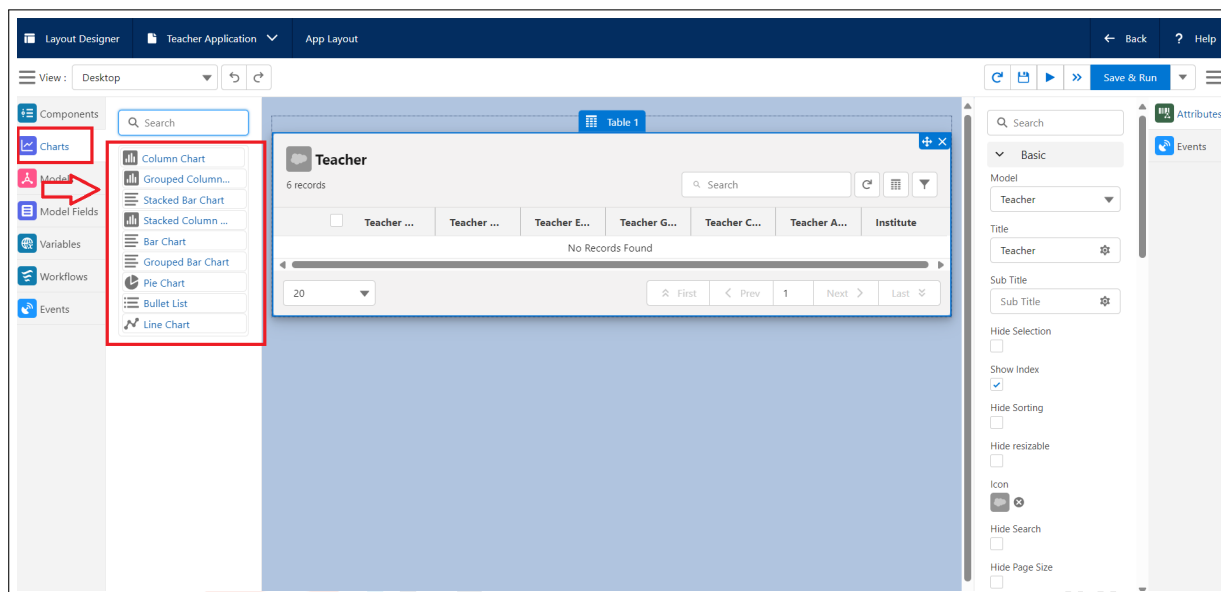


Figure 5.3.103: Charts

Types of Charts: There are nine types of chart

1. Column Chart
2. Group Column Chart
3. Stacked Bar Chart
4. Stacked Column Chart
5. Bar Chart
6. Grouped Bar Chart
7. Pie Chart
8. Bullet List
9. Line Chart

- **Column Chart:** A column chart is a type of graph that uses vertical bars to represent data values. The length of each column corresponds to the magnitude of the data it represents. Column charts are particularly effective for comparing values across different categories or displaying the distribution of a dataset. The x-axis typically represents categories, while the y-axis represents the scale of the data. Column charts are straightforward, making them a popular choice for visually presenting simple comparisons and trends in data.



- **Group Column Chart:** A grouped column chart is a type of chart that displays multiple sets of data using vertical bars grouped together side by side. Each group of columns represents a distinct category, and within each group, individual columns represent different series or subcategories. This format allows for easy comparison of values within each category as well as across different categories. Grouped column charts are useful when you want to show variations and comparisons of multiple datasets simultaneously, making it visually clear and accessible for the audience to analyze and interpret the data.
- **Stacked Bar Chart:** A stacked bar chart is a type of graph that uses horizontal bars to represent data values, with each bar divided into segments that stack on top of each other. Each segment within a bar represents a different category or component, and the full bar represents the total value for that particular category. Stacked bar charts are useful for illustrating the total and how it is divided into subcategories. They provide a visual representation of both the individual components and the overall composition of each category, making it easy to compare the contribution of each segment to the whole.
- **Stacked Column Chart:** A stacked column chart is a type of graph that uses vertical bars to represent data values, and each bar is divided into segments that stack on top of each other. Each segment within a bar represents a different category or component, and the full height of the stacked column represents the total value for that particular category.
- **Bar Chart:** A bar chart is a graphical representation of data using rectangular bars or columns to show the values of different categories. The length of each bar corresponds to the magnitude of the data it represents. The bars can be either vertical (column chart) or horizontal (bar chart), depending on the orientation.

Key features of a bar chart:

- **Categories:** The x-axis typically represents categories or labels that are being compared
 - **Values:** The y-axis represents the scale of the data, indicating the values associated with each category
 - **Bars:** Each bar is drawn for a specific category, with its length proportional to the value it represents
- **Grouped Bar Chart:** A grouped bar chart is a variation of the standard bar chart that displays multiple sets of data using bars grouped together side by side. Each group of bars represents a distinct category, and within each group, individual bars represent different series or subcategories.

Key features of a grouped bar chart:

- **Categories:** The x-axis typically represents categories or labels
- **Values:** The y-axis represents the scale of the data, indicating the values associated with each category
- **Groups:** Bars within each group are positioned next to each other, allowing for easy visual comparison of values within the same category.

This type of chart is particularly useful when you want to compare values across different categories and within each category, compare the contributions of various



subcategories or series. It provides a clear visual representation of the relationships and differences between multiple data sets

- **Pie Chart:** A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions. In a pie chart, each slice represents a proportionate part of the whole, and the size of each slice is proportional to the quantity it represents. The entire circle represents 100%, and each slice represents a percentage of that whole. Key features of a Pie chart:

- **Slices:** Each slice represents a category or component of the data being visualized
- **Proportions:** The size of each slice is proportional to the quantity it represents relative to the whole
- **Totality:** The sum of all the slices adds up to 100%, representing the entirety of the data
- **Color or Patterns:** Different colors or patterns are often used to distinguish between slices

- **Bullet List:** A bullet list is a text formatting technique used to present information in a concise and organized manner. In a bullet list, items are typically preceded by small symbols (bullets) to visually separate and highlight individual points. Each item in the list is typically a short, standalone statement or phrase.

Key features of a Bullet list :

- **Bullets:** Small symbols, often dots or circles, precede each item in the list
- **Itemization:** Each point in the list is presented as a separate, concise statement
- **Ordering:** Bullet lists are often used for unordered or non-sequential items. For ordered or sequential items, numbers or letters may be used

- **Line Chart:** A line chart is a type of graph that displays data points over a continuous interval or time span, connecting them with straight lines. This visual representation helps to show trends, patterns, and relationships between variables. Line charts are effective for illustrating changes in data over time or across ordered categories.

Key features of a Line chart:

- **Axes:** The x-axis typically represents the independent variable (such as time or categories), while the y-axis represents the dependent variable (the data values)
- **Data Points:** Each data point is marked on the chart, and a line connects these points, emphasizing the trend or pattern
- **Trends:** Line charts are particularly useful for visualizing trends, showing whether values are increasing, decreasing, or remaining relatively constant
- **Multiple Lines:** You can have multiple lines on the same chart, making it easy to compare trends between different data series
- **Markers:** Data points are often marked with dots or other symbols to enhance visibility

V.III.1.7 Variables : Variables enable users to manipulate data by performing operations, calculations, comparisons, and transformations on the stored values. Variables facilitate the passing of values between different parts of a program, such as functions, procedures, or modules. They allow data to be shared and processed across different components of a program.

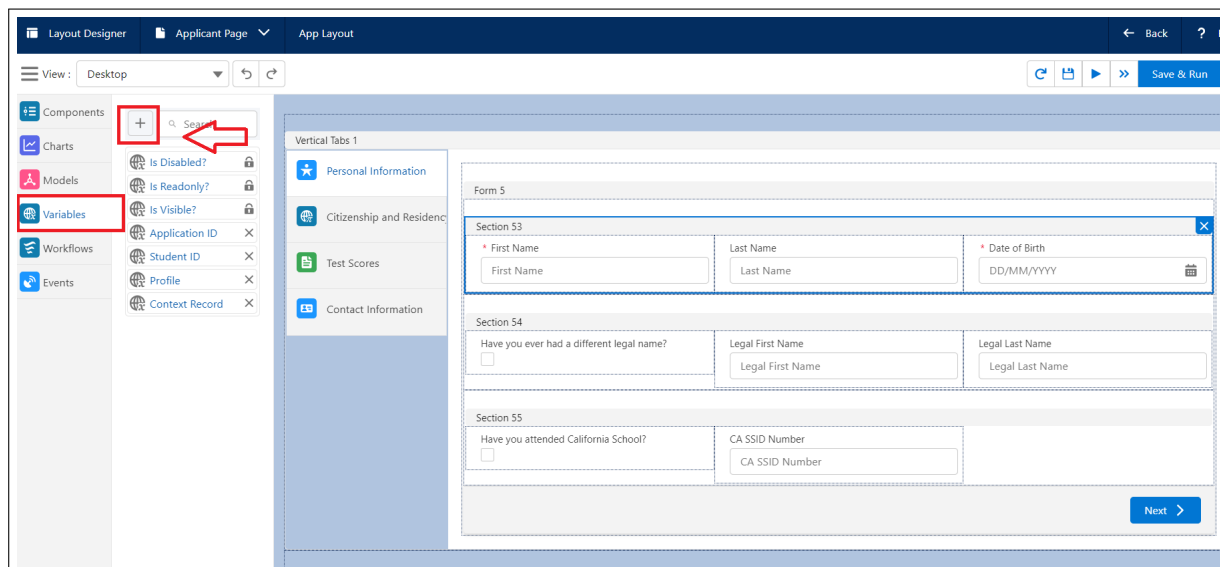


Figure 5.3.104: Variables

• **How to create Variable:**

1. Create a Layout
2. Go to the Variable tab in the Layout designer
3. Click on the + icon
4. Enter information, click Save

• **Data Types of Variable:**

1. **Text:** A data type used to store alphanumeric characters, including letters, numbers, and symbols. Text fields are versatile and can hold a wide range of information, such as names, descriptions, and comments.
2. **Checkbox:** A data type that represents a binary state, typically either checked or unchecked. Checkboxes are commonly used for boolean values or to indicate options that are selected.
3. **Integer:** A whole number without a fractional component. Integers are used to represent numerical values that do not contain decimals.
4. **Double:** A floating-point number with decimal precision. Doubles are used to represent numerical values that may have a fractional part.

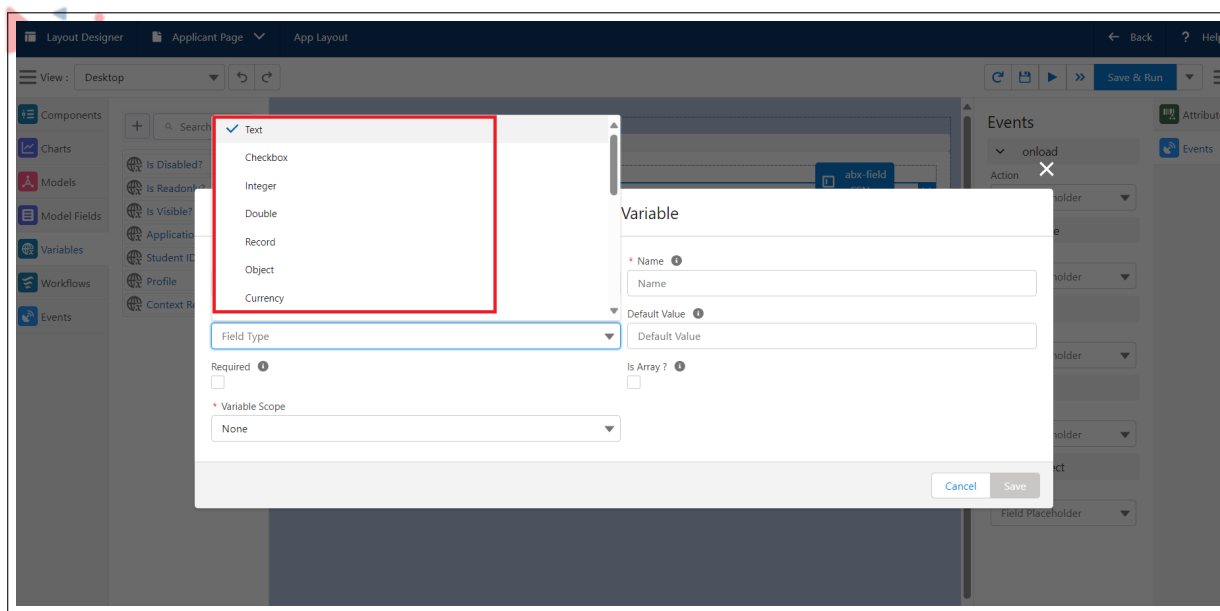


Figure 5.3.105: Field Type

5. **Record:** A collection of related data fields or attributes that represent a single entity or object. Records are commonly used in database management systems to store and organize structured data.
6. **Object:** A generic data type that can hold any type of data or a reference to an instance of a class. Objects are used for modeling complex data structures and can represent various entities or values.
7. **Currency:** A data type used to represent monetary values. Currency fields typically include formatting to denote the currency symbol and decimal precision.
8. **Date:** A data type used to represent calendar dates without a specific time zone. Dates consist of year, month, and day components.
9. **DateTime:** Similar to Date but also includes time information. DateTime data type represents a specific point in time, including both date and time components.
10. **Picklist:** Also known as a dropdown or select list, a picklist is a user interface element that allows users to choose one option from a predefined list of options.
11. **Tags:** A data type used to categorize or label records with keywords or descriptors. Tags are commonly used for organizing and filtering data.
12. **Email:** A data type used to store email addresses. Email fields often include validation rules to ensure that the entered value is a valid email address.
13. **Phone:** A data type used to store phone numbers. Phone fields often include formatting and validation rules to ensure data integrity.
14. **URL:** A data type used to store web addresses. URL fields often include validation rules to ensure that the entered value is a valid URL format.
15. **Radio:** A user interface element used to select a single option from a list of mutually exclusive options.
16. **Time:** A data type used to represent a specific time of day, independent of any particular date.



- ▶ 17. Percentage: A data type used to represent values as a percentage of a whole. Percentage fields typically store numerical values between 0 and 100.
- 18. Textarea: A data type used to store multiline text entries. Textarea fields allow users to input larger amounts of text than regular text fields.
- 19. Duration: A data type used to represent a length of time. Duration fields can store values such as hours, minutes, and seconds.
- 20. Multipicklist: Similar to a picklist, a multipicklist allows users to select multiple options from a predefined list of choices.



V.III.1.8 Workflow : Workflow is basically a container that automates certain actions based on particular criteria. If the criteria are met, the actions get executed. If they are not met, then no action will be executed. There are multiple actions in the Workflow. Eg: Push, Pop, Toaster, Show Spinner, etc.

- Workflow Creation:
 1. Go to the Workflow tab
 2. Click the + icon
 3. A new blank workflow screen opens
 4. Need to add all the details and, Click Save

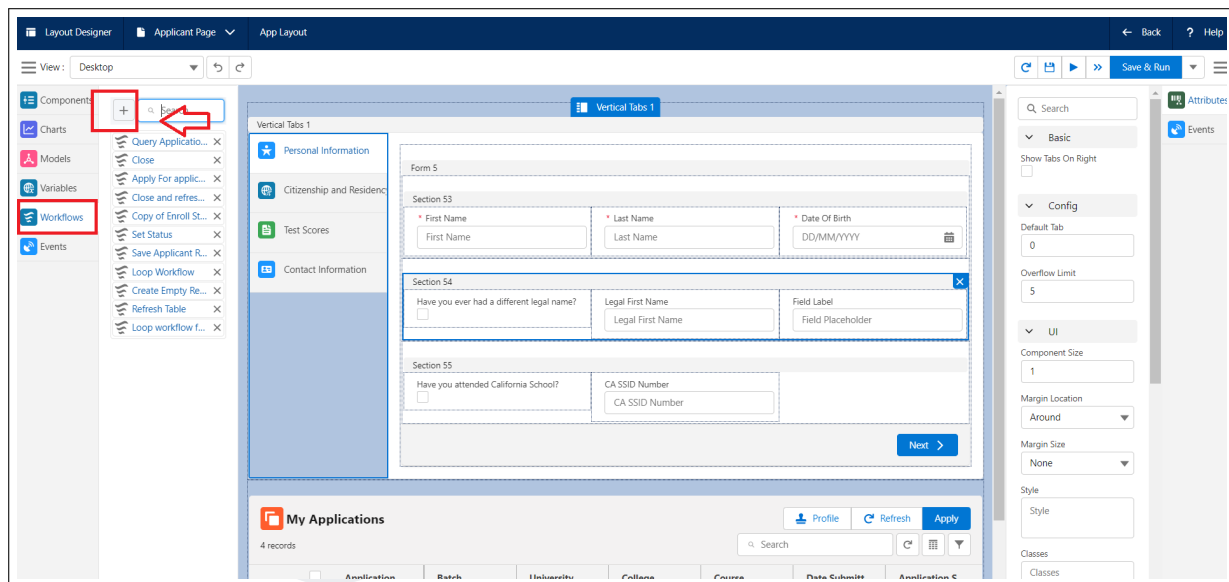


Figure 5.3.106: Workflows

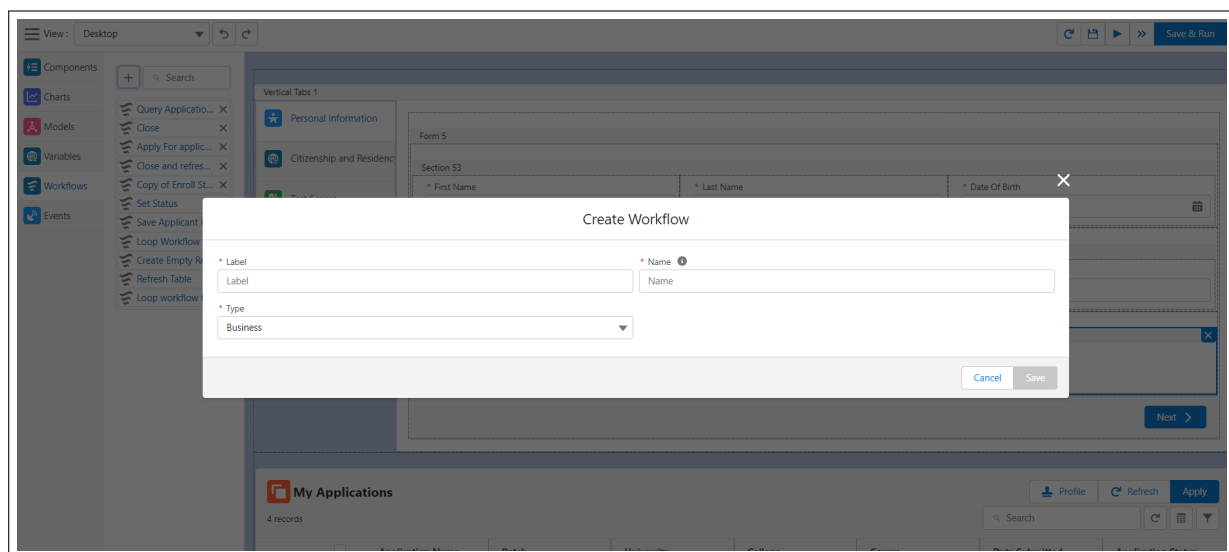


Figure 5.3.107: Create Workflow



- **Logic in Workflow:** There are several logic available in workflow, which can be used to create workflows.
 1. **Assignment:** This is the basic operation of assigning a value to a variable. For example, $x = 5$ assigns the value 5 to the variable x .
 2. **Decision:** Decisions, or conditional statements, allow the program to execute different blocks of code depending on whether a certain condition is true or false. Examples include if, else if, and else statements.
 3. **Switch:** A switch statement provides an alternative way to express multiple branching. It allows a variable to be tested for equality against a list of values. Depending on the matching value, a different block of code can be executed.
 4. **Loop:** Loops are used to repeat a block of code multiple times until a certain condition is met. Common types of loops include for, while, and do-while loops.
 5. **Validate:** Validation involves checking whether data meets certain requirements or constraints. This could include checking for the correct data type, length, format, or range.

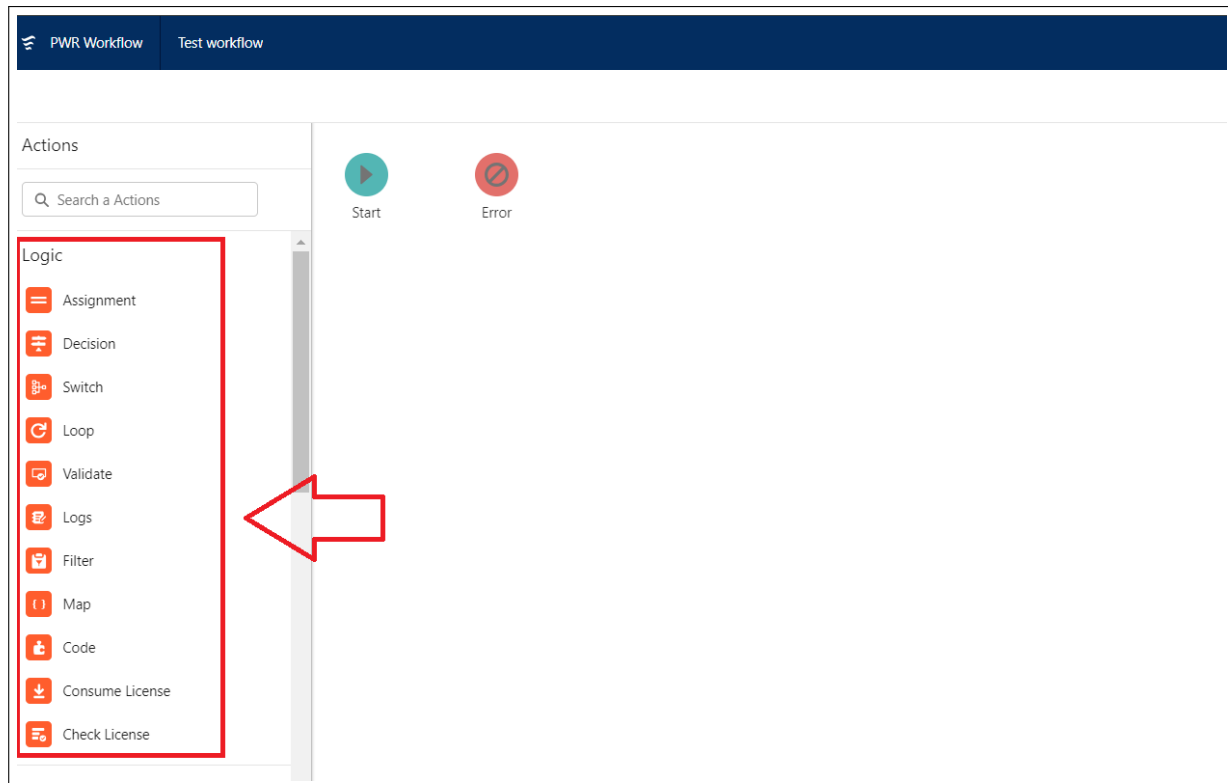


Figure 5.3.108: Workflow Logics

6. **Logs:** Logging involves recording events, messages, or data during the execution of a program. Logs are valuable for debugging, monitoring, and analyzing the behavior of the software.
7. **Filter:** Filtering involves selecting a subset of items from a larger set based on certain criteria or conditions. For example, filtering a list of numbers to only include even numbers.

8. Map: Mapping involves transforming each element of a collection based on a given function. It applies the function to each element and generates a new collection with the transformed values.
 9. Code: The term "code" refers to the instructions written in a programming language that define the behavior of a software application.
 10. Consume License: This refers to the process of using or activating a software license to gain access to a particular software product or service.
 11. Check License: Checking a license typically involves verifying whether a user or system is authorized to access or use a software product. This could involve validating license keys, checking subscription status, or verifying user permissions.
- Communication in Workflow: For Communication, there are several actions available in Workflows that can be used for communication with internal and external sources.

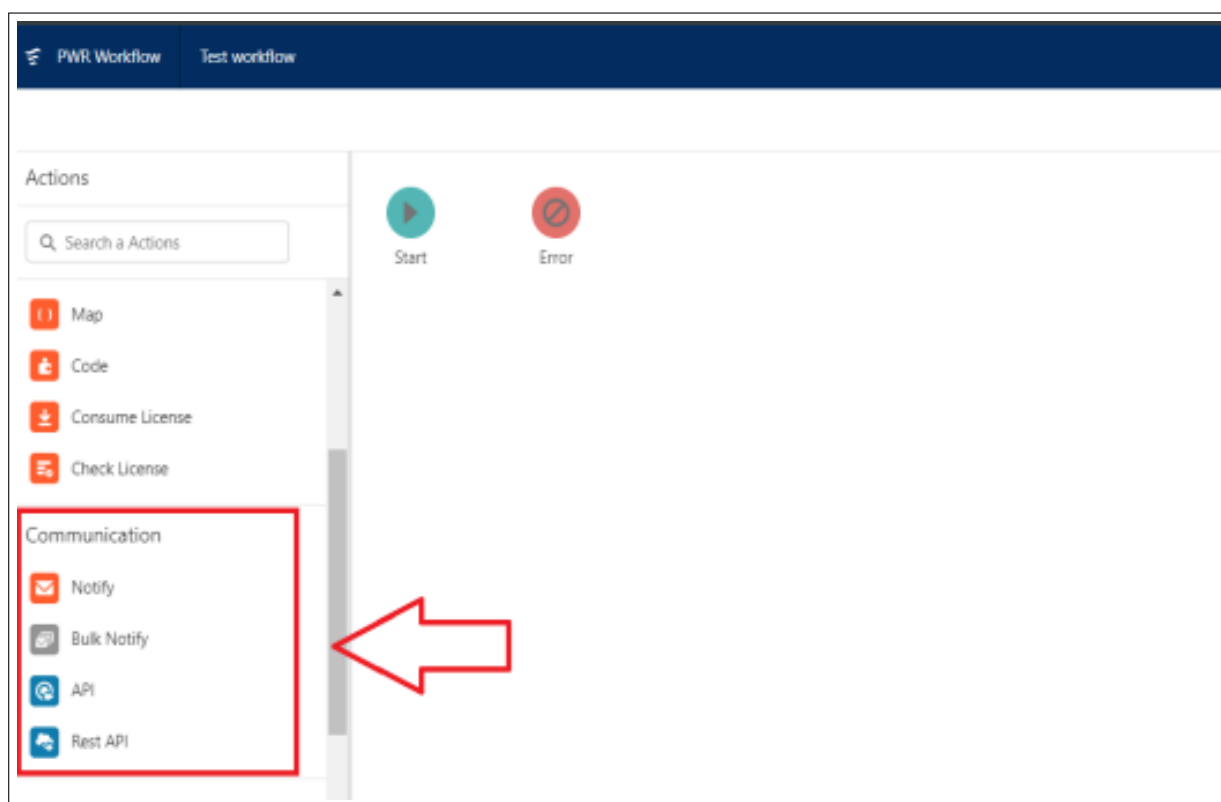


Figure 5.3.109: Workflow Communication actions

1. Notify: In workflows, notifications are used to alert users or relevant stakeholders about specific events, updates, or tasks within the process. These notifications can be sent through various channels such as email, SMS, in-app messages, or push notifications. For example, a workflow management system might notify team members when a new task is assigned to them or when a deadline is approaching.
2. Bulk Notify: Bulk notifications are particularly useful when you need to inform a large group of recipients about the same event or update. Instead of sending individual notifications to each recipient, bulk notifications allow you to send a single message to multiple recipients simultaneously.



3. API (Application Programming Interface): APIs play a crucial role in workflows by enabling different software systems or services to communicate with each other. In workflow automation, APIs can be used to integrate different tools, platforms, or applications, allowing them to exchange data and trigger actions seamlessly. For example, an e-commerce workflow might use APIs to retrieve the product information from a database, process payments through a payment gateway, or update inventory levels in real time.
 4. REST API (Representational State Transfer Application Programming Interface): RESTful APIs follow the principles of REST architecture and are commonly used for building web services that allow clients to interact with server resources using standard HTTP methods. In workflows, REST APIs can be used to access and manipulate data stored on remote servers or cloud services. For example, a workflow might use a REST API to retrieve weather information from a third-party service, fetch data from a social media platform, or perform operations on a cloud-based storage system.
- Data Actions in Workflow:
 1. Query Records: This action involves retrieving specific data records from a database or data source based on predefined criteria or conditions. In workflows, querying records is often used to gather information needed for further processing or analysis. For example, in a customer relationship management (CRM) system, a workflow might query customer records to identify those who haven't made a purchase in the last six months for a re-engagement campaign.
 2. Create Records: Creating records involves adding new data entries to a database or data repository. In workflows, creating records is typically used to capture and store information generated during the execution of a process. For instance, in an online registration workflow, new user accounts could be created and added to a user database when individuals sign up for a service or event.

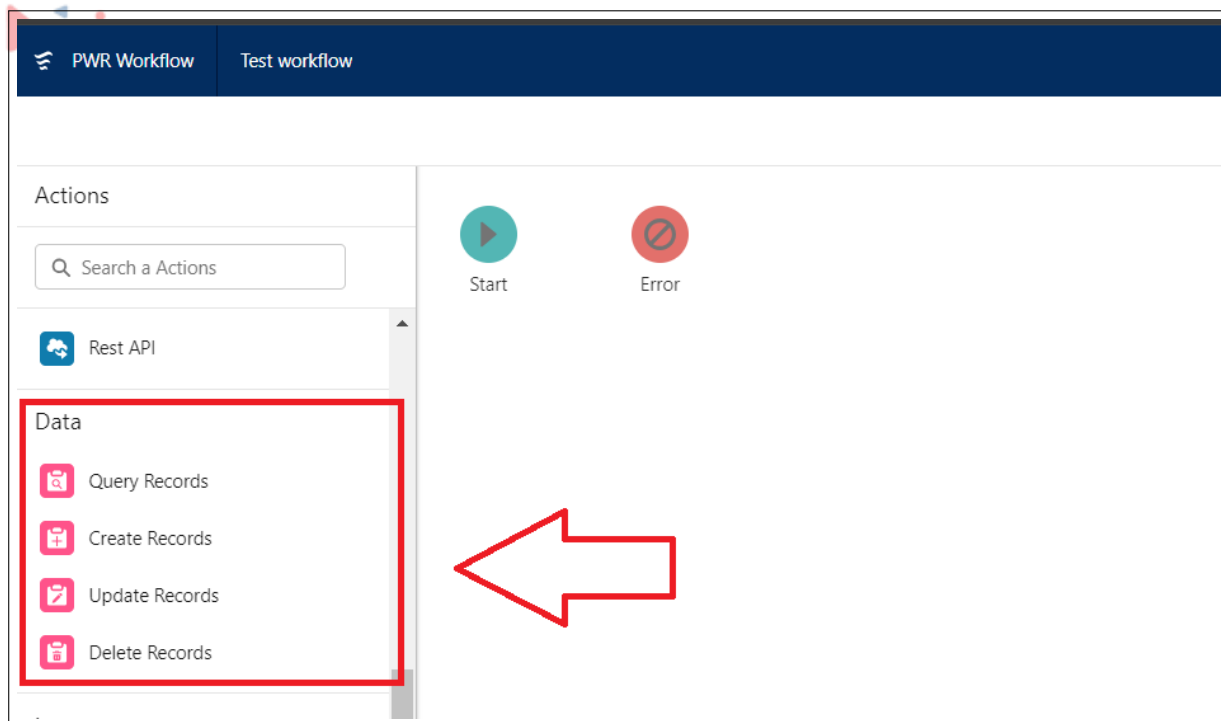


Figure 5.3.110: Workflow Data actions

3. **Update Records:** Updating records involves modifying existing data entries in a database or data source. In workflows, updating records is often used to reflect changes or updates to information over time. For example, in an inventory management system, a workflow might update product quantities to reflect recent sales or incoming shipments.
4. **Delete Records:** Deleting records involves removing data entries from a database or data repository. In workflows, deleting records is used to eliminate outdated, redundant, or unnecessary information. For instance, in a document management system, a workflow might delete expired documents or files that are no longer needed.

- **Layout Actions in Workflow:**

1. **Emit Event:** Emit Event is used for communication between two layouts or two workflows by Emitting Events from the source Layout. emit event is fired on any events like click event on Button or any other events. For these Events, we have to create Events on Layouts. Through Emit Event you can pass output (Parameter)
 - ☐ **Subscribe Event:** Subscribe Event is used to communicate between two Layouts or two workflows by Listening to an Event Emitted by the source Layout. For Subscribe Event you have to select Layout and respective Event. You can assign a Workflow that fire/emit an Event. Through Subscribe Event you can catch Input(Parameter)

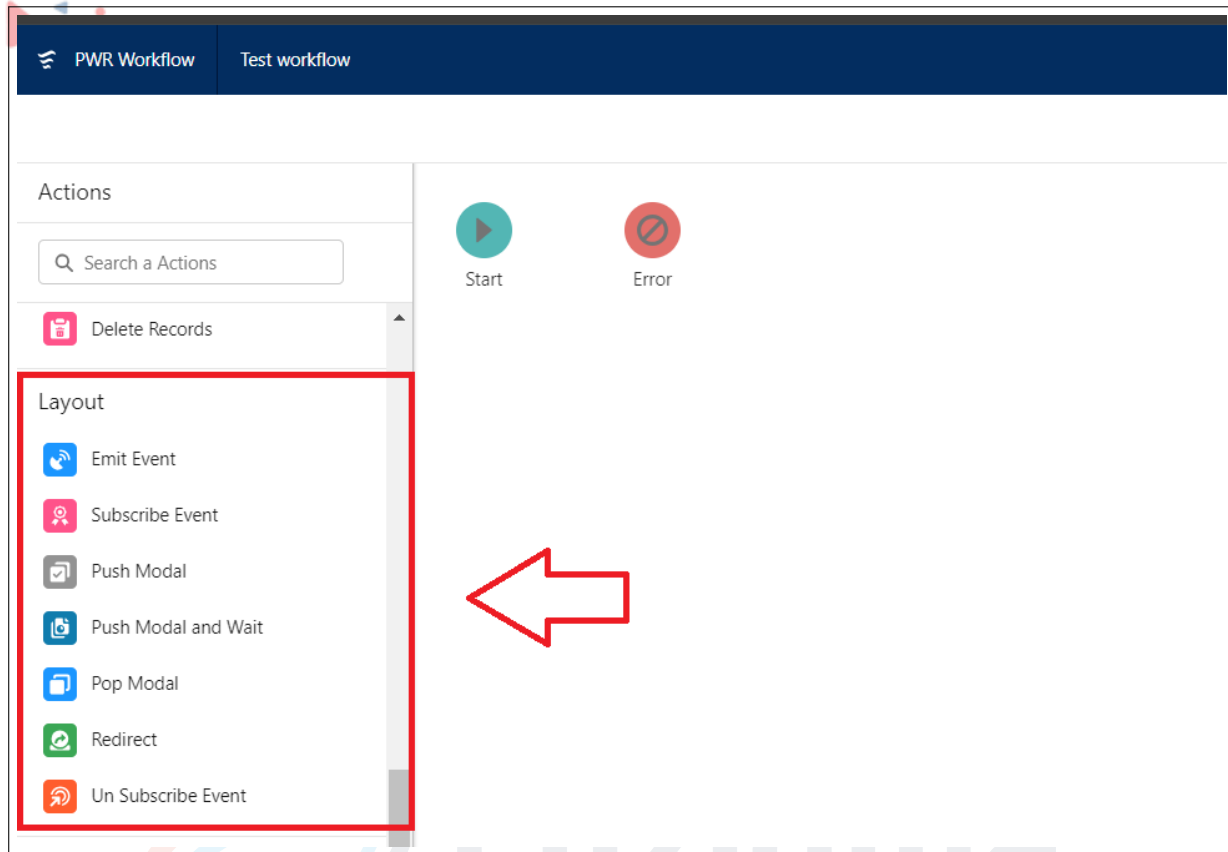


Figure 5.3.111: Workflow layout actions

2. Push Modal: Push Modal is a container that contains fields. This modal lets you create a record. Push Modal is created on the Application layout.
3. Push Modal and Wait: Push Modal and Wait is used to open the Layout in Modal(Pop Up) to perform various actions.e.g.for Create/Updating a new record, viewing a record, etc. this action preserves the next action until the Modal is closed(Pop Modal). You can configure modals with different sizes, Padding, and Header and are also able to pass parameters through Input.
4. Pop Modal: Pop modal is used to push back Layout when we save or cancel a record.
5. Redirect: Users can be redirected to layouts or to any URL using a redirect workflow
6. Un Subscribe Event: The Un Subscribe Event is used to stop subscription of any particular event

V.III.1.9 Listview Creation and its Config : List View is used for showing the records of different objects from where the user can perform CRUD operations. List View is an easily configurable component that can be used to display the ListView on the Home Page and Record Page. List View uses the Standard List View and the User can configure it in order to display it on the UI by adding extra fields to the existing List View.

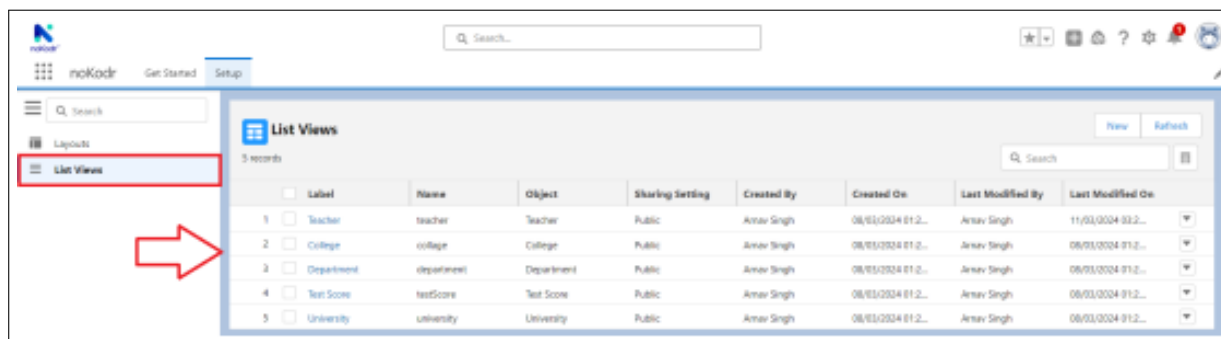


Figure 5.3.112: List View

- ListView Homepage Operations:

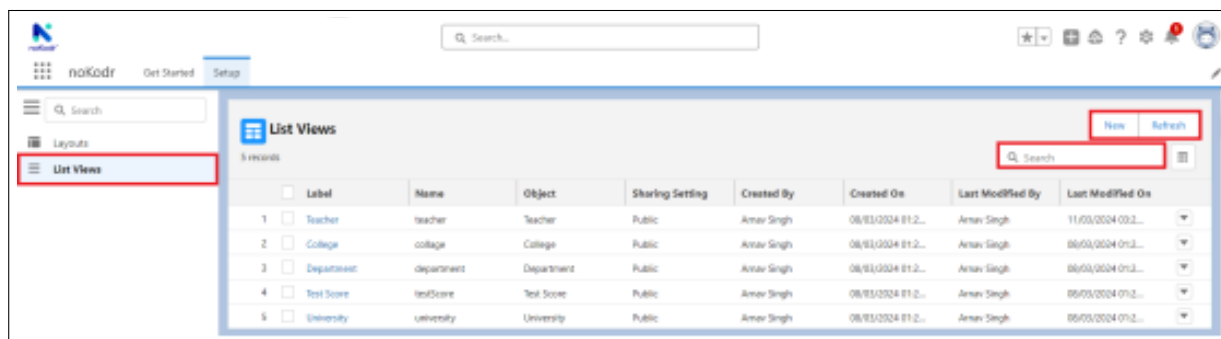


Figure 5.3.113: List View Operations

1. New: Here are the steps required to configure List View. To create a List View, Click on Setup, then click on List View then navigate to the List View page.
2. Refresh: A refresh button is often used to update the content of a Listview interface with fresh data. refresh button gives users a sense of control over the application's behavior.
3. Search: Select the field in which you want to search the records in the list view

- List View Actions:

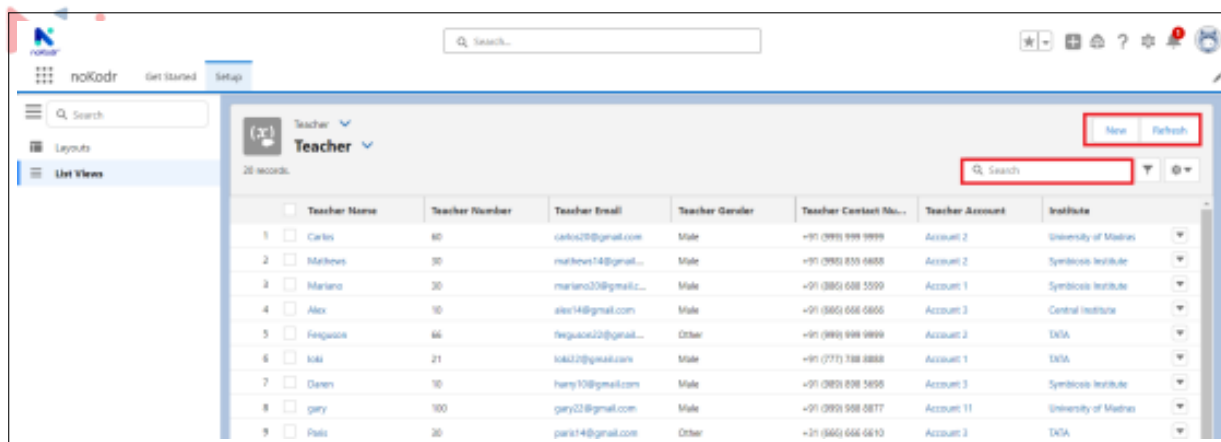


Figure 5.3.114: Listview page action

You can only show two actions, which are visible in the top right corner i.e. New, and Refresh. These actions are mainly used to perform operations

1. New: To create a New record you can use the New action
 2. Refresh: To Refresh the record you can use the Refresh action
 3. Search: Select the field in which you want to search the records in the list view
- Filter Designer: The filter is used to filter the records as per the given criteria. Filter that reads data in and manipulates the data to fit another output pattern or removes data that may not be needed.

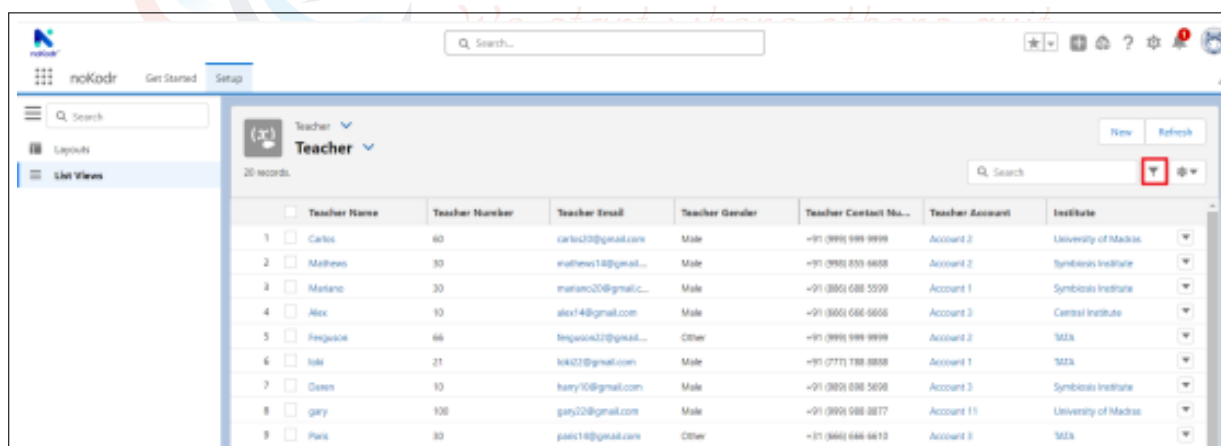


Figure 5.3.115: Filter Designer

Logical Operators: Logical operators are used to combine multiple conditions.

1. AND: The logical AND operator is an operator that performs a logical combination of two statements. It receives the value "TRUE" only when both statements are true. If one of the two statements is false, the logical AND operator returns the value "FALSE".
2. OR: The OR operator is a Boolean operator that returns TRUE if one or both operands are TRUE



3. NOT: Operators that can be used to create a new compound statement from two or more statements. It reverses the truth value of any statement it appears before.

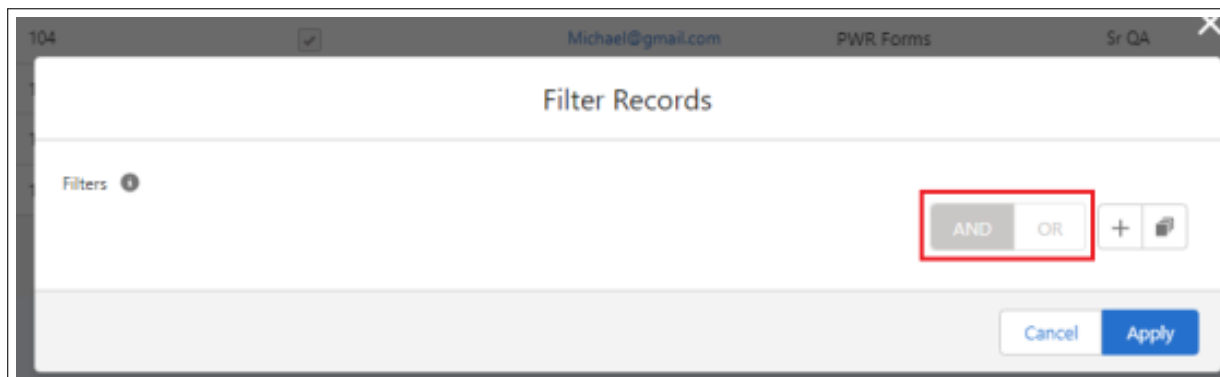


Figure 5.3.116: Workflows



Figure 5.3.117: Workflows

- Add Condition: This is used to add multiple conditions using the “+” icon



Figure 5.3.118: Add Condition

- Add Group Condition: It is used to add multiple conditions to a group using the “Add Group” icon

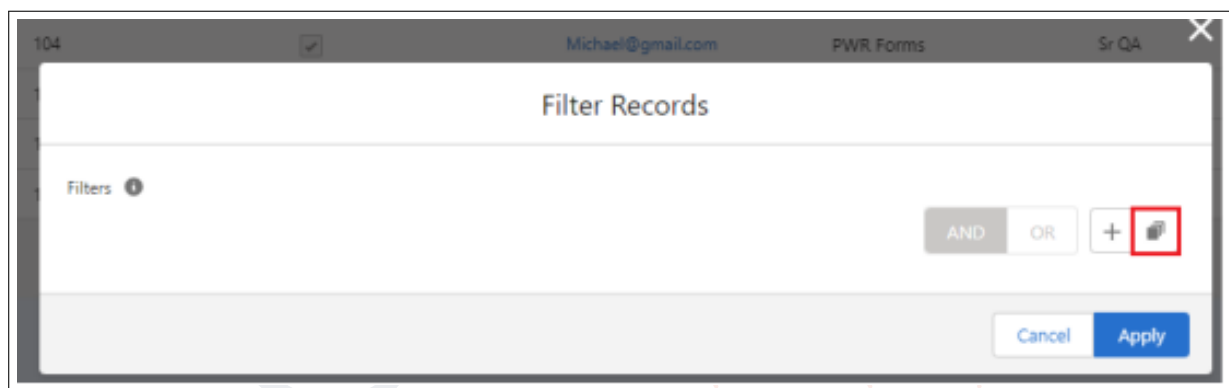


Figure 5.3.119: Workflows

- Filter Operators:

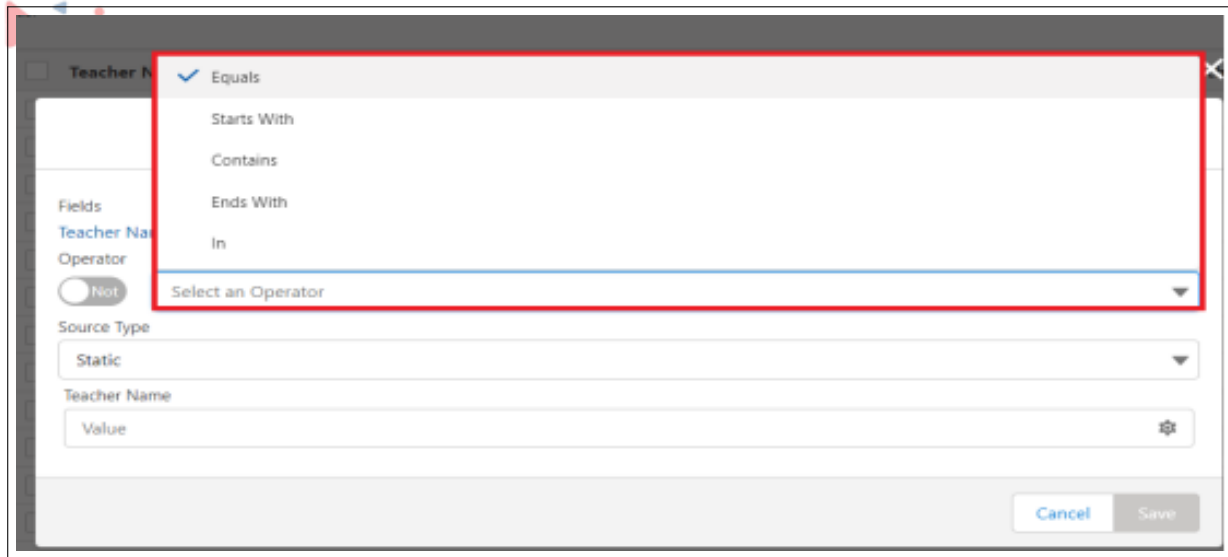


Figure 5.3.120: Filter Operators

1. Equals: The equal operator is used to compare two values or expressions. It is used to compare numbers, strings, Boolean values, variables, objects, etc. The result is TRUE if the expressions are equal otherwise it's FALSE.
 2. Start With: It returns TRUE if a string or number starts with the specified character otherwise it returns FALSE
 3. Contains: The contains operator returns TRUE if the value on the left contains the value on the right, and otherwise FALSE.
 4. End With: It returns TRUE if a string or number ends with the specified character otherwise it returns FALSE
 5. IN: IN operator allows you to easily test if the expression matches any value in the list of values. Determines whether the value of an expression is equal to any of several values in a specified list.
- Source Type:

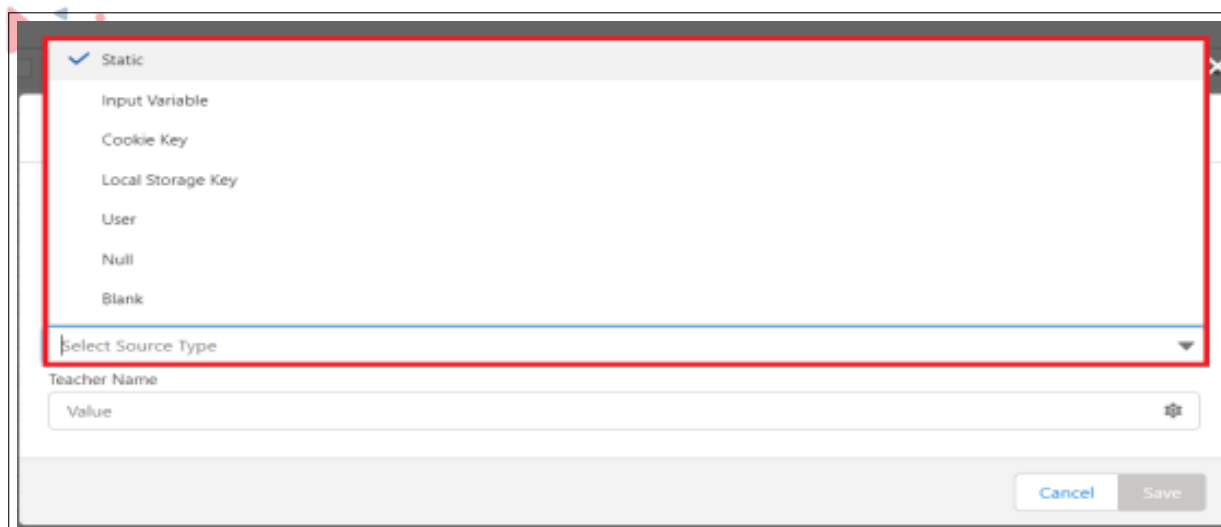


Figure 5.3.121: Source Type

1. Static: By using the Static source type, results include only records from the selected field. Result fetching by the value which has been added under the Static. Steps To Do:
 - (a) Click on “Filter”
 - (b) Select the [+] icon to create a New Condition.
 - (c) Select an appropriate “Field & Operator” from the drop-down.
 - (d) Choose the Source Type as a “Static”.
 - (e) Input value which needs to fetch in result & then “Save”.
2. Input Variable: By using an Input Variable, the result is fetched by the value that has been added under the Input Variable, But have to make sure that the value that we are adding is from the value of the input variable. Steps To Do:
 - (a) We need to create an input variable first.
 - (b) Click on “Filter”
 - (c) Select the [+] icon to create a New Condition.
 - (d) Select an appropriate “Field & Operator” from the drop-down.
 - (e) Choose the Source Type as an “Input Variable”.
 - (f) Select an input variable from the drop-down that needs to fetch in the result
 - (g) Click on Save
3. Cookie Key: By using Cookie Key, the result is fetched by the value that has been added under the Cookie Key. But, have to make sure that the value that we are adding is from the value of the cookie. Steps To Do:
 - (a) Click on “Filter”
 - (b) Select the [+] icon to create a New Condition.
 - (c) Select an appropriate “Field & Operator” from the drop-down.
 - (d) Choose Source Type as a “Cookie Key”.
 - (e) Open the Inspect by right-clicking on the respected object’s list view.
 - (f) Find the “Cookies” from the Application tab.



- (g) Paste Cookies value which needs to fetch in result & then “Save”.
- 4. Local Storage Key: By using the Local Storage Key, the result is fetched by the value that has been added under the Local Storage Key. But, have to make sure that the value that we are adding is from the Local storage value. Steps To Do:
 - (a) Click on “Filter”
 - (b) Select the [+] icon to create a New Condition.
 - (c) Select an appropriate “Field & Operator” from the drop-down.
 - (d) Choose the Source Type as a “Local Storage Key”.
 - (e) Open the Inspect by right-clicking on the respected object’s list view.
 - (f) Find the “Local Storage” from the Application tab.
 - (g) Paste in the value that needs to fetch in the result & then “Save”
- 5. User: By using the User source type, the result is fetched by the value that has been added under the User. Steps To Do:
 - (a) Click on “Filter”
 - (b) Select the [+] icon to create a New Condition
 - (c) Select an appropriate “Field & Operator” from the drop-down
 - (d) Choose the Source Type as a “User”
 - (e) Select a value from the drop-down that needs to fetch in the result & then save
- 6. Null: By using the Null source type, the result is fetched by the value that has been Null Value. Steps To Do:
 - (a) Click on “Filter”
 - (b) Select the [+] icon to create a New Condition.
 - (c) Select an appropriate “Field & Operator” from the drop-down
 - (d) Choose the Source Type as a “Null”
 - (e) Select a value from the drop-down which needs to fetch in the result & then Save
- 7. Blank: By using the Blank source type, the result is fetched by the value that has been Blank Value.
Steps To Do:
 - (a) Click on “Filter”
 - (b) Select the [+] icon to create a New Condition.
 - (c) Select an appropriate “Field & Operator” from the drop-down
 - (d) Choose the Source Type as a “Blank”
 - (e) Select a value from the drop-down that needs to fetch in the result & then Save

• **List View Control:**

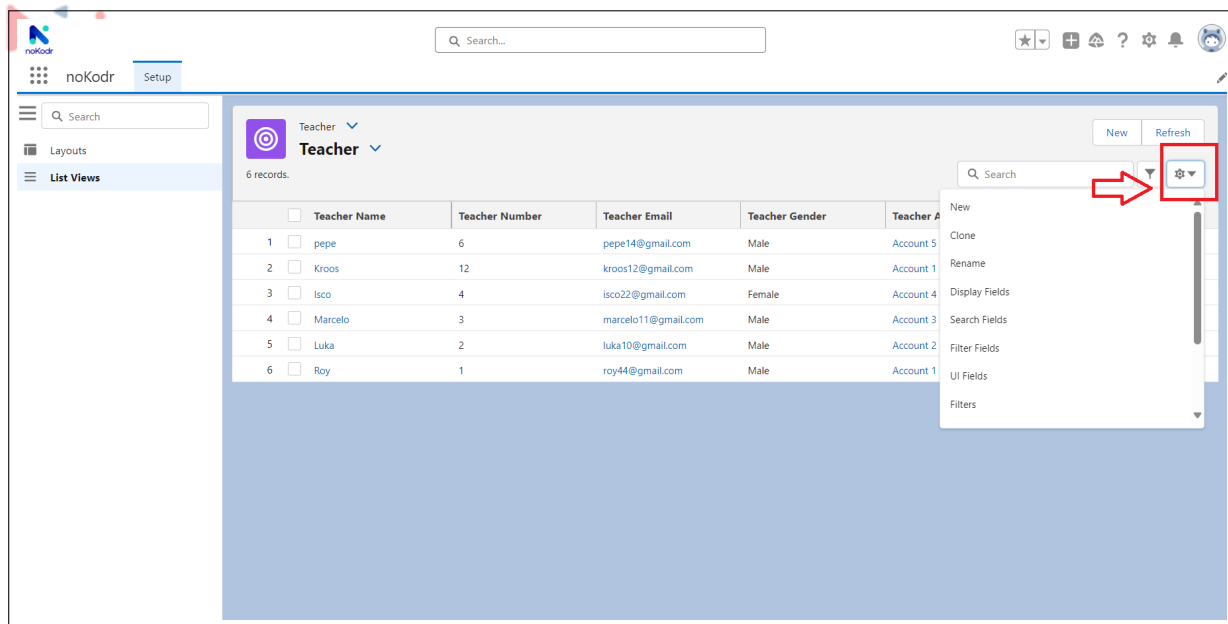


Figure 5.3.122: List View Control

1. New: Under List View Controls, select New to create a new of the current list view
2. Clone: Under List View Controls, click Clone to create a copy of the current list view
3. Rename: Edit the label of the list view, and then save your changes
4. Display Fields: Users can display the fields according to their requirements, With just by single click users can switch the order of the fields on List View
5. Search Fields: The user can search any fields that are displayed on List View by using the search box when configuring from search fields, just by single click.
6. Filter Fields: Select the field you want to apply the filter to in the list view and Click Save.
7. UI Field: Using UI fields the user can display the fields in the List view on the home page. UI Fields created by the user as per their requirements, after creating any UI Fields, all those UI fields are visible in the Display Fields, and from display fields, the user can display the fields on the home page
8. Filters: Users can give the filters as per their requirements, for any of the fields. this is a permanent filter used to sort records
9. Delete: To delete the current list view

• **List View Controls Config:**

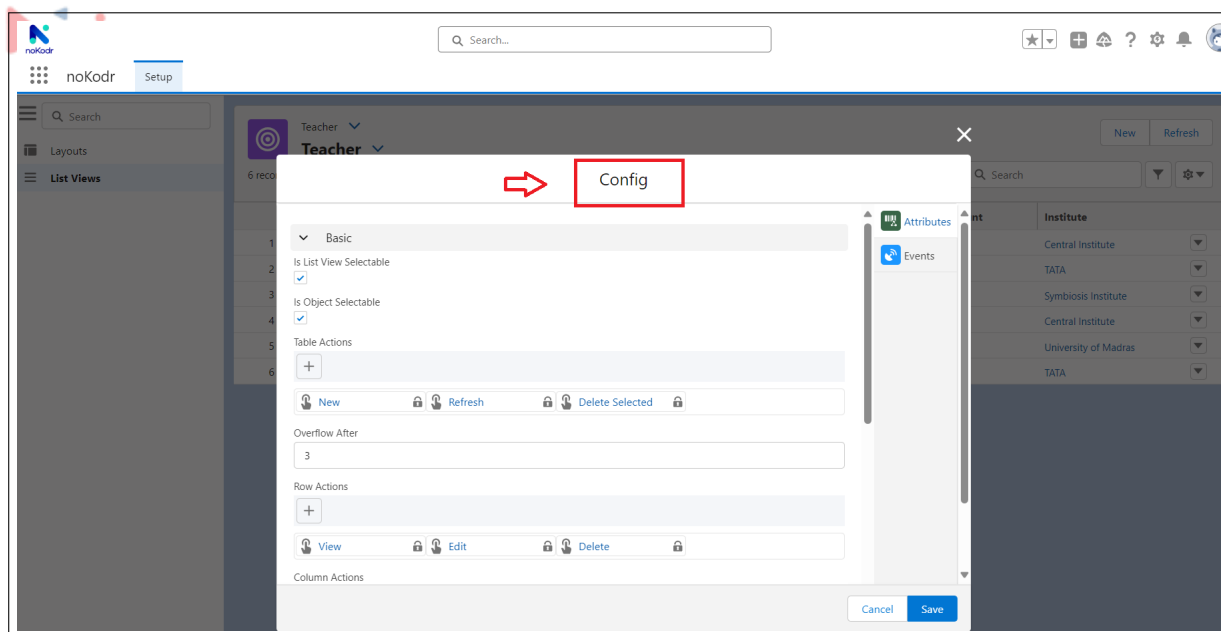


Figure 5.3.123: List View Config

- **Basic:**

- **Is List View Selectable:** This allows the end user to change the List View at runtime. If selected it shows a drop-down icon next to the object selector.
- **Is Object Selectable:** This allows the end user to change the object at runtime. If selected it shows a drop-down icon next to the List View selector.
- **Table actions:** You can create multiple table actions on the List View. You can only show two actions, which are visible in the top right corner, and the rest of the actions are displayed under a drop-down at the right. These actions are mainly used to perform operations such as create, refresh, etc. on the selected records or unselected records. Each action has a workflow that actually performs an operation and the following are the steps to create Table Action-
 - * Add the table actions by clicking on the '+' icon
 - * Fill in all the details in the Create Table Action model and click the Save button
 - * For table actions, you need to create a workflow
- **Overflow After:** Overflow After an attribute is used to display the actions in list format after reaching its entered limit. By default, the value is 3 which means the three actions will displayed on a section header. If you added the new action despite having 3 actions then the new action will appear in the drop-down list section.
- **Row Actions:** These Actions are to be performed at the record level for the respective record. These actions are mainly used to read, update, or delete a single record. Each action has a workflow that actually performs an operation. Default row actions are edit, delete, and view, but you can create more actions as per your needs as shown below and by following the below steps you can create Row Action-
 - * You can add the row actions by clicking on the '+' icon
 - * Fill in all the details in the Create Row Action model and click the Save button



- * For row actions, you need to create a workflow
- Column Actions: You can set the actions at the column level on the List View. You can assign only one action to any column. When you click record in the column action gets executed. Each action has a workflow that actually performs an operation. When you set column action, that record in the cell becomes a link. When the user clicks such a record, the action bound to that column gets executed, and the following are the Steps to create Column Action:
 - * You can add the column actions by clicking on the '+' icon
 - * Fill in all the details in the Create Column Action model and click the Save button
- Layout For New: You can set the layout for table action New. When you click the New button layout that you set will open.
- Layout For Edit: You can set the layout for row action Edit. When you click Edit from the row action layout that you set will open.
- Is Same As New: Check this checkbox, if you want to keep the same layout for new and edit.
- Offset: You can set the offset for the query on the object. If the offset is "n" then the query will take the records "n+1" onwards. For e.g. if there are 200 records and you set offset 100 then it will show records from the 101st record on the List View.
- Limit: Number of the records to be queried at once. e.g. if there are 200 records of an object and you set a limit of 100 then it will query the first 100 records and show them on the List View.
- Order By: You can set the order by on the fields to records to be queried and displayed in the List View. You can order records in ascending or descending manner.
- Config:
 - Icon: You can select the icon to be displayed on the List View. Generally icon displays on the left side of the header and subheader. You can also remove the icon by clicking on "x" which appears on the icon
 - Show Index: To show the index of records on the List View
 - Is Export Supported?: If you check that checkbox then the download icon will appear on the right side of the search box
 - Sharing Settings: The user can give the sharing setting to that particular List view. There are three types of sharing settings as:
 - * Only Me: Only users can see the List View.
 - * Public: All users can see the List View.
 - * Specific User and Profiles: Only the particular user can see the list view, depending on which users and profiles they give access to.

V.III.1.10 Publish Layout and List Views : noKodr gives the option to Publish Layout and List Views. Publishing helps the users to use Layouts and List Views on the Home Page or Community, without publishing you cannot use Layouts and List Views on the Home Page and Community.

1. Layout:

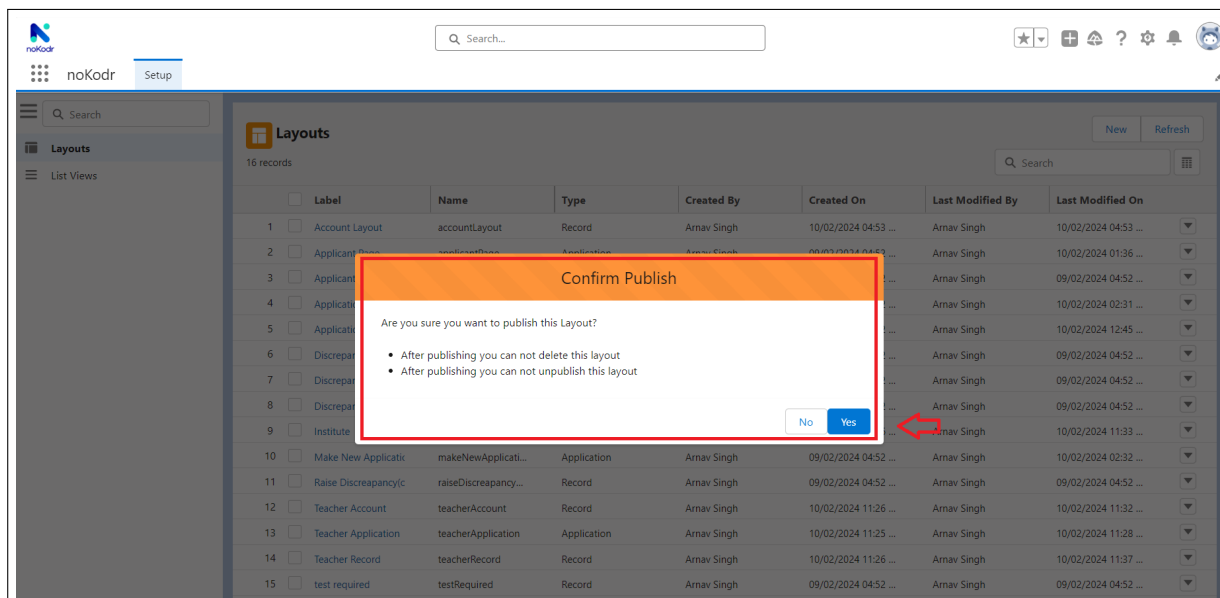


Figure 5.3.124: Publish Layout

2. Listview:

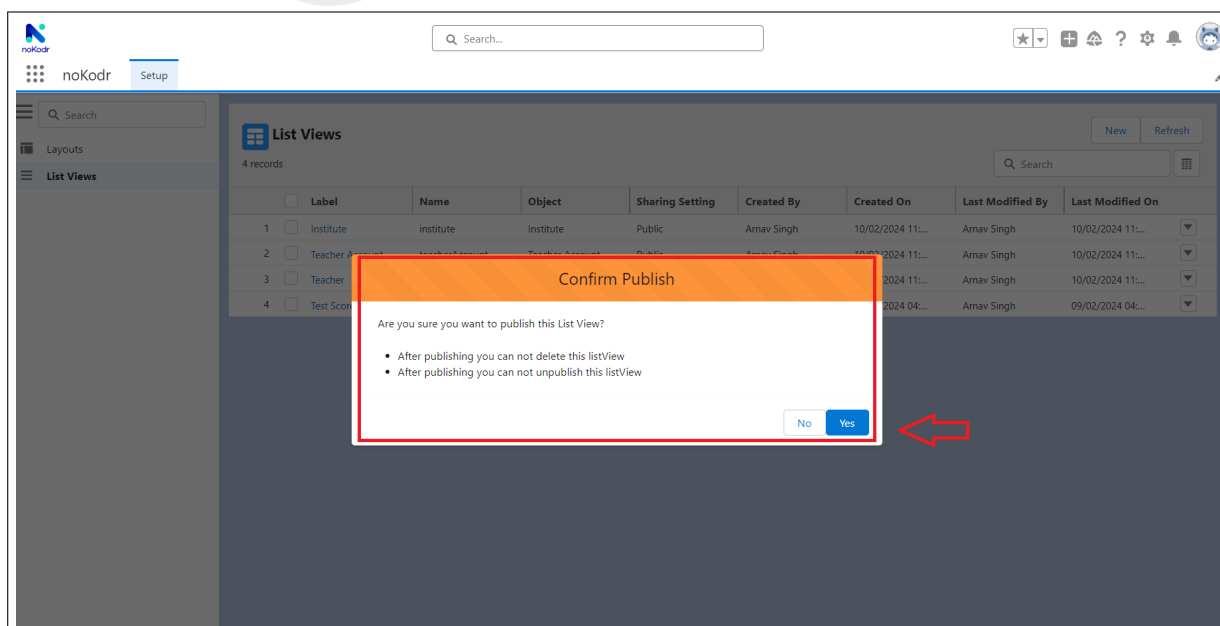


Figure 5.3.125: Publish Listview



V.III.1.11 Add Publish Layouts & List Views to Flexi Pages : Configuration for Publishing the Layouts by going to the Layouts row actions, and clicking on Publish. (Note: Once published the layout then it cannot be unpublished.)

- **Lightning Home Page:** Only the Published layouts and List Views should be able to drag and drop on the home page.
- Configuration to add Layouts and List Views on the Home Page:
 - Go to the Home page, click on the setup icon, and go to the edit page.

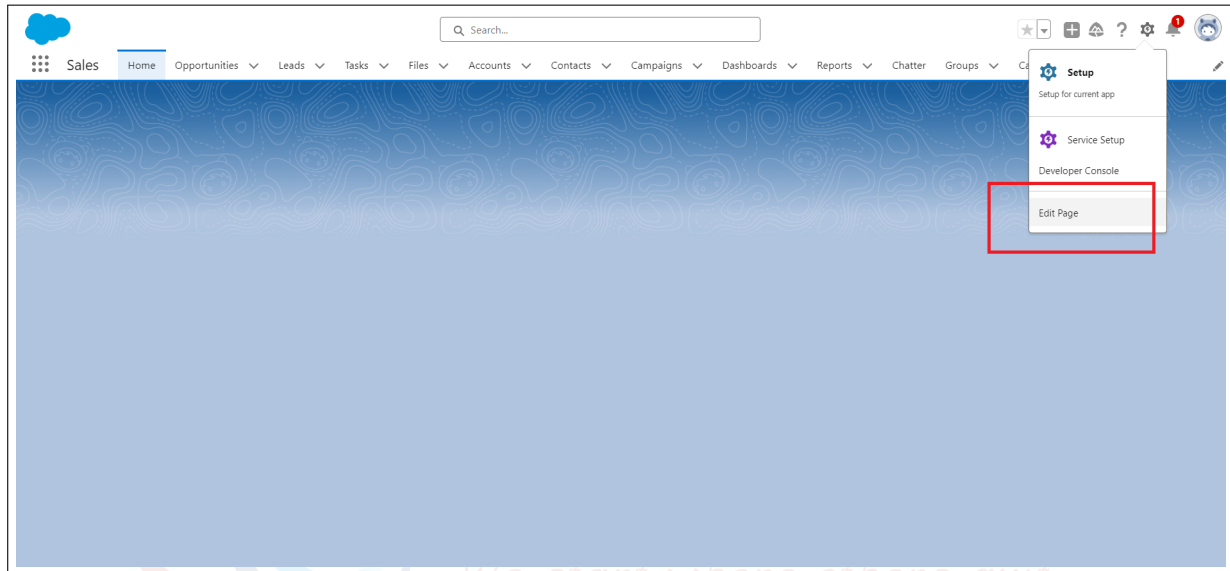


Figure 5.3.126: Edit Page

- On the edit page need to drag and drop the Layout and List view in the respective components.

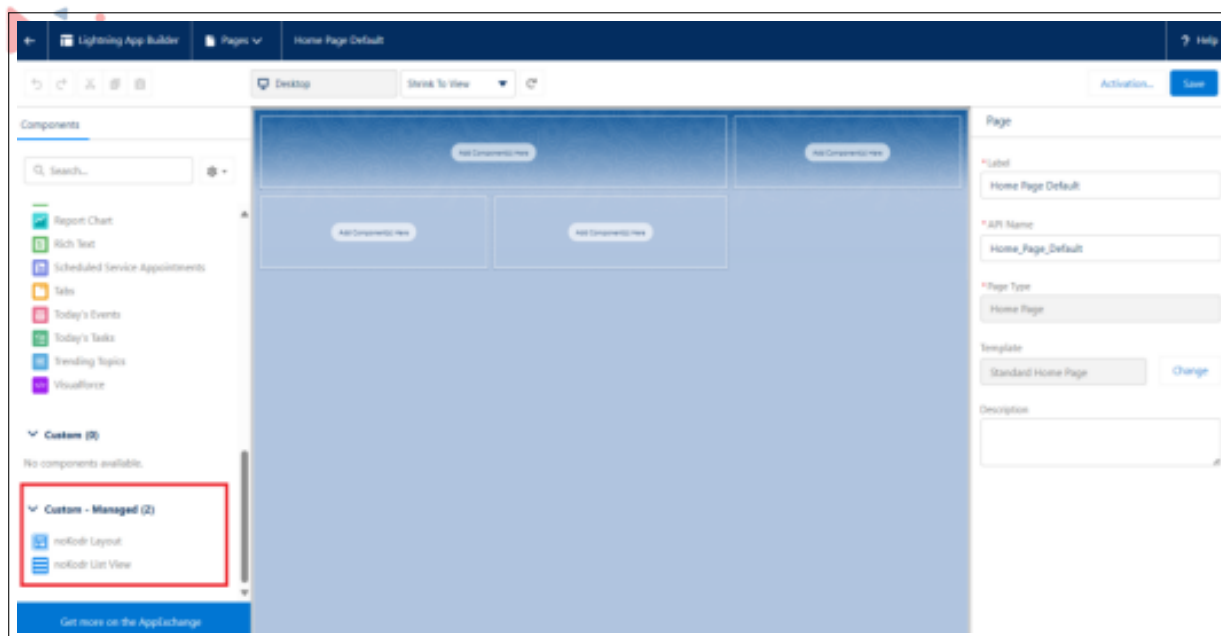


Figure 5.3.127: Custom Component

- On the edit page need to drag and drop the Layout and Listview from the dropdown where you will get the list of all the published Layouts and List Views. From where you can perform CRUD operation on Layout and List View on Home Page.

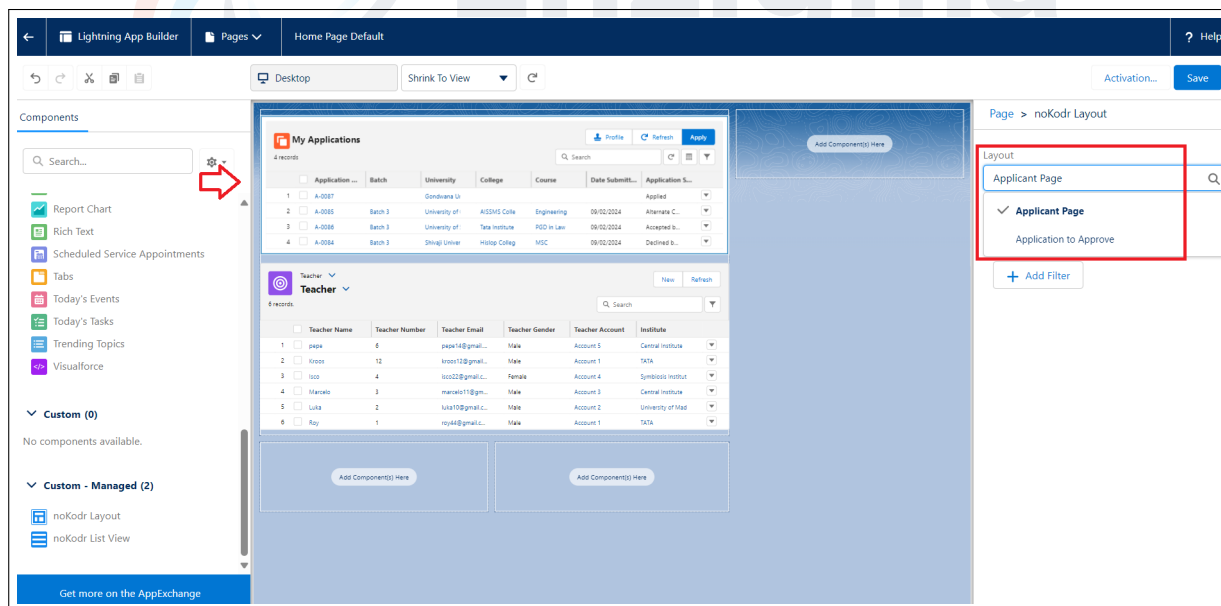


Figure 5.3.128: Published Layout

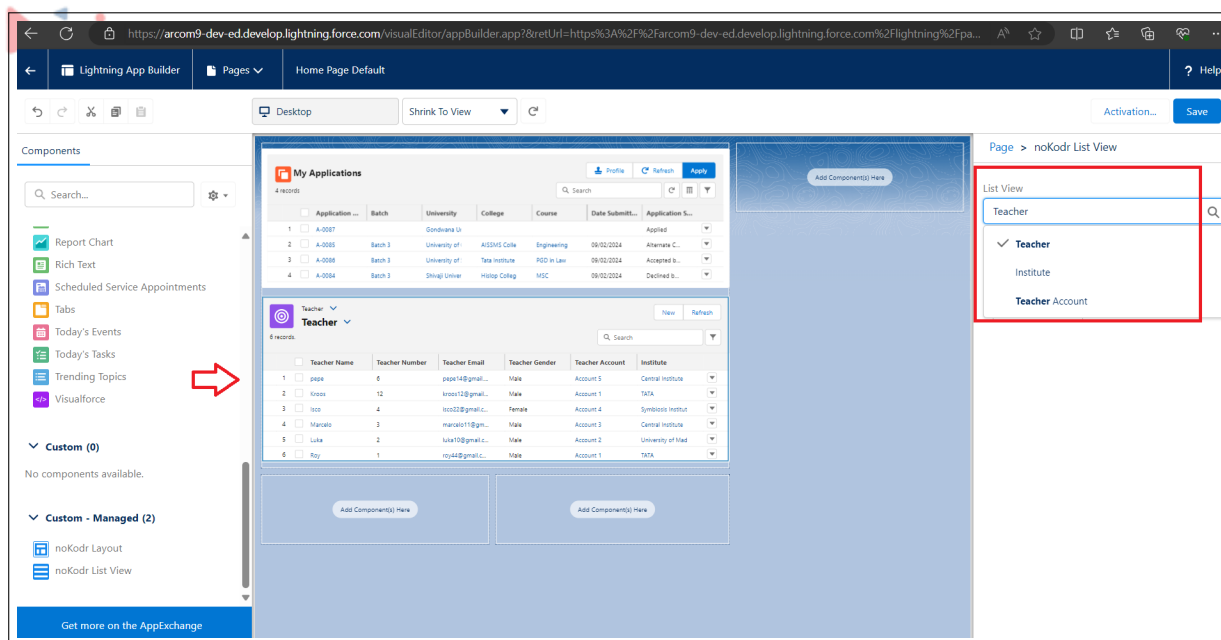


Figure 5.3.129: Published ListView

- **Community:** Using custom components in a Salesforce Community allows you to extend the functionality and user experience beyond what's available out-of-the-box. Salesforce Communities support the use of custom components, which are reusable building blocks that you can create using the Lightning Component Framework. **Here's how you can use custom components in a Salesforce Community:** Only the Published layouts and List Views should be able to drag and drop on the community. Configuration to add Custom Components to Community Builder:
 - Once you've created your custom components, you can add them to your community using the Community Builder tool provided by Salesforce.
 - Navigate to the Community Builder by going to Setup > All Sites

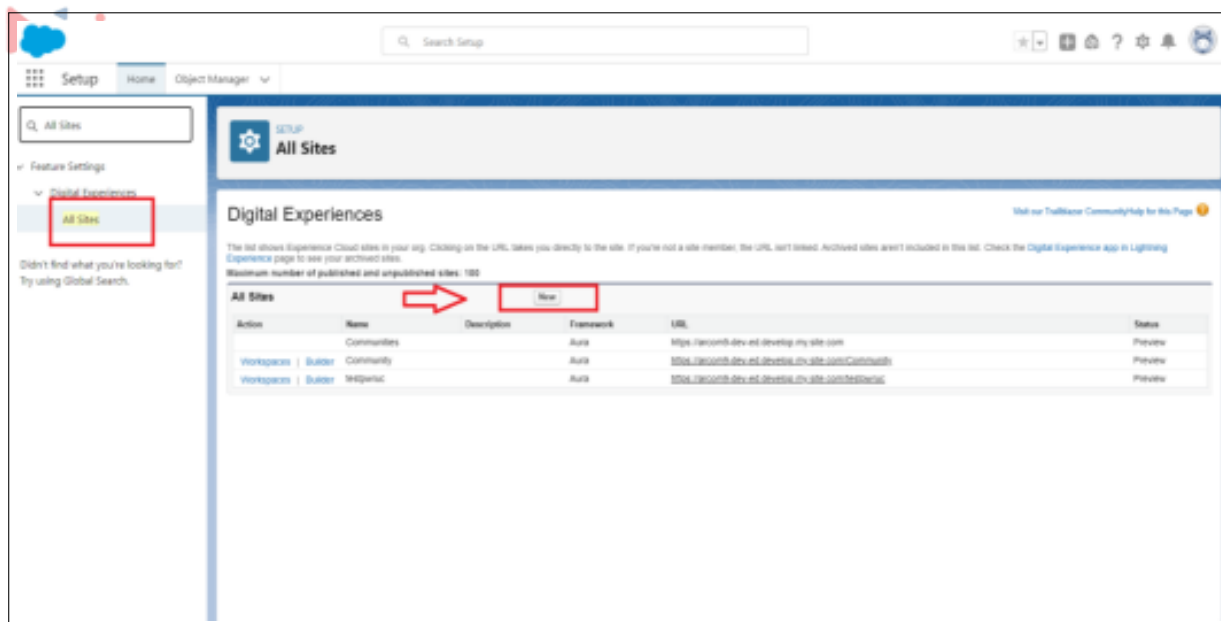


Figure 5.3.130: Community Builder

- Then by creating a New site choose the appropriate template and select >Builder

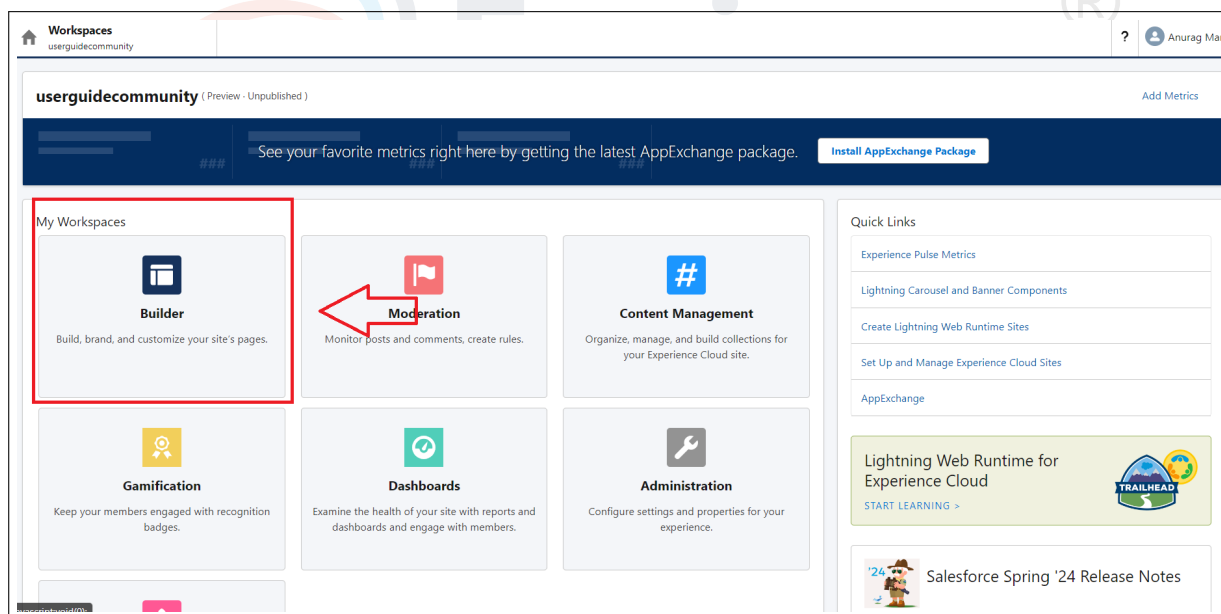


Figure 5.3.131: Builder

- In the Community Builder, you can drag and drop your custom components onto pages to incorporate them into your community's user interface.

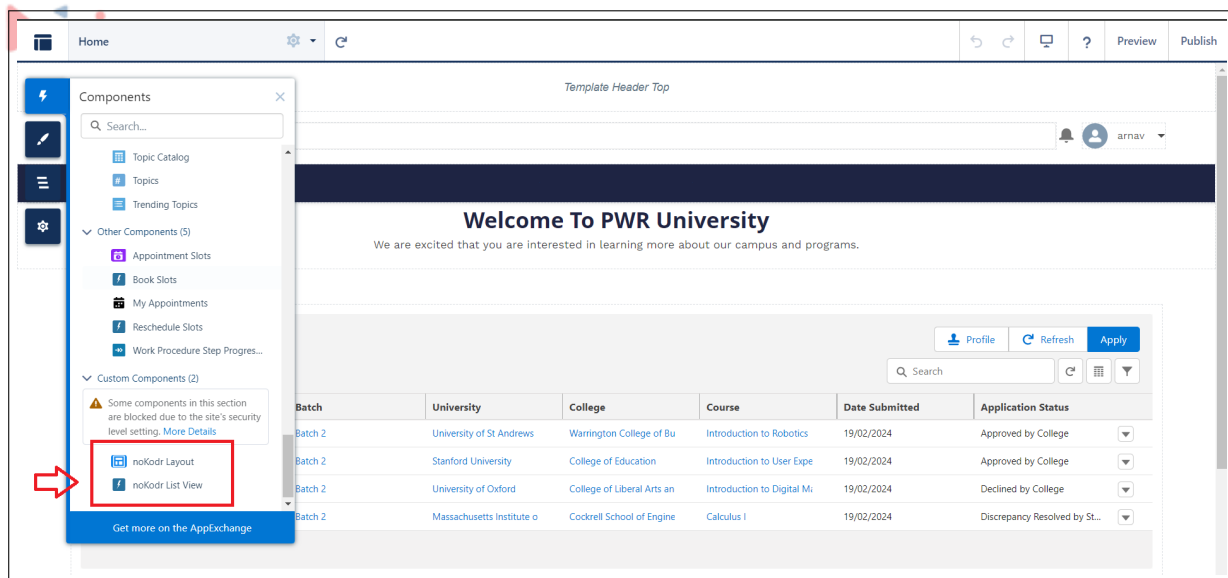


Figure 5.3.132: Components on Community

- After dragging and dropping Layout and Listview on the community once you publish it you can perform configured operations on them.

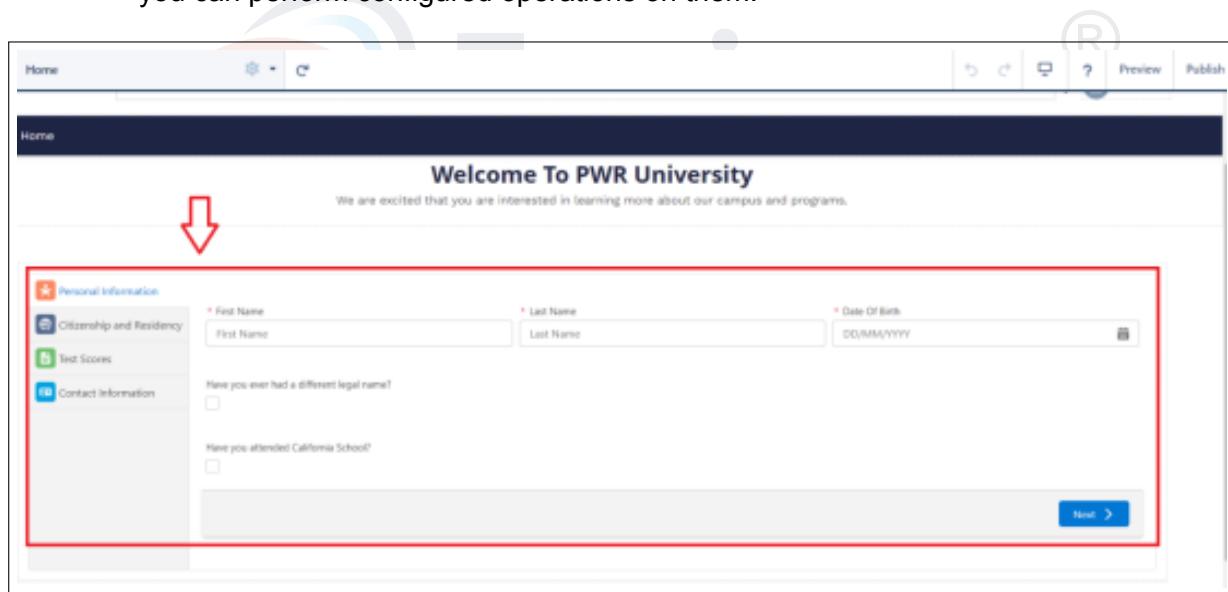


Figure 5.3.133: Layout on Community



Home

Welcome To PWR University

We are excited that you are interested in learning more about our campus and programs.

Personal Information

* First Name:

* Last Name:

* Date Of Birth:

Have you ever had a different legal name?
☐

Have you attended California School?
☐

Next >

Figure 5.3.134: Community





V.III.1.12 REST API Configuration : REST API acts as a communication link between two platforms, enabling interaction through methods like GET (retrieve), POST (create), PUT (update), and DELETE (remove).

- Ways to Configure REST API in noKodr: Rest API can be executed using Model and Workflows
 - noKodr platform > Login > Layout > Open a layout > Workflow > Rest API Action
 - noKodr platform > Login > Layout > Open a layout > Model
- Prerequisite: Create a Named Credentials, Auth Provider and establish a connection. Refer to Named Credentials and Auth Provider.
- Steps to Create Named Credentials:
 1. In Salesforce Setup, navigate to "Named Credentials" under "Security."
 2. Click "New Named Credential."
 3. Enter the details, including the name, URL, and identity type.
 4. Configure the authentication settings based on the requirements (e.g., password, OAuth, or JWT).
 5. Save the Named Credential.
- Steps to Create Auth Provider:
 1. In Salesforce Setup, go to "Auth. Providers" under "Security Controls."
 2. Click "New" to create a new Auth. Provider.
 3. Select the type of provider (e.g., OpenID Connect, SAML, etc.).
 4. Fill in the necessary details such as the name, callback URL, and other provider-specific settings.
 5. Configure the authentication settings and set the appropriate scopes or permissions.
 6. Save the Auth. Provider configuration.

By following these steps, you can create named credentials and an auth provider in Salesforce to establish secure external connections and authenticate users for various integration and external services.
- Steps to create a connection with external source in noKodr:
 1. Create an Account to the respective web application you want to integrate with noKodr
 2. Get the 'Client ID' and 'Client secret key' of that web application.
 3. Create the auth Provider in salesforce.
 4. Create the Named Credential in Salesforce using Legacy
 5. Open noKodr and Create a Model using Connection as API
 6. Request for the operation you want to perform by clicking the 'Request' tab of the 'Update Model'
 7. Click on the 'Response' tab of the 'Update Model' to generate schema as per the response you want to get.



8. Click on Merge.

- API Type Model: The API type model acts as a communication link between two platforms, enabling interaction through methods like GET (retrieve), POST (create), PUT (update), and DELETE (remove).

How to configure API type model:

- Prerequisite: Create a Connected App, Named Credentials, Auth Provider and establish a connection as mentioned above
 1. Create of Model with the following field values:
 - (a) Type: API
 - (b) Connection: Choose the desired connection to fetch records from
 - (c) Label: Auto-populates, mirroring the connection Label
 - (d) Name: Auto-populates, mirroring the connection name
 - (e) Execute on load: If checked, records will be fetched automatically upon loading the component

Figure 5.3.135: Update Model

2. Click on the Request tab and choose the values respectively.
 - (a) Method: Choose from HTTP methods like Get, Post, Put, Delete, Head, or Patch based on the operation you want to perform
 - (b) URL: Add the relevant URL that corresponds to the connection from which you wish to get, create, update, or delete records
 - (c) Record Count: Set the record count to either Multirecord or Single, depending on whether you are dealing with multiple records or a single record in the data operation
 - (d) Body>Schema Designer:

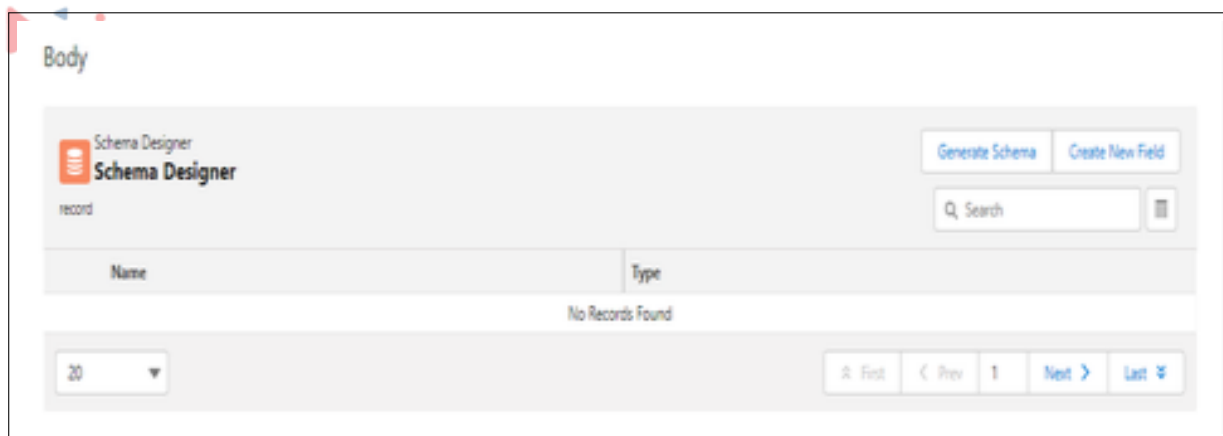


Figure 5.3.136: Schema Designer

There are two ways to provide the request body.

- i. Generate Schema: Users can define fields using JSON format. For example "Date": "05/12/2023", "Date and Time": "05/12/2023 11:40 AM", "Double": 51.21, "Integer": 399, "Object": "", "Text": "Smith"
- ii. Create New Fields: Users can create fields using the create field Model

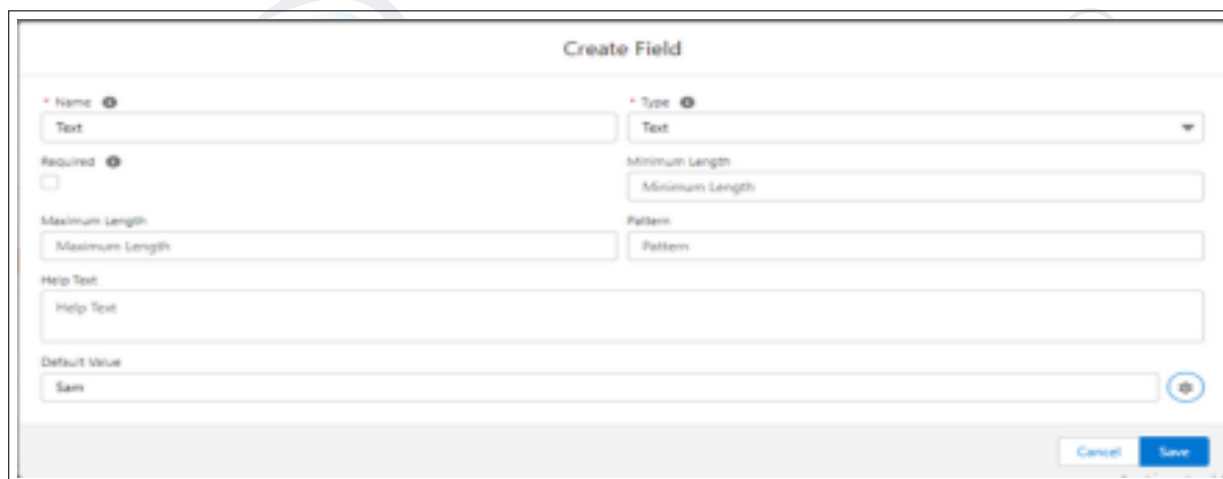


Figure 5.3.137: Create Field

3. Click on the Response tab and choose the values respectively.
 - (a) Response: The user can create a new response with the help of the status code
 - (b) Schema Designer:

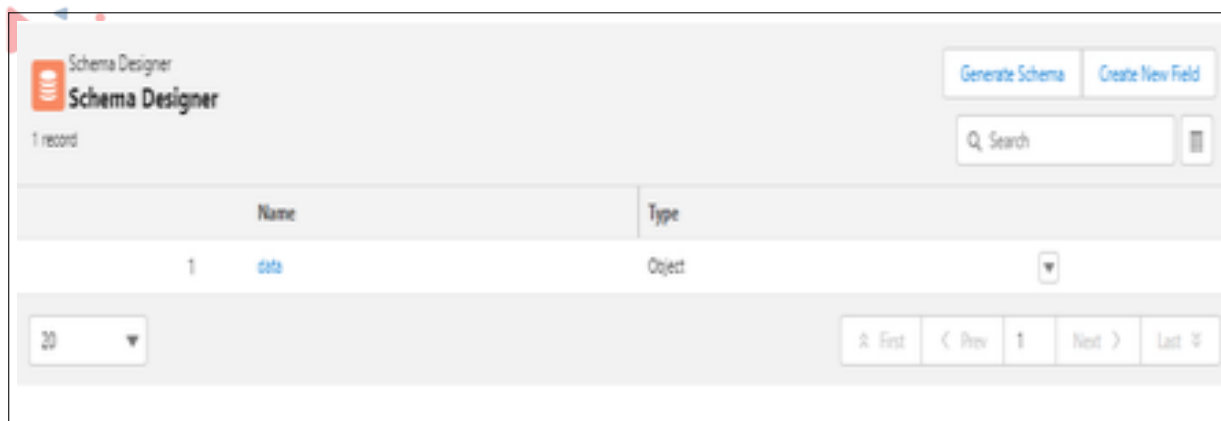


Figure 5.3.138: Schema Designer

- i. Generate Schema: Users can define fields using JSON format. For example "Date": "05/12/2023", "Date and Time": "05/12/2023 11:40 AM", "Double": 51.21, "Integer": 399, "Object": "", "Text": "Smith", "Array": []

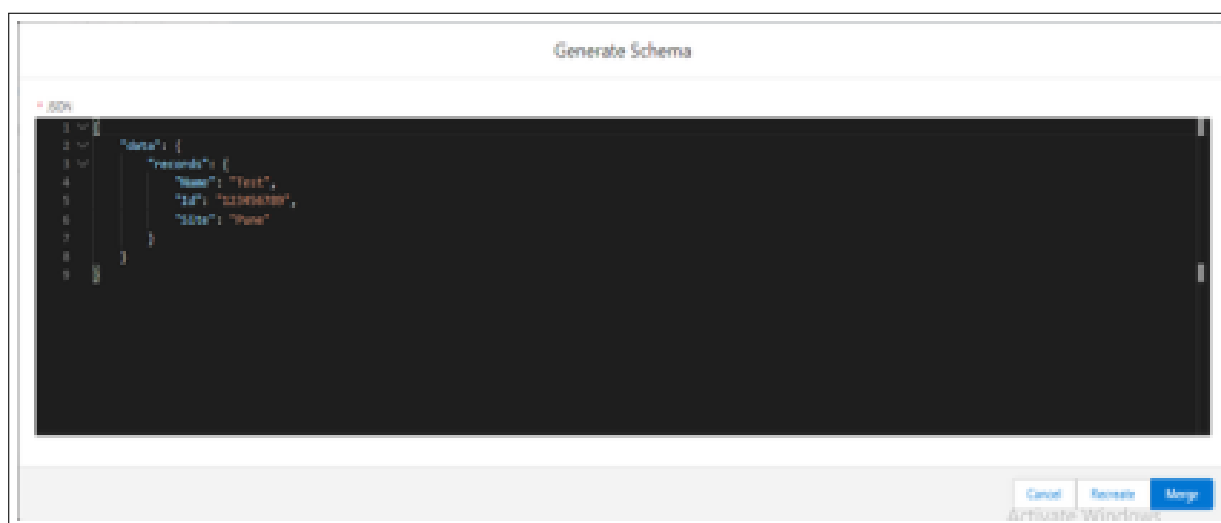


Figure 5.3.139: Generate Schema

- ii. Create New Fields: Users can create fields using the create field Model

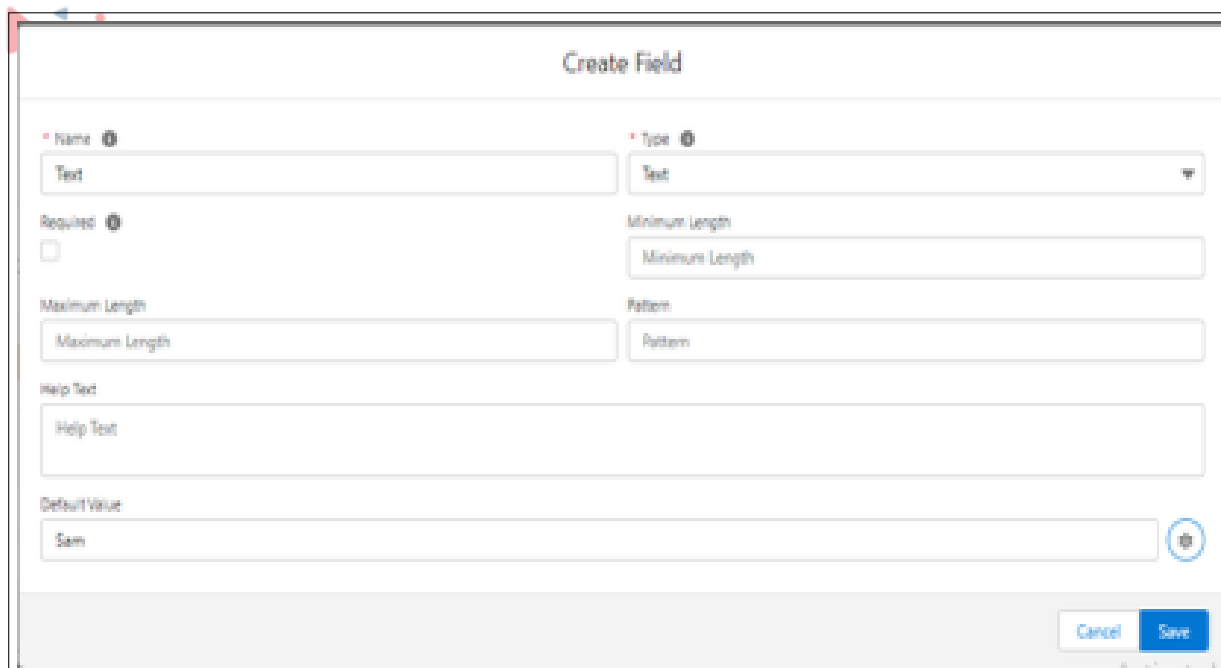


Figure 5.3.140: Create Field

- 'CRUD' using Rest API with Model in Workflow:

- To Fetch Records: Use Get Method >

1. **Create a new layout**

Table Setup:

- (a) Drag and drop a Table component onto the layout Designer
- (b) Add Table action as New and Refresh
- (c) Add row action on Table as Edit and Delete

2. **Assign Model to Table Component**

Assign a model to the Table with the following details:

- (a) Type: API
- (b) Connection: Choose a connection e.g. Sales Connect
- (c) Label gets Auto-populates e.g. Sales Connect
- (d) Name gets Auto-populates e.g. Sales Connect

3. **Request**

Request with the following details:

- (a) Method: GET
- (b) URL: /services/data/v56.0/query?q=select id,name,site from account
- (c) Record Count: Multirecord

4. **Response:** Define or create a schema to fetch the response. This schema will structure the data received from the API call "data": "records": "Name": " ", "Id": " ", "Site": " "

* In Component Attributes:

- Model: Select the model e.g. Sales Connect
- Schema Source: Select "Response" as the source

- Response Code: Set the response code to 200
- * In Fields:
 - Select data > record
 - Click on the Table component, and the Model fields will be displayed on the left side
 - Drag-drop the fields on the Table component
 - Create a new workflow (defining Label and Name)
- * Show Spinner Action:
 - Drag and drop the "Show Spinner Action"
 - Define layout item (e.g., Table), variant, and size
- * Rest API Action:
 - Drag and drop the "Rest API Action"
 - Set the Source API Type to Model
 - Choose the Model (e.g., Sales Connect) assigned to the Table
- * Hide Spinner Action:
 - Drag and drop the "Hide Spinner Action"
 - Define the layout item
- * Connect Actions:
 - Connect the actions in the workflow

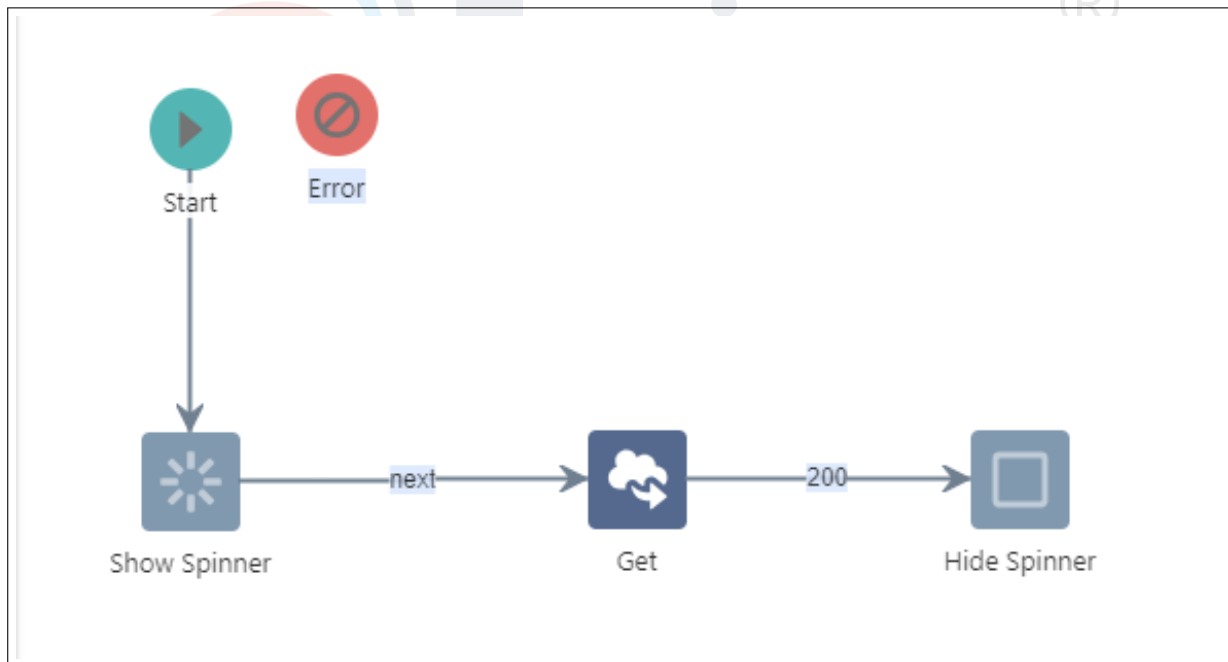


Figure 5.3.141: Workflow

5. Call the workflow on component events

- (a) Refresh: table Action
- (b) Select Action as Workflow
- (c) Select Config and define Workflow: (Select the workflow created for Get e.g. Get and Save)



(d) Save & Run the Layout





6. Preview

- (a) On preview, click on the Refresh Button.
- (b) The Get workflow will be executed, fetching the records

– To Create Records: To POST >

1. Create a new layout

Form Setup:

- * Drag and drop a Form component onto layout Designer
- * Add Form Action as Cancel and Save

2. Assign Model to Form Component

Assign a model to the Form with the following details:

- (a) Type: API
- (b) Connection: Choose a connection e.g. Sales Connect
- (c) Label gets Auto-populates e.g. Sales Connect
- (d) Name gets Auto-populates e.g. Sales Connect

3. Request

Request with the following details:

- (a) Method: Post
- (b) URL: /services/data/v56.0/subjects/Account
- (c) Record Count: Single
- (d) Body > Schema Designer Define or create a schema to fetch the response.
This schema will structure the data received from the API call
"Name": " ", "Site": " "

4. Response

Response with the following details

- (a) No need to define response for POST
- (a) In Component Attributes:
 - Basic:
 - i. Model: Select the model e.g. Sales Connect
 - ii. Schema Source: Select "Request" as the source
 - iii. Now Fields will be displayed at the left corner drag and drop the fields on the form
 - iv. Create a new workflow (defining Label and Name)
- (b) Rest API Action
 - i. Drag and drop the "Rest API Action"
 - ii. Set the Source API Type to Model
 - iii. Choose the Model (e.g., Sales Connect) assigned to the Form
- (c) Toaster Action:
 - i. Drag and drop the "Toaster Action."
 - ii. Fill Type with "Success" and Message with "Record created successfully!"
 - iii. Save the Toaster Action
- (d) Emit Action:
 - i. Drag and drop the "Emit Action"

- ii. Add the event you have created
- iii. Save the Emit Action
- (e) Connect Actions:
Connect the actions in the workflow

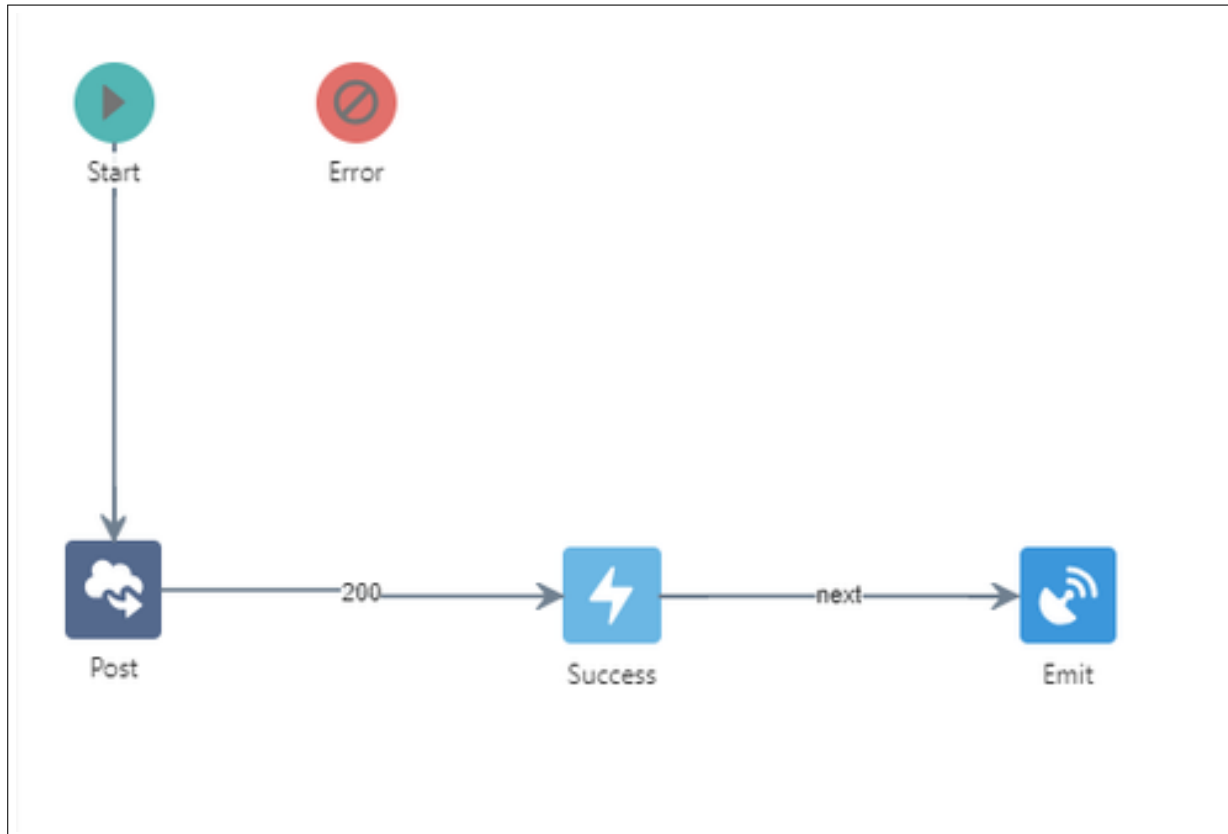


Figure 5.3.142: Workflow

5. Call the workflow on component events

- (a) Click on the form and navigate to Events > Save : Action.
- (b) Select Action as Workflow
- (c) Select Config and define Workflow: (Select the workflow created for Post e.g. Post and Save)
- (d) Save & Run the Layout
- 6. **Preview:** On preview, check if the record is getting saved by clicking on the Save Button

Note: To execute CRUD for the Rest API go to the layout where you have drag dropped the table.

Follow these steps on the layout where you have drag-dropped the table

- 1. Create a Pop and refresh workflow
- 2. Drag and drop POP action and add a label and Save
- 3. Drag and drop Workflow action and select the workflow created for the Refresh Button

4. Connect the actions and Save the workflow

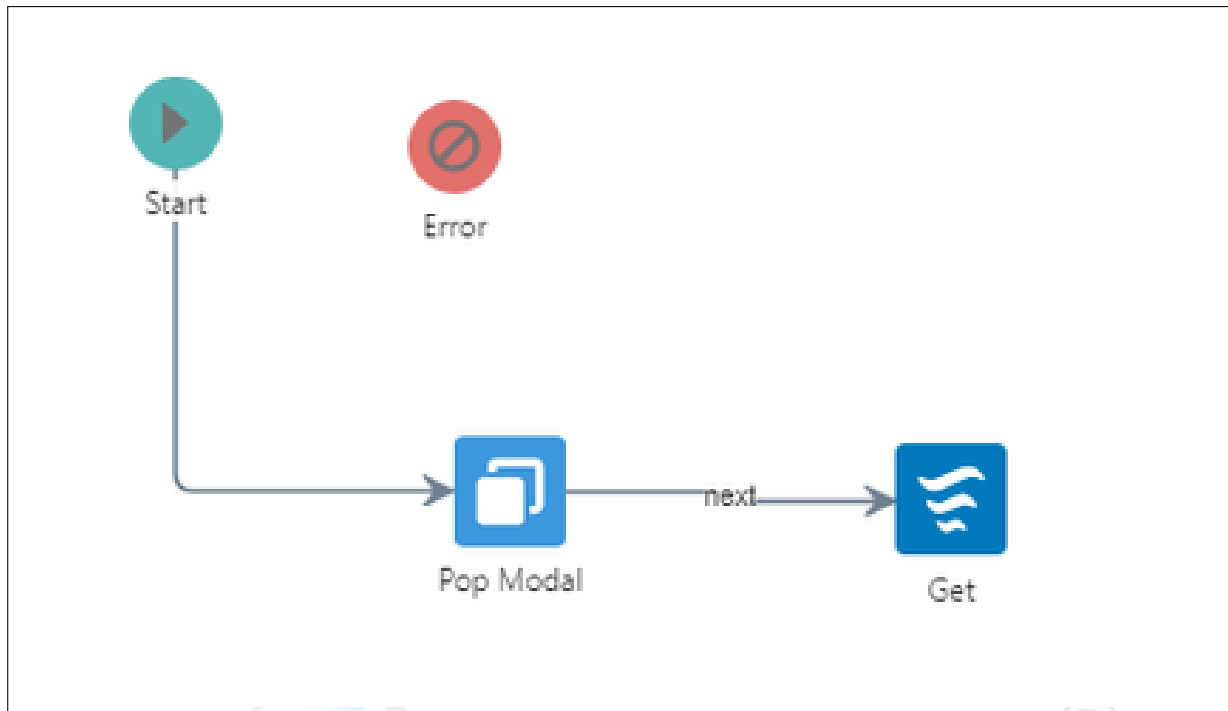


Figure 5.3.143: Workflow

5. Click on the table and navigate to Events > New : tableAction
6. Select Action as Push Modal
7. Select Config and define Push Modal model
8. Layout: Choose the layout created for the POST method e.g Connection Post
9. Event: Select the event created on the POST Method layout
10. Workflow: Select the Pop and Refresh workflow and Save
11. Save & Run the Layout

Preview:

1. On preview, click on the New Button.
2. And Fill the data in the fields and save
3. Check the same on the Connected app i.e. Salesforce Account >navigate to Accounts from app launcher > view the change

• To Update Records: Patch >

1. Create a new layout

Form Setup:

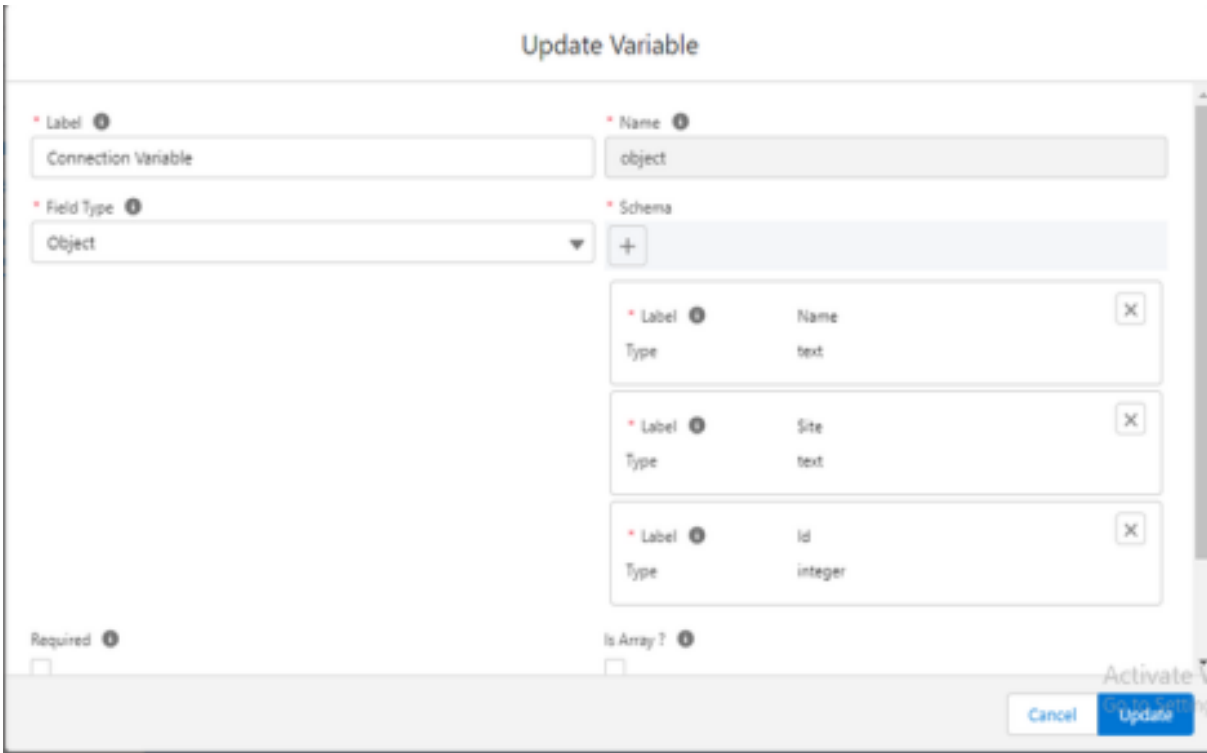
- (a) Drag and drop a Form component onto the layout Designer
- (b) Add Form Action as Cancel and Update

2. Create a variable:

- (a) Add Label & Name
- (b) Type: Object

3. Add Schema:

- (a) Add Label and Name
- (b) Field Type: Object
- (c) Schema: Add Schema Id, Name, and Site
- (d) Click Save



The screenshot shows the 'Update Variable' dialog box. The 'Label' field is set to 'Connection Variable' and the 'Name' field is set to 'object'. The 'Field Type' is set to 'Object'. The 'Schema' section shows a list of three fields: 'Name' (text), 'Site' (text), and 'Id' (integer). Each field has a 'Label' and a 'Type' field. The 'Required' checkbox is unchecked, and the 'Is Array?' checkbox is also unchecked. The 'Activate Variable' checkbox is checked. The 'Cancel' and 'Update' buttons are at the bottom right.

Figure 5.3.144: Add Schema

4. Assign Model to Form Component

Assign a model to the Form with the following details:

- (a) Type: API
- (b) Connection: Choose a connection e.g. Sales Connect
- (c) Label gets Auto-populates e.g. Sales Connect
- (d) Name gets Auto-populates e.g. Sales Connect

5. Request

Request with the following details

- (a) Method: Patch
- (b) URL: /services/data/v56.0/objects/Account/variable:object.Id (Note: Use merge text for Id)

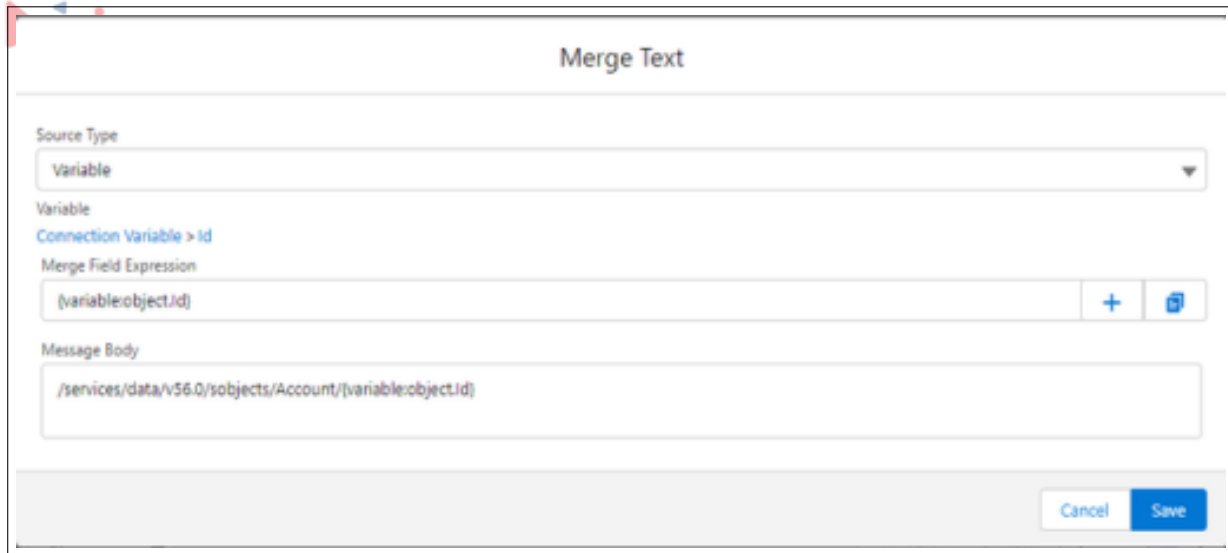


Figure 5.3.145: Merge Text

- (c) Record Count: Single
- (d) Body > Schema Designer Define or create a schema to fetch the response. This schema will structure the data received from the API call "Name": " ", "Site": " "
- (e) Response: No need to define response for Patch
 - In Component Attributes:
 - Basic**
 - (a) Model: Select the model e.g. Sales Connect
 - (b) Schema Source: Select "Request" as the source
 - (c) Now Fields will be displayed at the left corner drag and drop the fields on the form
 - (d) Create a new workflow (defining Label and Name)
 - (e) Workflow 1: For e.g. name it as Put Data
 - Log Action:
 - (a) Drag and drop the Log Action
 - (b) In the Config section, add Log Source: Variable and choose the Variable: Select the variable you created
 - (c) Add Action details and save the log action

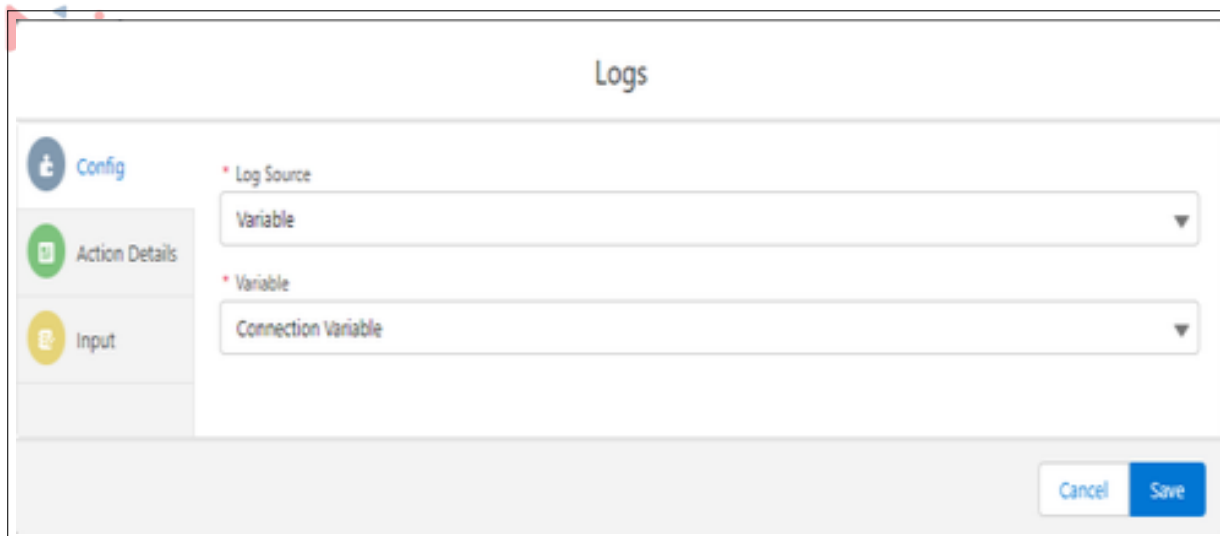


Figure 5.3.146: Logs

- Assignment Action:
 - (a) Drag and drop the Assignment Action
 - (b) Add conditions
 - Condition 1:**
 - i. Destination Type: Model
 - ii. Model: Select the model assigned to the Form
 - iii. Schema Source: Request
 - iv. Fields: Keep it blank, but ensure that fields are visible
 - v. Operator: Set
 - vi. Source Type: Variable

vii. Variable: Select the variable you created at the start

Condition 2:

- i. Destination Type: Model
- ii. Model: Select the model assigned to the Form
- iii. Schema Source: Request
- iv. Fields: Id
- v. Operator: Set
- vi. Source Type: Null

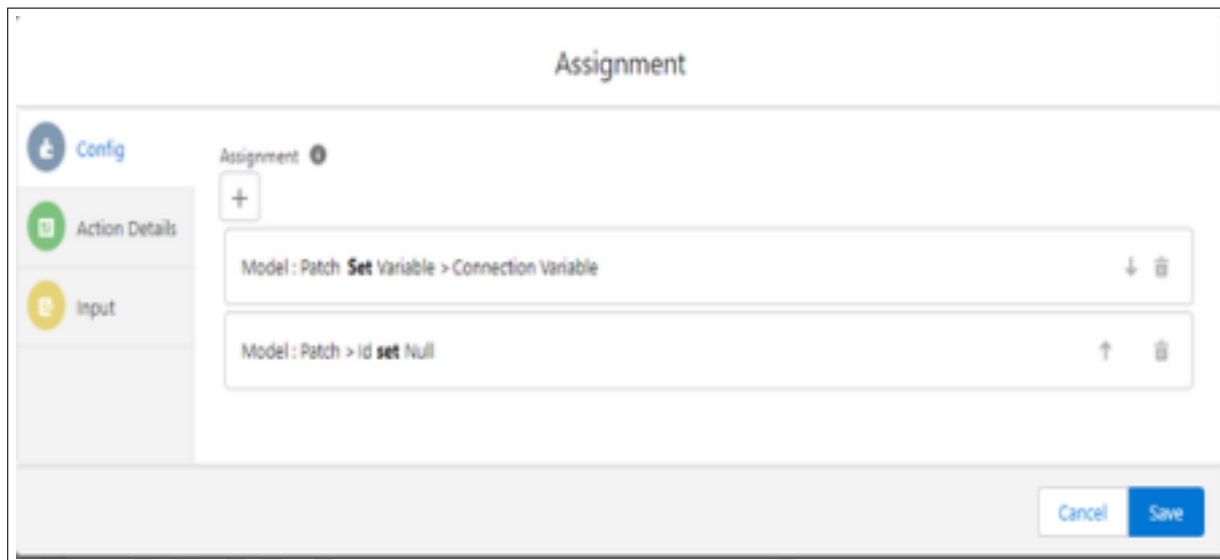


Figure 5.3.147: Assignment

6. Connect Actions

- (a) Connect the actions in the workflow

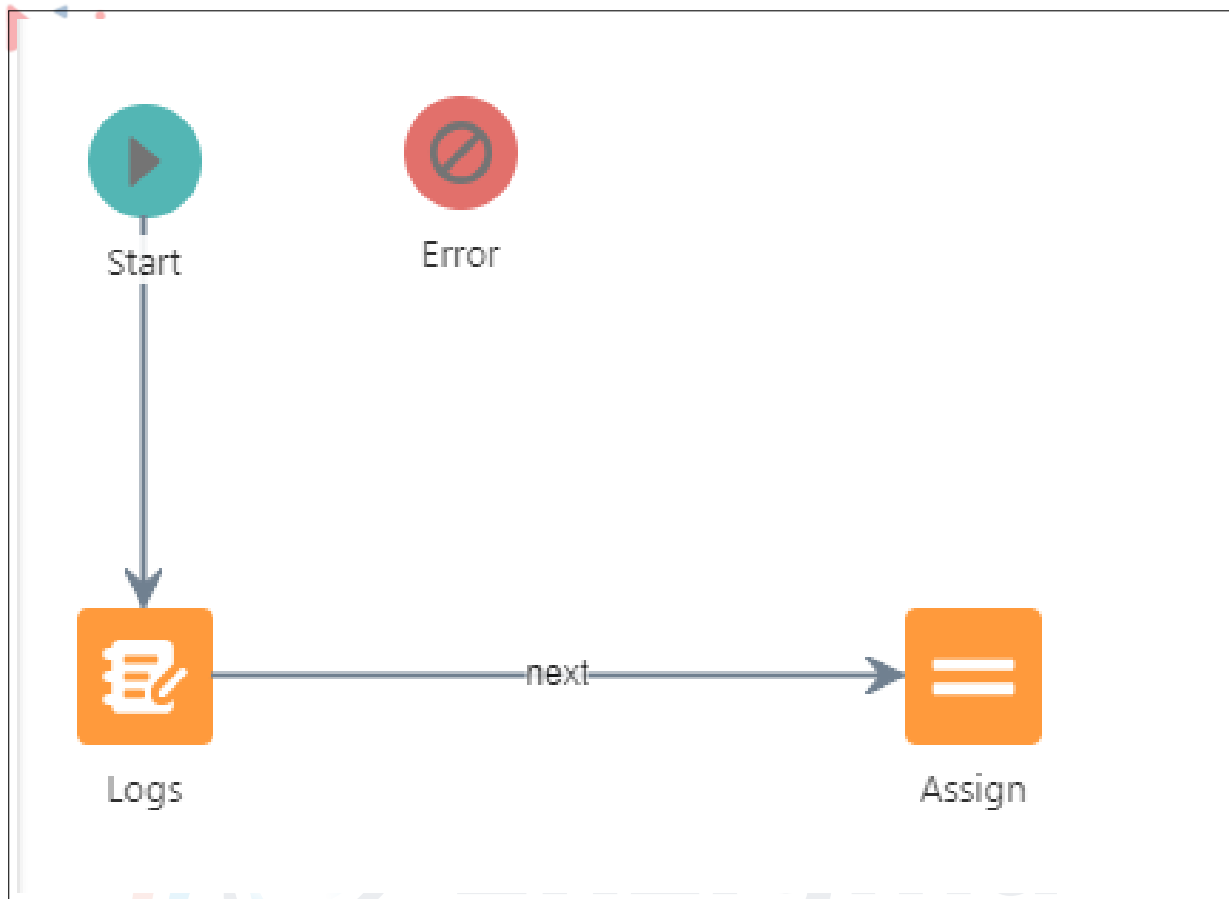


Figure 5.3.148: Add Schema

- (b) Save the workflow
- (c) Assign Workflow 1 on the on-load event of the form Workflow 2. For e.g. name it as Patch Records
 - Rest API Action:
 - * Drag and drop the "Rest API Action"
 - * Set the Source API Type to Model
 - * Choose the Model (e.g., Sales Connect) assigned to the Form
 - Toaster Action:
 - * Drag and drop the "Toaster Action."
 - * Fill Type with "Success" and Message with "Record updated successfully!"
 - * Save the Toaster Action
 - Emit Action:
 - * Drag and drop the "Emit Action"
 - * Add the event you have created
 - * Save the Emit Action
 - * Connect Actions: Connect the actions in the workflow

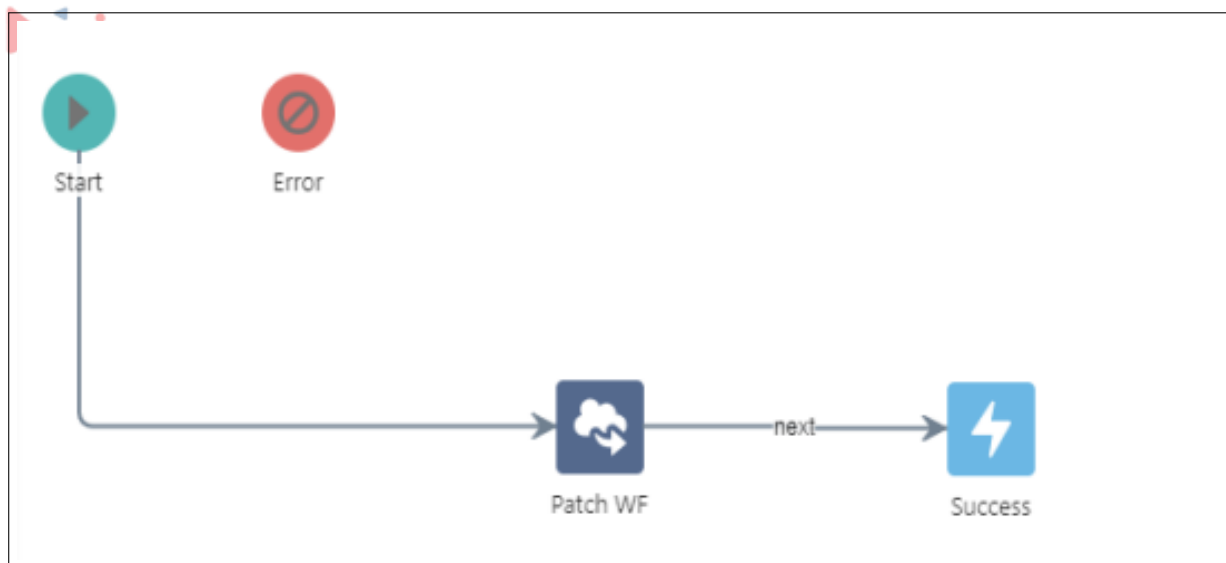


Figure 5.3.149: Workflow

- * Save the workflow
 - * Assign Workflow 2 on the Update button
 - * Save & Run the Layout
7. **Preview:** On preview, check if the record is getting saved by clicking on the Update Button.
- Note:** To execute CRUD for the Rest API, go to the layout where you have drag-dropped the table. Follow these steps on the layout where you have drag-dropped the table:
- (a) Create a Pop and refresh workflow
 - (b) Drag and drop POP action and add a label and Save
 - (c) Drag and drop the Workflow action and select the workflow created for Refresh Button
 - (d) Connect the actions and Save the workflow

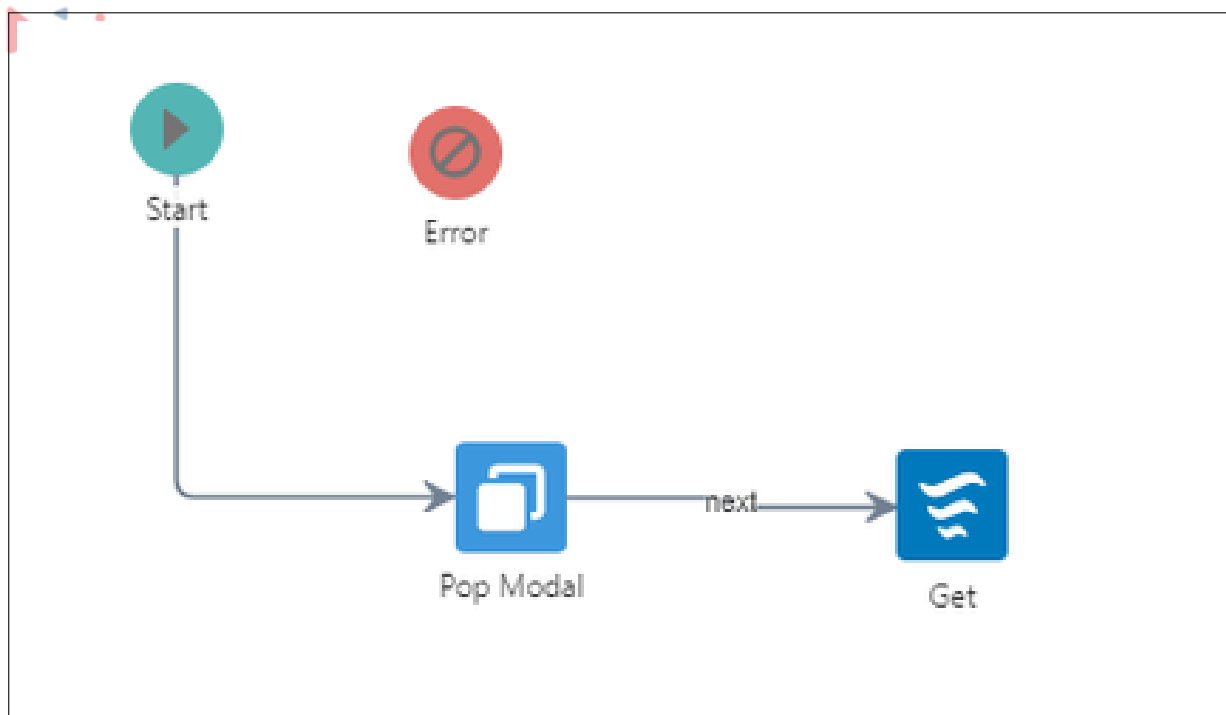


Figure 5.3.150: Workflow

- (e) Click on the table and navigate to Events > Edit : tableAction
- (f) Select Action as Push Modal
- (g) Select Config and define the Push Modal model
- (h) Layout: Choose the layout created for the Patch method e.g Patch Records
- (i) Event: Select the event created on the Patch Method layout
- (j) Workflow: Select the Pop and Refresh workflow
- (k) Click Save
- (l) On Events > Add mapping on Edits workflow
- (m) Input Variable: Select the variable created on the patch layout
 - Operator: Set
 - Source Type: Output Variable
 - Output Variable: Record



The 'Input Mapping' dialog box has a title bar 'Input Mapping'. Below the title bar is a list of mappings. A plus sign button is on the left. The list contains one item: 'Input > Connection Variable **Set** Output > Record'. A trash icon is on the right of the list. At the bottom right are 'Cancel' and 'Save' buttons.

Figure 5.3.151: Input Mapping

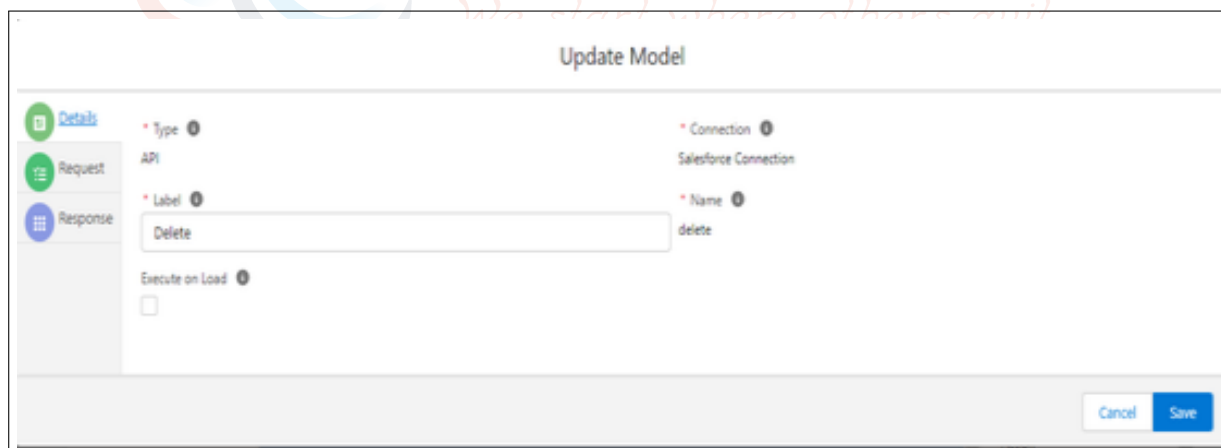
(n) Save Run the Layout

Preview:

- (a) On preview, click on the Edit Button i.e. row action
- (b) Fill the data in the fields and update
- (c) Check the same on the Connected app i.e. Salesforce Account >navigate to Accounts from app launcher > view the change

• To Delete Records: Delete

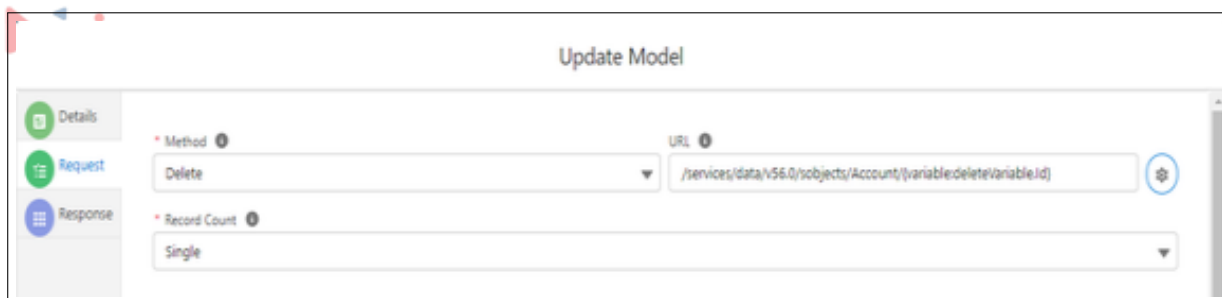
1. Create a workflow in Rest API Presentation where you have dragged and dropped the table details



The 'Update Model' dialog box has a title bar 'Update Model'. On the left is a sidebar with 'Details' (selected), 'Request', and 'Response'. The main area has two columns. The left column has 'Type' (API), 'Label' (Delete), and 'Execute on Load' (checkbox). The right column has 'Connection' (Salesforce Connection) and 'Name' (delete). At the bottom right are 'Cancel' and 'Save' buttons.

Figure 5.3.152: Model

2. Request:



Update Model

Details

Request

Response

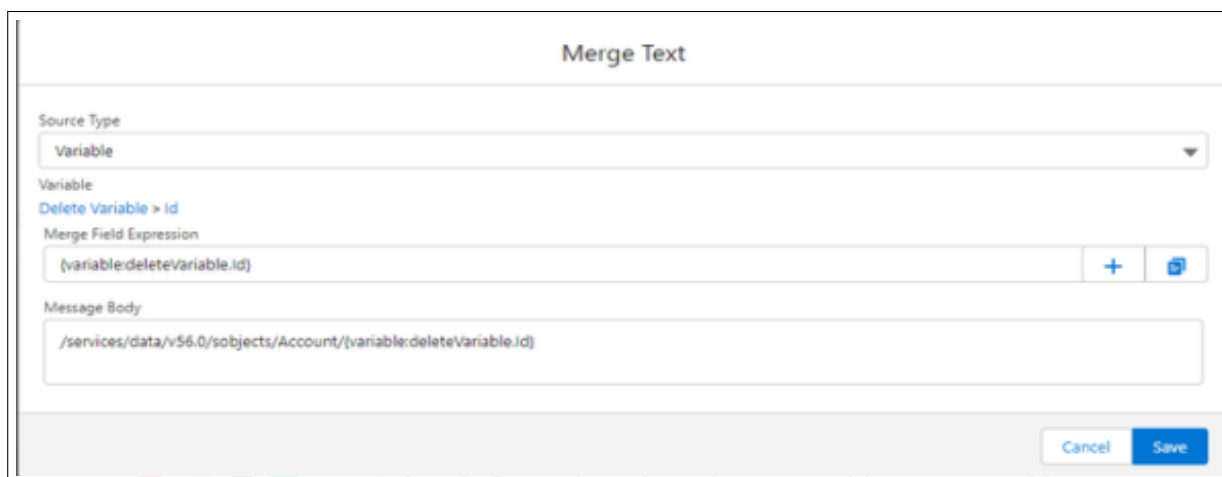
Method: Delete

URL: /services/data/v56.0/subjects/Account/{variable:deleteVariable.Id}

Record Count: Single

Figure 5.3.153: Request

For URL merge text follow



Merge Text

Source Type: Variable

Variable: Delete Variable > Id

Merge Field Expression: {variable:deleteVariable.Id}

Message Body: /services/data/v56.0/subjects/Account/{variable:deleteVariable.Id}

Cancel Save

Figure 5.3.154: Merge Text

Note : No need to define Schema for delete

3. Response: No need to define response for Delete
4. Workflow: Delete Records
 - Prompt Action: Drag and drop the prompt action

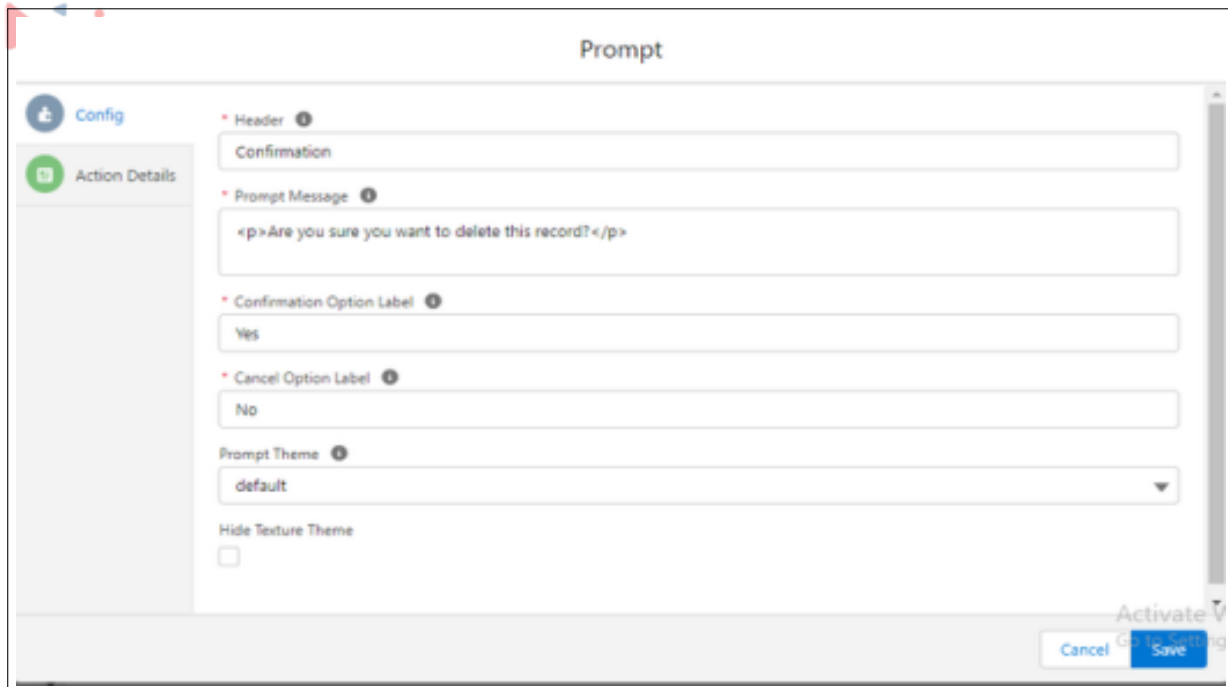


Figure 5.3.155: Prompt

- Rest API Action: Drag and drop the Rest API action

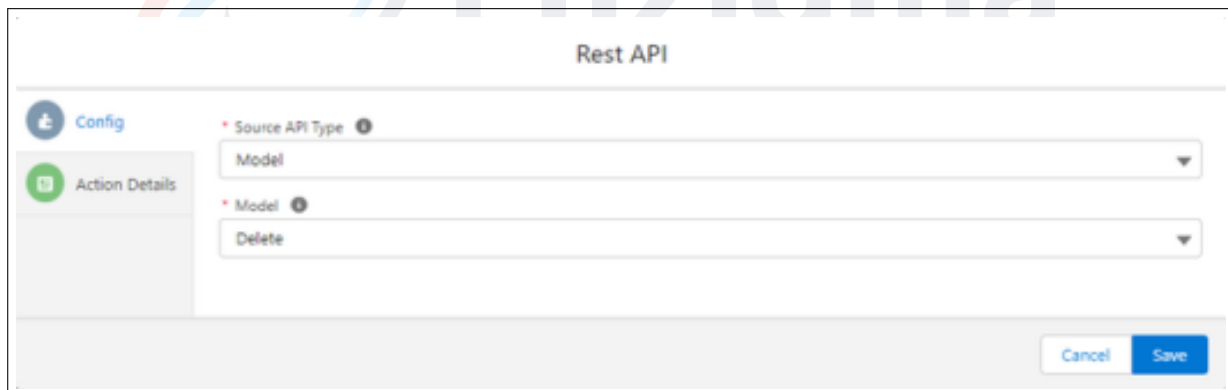


Figure 5.3.156: Rest API

- Toaster Action: Drag and drop the Toaster action



Figure 5.3.157: Toaster

- Connect Actions: Connect the actions in the workflow

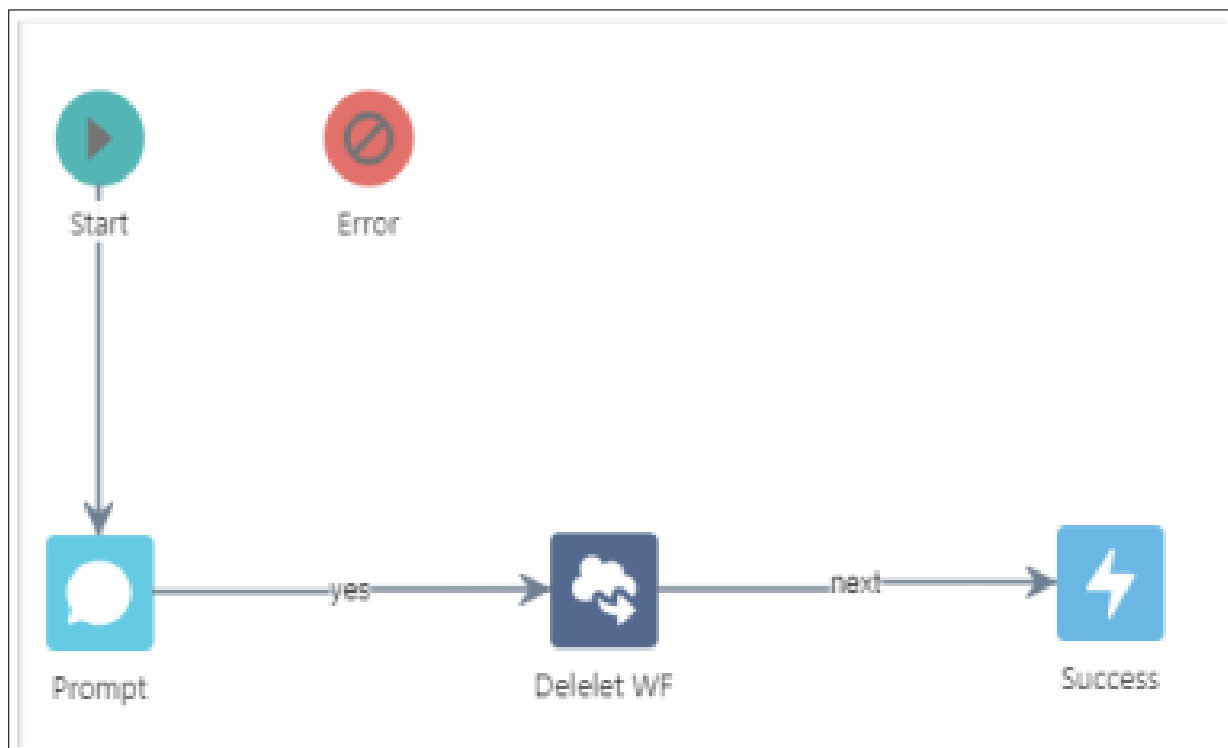


Figure 5.3.158: Connect Action

5. Save the workflow
6. Assign the workflow on table action Delete
7. Click on the table > Events > Delete :rowAction

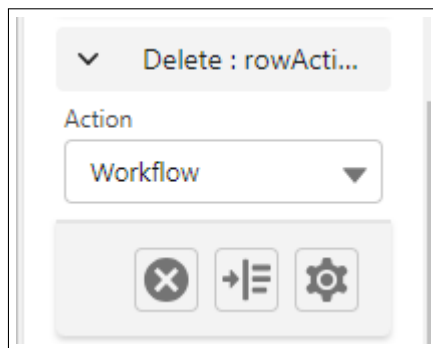


Figure 5.3.159: Delete Row Action

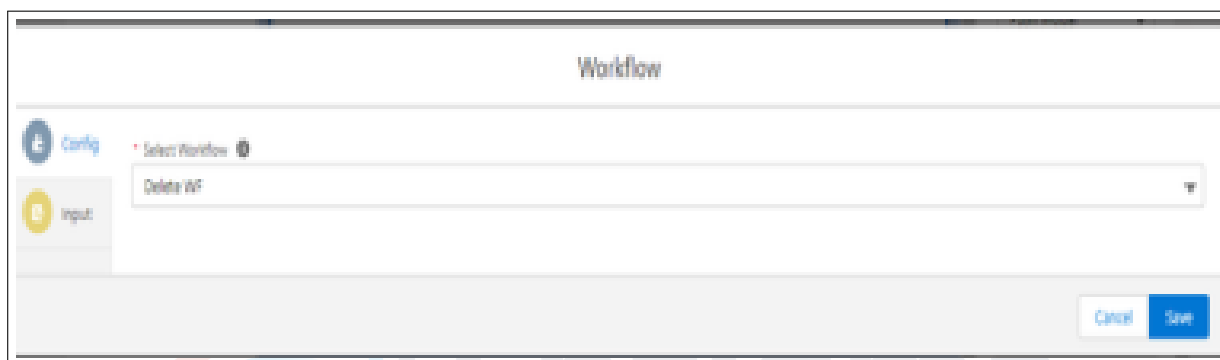


Figure 5.3.160: Workflow

8. Add mapping on the workflow *start where others quit*

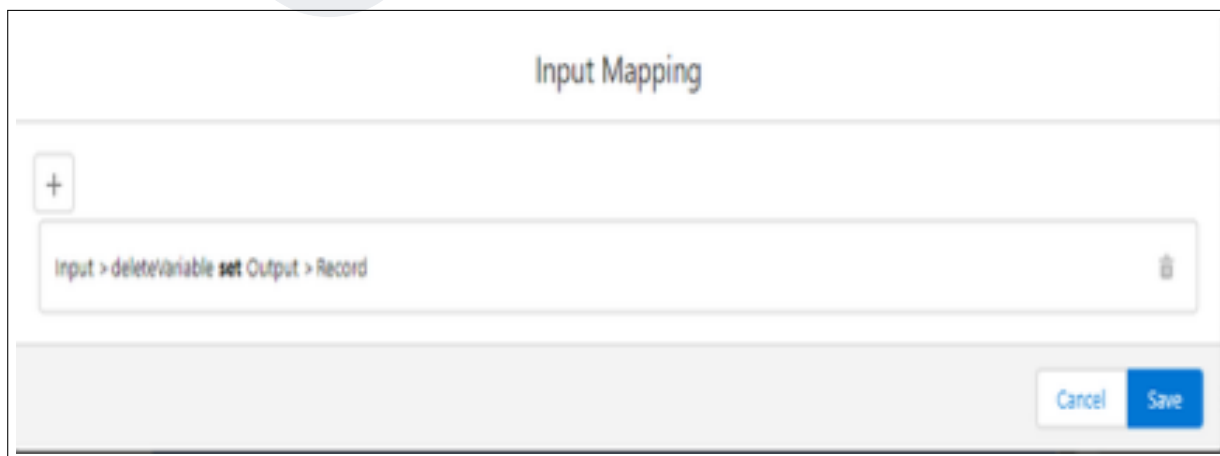


Figure 5.3.161: Input Mapping

9. Save and Run the layout
10. Preview: On preview, check if the record is getting deleted by clicking on the delete button Button.
Check the same on the Connected app eg. Salesforce Account >navigate to Accounts from the app launcher > view the change.



VI Contact Us

For more information visit www.orektic.com, www.noKodr.com

You can also schedule a product demo to know more about **noKodr** simply by filling out the form [here](#).

If you have any concerns or queries then please contact us at support@orektic.com

