vDesigner User Guide
Version No: 1.0
Approved By:

Decimal Technologies
8th Floor, Tower D Pioneer Urban Square,
Golf Course Ext Rd, Sector 62, Gurugram, Haryana-122102

Document Control
This document contains proprietary information of Decimal Technologies Pvt. Ltd. No part of this document may be reproduced, stored, copied, or transmitted in any form or by means electronic, mechanical, photocopying or otherwise, without the explicit written permission of Decimal Technologies.

Document Control

S.No	Type of Information	Document Data
1.	Title	vDesigner User Guide
2.	Document No	VUG_01
3.	Document Version No	1.0
4.	Document Owner	Decimal Technologies Product Team
5.	Document Author(s)	Kumar Saurabh/Lakshay/Ashmit/Deepesh
6.	Document Approver	

Document Update Summary

Version No	Revision Date	Nature of Change	Reviewer	Date Approved
V 1.0	18/May/2020	1st Draft	N/A	N/A

Table of Contents

vDesigner User Guide

1. Introduction

1.1 Document Purpose

The purpose of this document is to explain the features and functions that a user can use to design interactive mobile apps. VDesigner is a web based application to design mobile apps. The document broadly explains several but important “how-to” functions by answering them in simple written simply written step-by-step instructions.

1.2 Document Scope

The functional scope of this document includes the following sections:

Section1:- This is the current section of the document, which provides general information as follows:

- **Purpose of the document**
- **Functional scope of the document (Current heading section)**
- **Audience that can access the content of document**
- **List of abbreviated terms along with their full description, and**
- **Reference document (If any)**

Section2:- This section covers the functional description of the vDesigner application. The document attempts to cover all the important functions by describing them through small modules. These modules help users perform the functions to design a mobile app and its intended functional behavior.

1.3 Intended Audience

This document is mainly written for mobile app designer team. In Decimal Technologies, the mobile app designer team, software development team can access this document. Apart from Decimal technologies’ experts and team members, the mobile app development team from channel partners, client organizations can source this document to learn several functions that they therefore can perform on the vDesigner application.

1.4 Acronyms and Abbreviation

The following table contains the list of abbreviated terms that have been repeatedly used in the document. The table also contains the full description of the document.

Term	Description
API	Application programming Interface
JSON	Java Script Object Notation
OTP	One Time Password
UI	User Interface
KYC	Know Your Customer

1.5 Reference Document

Through different modules, this document covers almost every functionality that the vDesigner application provides to the user to design and develop a mobile app. In conjunction of the features and functions of the vDesigner application, this document sources an important chunk of information from the following document.

Document Name	Version	Date	Company/Organization
vDesigner Navigation Manual v1.0	1.0	25th March, 2020	Decimal Technologies Pvt. Ltd.

To source the vDesigner Navigation Manual v1.0, you can ask Decimal technologies' product/technical team.

2. vDesigner Application Overview

vDesigner is a web based application that allows you to design and develop interactive mobile apps. It incorporates multiple common controls and pre-defined templates to design app layout and screens. You can efficiently use it to place and manage multiple UI (User interface) objects and controls across different screens.

The main advantage of using the vDesigner application is that it requires minimal coding efforts to design a mobile app. By using simple PnP (Plug and Play) types of features, you can design robust mobile apps and implement complex and most prevalent features instantly and cost-effectively. Conclusively, it separates the design concern from the implementation aspect of mobile app development so that you can efficiently focus on the functional behavior of the mobile app.

The vDesigner application mainly provides the following functional benefits:

- **Creating dynamic form**
- **Providing drag and drop elements and widgets**
- **Applying validation, regex, and business rules**
- **Building mobile app workflow based on customer specific requirement**
- **Instantly deploying app and downloading APK file**
- **Low cost ownership**

To proficiently perform the function on the vDesigner application, you mandatorily must be have the working knowledge of the following tools and concepts:

- **Basic Knowledge**
    - **JSON Objects**
    - **API(s) and Signatures**

- **In-depth Knowledge**
    - **Data Modeling**
    - **Business Requirements, and rules and validation**

Different features of the vDesigner application are described in the following modules:

3. vDesigner Modules

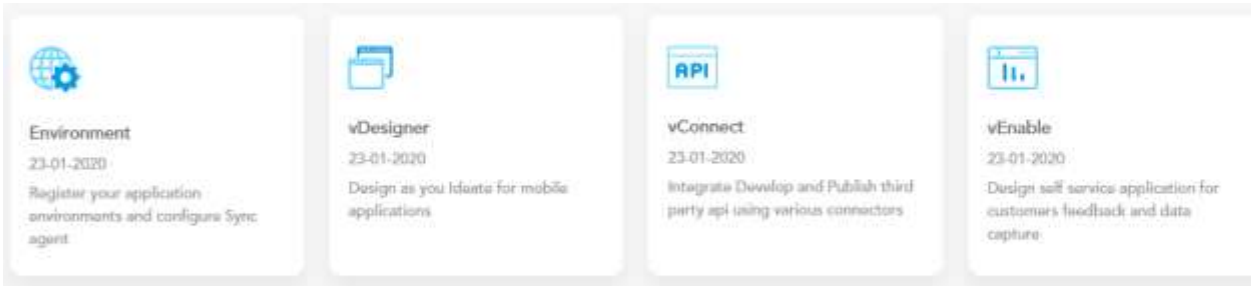
3.1 Configuring a Form and Adding Controls

This section describes how to add and configure a form or screen. After you access the vDesigner application, it, by default, creates and displays a home page. Apart from the home page, you can add new forms and then configure them based on the mobile app related requirements.

3.1.1 Configuring a New Form

To add and configure a new form:

1. On Vahana's project management workbench, locate the **vDesigner** tile **(Fig 3.1.1 (a))**.



Environment
23-01-2020
Register your application environments and configure Sync agent

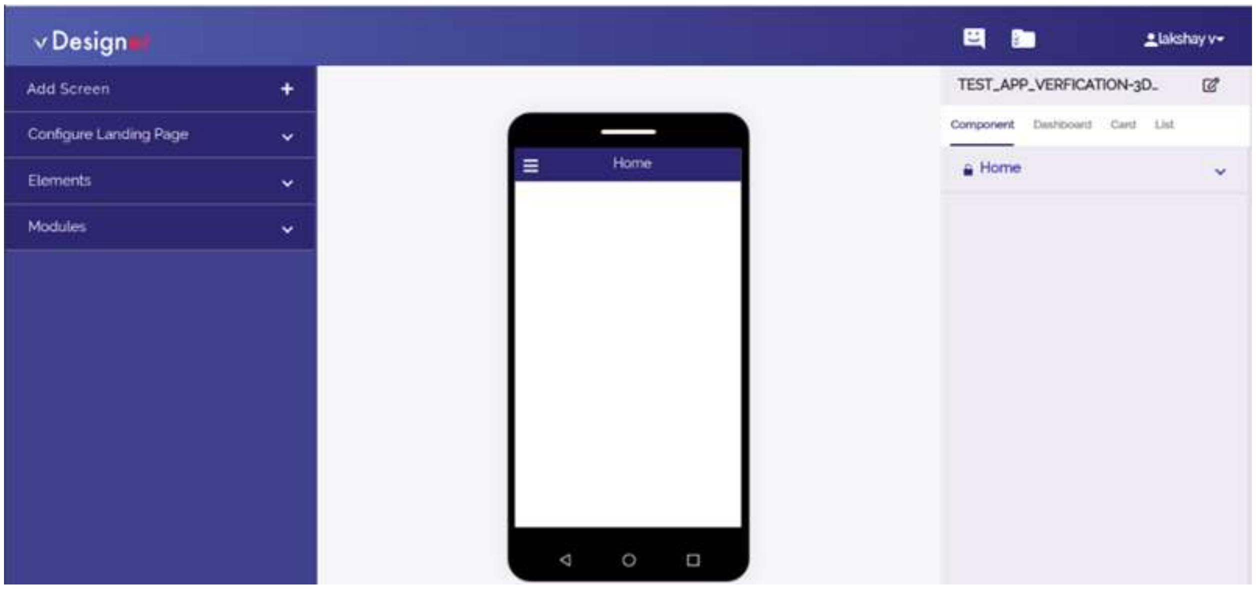
vDesigner
23-01-2020
Design as you Ideate for mobile applications

vConnect
23-01-2020
Integrate Develop and Publish third party api using various connectors

vEnable
23-01-2020
Design self service application for customers feedback and data capture

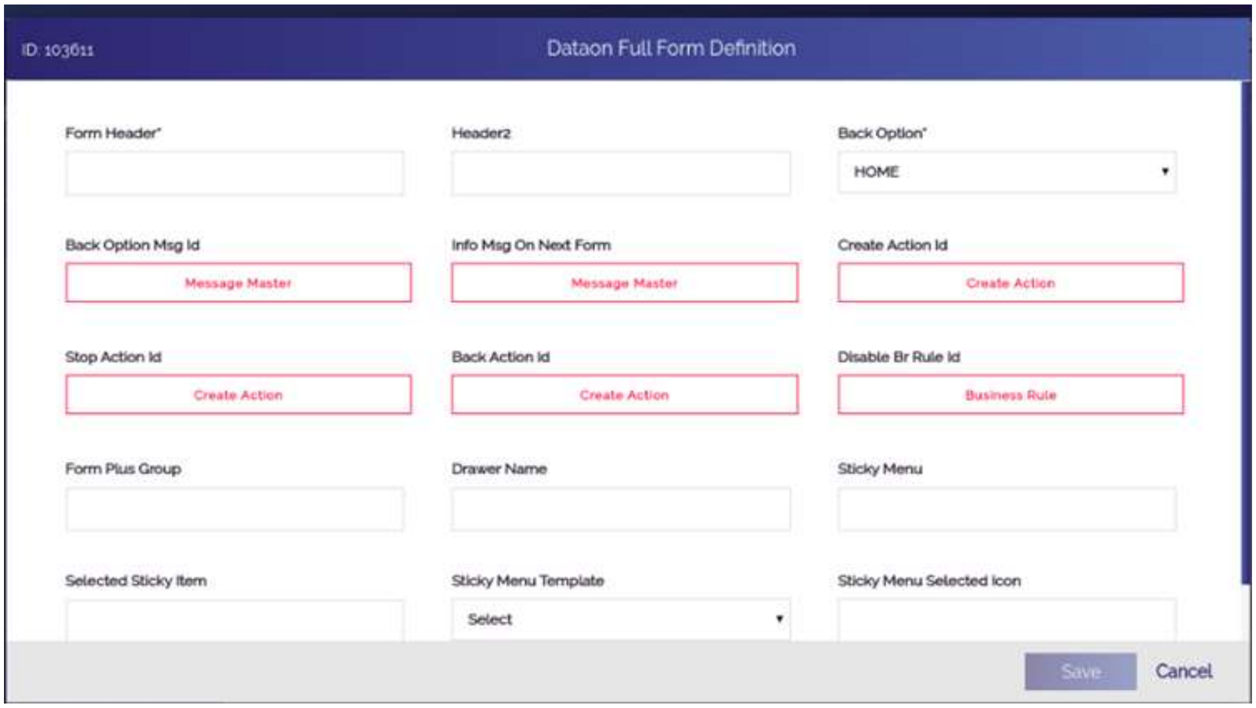
(Fig 3.1.1 (a))

2. Click the **vDesigner** tile, vDesigner's home page **(Fig 3.1.1 (b))** opens.



(Fig 3.1.1 (b))

3. In the left navigation pane, click **Add Screen**, the **Dataon Full Form Definition** dialog box **(Fig 3.1.1 (c))** opens.


ID: 103611

Dataon Full Form Definition

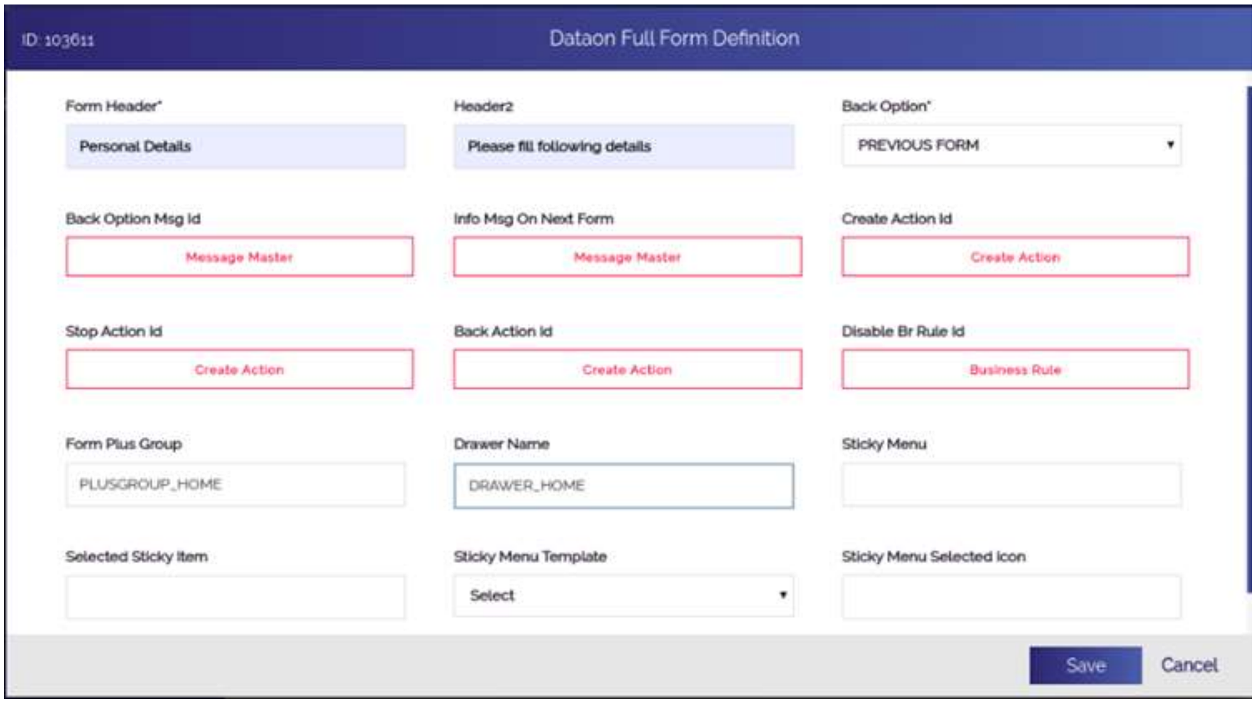
Form Header*	Header2	Back Option*
		HOME
Back Option Msg Id	Info Msg On Next Form	Create Action Id
Message Master	Message Master	Create Action
Stop Action Id	Back Action Id	Disable Br Rule Id
Create Action	Create Action	Business Rule
Form Plus Group	Drawer Name	Sticky Menu
Selected Sticky Item	Sticky Menu Template	Sticky Menu Selected Icon
	Select	

(Fig 3.1.1 (c))

Note:-
The **Dataon Full Form Definition** dialog box can be referred to as property sheet of any entity and/or element. In the **Dataon Full Form Definition** dialog box or property sheet, you configure the entity.

4. In the **Dataon Full Form Definition** dialog box, enter the values in the respective boxes to configure a form as follows:

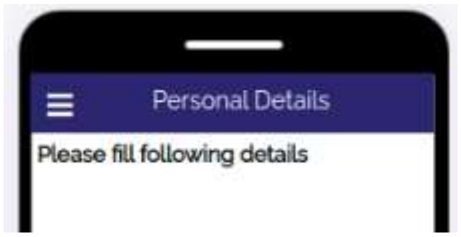
Box	Description
Form Header	In this box, enter the title of the form (For example:- Personal Details)
Header 2	In this box, enter the sub-heading of the form (For example: - Please fill following details). The sub-heading of the form is displayed under the title of the form.
Back Option	Click this list to configure the back option. The back option allows you to navigate to the desired page after you tap the back option on the current form/page. For instance: - *In the **Back Option** list, if you select **Home**. The back option will take you to the home page after you tap the back option on the current form.*
Form Plus Group	Click in this box and then select **PLUSGROUP_HOME** to add a plus group on the form.
Drawer Name	Click this box and then select **DRAWER_HOME** to add new hamburger drawer.



Form Header*: Personal Details
Header2: Please fill following details
Back Option*: PREVIOUS FORM
Back Option Msg Id: Message Master
Info Msg On Next Form: Message Master
Create Action Id: Create Action
Stop Action Id: Create Action
Back Action Id: Create Action
Disable Br Rule Id: Business Rule
Form Plus Group: PLUSGROUP_HOME
Drawer Name: DRAWER_HOME
Sticky Menu:
Selected Sticky Item:
Sticky Menu Template: Select
Sticky Menu Selected Icon:

(Fig 3.1.1 (d))

5. After you enter or select the values in the respective box/list, click **Save (Fig 3.1.1 (d))**, the form is successfully configured.



(Fig 3.1.1 (e))

3.1.2 Adding Controls

After you successfully add and configure a form, you can add multiple controls to the form. These controls are added to the form to impart the functionality to the respective form. The selection of the control, which has to be added to the form, entirely depends on the workflow of the form and the mobile app.

The vDesigner application offers you two categories of controls or elements. These categories are: ***Standard*** and ***Custom***.

1. Standard Elements

The **Standard** category of elements includes commonly used elements/controls such as text field (text box), label, radio button, check box, bottom button, dropdown list, date, hyperlink, and others. You can add these controls to the form based on the functional requirement on the respective form.

2. Custom Elements

The **Custom** category of element provides you a few special elements, which are used for specific jobs. These elements are AADHAAR Biometric Auth, AADHAAR Biometric KYC, audio recorder, date time slot, email, email with domain, and drop down with button, google map, mobile number, OTP, and others.

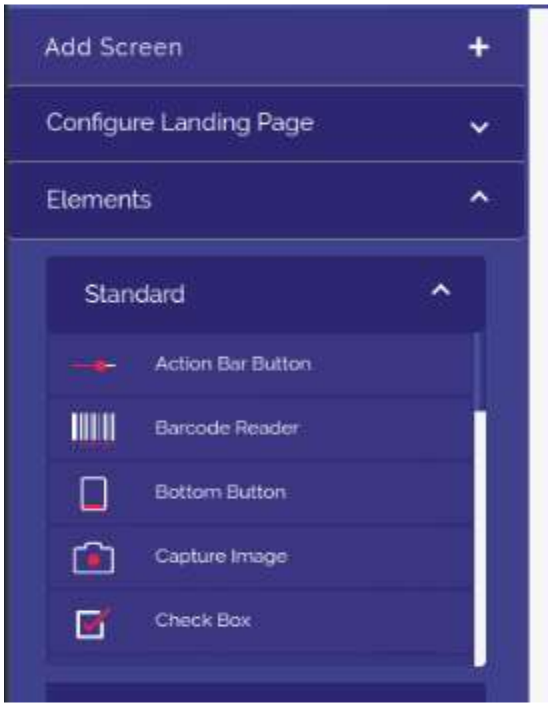
To know the functionality of each element, source the vDesigner Navigation Manual for your reference.

The following heading sections describe to add a few but prevalent controls that are frequently used to design and develop a mobile app.

3.1.2.1 Adding a Text Field

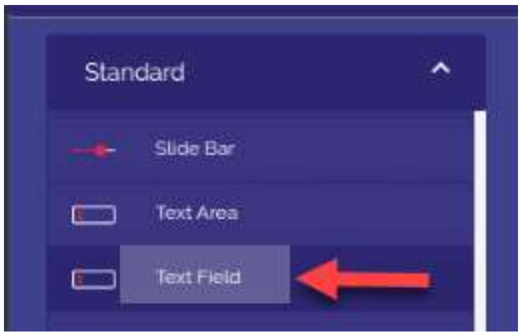
To add a text field:

1. On the left navigation pane, click **Standard**, the **Standard** menu **(Fig 3.1.2.1 (a))** expands.



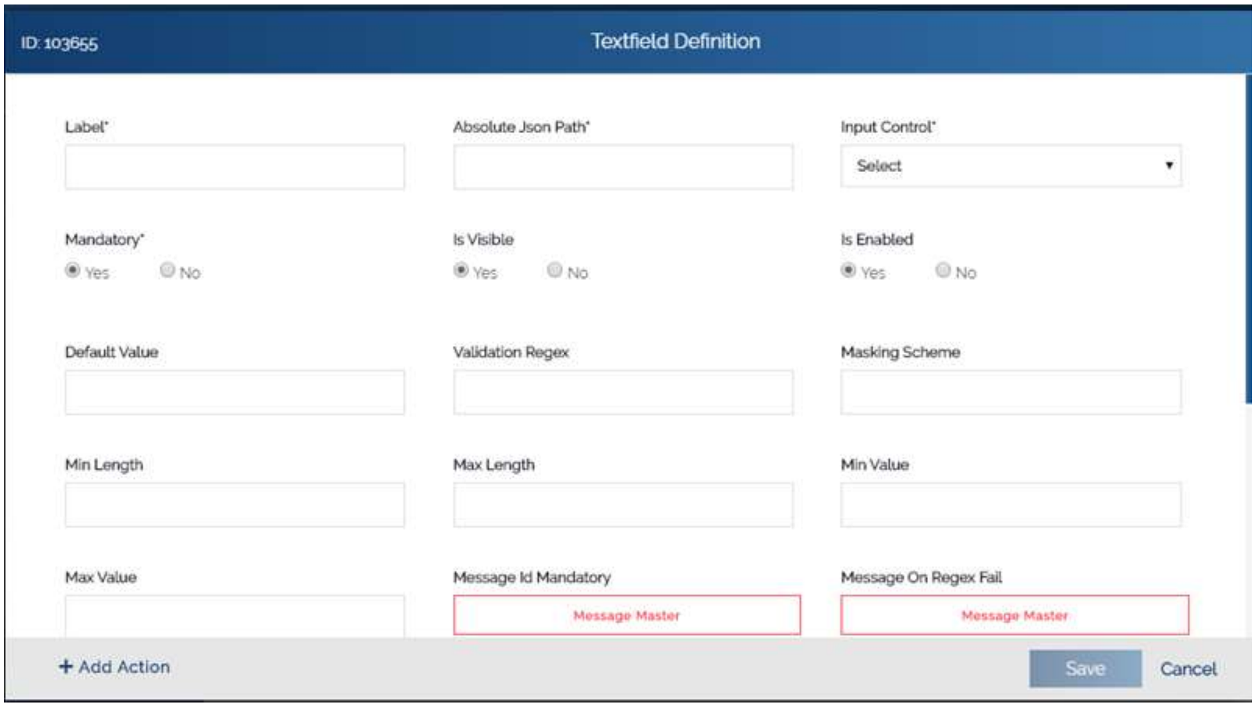
(Fig 3.1.2.1 (a))

2. Under **Standard**, scroll down and locate the **Text Field** element **(Fig 3.1.2.1 (b))**.



(Fig 3.1.2.1 (b))

3. Click the **Text Field** element **(Fig 3.1.2.1 (b))**, the **TextField Definition** dialog box **(Fig 3.1.2.1 (c))** opens.


ID: 103655

Textfield Definition

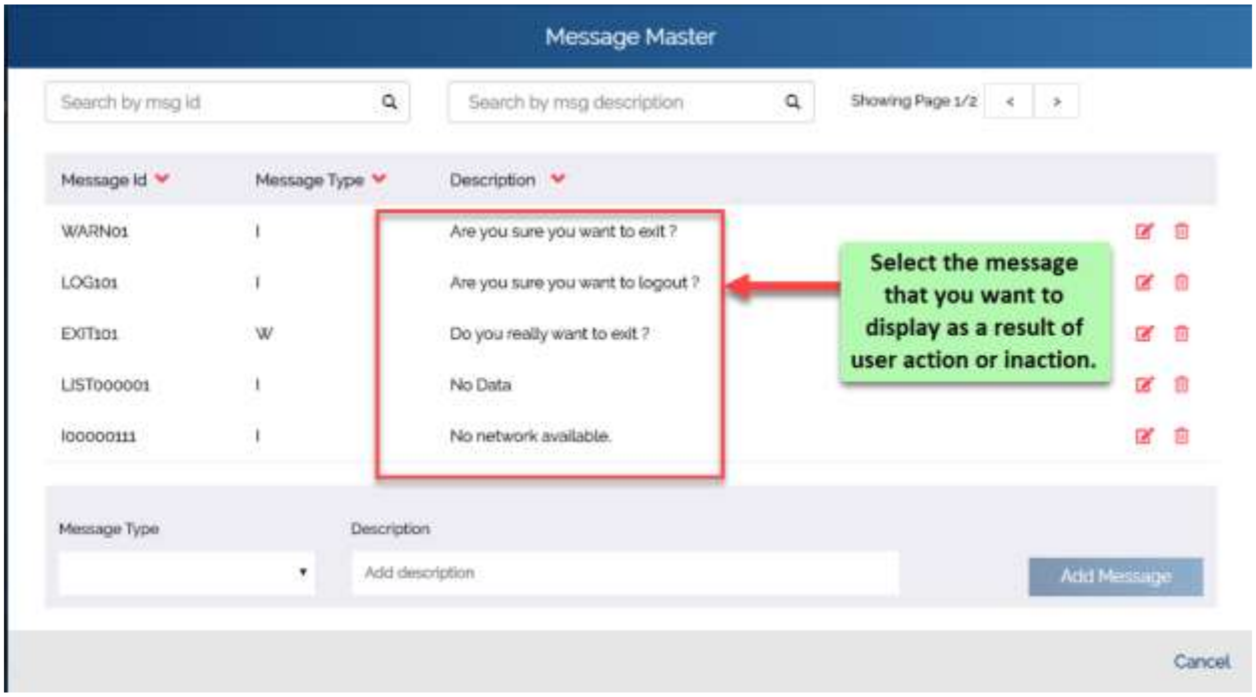
Label*	Absolute Json Path*	Input Control*: Select
Mandatory*: [✓ Yes] No	Is Visible: [✓ Yes] No	Is Enabled: [✓ Yes] No
Default Value	Validation Regex	Masking Scheme
Min Length	Max Length	Min Value
Max Value	Message Id Mandatory: Message Master	Message On Regex Fail: Message Master

(Fig 3.1.2.1 (c))

4. On the **TextField Definition** dialog box, enter values in the respective boxes as follows:

Box/List	Description
Label	In this box, enter label (For example:- Name) of the text field that is displayed to the left of the text field
Absolute JSON Path	In this box, enter the absolute json path as follows: $.POD.name. This json path denotes the location where the configuration details of this element are stored.
Input Control	Click this list to select the condition to enter the value in the text field. For example: - *If you select **Char With Title Case**, it means that the user can enter only alphabetic characters with first character in capital letters*.
Mandatory	Click **Yes** if you want to make the text field as a mandatory element.
Is Visible	Click **Yes** if you want to make the element visible after the respective form is displayed.
Is Enable	Click **Yes** if you want to make the element as active element. In the active element, the user can enter the value.
Message ID Mandatory	This feature is used to configure the message against the user action on the text field. To configure the message: ➢ Click **Message Master**, the **Message Master** dialog box **(Fig 3.1.2.1 (d))** opens. ➢ In the **Message Master** dialog box, select the message that you want to display if the user forgets to fill the value in the text field. If you want to add the new message: ➢ Click the **Message Type** list and then select: ○ **W** to add a warning message.

	○ **E** to add an error message. ○ **I** to add information type message. ➢ After you select the message type, click the **Description** box. ➢ In the **Description** box, enter the text of message (For example: - Please fill correct value.) ➢ After you enter the text/description of the message, click **Add Message**, the message is successfully configured.


Message Master

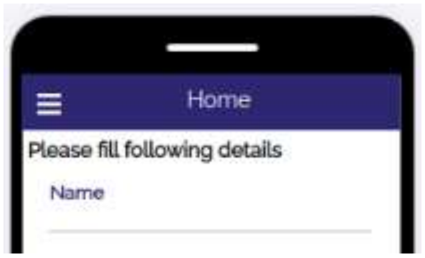
Search by msg id | Search by msg description | Showing Page 1/2

Message Id	Message Type	Description
WARN01	I	Are you sure you want to exit ?
LOG101	I	Are you sure you want to logout ?
EXIT101	W	Do you really want to exit ?
LIST000001	I	No Data
I00000111	I	No network available.

Select the message that you want to display as a result of user action or inaction.

(Fig 3.1.2.1 (d))

5. After you enter values in the respective boxes, click **Save (Fig 3.1.2.1 (c))**, the text field **(Fig 3.1.2.1 (e))** is successfully configured.



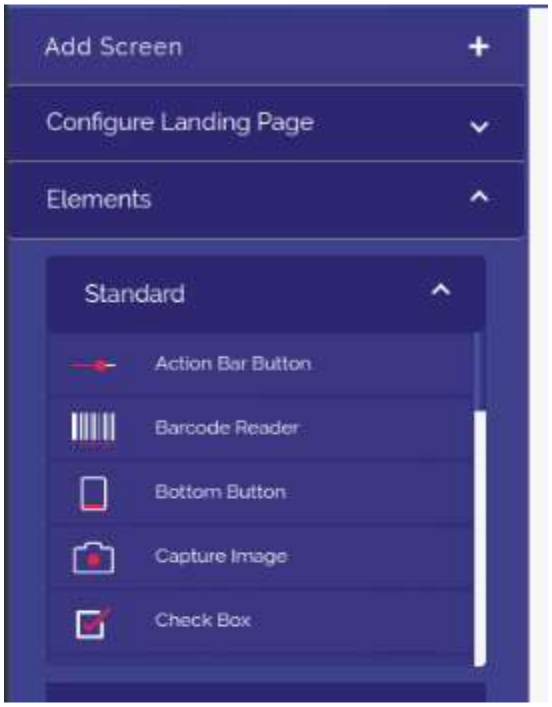
(Fig 3.1.2.1 (e))

3.1.2.2 Adding Capture Image Element

This section describes how to add Capture Image control on the form. After you add Capture Image control, you can configure this element to capture the image from your mobile device.

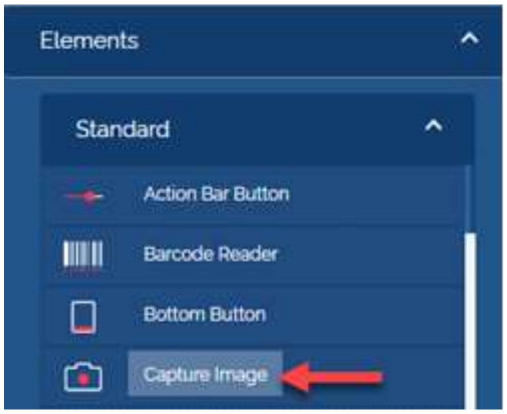
To add the image element:

1. On the left navigation pane, click **Standard**, the **Standard** menu **(Fig 3.1.2.2 (a))** expands.



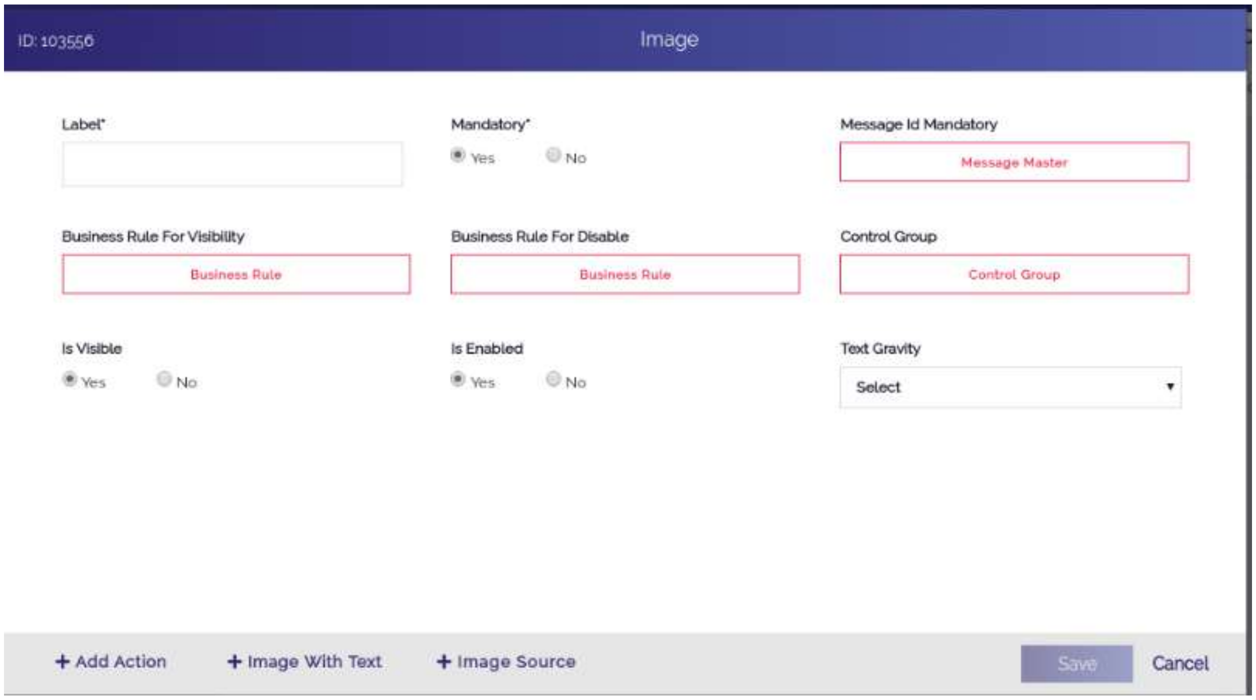
(Fig 3.1.2.2 (a))

2. Under **Standard**, scroll down and locate the **Capture Image** element **(Fig 3.1.2.2 (b))**.



(Fig 3.1.2.2 (b))

3. Click the **Capture Image** element, the **Image** dialog box **(Fig 3.1.2.2 (c))** opens.

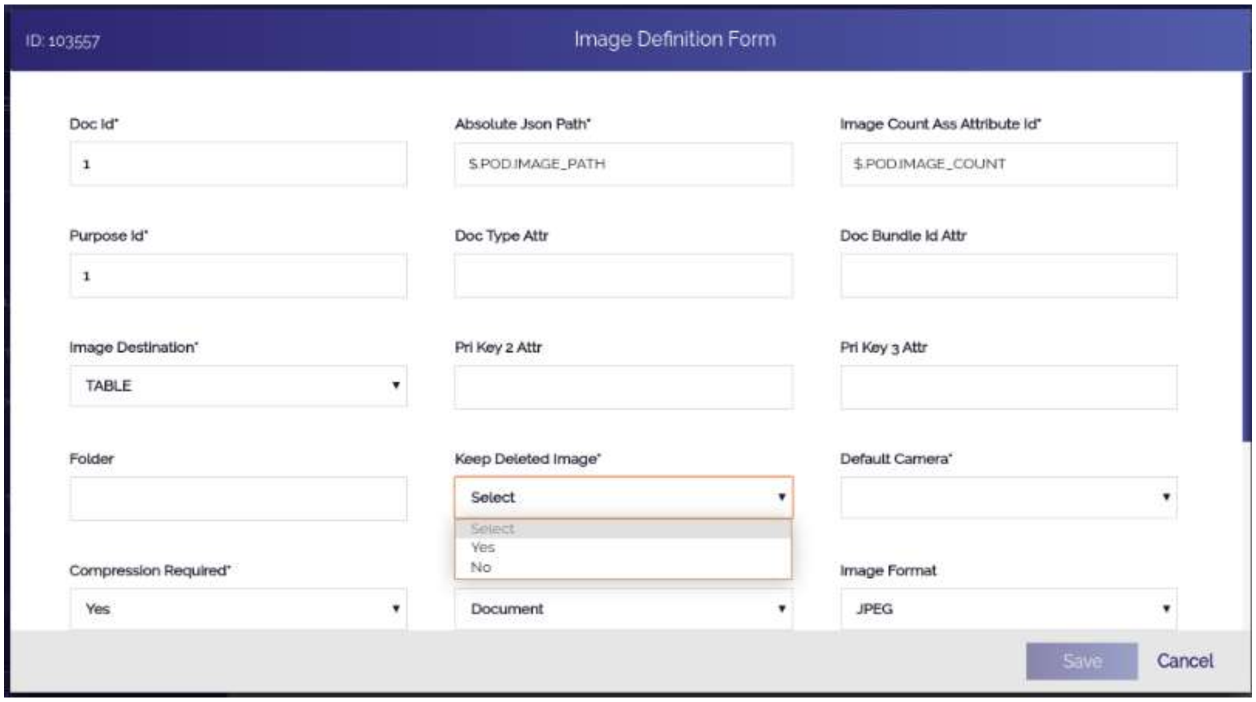


(Fig 3.1.2.2 (c))

4. In the **Image** dialog box, perform the functions as follows:

Box	Description
Label	In this box, enter the label name or title of the image (For example: - Upload Your Image).
Mandatory	Click **Yes** if you want to make the capture image function as a mandatory activity.
Message Id Mandatory	This feature is used to configure the message if the user forgets to perform the action on the element. To configure a new message: ➢ Click **Message Master**, the **Message Master** dialog box opens. ➢ In the **Message Master** dialog box, click the **Message Type** list and then select **I** to add information type message. ➢ After you select the message type, click the **Description** box. ➢ In the **Description** box, enter the text of message (For example: - Please upload your image.) ➢ After you enter the text/description of the message, click **Add Message**, the message is successfully configured.

5. After you perform these functions on the **Image** dialog box, click **Image Source (Fig 3.1.2.2 (c))**, the **Image Definition Form** dialog box **(Fig 3.1.2.2 (d))** opens.



Doc Id*: 1	Absolute Json Path*: $.POD.IMAGE_PATH	Image Count Ass Attribute Id*: $.POD.IMAGE_COUNT
Purpose Id*: 1	Doc Type Attr:	Doc Bundle Id Attr:
Image Destination*: TABLE	Pri Key 2 Attr:	Pri Key 3 Attr:
Folder:	Keep Deleted Image*: Select (Select / Yes / No)	Default Camera*:
Compression Required*: Yes	Document	Image Format: JPEG

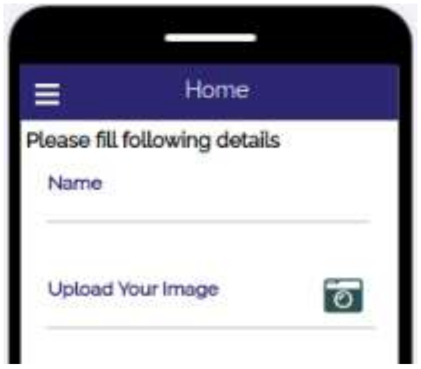
(Fig 3.1.2.2 (d))

6. In the **Image Definition Form** dialog box, enter values in the respective mandatory boxes as follows:

Box	Description
Absolute JSON Path	In this box, enter the path (For example: - $.POD.IMAGE_PATH) where the value of the image will be stored. **Note**:- Following the POD object, you can enter any random name (For example:- IMAGE_PATH) where the value of image is stored.
Image Count Ass Attribute	In this box, enter or define the attribute ID (For example: - $.POD.IMAGE_COUNT, in which the total count of captured images is stored. In this expression, the total count of captured images is stored in the IMAGE_COUNT variable.
Keep Deleted Image	In this list, click to select: ➢ **Yes** if you do not want to store the image file locally after the image is captured. ➢ **No** if want to store the image file locally after the image is captured.
Default Camera	In this list, click to select: ➢ **Front** if you want to use the front camera to capture the image. ➢ **Back** if you want to use the back camera to capture the image.
Compression Required	In this list, click to select: Yes if you want to compress the image after it is captured. No if you do not want to compress the image after it is captured.

Camera Template Id	Click this list to select the type of shape, in which you want the image should be displayed after it is captured. For instance: - If you select **Document**, the image is displayed in the rectangular shape after it is captured.
Max Image Allowed	In this box, enter the numeric value (For example:- 4) that specifies the maximum number of image the user can capture of single object/document.
Resolution	In this box, enter the resolution of the captured image in the following format: Height of image in pixel unit*Width of image in pixel unit (For example: - 320*640).
Image Size Kb	In this box, enter the numeric value (For example: - 1000) in the kilo byte unit. The entered value specifies the maximum amount of size of an image that you can capture.

7. After you enter values in the respective boxes, click **Save**, the image source is successfully configured.
8. On the **Image** dialog box, click Save, the **Capture Image** control **(Fig 3.1.2.2 (e))** is successfully configured.

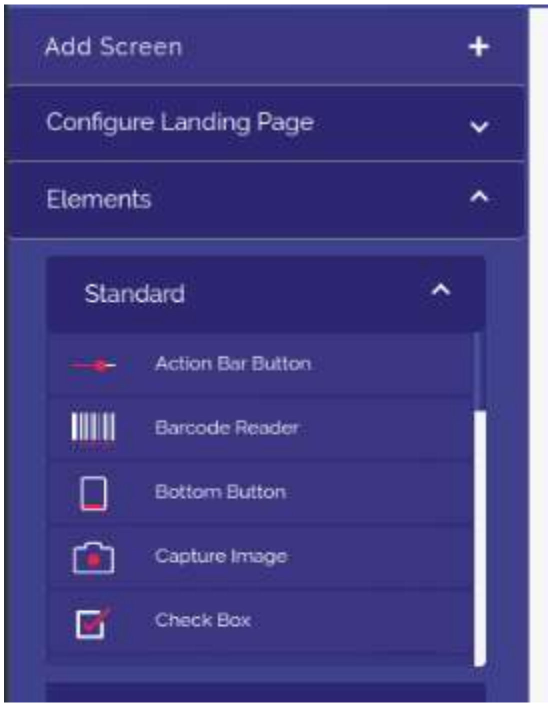


(Fig 3.1.2.2 (e))

3.1.2.3 Adding Date Element

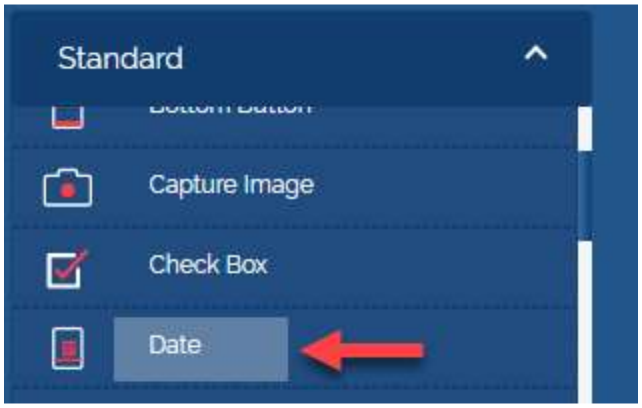
To add a date control/element:

1. On the left navigation pane, click **Standard**, the **Standard** menu **(Fig 3.1.2.3 (a))** expands.



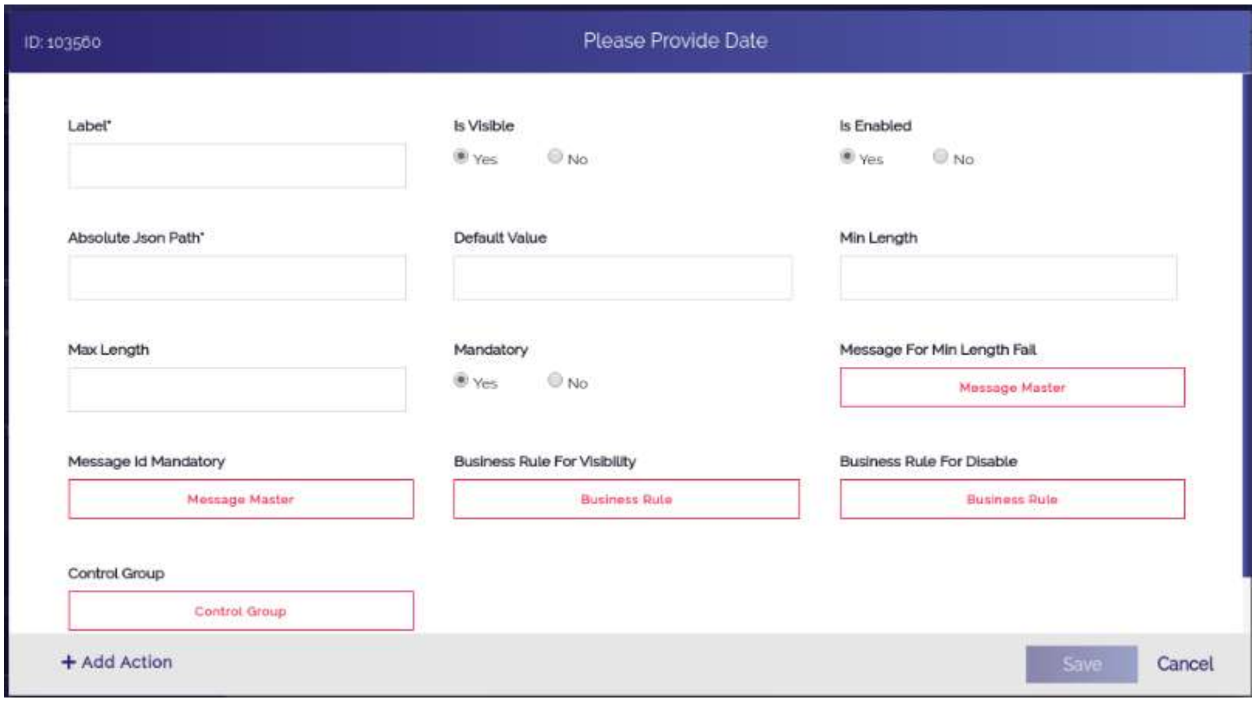
(Fig 3.1.2.3 (a))

2. Under **Standard**, scroll down and locate the **Date** element **(Fig 3.1.2.3 (b))**.



(Fig 3.1.2.3 (b))

3. Click the **Date** element, the **Please Provide Date** dialog box **(Fig 3.1.2.3 (c))** opens.


ID: 103560 Please Provide Date

Label*
Is Visible: (● Yes) ○ No
Is Enabled: (● Yes) ○ No
Absolute Json Path*
Default Value
Min Length
Max Length
Mandatory: (● Yes) ○ No
Message For Min Length Fail: Message Master
Message Id Mandatory: Message Master
Business Rule For Visibility: Business Rule
Business Rule For Disable: Business Rule
Control Group: Control Group
+ Add Action
Save
Cancel

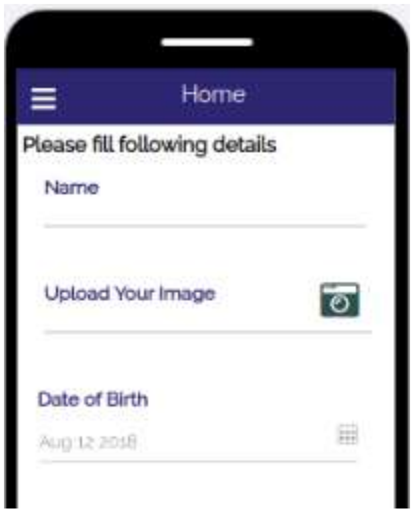
(Fig 3.1.2.3 (c))

4. In the **Please Provide Date** dialog box, enter values in the respective boxes as follows:

Box	Description
Label	In this box, enter the label name (For example: - Date of Birth) of the Date element.
Is Visible	Click **Yes** if you want to make the element visible after the respective form is displayed.
Is Enable	Click **Yes** if you want to make the element as active element. In the active element, the user can enter the value.
Absolute Json Path	In this box, enter the absolute json path as follows: $.POD.DOB. In the **$.POD.DOB** element, the DOB variable stores the value of the **Date of Birth** element.
Min Length	In this box, enter value: CD0-100Y. The value: **CD0-100Y** specifies that you can select the date for the last hundred years in the digital calendar.
Max Length	In this box, enter value: CD0. The value: **CD0** specifies that you can select the date up to current date in the digital calendar.
Mandatory	Click **Yes** to make this element as the mandatory field/element.
Message For Min Length Fall	This feature is used to configure the message after the user selects the wrong date. To configure the message: ➢ Click **Message Master**, the **Message Master** dialog box opens. ➢ In the **Message Master** dialog box, click the **Message Type** list and then select **E** to add error message. ➢ After you select the message type, click the **Description** box.

	➢ In the **Description** box, enter the text of message (For example: - Please enter correct date.) ➢ After you enter the text/description of the message, click **Add Message**, the message is successfully configured.
Message ID Mandatory	This feature is used to configure the message if the user forgets to enter the date. To configure the message: ➢ Click **Message Master**, the **Message Master** dialog box opens. ➢ In the **Message Master** dialog box, click the **Message Type** list and then select **E** to add error message. ➢ After you select the message type, click the **Description** box. ➢ In the **Description** box, enter the text of message (For example: - Please enter the date.) ➢ After you enter the text/description of the message, click **Add Message**, the message is successfully configured.

5. After you enter values in the respective boxes, click **Save**, the Date element is successfully configured.



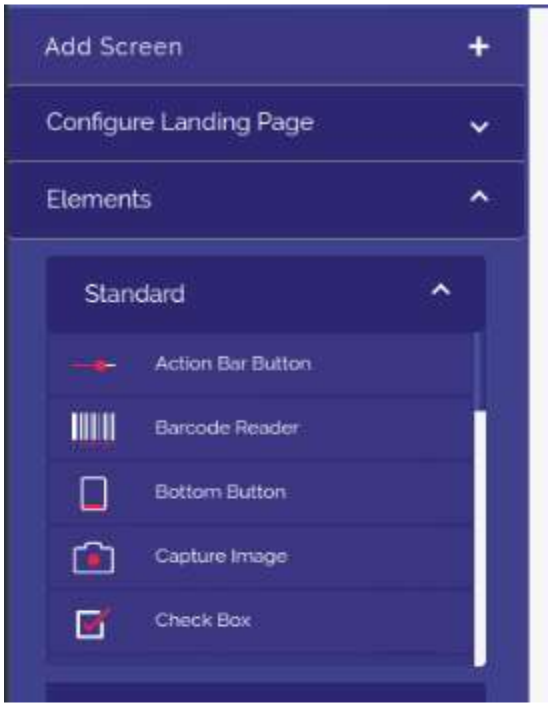
(Fig 3.1.2.3 (d))

3.1.2.4 Adding Radio Button Element

The radio button element works as a mutually exclusive option button. In the mutually exclusive option buttons, you can select only one option at a given point of time.

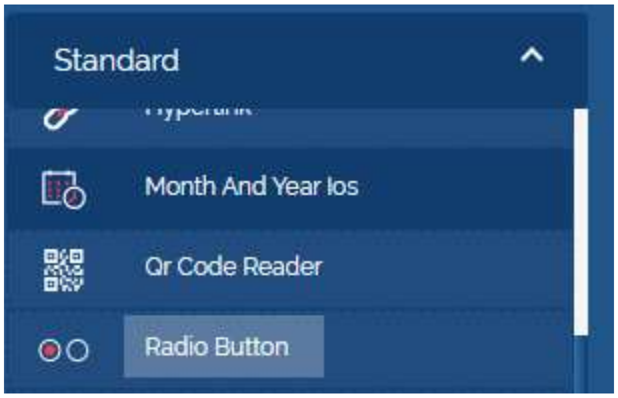
To add the radio button element:

1. On the left navigation pane, click **Standard**, the **Standard** menu **(Fig 3.1.2.4 (a))** expands.



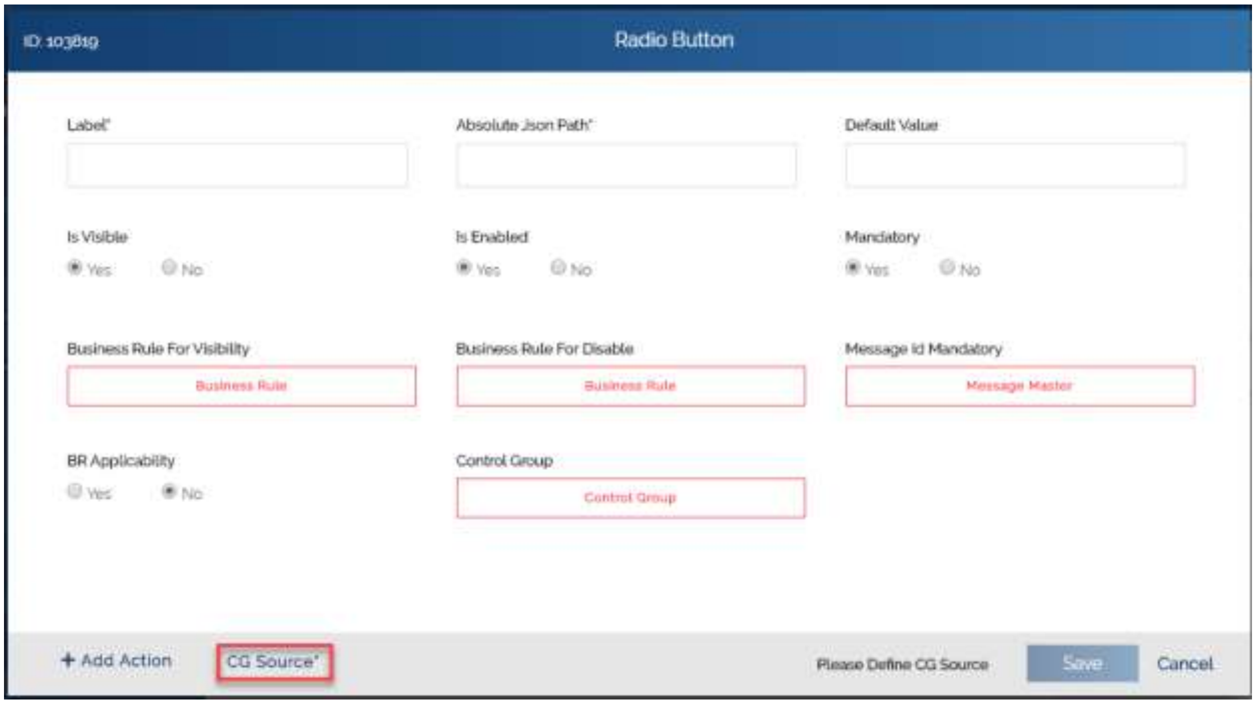
(Fig 3.1.2.4 (a))

2. Under **Standard**, scroll down and locate the **Radio Button** element **(Fig 3.1.2.4 (b))**.



(Fig 3.1.2.4 (b))

3. Click the **Radio Button** element, the **Radio Button** dialog box **(Fig 3.1.2.4 (c))** opens.

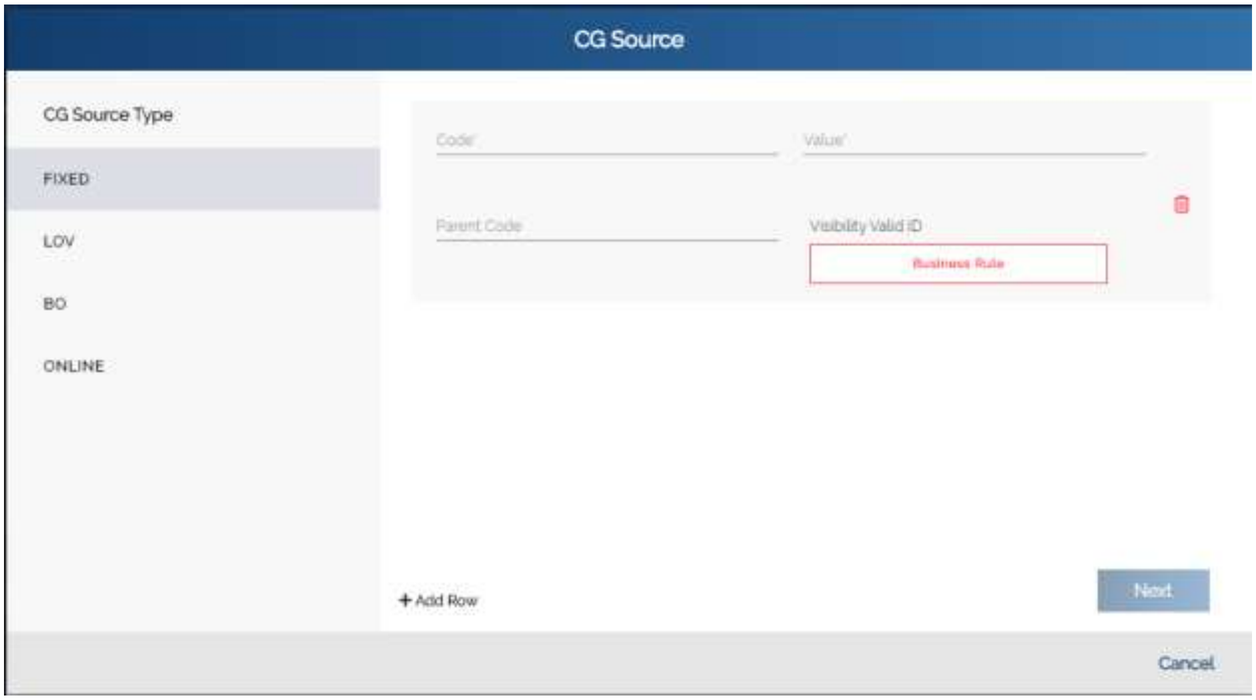


(Fig 3.1.2.4 (c))

4. On the **Radio Button** dialog box, enter values in the respective boxes as follows:

Box	Description
Label	In this box, enter the label (For example: - Gender) of the radio button. This label name can contain multiple options as radio button.
Absolute Json Path	In this box, enter the absolution Json path as follows: $.POD.GENDER In the Json path, the **Gender** variable stores the value of radio button.
Is Visible	Click **Yes** if you want to make the element visible after the respective form is displayed.
Is Enable	Click **Yes** if you want to make the element as an active element. The user can perform the function on the active element.
Mandatory	Click **Yes** if you want to make the element as a mandatory element.

5. After you enter values in the respective boxes, click **CG Source (Fig 3.1.2.4 (c))**, the **CG Source** dialog box **(Fig 3.1.2.4 (d))** opens.



(Fig 3.1.2.4 (d))

6. On the **CG Source** dialog box, locate the **CG Source** left panel **(Fig 3.1.2.4 (d))**.
7. Under **CG Source**, click **Fixed (Fig 3.1.2.4 (d))**, a group of different fields opens.
8. In the group of fields, enter values in the respective boxes as follows:

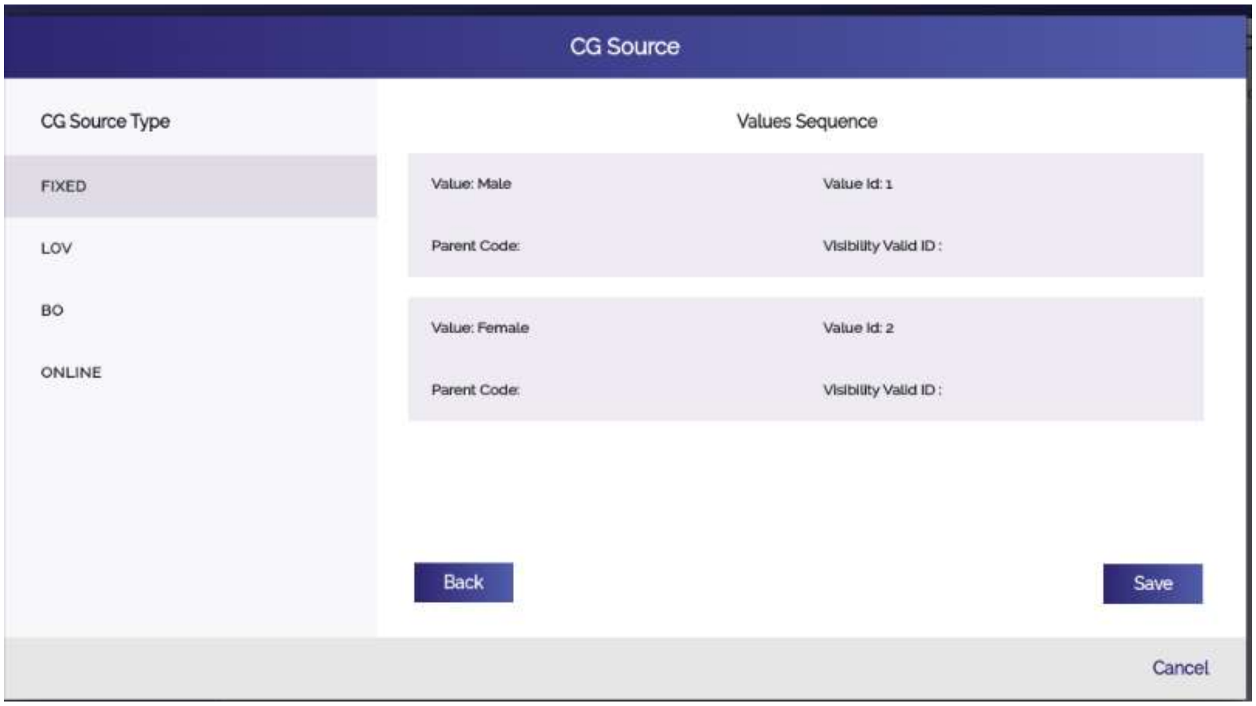
Box	Description
Code	In this box, enter abbreviated or one character code (For example: - 1) of the actual value. For instance: - You can enter 1 numeric value as a code for the radio button with value: **Male**.
Value	In this box, enter the actual value (For example: - Male) of the radio button that is displayed on the form.

9. To add the next radio button, click **Add Row (Fig 3.1.2.4 (d))**, another group of fields opens.
10. In the group of fields, enter values as follows:

➢ In the **Code** box, enter **2**.

➢ In the **Value** box, enter **Female**.

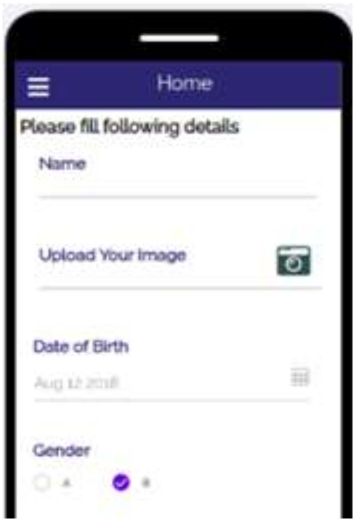
11. Repeat the steps 9 and 10 to add other radio buttons.
12. After you add all the radio buttons, click **Next (Fig 3.1.2.4 (d))**, the **Value Sequence** panel **(Fig 3.1.2.4 (e))** displays the details of radio buttons that you have added.



Value: Male	Value Id: 1
Parent Code:	Visibility Valid ID :
Value: Female	Value Id: 2
Parent Code:	Visibility Valid ID :

(Fig 3.1.2.4 (e))

13. After you ensure that the details of added radio buttons are correct, click **Save**, the **CG Source** dialog box is closed.
14. On the Radio Button dialog box, click Save, the **Radio Button** element is successfully configured.



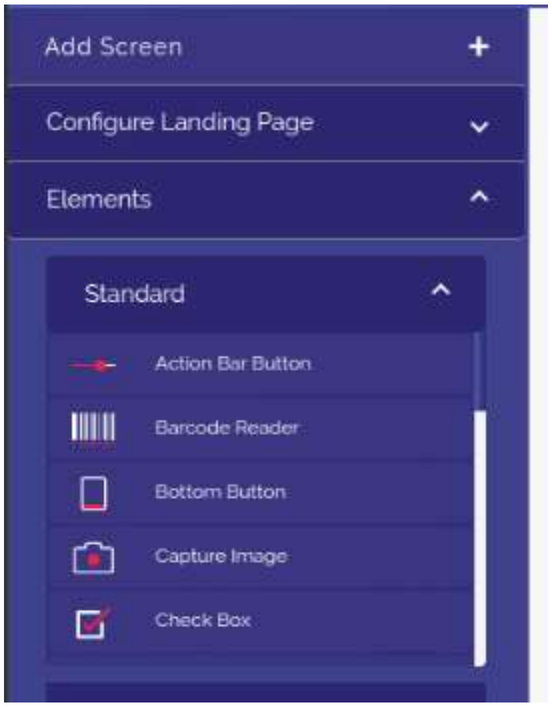
(Fig 3.1.2.4 (e))

3.1.2.5 Adding Bottom Button

In the mobile app, bottom button works similar to a push button. The bottom button remains static and is placed at the bottom of the mobile app after it is configured in the vDesigner application.

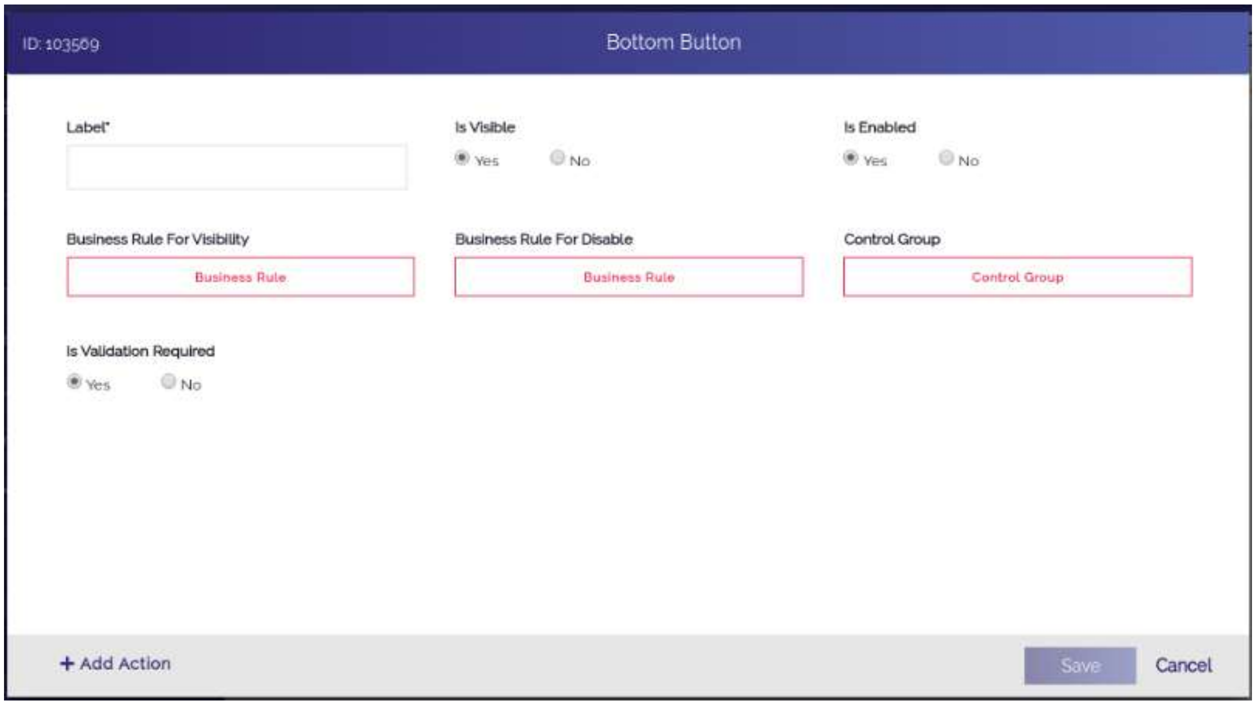
To add a bottom button:

1. On the left navigation pane, click **Standard**, the **Standard** menu **(Fig 3.1.2.5 (a))** expands.



(Fig 3.1.2.5 (a))

2. Under **Standard**, click the **Bottom Button** link, the **Bottom Button** dialog box **(Fig 3.1.2.5 (b))** opens.


ID: 103569

Bottom Button

Label*

Is Visible: [✓ Yes] No

Is Enabled: [✓ Yes] No

Business Rule For Visibility: Business Rule

Business Rule For Disable: Business Rule

Control Group: Control Group

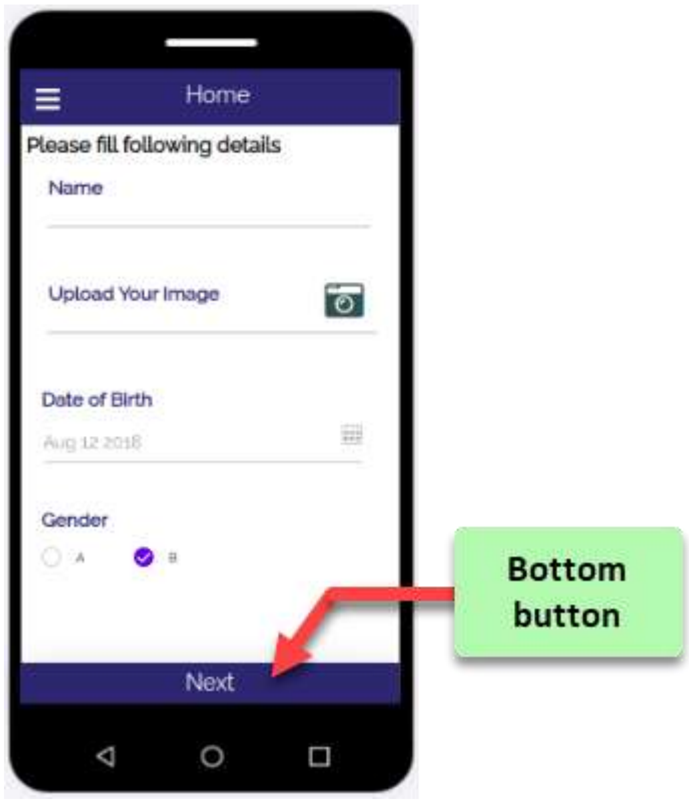
Is Validation Required: [✓ Yes] No

(Fig 3.1.2.5 (b))

3. On the **Bottom Button** dialog box **(Fig 3.1.2.5 (b))**, enter or select values in the respective boxes as follows:

Box	Description
Label	In this box, enter the label name of the element.
Is Visible	Click **Yes** if you want to make the element visible after the respective form is displayed.
Is Enable	Click **Yes** if you want to make the element as an active element. The user can perform the function on the active element.

4. After you enter the values in the respective boxes, click **Save**, the bottom button is successfully configured.


Home
Please fill following details
Name
Upload Your Image
Date of Birth
Aug 12 2018
Gender
A
B
Next
Bottom button

(Fig 3.1.2.5 (c))

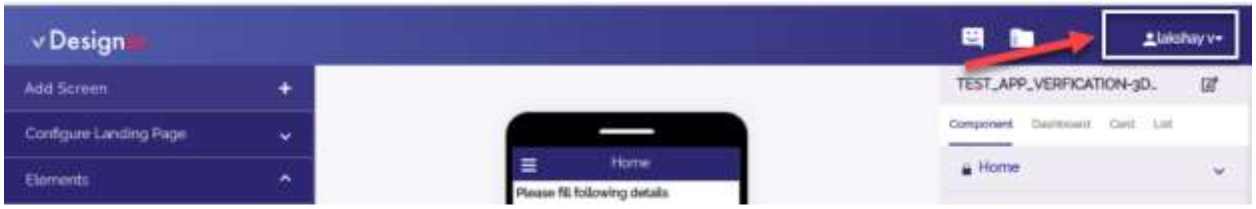
3.1.2.6 Component Migration

After you add new screens and controls or modify the existing screen(s) or controls by using the vDesigner application, you mandatorily need to migrate the added or updated components from the local environments to the cloud environment. If you do not migrate the component, the mobile app will not display newly added screens and controls after you open and then access the mobile app.

The feature of component migration pushes newly added and/or updated components to the cloud environment so that when you access the application, you can view and observe the newly added/updated screens and controls.

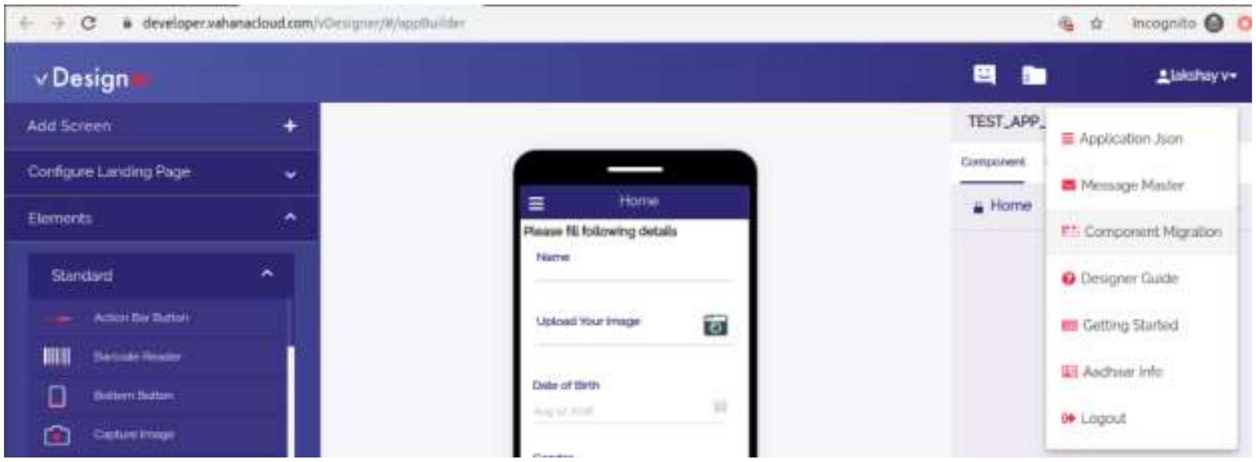
To migrate the component:

1. On the right top corner of the vDesigner dashboard, locate the user name icon **(Fig 3.1.2.6 (a))**.


vDesign
Add Screen
Configure Landing Page
Elements
lakshay v
TEST_APP_VERIFICATION-3D_
Component Dashboard Card List
Home
Please fill following details

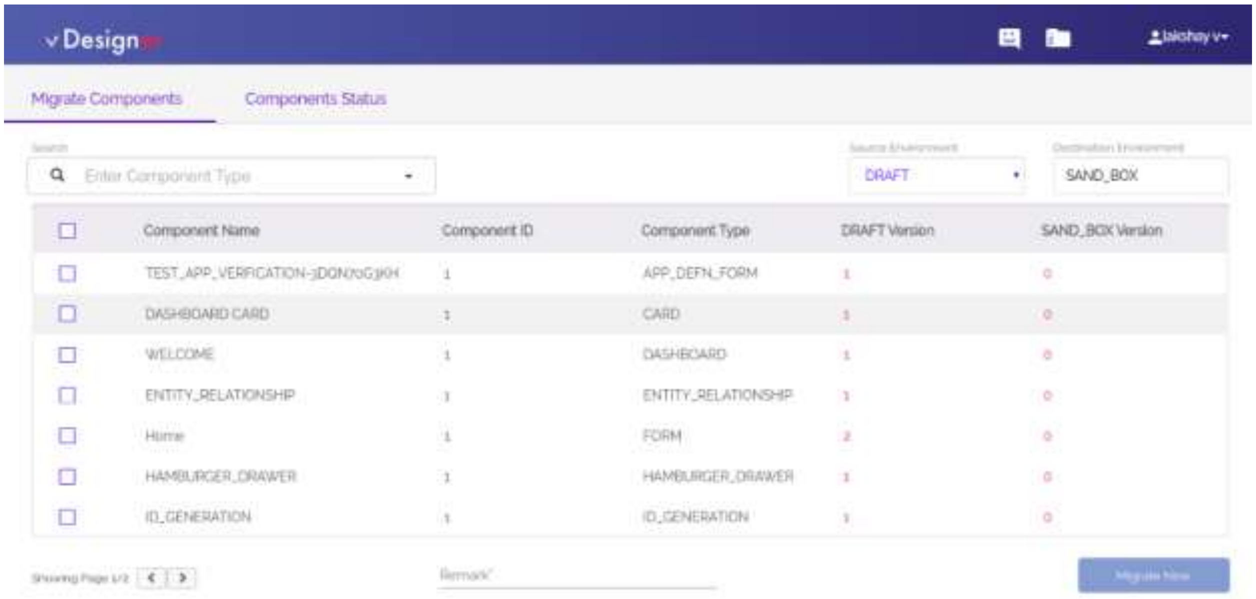
(Fig 3.1.2.6 (a))

2. Click the arrow right to the user name **(Fig 3.1.2.6 (a))**, a menu **(Fig 3.1.2.6 (b))** expands.



(Fig 3.1.2.6 (b))

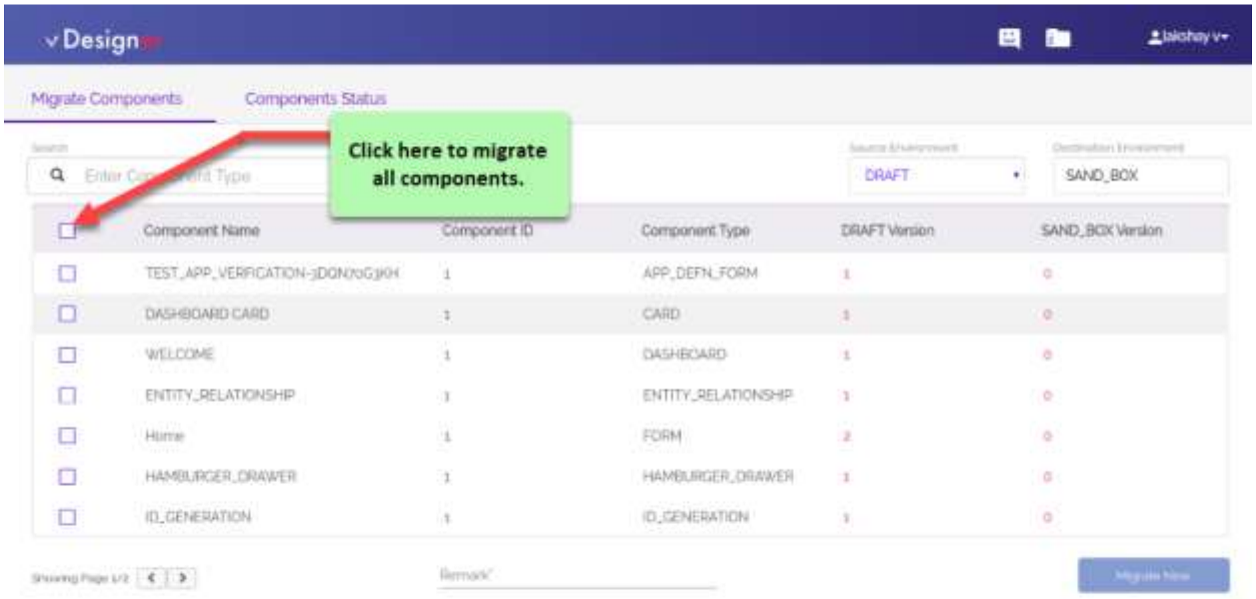
3. In the menu, click **Component Migration**, the **Component Migration** dashboard **(Fig 3.1.2.6 (c))** opens.



☐	Component Name	Component ID	Component Type	DRAFT Version	SAND_BOX Version
☐	TEST_APP_VERFICATION-3DQN70G3KH	1	APP_DEFN_FORM	1	0
☐	DASHBOARD CARD	1	CARD	1	0
☐	WELCOME	1	DASHBOARD	1	0
☐	ENTITY_RELATIONSHIP	1	ENTITY_RELATIONSHIP	1	0
☐	Home	1	FORM	2	0
☐	HAMBURGER_DRAWER	1	HAMBURGER_DRAWER	1	0
☐	ID_GENERATION	1	ID_GENERATION	1	0

(Fig 3.1.2.6 (c))

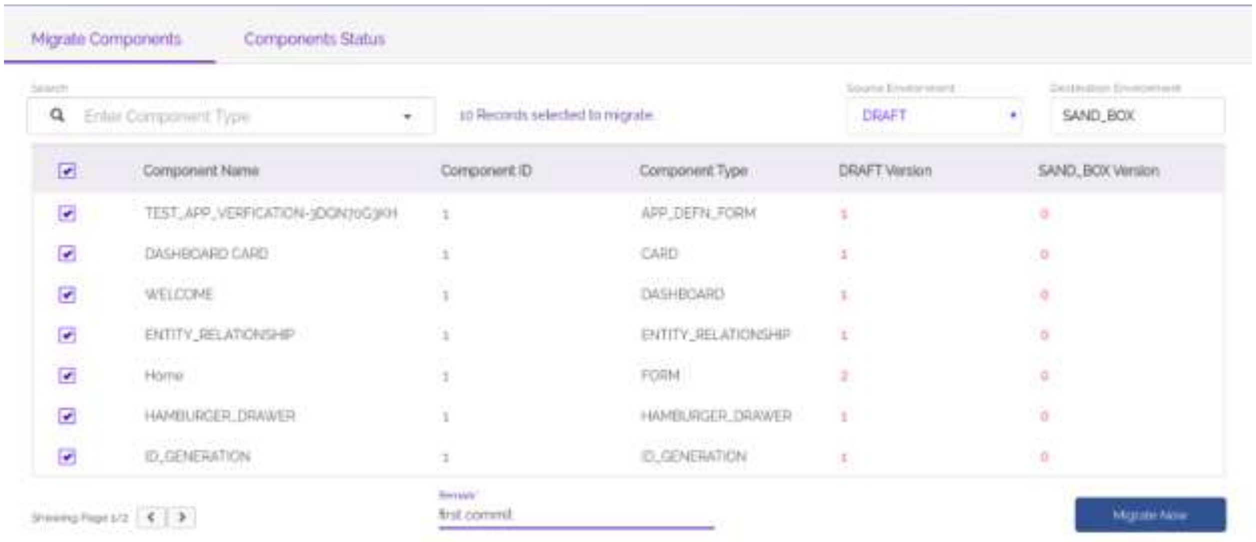
4. In the **Component Migration** dashboard **(Fig 3.1.2.6 (c))**, the **Migrate Components** tab displays the list of all components.
5. To migrate all the newly added or modified components, locate the check box left to the **Component Name** column **(Fig 3.1.2.6 (d))**.



☐	Component Name	Component ID	Component Type	DRAFT Version	SAND_BOX Version
☐	TEST_APP_VERFICATION-3DQN70G3KH	1	APP_DEFN_FORM	1	0
☐	DASHBOARD CARD	1	CARD	1	0
☐	WELCOME	1	DASHBOARD	1	0
☐	ENTITY_RELATIONSHIP	1	ENTITY_RELATIONSHIP	1	0
☐	Home	1	FORM	2	0
☐	HAMBURGER_DRAWER	1	HAMBURGER_DRAWER	1	0
☐	ID_GENERATION	1	ID_GENERATION	1	0

(Fig 3.1.2.6 (d))

6. Click the check box left to the **Component Name** column **(Fig 3.1.2.6 (d))**, all components that were added or modified are also selected **(Fig 3.1.2.6 (e))**.

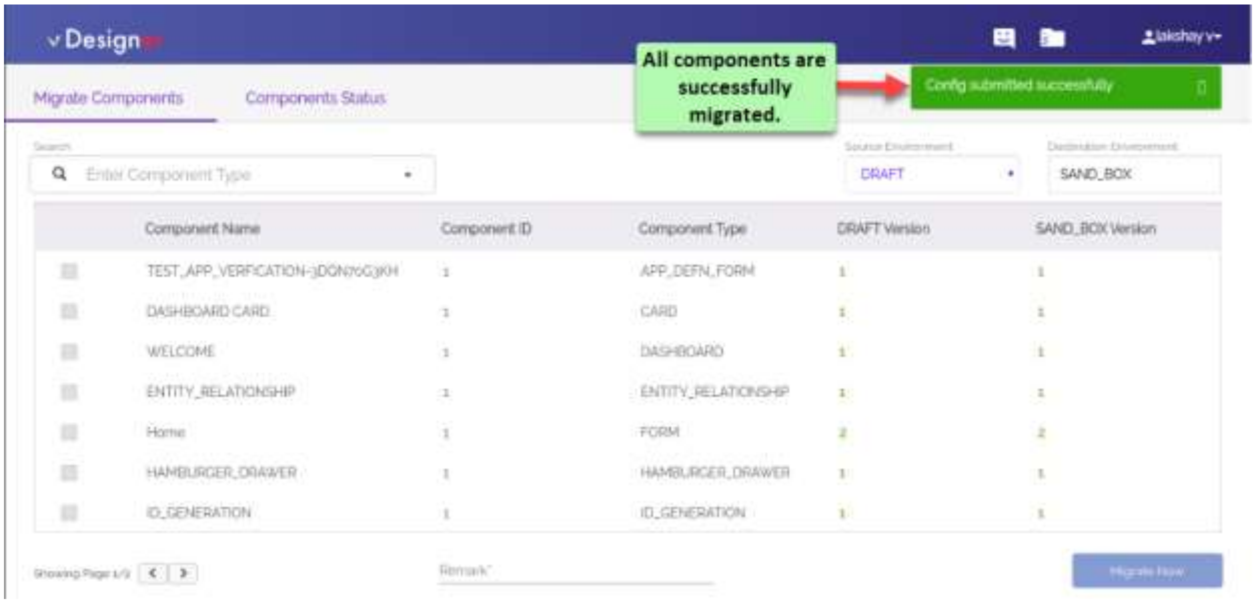


☑	Component Name	Component ID	Component Type	DRAFT Version	SAND_BOX Version
☑	TEST_APP_VERFICATION-3DQN70G3KH	1	APP_DEFN_FORM	1	0
☑	DASHBOARD CARD	1	CARD	1	0
☑	WELCOME	1	DASHBOARD	1	0
☑	ENTITY_RELATIONSHIP	1	ENTITY_RELATIONSHIP	1	0
☑	Home	1	FORM	2	0
☑	HAMBURGER_DRAWER	1	HAMBURGER_DRAWER	1	0
☑	ID_GENERATION	1	ID_GENERATION	1	0

Remark*: first commit

(Fig 3.1.2.6 (e))

7. In the **Remark** field, enter brief text (For example: - First Commit) or description about component migration.
8. Click **Migrate Now (Fig 3.1.2.6 (e))**, the all selected components are successfully migrated.



Config submitted successfully

	Component Name	Component ID	Component Type	DRAFT Version	SAND_BOX Version
☐	TEST_APP_VERFICATION-3DQN70G3KH	1	APP_DEFN_FORM	1	1
☐	DASHBOARD CARD	1	CARD	1	1
☐	WELCOME	1	DASHBOARD	1	1
☐	ENTITY_RELATIONSHIP	1	ENTITY_RELATIONSHIP	1	1
☐	Home	1	FORM	2	2
☐	HAMBURGER_DRAWER	1	HAMBURGER_DRAWER	1	1
☐	ID_GENERATION	1	ID_GENERATION	1	1

(Fig 3.1.2.6 (f))

3.2 Creating New Entity and Defining Relationship between Entities

The vDesigner application provides you the **Entity and Relationship** feature that contains two different sub-modules: ***Entity*** and ***Relationship***. The ***entity*** module allows you to create entity. The ***relationship*** module allows to establish the relationship between two entities.

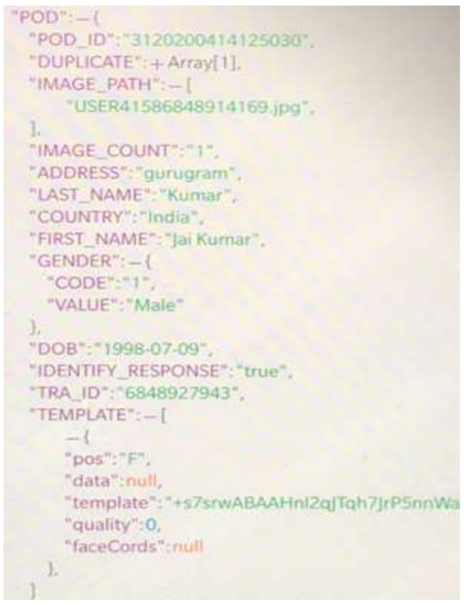
While establishing relationship between two entities, you make one entity as a parent entity and therefore can define another entity as child entity of the parent entity. That is how you can define multiple child entities under single parent entity.

The concept of creating an entity and defining relationship between two entities is broadly described as below:

3.2.1 Creating New Entity

The vDesigner application provides POD as a default entity/object. Apart from POD, you can create a new entity and use it later. You can create a new entity as a parent entity or a child entity. Under POD, you can also create a child entity. **The main objective of creating an entity is to store the entire data of mobile app in the JSON format**.

The concept of creating and using an entity can be better interpreted by observing the following screen capture: **(Fig 3.2.1 (a))**



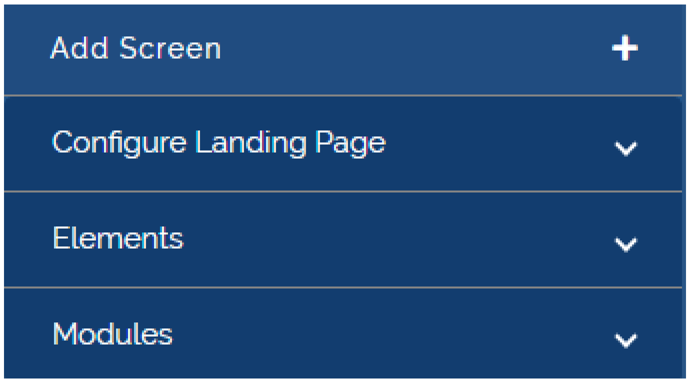
```
"POD":—{
  "POD_ID":"3120200414125030",
  "DUPLICATE":+Array[1],
  "IMAGE_PATH":—[
      "USER41586848914169.jpg",
  ],
  "IMAGE_COUNT":"1",
  "ADDRESS":"gurugram",
  "LAST_NAME":"Kumar",
  "COUNTRY":"India",
  "FIRST_NAME":"Jai Kumar",
  "GENDER":—{
    "CODE":"1",
    "VALUE":"Male"
  },
  "DOB":"1998-07-09",
  "IDENTIFY_RESPONSE":"true",
  "TRA_ID":"6848927943",
  "TEMPLATE":—[
      —{
      "pos":"F",
      "data":null,
      "template":"+s7srwABAAHnl2qJTqh7JrP5nnWa
      "quality":0,
      "faceCords":null
    },
  ]
```

(Fig 3.2.1 (a))

In the screen capture: **(Fig 3.2.1 (a))**, the **POD** object stores the data of mobile app in the JSON format. In the JSON code, the **POD_ID** variable stores the unique ID that can be used to manage the data of mobile app in the JSON format.

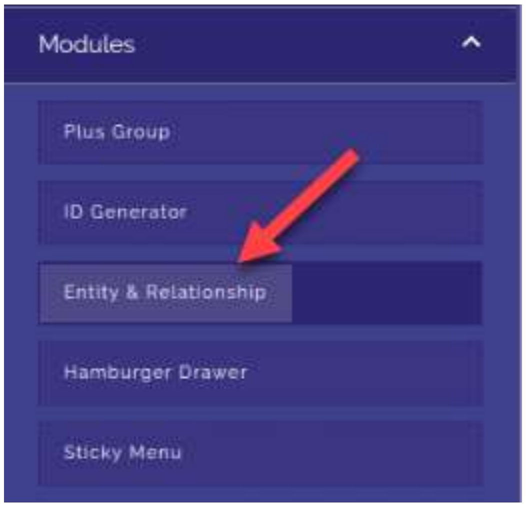
You can create an entity as follows. To create an entity:

1. On the vDesigner dashboard, locate the left navigation pane **(Fig 3.2.1 (b))**.



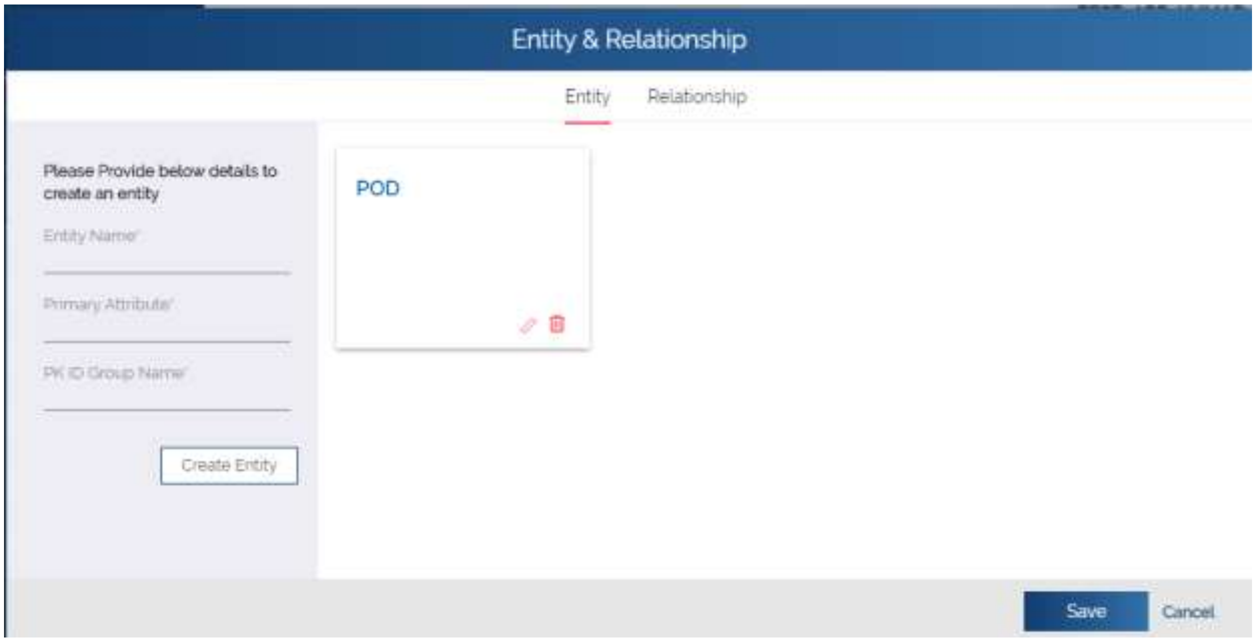
(Fig 3.2.1 (b))

2. In the left navigation pane, click **Modules**, the navigation pane **(Fig 3.2.1 (c))** expands.



(Fig 3.2.1 (c))

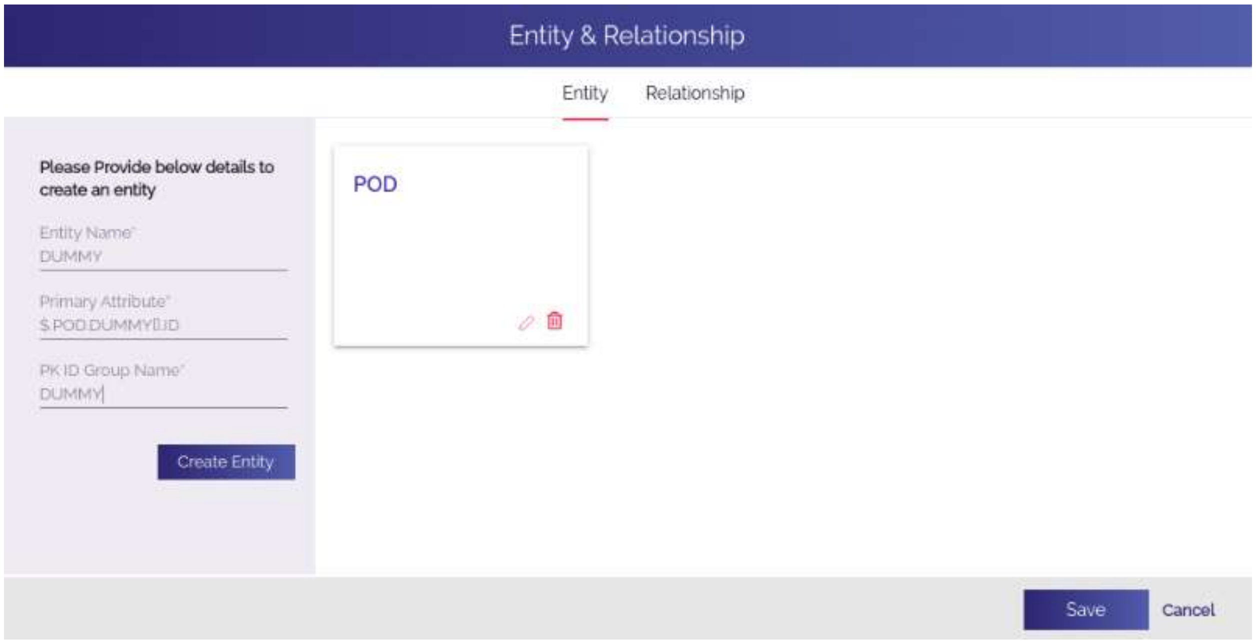
3. Under **Modules**, click **Entity & Relationship**, the **Entity & Relationship** dialog box **(Fig 3.2.1 (d))** opens.



(Fig 3.2.1 (d))

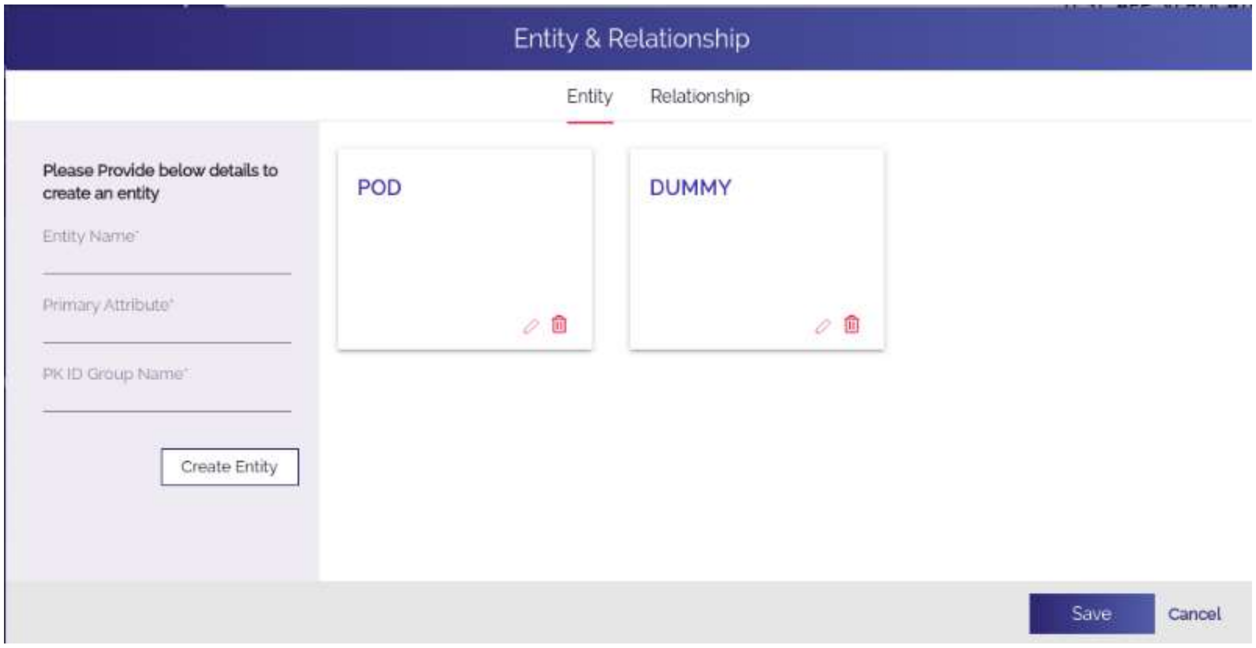
4. In the **Entity & Relationship** dialog box, enter the values to create an entity as follows:

Box	Description
Entity Name	In this box, enter the name of entity (For example: - DUMMY).
Primary Attribute	In this box, enter the value as follows: $.POD.DUMMY[].ID In this hardcoded value, **DUMMY[]** is the array type object that stores the value of the **ID** variable. The POD entity is used to store the data in the JSON object.
PK ID Group Name	In this box, you can perform function with any of two options: ➢ **Option1:-** (If you want to configure the ID generator for the new entity) In this box, enter the name of entity: **Dummy** and then define the ID generator rule as described in the heading section: Configuring ID Generator Rule. ➢ **Option2:-** (If you do not want to configure the ID generator for the new entity) In this box, only enter the name of entity: **Dummy** that you are creating. In this case, a new ID will not be created.



(Fig 3.2.1 (e))

5. After you enter the value in the respective boxes **(Fig 3.2.1 (e))**, click **Create Entity**, the entity **(Fig 3.2.1 (f))** is successfully created.



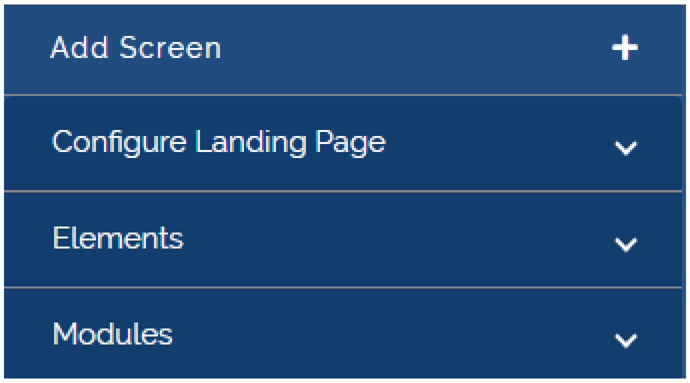
(Fig 3.2.1 (f))

3.2.2 Defining Relationship between Two Entities

As described earlier in the main heading section: Creating New Entity and Defining Relationship, you can define the relationship between two entities. While defining the relationship, you can make one entity as a child entity of another entity. You can define the relationship between two entities as follows:

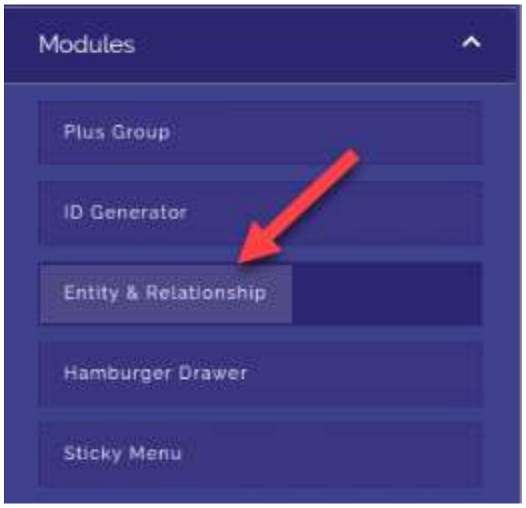
To define the relationship:

1. On the vDesigner dashboard, locate the left navigation pane **(Fig 3.2.2 (a))**.



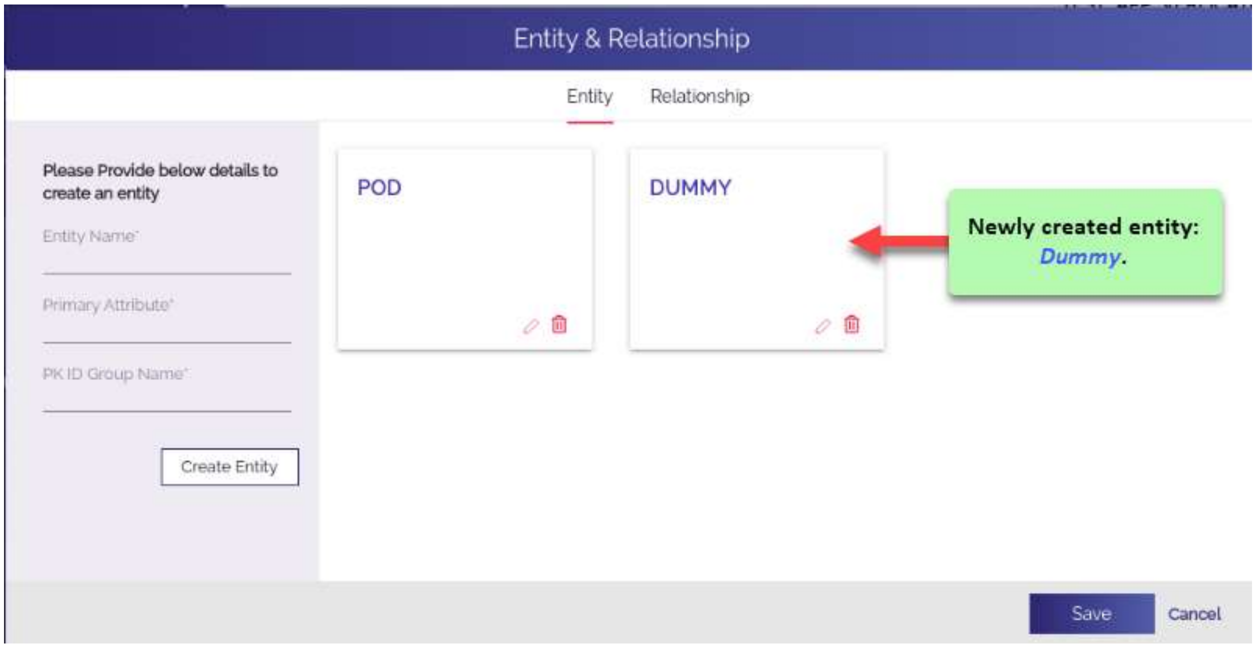
(Fig 3.2.2 (a))

2. In the left navigation pane, click **Modules**, the navigation pane **(Fig 3.2.2 (b))** expands.



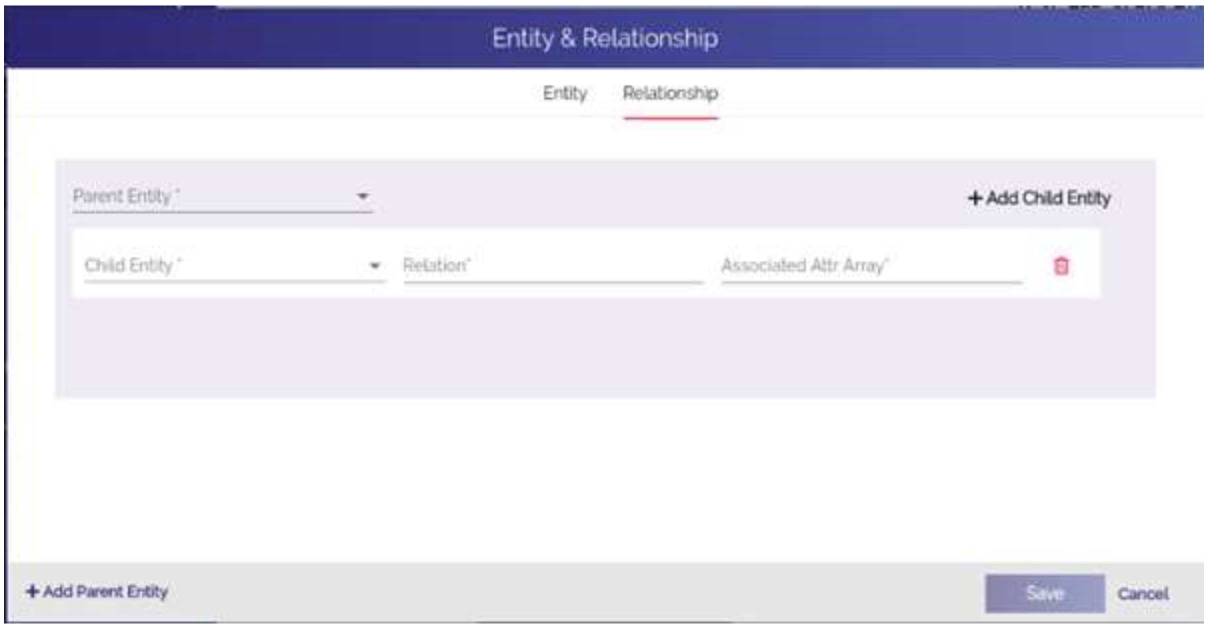
(Fig 3.2.2 (b))

3. Under **Modules**, click **Entity & Relationship**, the **Entity & Relationship** dialog box **(Fig 3.2.1 (c))** opens.



(Fig 3.2.2 (c))

4. In the **Entity & Relationship** dialog box, click the **Relationship** tab **(Fig 3.2.2 (c))**, the **Entity & Relationship** dialog box displays the following fields (boxes) **(Fig 3.2.2 (d))**:
   - ➢ **Parent Entity**
   - ➢ **Child Entity**
   - ➢ **Relation**
   - ➢ **Associated Attr Array**



(Fig 3.2.2 (d))

5. In these fields **(Fig 3.2.2 (d))**, enter/select values as follows:

Box/List	**Description**
Parent Entity	Click this list, it displays the currently existing entities. In the list, select the entity (For example: - POD) that you want to make as a parent entity.
Child Entity	Click this list, it also displays the currently existing entities. In the list, select the entity (For example: - DUMMY) that you want to make as a child entity.
Relation	In this box, enter the following value: 1*n This value specifies the one-to-many relationship between parent and the child entity.
Associated Attr Array	In this box, enter the hardcoded value as follows: $.POD.DUMMY[] This hardcoded expression specifies that the DUMMY[] array type JSON object has been defined as a child object under POD entity.

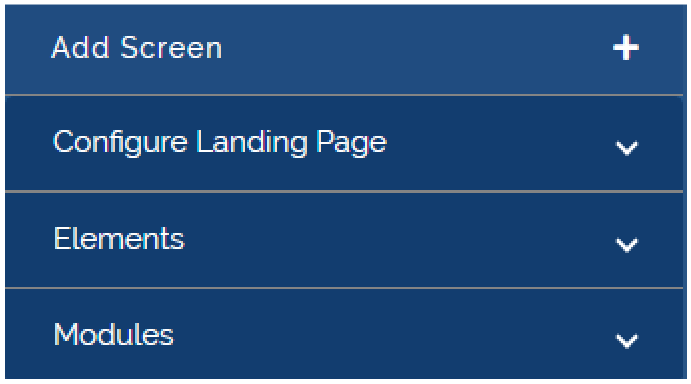
6. After you enter/select the value in the respective box/list, click **Save (Fig 3.2.2 (d))**, the parent-child relationship between two entities is successfully defined.

3.2.3 Configuring ID Generator

This feature allows you to define the ID generator rule for newly created entity. After you configure ID generator rule, it automatically generates a unique ID for the respective entity when **Load New Object** task executes as a result of specific action.

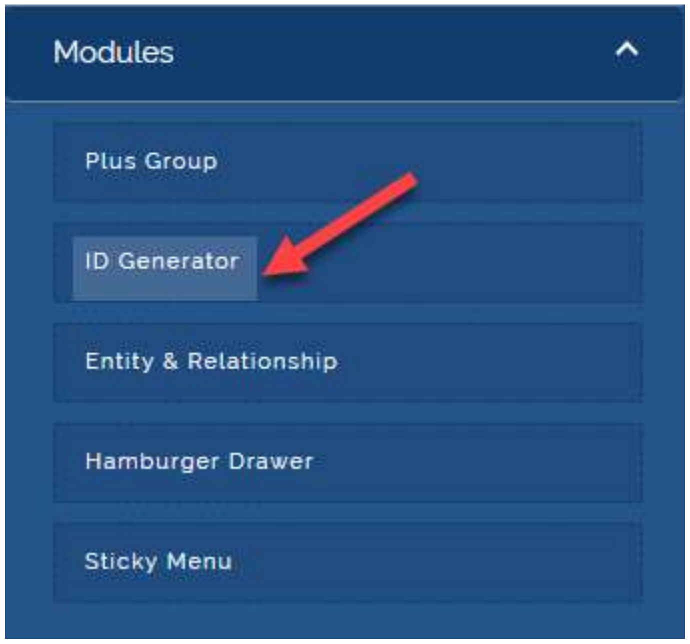
To configure ID generator:

1. On the vDesigner dashboard, locate the left navigation pane **(Fig 3.2.3 (a))**.



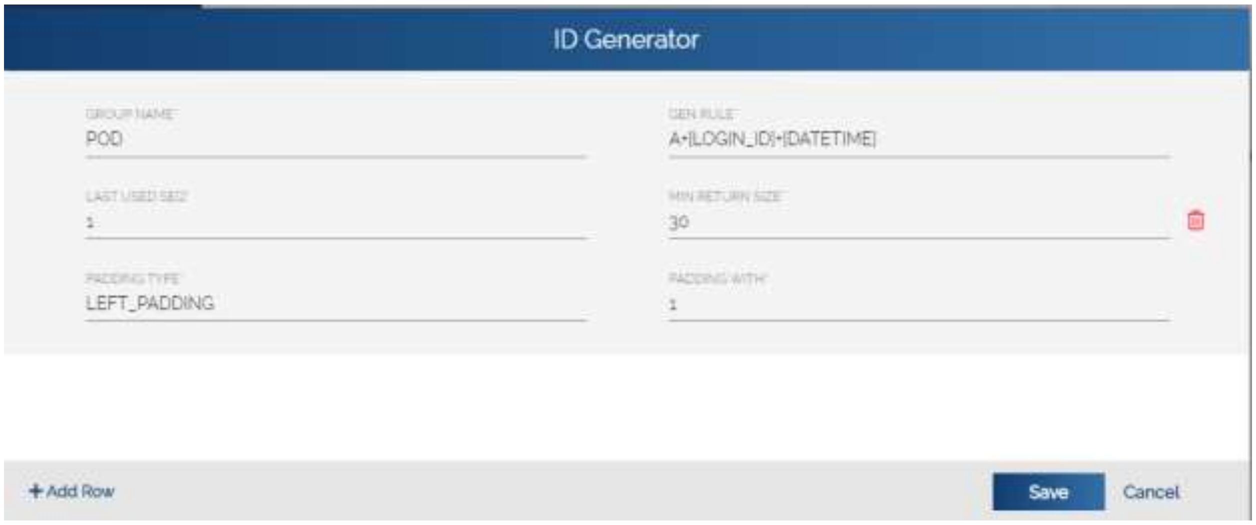
(Fig 3.2.3 (a))

2. In the left navigation pane, click **Modules**, the navigation pane **(Fig 3.2.3 (b))** expands.



(Fig 3.2.3 (b))

3. Under **Modules**, click **ID Generator (Fig 3.2.3 (b))**, the **ID Generator** dialog box **(Fig 3.2.3 (c))** opens.


ID Generator

GROUP NAME: POD

GEN RULE: A+[LOGIN_ID]+[DATETIME]

LAST USED SEQ: 1

MIN RETURN SIZE: 30

PADDING TYPE: LEFT_PADDING

PADDING WITH: 1

(Fig 3.2.3 (c))

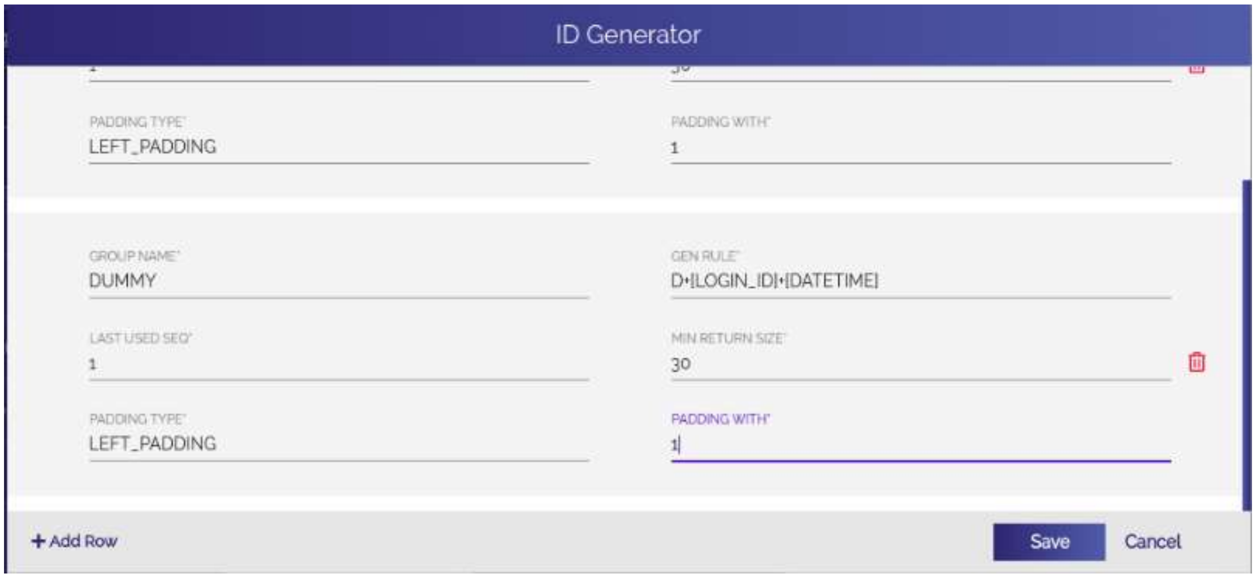
4. In the **ID Generator** dialog box, click **Add Row (Fig 3.2.3 (c))**, the **ID Generator** dialog box displays the group of the following fields:

- ➢ **GROUP NAME**
- ➢ **LAST USED SEQ**
- ➢ **PADDING TYPE**
- ➢ **GEN RULE**
- ➢ **MIN RETURN SIZE**
- ➢ **PADDING WITH**

5. In these fields (boxes), enter values as follows:

Box/Field	**Description**
Group Name	In this enter the name of entity (For example: - DUMMY) for which you want to configure the ID generator rule.
Last Used Seq	In this box, enter the numeric value: **1**.
Padding Type	In this box, enter: **LEFT_PADDING**. The value: **LEFT_PADDING** specifies that the ID generator rule will start generating the unique ID from the left side character.
Gen Rule	In this box, enter/write the expression of ID generator rule that you want to apply to create unique ID. **#<u>Sample ID generator rule</u>:** D+[LOGIN_ID]+[DATETIME] When this ID generator rule executes to create unique ID, the unique ID contains: ➢ The character: **D**, which is prefixed to the entire value of the unique ID. ➢ **LOGIN_ID**, which denotes the login Id/login name of the mobile app user. ➢ **DATETIME**, which denotes the date and time at which the unique ID is generated.
Min Return Size	In this box, enter the numeric value: **30**.

Padding With	In this box, enter the numeric value: **1**.



PADDING TYPE*	PADDING WITH*
LEFT_PADDING	1
GROUP NAME*	GEN RULE*
DUMMY	D+[LOGIN_ID]+[DATETIME]
LAST USED SEQ*	MIN RETURN SIZE*
1	30
PADDING TYPE*	PADDING WITH*
LEFT_PADDING	1

(Fig 3.2.3 (d))

6. After you enter values in the respective boxes, click **Save (Fig 3.2.3 (d))**, the ID generator rule is successfully configured.

3.3 Integrating Different API(s) with vDesigner

In the vDesigner application, you can integrate different API to bind an action to different elements and controls. You can integrate two types of API(s), which are given as below:

- **Sync API**
- **Other API(s)**

The sync API is specially used to insert the mobile app data into two tables: ***tb_cop_buss_obj_txn*** and ***tb_image_data_obj***. The ***tb_cop_buss_obj_txn*** table stores the text and numeric data, while the ***tb_image_data_obj*** table stores image related data.

API(s) that falls in the Other API(s) category captures the response from third party server, apart from manipulating data on the database level such as fetching data from the database table. The database API(s) are mainly used to retrieve the old data from the database tables. In the database API(s), internally implemented stored procedure executes and then calls corresponding service to perform the function.

In the vDesigner application, specific API is integrated based on the functional requirement of the mobile app. You can integrate the API as follows:

3.3.1 Integrating Other API(s)

When you integrate an API in the vDesigner application, you map two types of parameters: ***Request Parameters*** and ***Response Parameters***. In the vDesigner application, you map the request parameters on the basis of template body parameters of an API that you define in the vConnect portal.

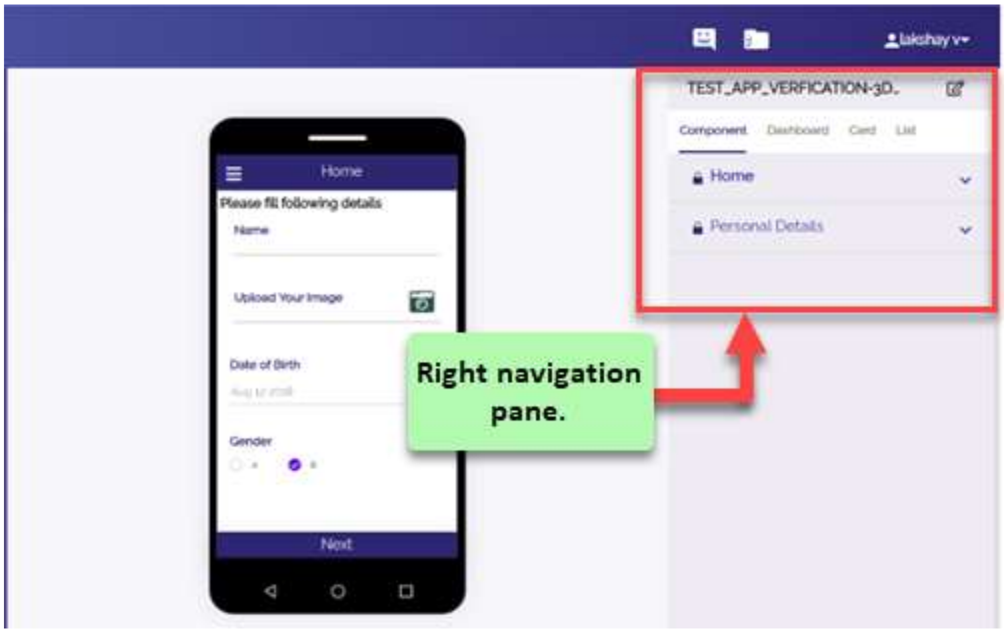
On the other hand, you configure response parameters on the basis of keys and attributes that an API receives in the response of third party API.

You can perform the mapping of the request parameters and the response parameters as follows:

3.3.1.1 Mapping Request Parameters

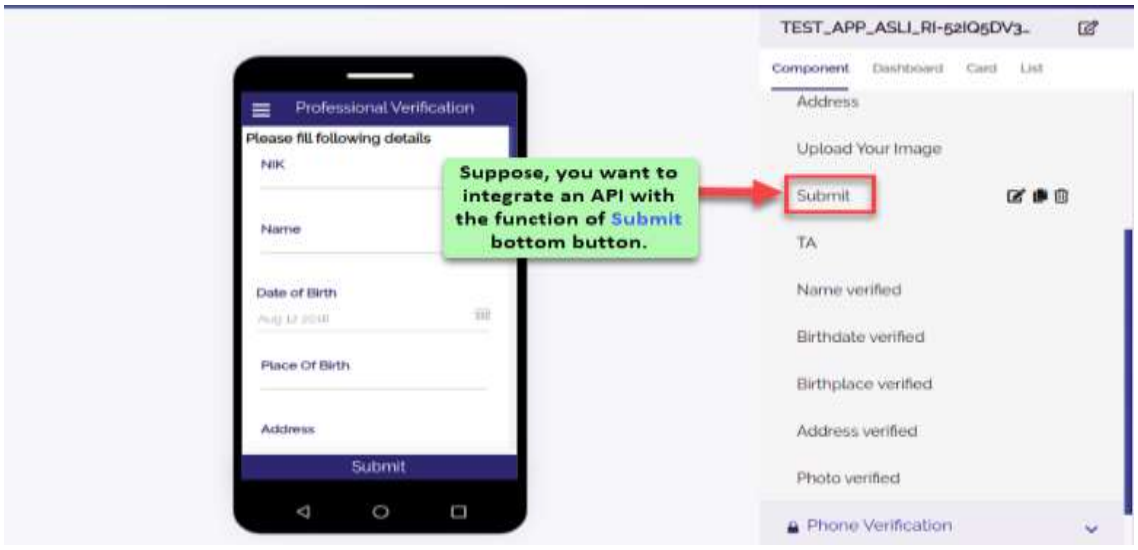
To map request parameters:

1. In the vDesigner application, locate the control/element (For example: - Submit (Bottom button)) where you want to integrate the API.
2. To locate the control, locate the right navigation pane **(Fig 3.3.1.1 (a))**.



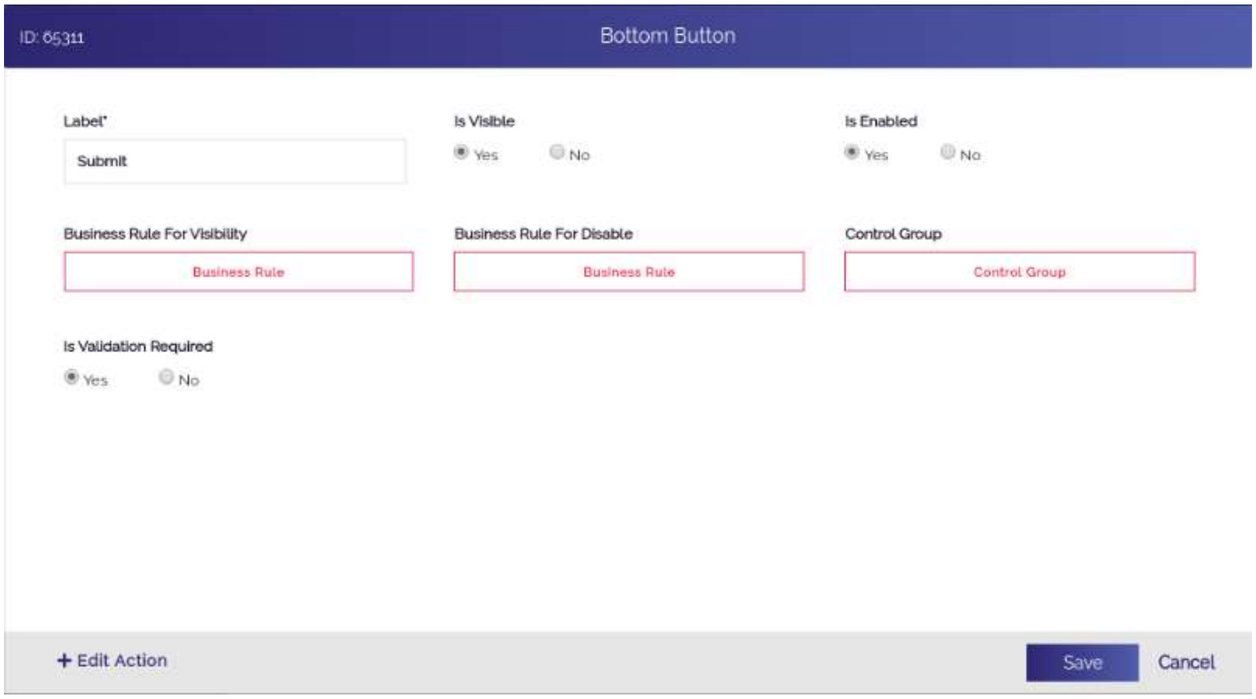
(Fig 3.3.1.1 (a))

3. In the right navigation pane, click the tab (For example: - Component) to locate the form/control **(Fig 3.3.1.1 (b))** where you want to integrate the API.



(Fig 3.3.1.1 (b))

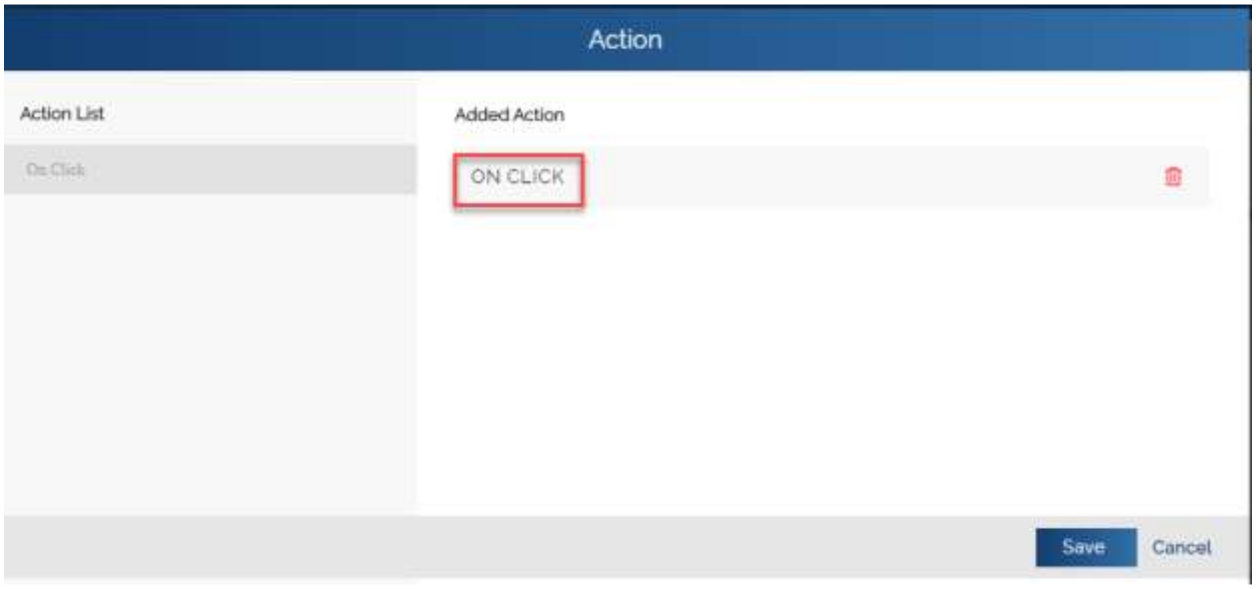
4. After you locate the element/control (For example: - Submit) **(Fig 3.3.1.1 (b))**, double-click the name of element in the right pane, element's dialog box (For example: - Bottom Button dialog box) **(Fig 3.3.1.1 (c))** opens.



Label*: Submit	Is Visible: [✓ Yes] No	Is Enabled: [✓ Yes] No
Business Rule For Visibility	Business Rule For Disable	Control Group
Is Validation Required: [✓ Yes] No		

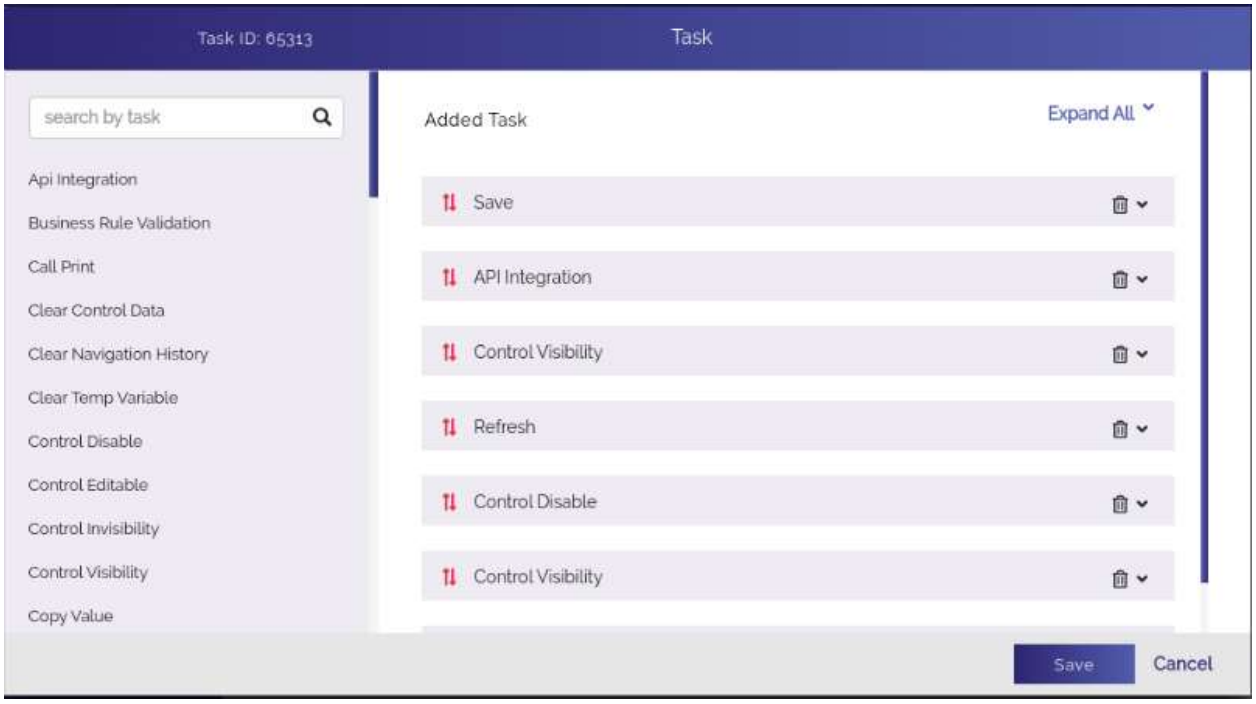
(Fig 3.3.1.1 (c))

5. In the **Bottom Button** dialog box **(Fig 3.3.1.1 (c))**, click **Edit Action**, the **Action** dialog box **(Fig 3.3.1.1 (d))** opens.



(Fig 3.3.1.1 (d))

6. In the **Action** dialog box, locate the **Added Action** area **(Fig 3.3.1.1 (d))** in the right pane.
7. Under **Added Action**, click **ON CLICK**, the **Task** dialog box **(Fig 3.3.1.1 (e))** opens.

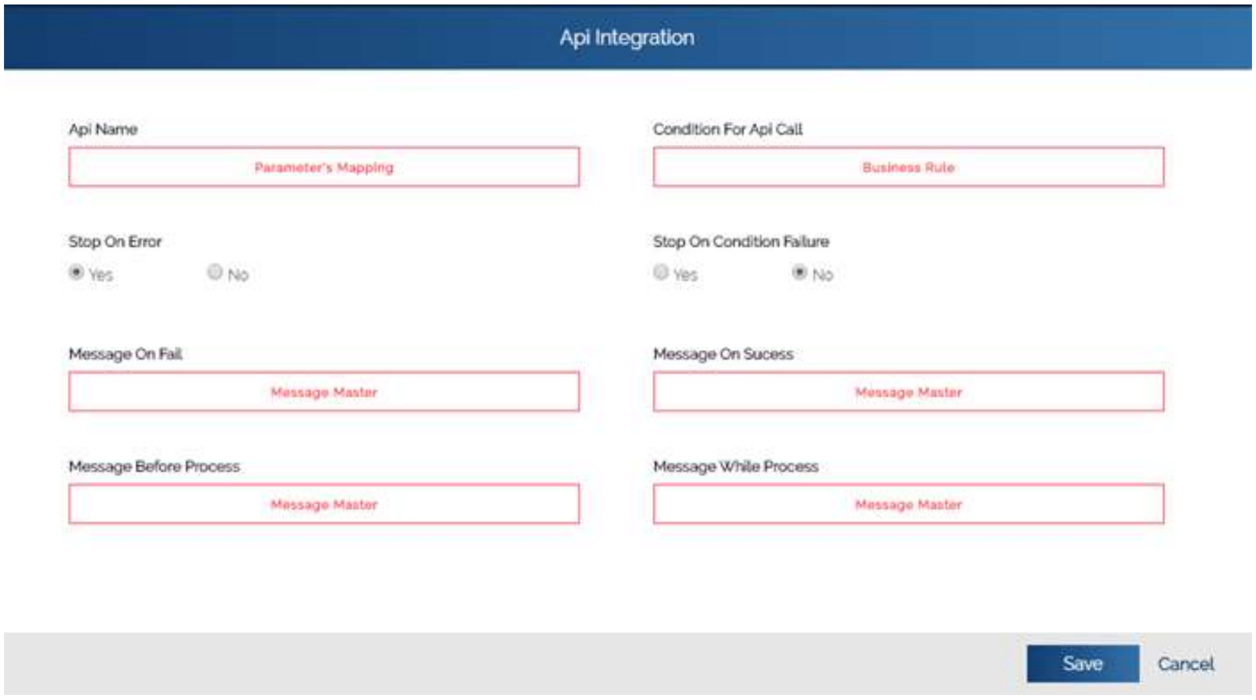


(Fig 3.3.1.1 (e))

Note:-
In the screen capture **(Fig 3.3.1.1 (e))**, **if you view the list of added tasks in the right pane**, you will find that the **Save** task has been configured on the priority basis. The reason of configuring the **Save** task is described as below:

On specific form, If you are taking input from the mobile app user and you want that the form continues to hold the input data after the user navigates to another page, you need to apply the **Save** task on the **Next** button or **Submit** button of that form where you have asked the user to enter the data.

8. In the left pane **(Fig 3.3.1.1 (e))**, locate the **Search by task** box.
9. In the **Search by task** box, enter **API Integration**.
10. After you locate the **Api Integration** task, click it, the **Api Integration** dialog box **(Fig 3.3.1.1 (f))** opens.


Api Integration

Api Name: Parameter's Mapping
Condition For Api Call: Business Rule

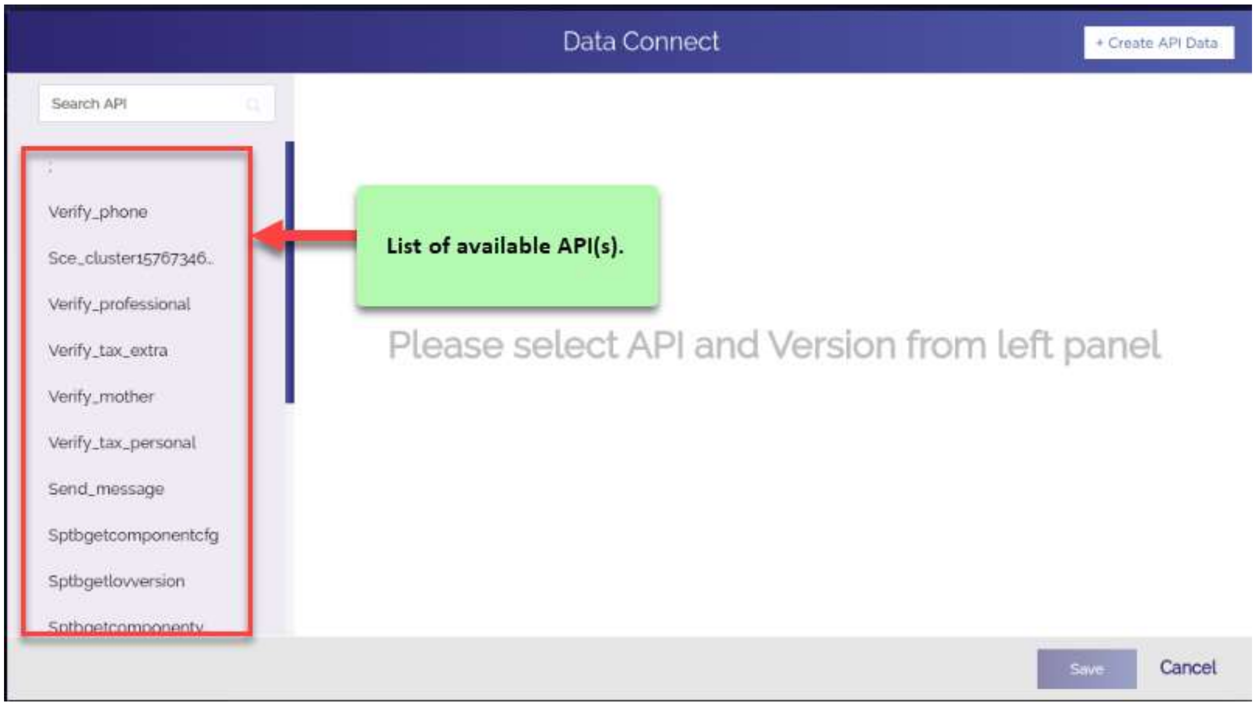
Stop On Error: [✓ Yes] No
Stop On Condition Failure: Yes [✓ No]

Message On Fail: Message Master
Message On Sucess: Message Master

Message Before Process: Message Master
Message While Process: Message Master

(Fig 3.3.1.1 (f))

11. In the **Api Integration** dialog box **(Fig 3.3.1.1 (f))**, click **Parameter's Mapping**, the **Data Connect** dialog box **(Fig 3.3.1.1 (g))** opens.



(Fig 3.3.1.1 (g))

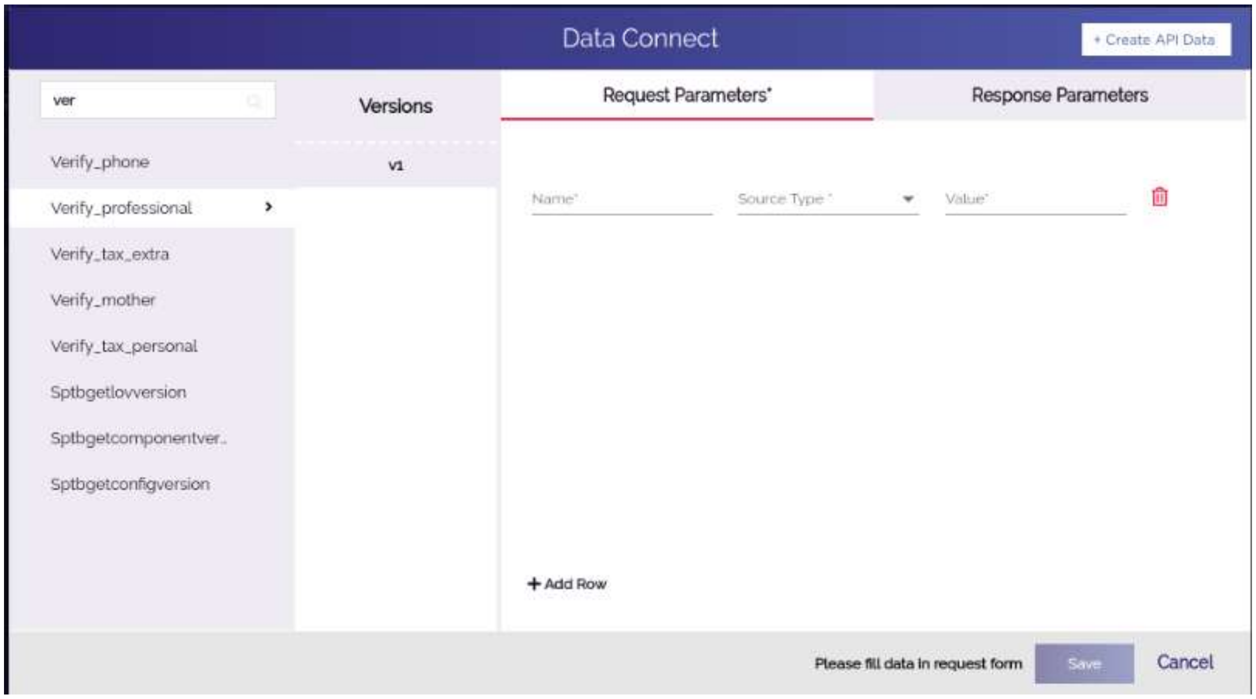
12. In the **Data Connect** dialog box, the left navigation pane displays the list of available API(s).

Note:-
The left navigation pane displays the list of API(s) that are deployed on the vConnect platform.

13. In the **Search API** box **(Fig 3.3.1.1 (g))**, enter the name of API (For example: - Verify_professional) that you want to integrate.

Note:-
On searching specific API, if you find multiple versions of the API, select that version of the API that you want to integrate.

14. After you search the API (For example: - Verify_professional), click to select that API.
15. After you select the Verify_professional API, click the version number (For example: - v1) **(Fig 3.3.1.1 (h))** of the API, the right pane displays the **Request Parameters** and **Response Parameters** tabs.

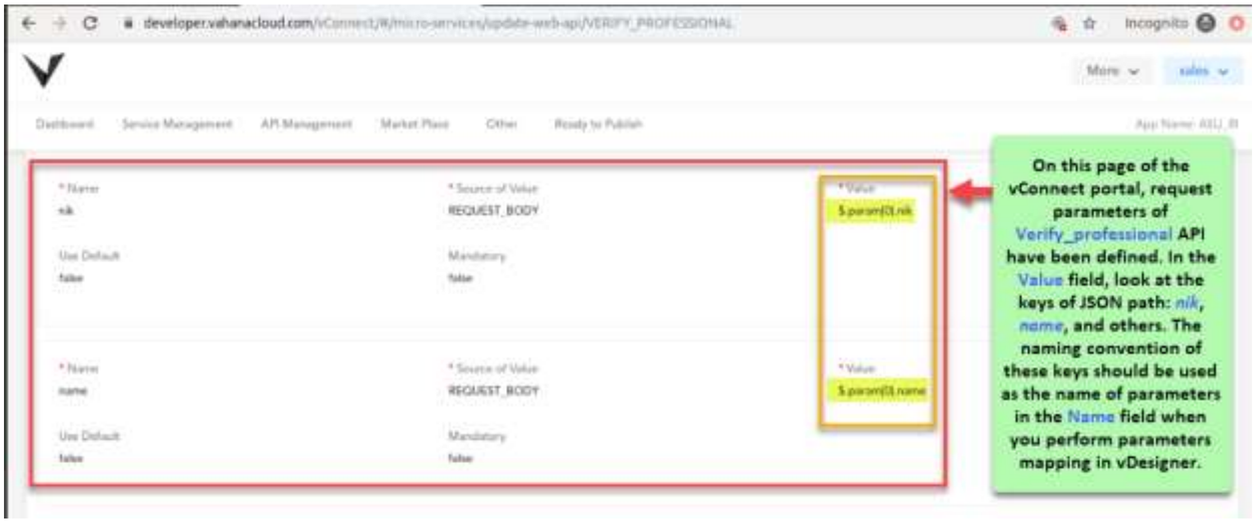


(Fig 3.3.1.1 (h))

16. Click the **Request Parameters** tab (If not selected) **(Fig 3.3.1.1 (h))**.
17. The **Request Parameters** tab **(Fig 3.3.1.1 (h))** displays the following fields:
    - **Name**
    - **Source Type**
    - **Value**
18. In these fields, enter/select values as follows:

Fields	Description
Name	In this field, enter the name of parameter (For example: - nik or name). **Note**:- In the **Name** field, the name of parameter must match with the name of parameter that you define in the **Value** field in the vConnect portal. On the vConnect portal, you enter the name of parameter in the **Value** field **(Fig 3.3.1.1 (i))** when you define the template body parameter while configuring an API.
Source Type	Click this list to select any of the following values: ➢ **Special Value** ➢ **Fixed** ➢ **JSON** ➢ **Table** ➢ **Control**

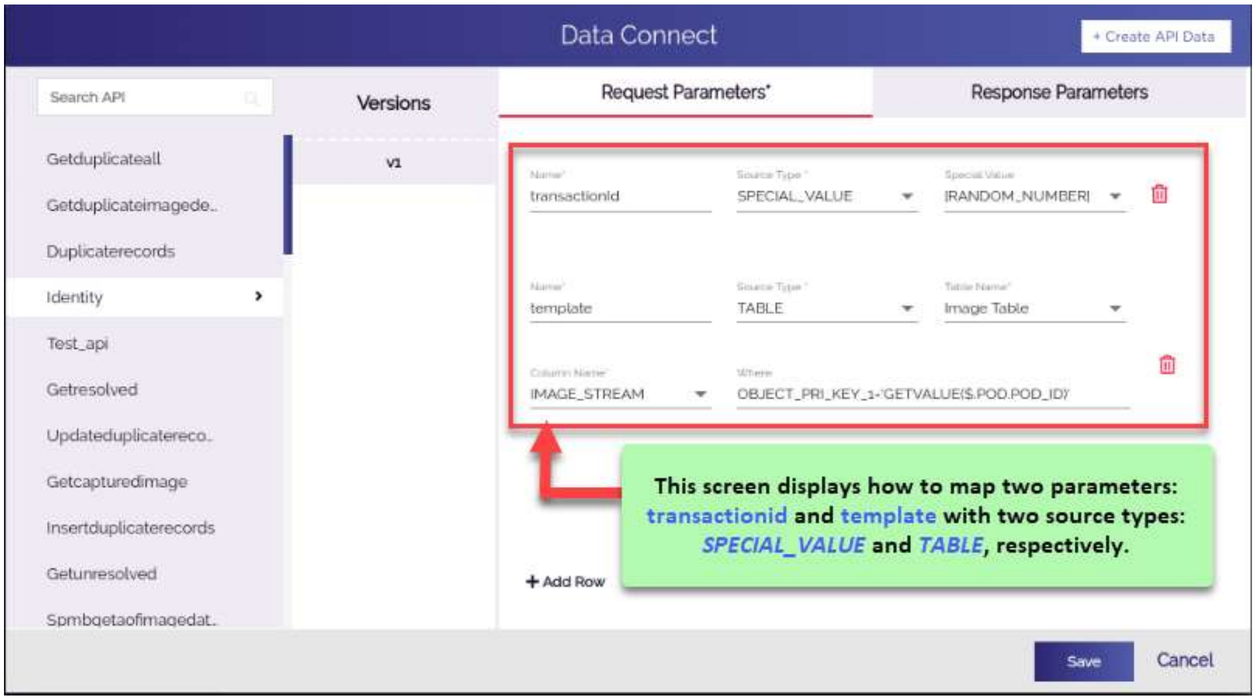
	Selecting any of these values as source type determines the type of data source or format from where the parameter will pick the data or in which insert the data.
Value	In this box, enter the value as follows. In the **Source Type** list, if: ➢ You select: **Special Value** ○ In the **Value** box, enter hardcoded value: **[Random_number]**. In case of special value, the API will fetch the random number. ➢ You select: **Fixed** ○ In the **Value** box, enter any constant value that the respective parameter will store. ➢ You select: **JSON** ○ In the **Value** box, enter the json path (For example: - $.POD.nik) **(Fig 3.3.1.1 (k))**. The API will pick the data from the JSON path. ➢ You select: **Table** ○ Click the **Table Name** list and then select the table (For example: - Image Table) from where the API will fetch the data. ○ After you select the table, the **Column Name** list and the **Where** box appear. ○ Click the **Column Name** list **(Fig 3.3.1.1 (j))** and then select the column (For example:- Image_stream) from where API will fetch the data. ○ In the **Where** box **(Fig 3.3.1.1 (j))**, enter/write the where clause of the SQL query if you want that the API fetch the data from the selected column with special conditions. ➢ You select: **Control** ○ In the **Value/Control** box, enter the control ID (For example: - 79229) of the control/element (For example: - Text box/Text field) from where API will pick the data and store in the respective parameter.



* Name: nik	* Source of Value: REQUEST_BODY	* Value: $.param[0].nik
Use Default: false	Mandatory: false	
* Name: name	* Source of Value: REQUEST_BODY	* Value: $.param[0].name
Use Default: false	Mandatory: false	

On this page of the vConnect portal, request parameters of Verify_professional API have been defined. In the Value field, look at the keys of JSON path: nik, name, and others. The naming convention of these keys should be used as the name of parameters in the Name field when you perform parameters mapping in vDesigner.

(Fig 3.3.1.1 (i)) (The page of vConnect portal where request parameters are defined)



Name*: transactionId | Source Type*: SPECIAL_VALUE | Special Value: |RANDOM_NUMBER|

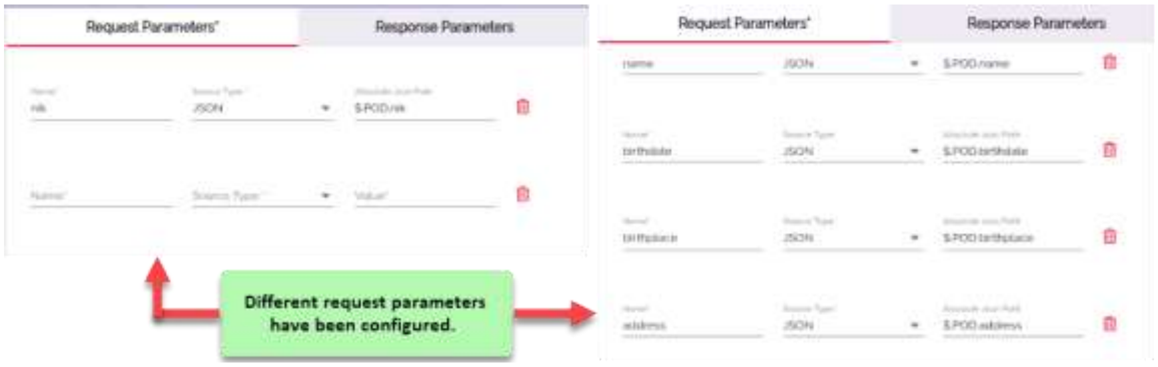
Name*: template | Source Type*: TABLE | Table Name*: Image Table

Column Name*: IMAGE_STREAM | Where: OBJECT_PRI_KEY_1='GETVALUE($.POD.POD_ID)'

This screen displays how to map two parameters: transactionid and template with two source types: *SPECIAL_VALUE* and *TABLE*, respectively.

(Fig 3.3.1.1 (j))

19. After you map the first parameter by following steps as described in the last table, map other parameters.
20. To map other parameters, click **Add Row (Fig 3.3.1.1 (h))**, another row of the following fields appears.
    - **Name**
    - **Source Type**
    - **Value**
21. In these fields, enter values to map next parameter as described in the last table.



Name*: nik | Source Type*: JSON | Absolute Json Path: $.POD.nik

Name* | Source Type* | Value*

name | JSON | $.POD.name

Name*: birthdate | Source Type*: JSON | Absolute Json Path: $.POD.birthdate

Name*: birthplace | Source Type*: JSON | Absolute Json Path: $.POD.birthplace

Name*: address | Source Type*: JSON | Absolute Json Path: $.POD.address

Different request parameters have been configured.

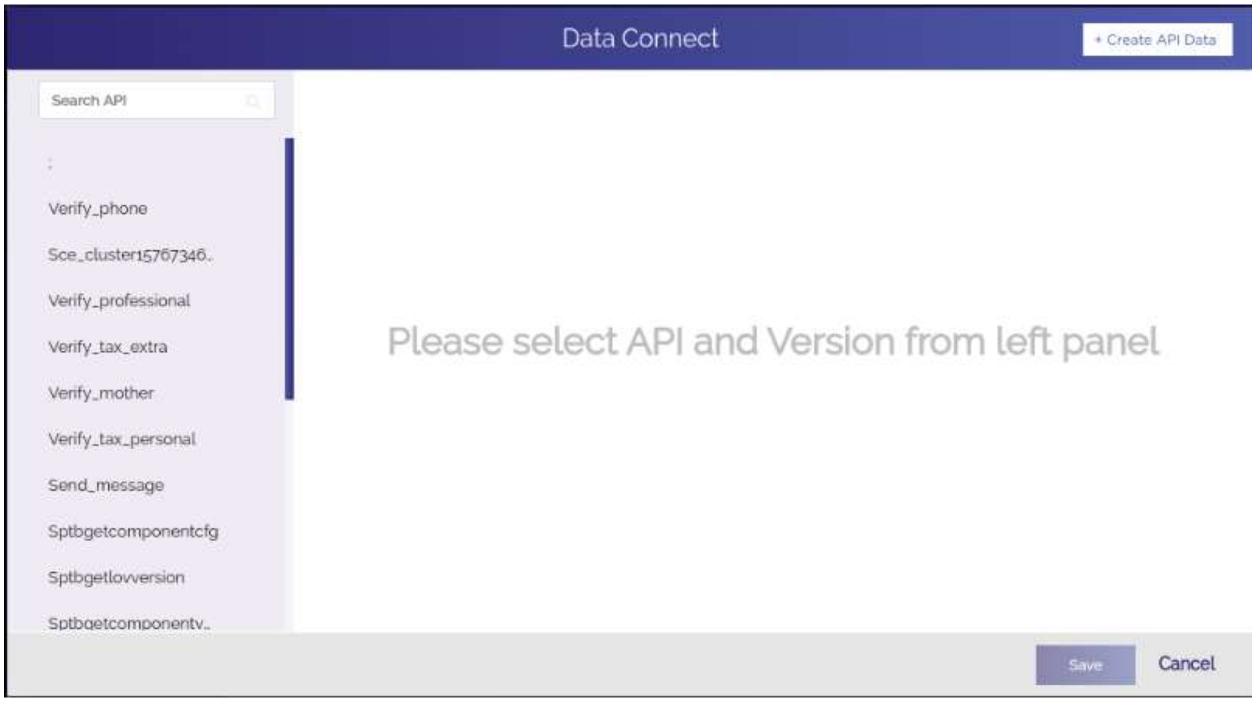
(Fig 3.3.1.1 (k))

22. After you map all the request parameters, click **Save** (**Fig 3.3.1.1 (j))**, the **Data Connect** dialog box closes.
23. In the **API Integration** dialog box, click **Save**, the **API Integration** dialog box closes.
24. In the **Task** dialog box, click **Save,** the **Task** dialog box closes.
25. In the **Action** dialog box, click **Save**, the **Action** dialog box closes.
26. In the **Bottom Button** dialog box, click **Save**, the **Bottom Button** dialog box closes and all request parameters are successfully mapped.

3.3.1.2 Mapping Response Parameters

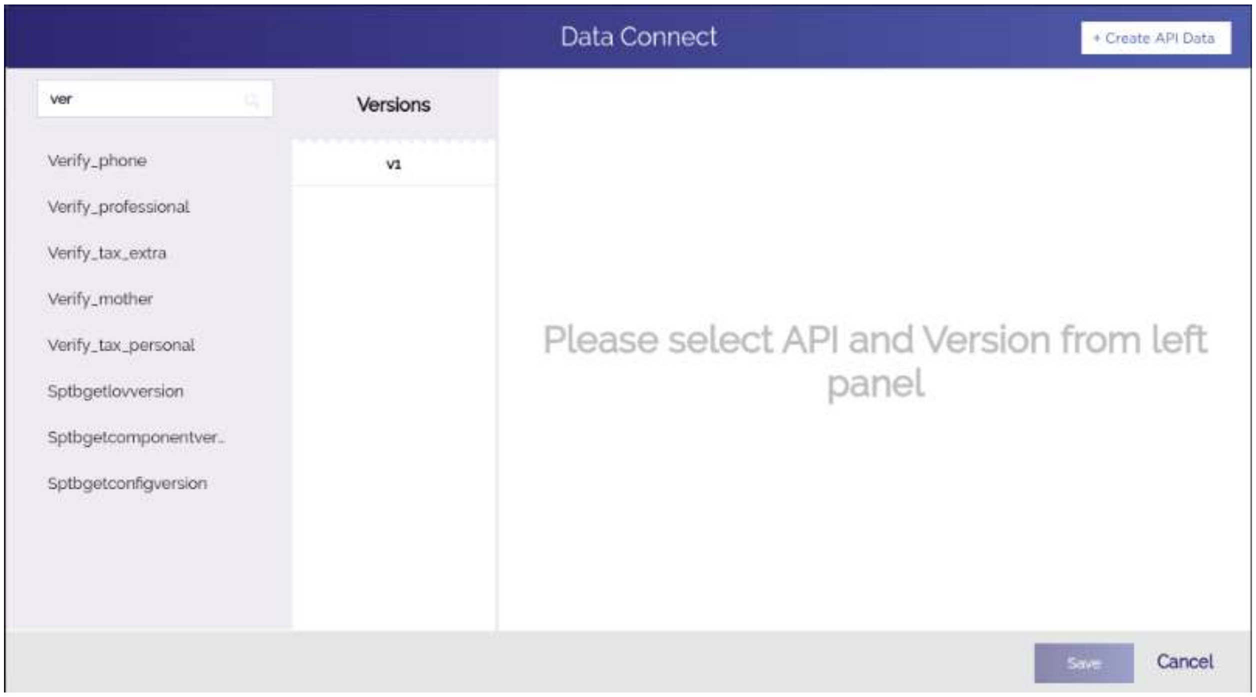
To map response parameters:

1. Perform steps from the step1 to step11 to access the **Data Connect** dialog box **(Fig 3.3.1.2 (a))** where you map the response parameters.



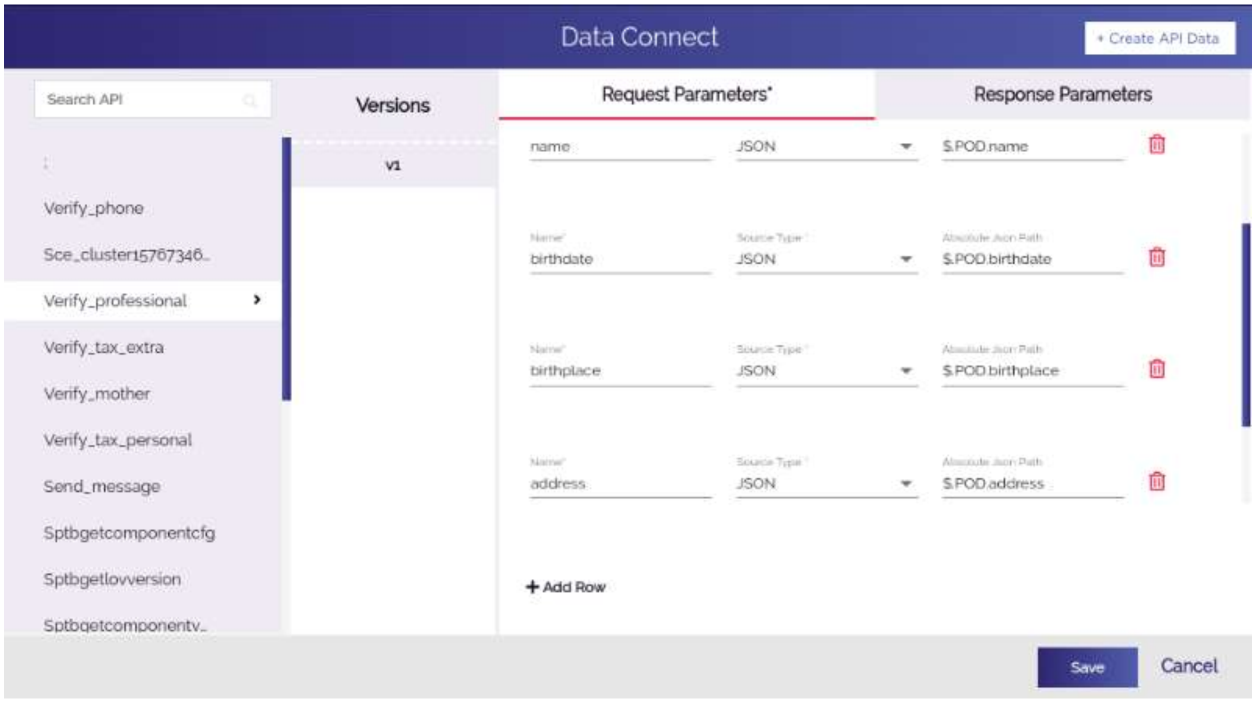
(Fig 3.3.1.2 (a))

2. In the **Data Connect** dialog box, locate the left navigation pane **(Fig 3.3.2.1 (a))**.
3. In the **Search API** box **(Fig 3.3.1.2 (a))**, enter the name of API, the left pane displays the name of API (For example: - Verify_professional).
4. After you search the API, click it, the **Data Connect** dialog box displays the name of API **(Fig 3.3.1.2 (b))** along with its available version.



(Fig 3.3.1.2 (b))

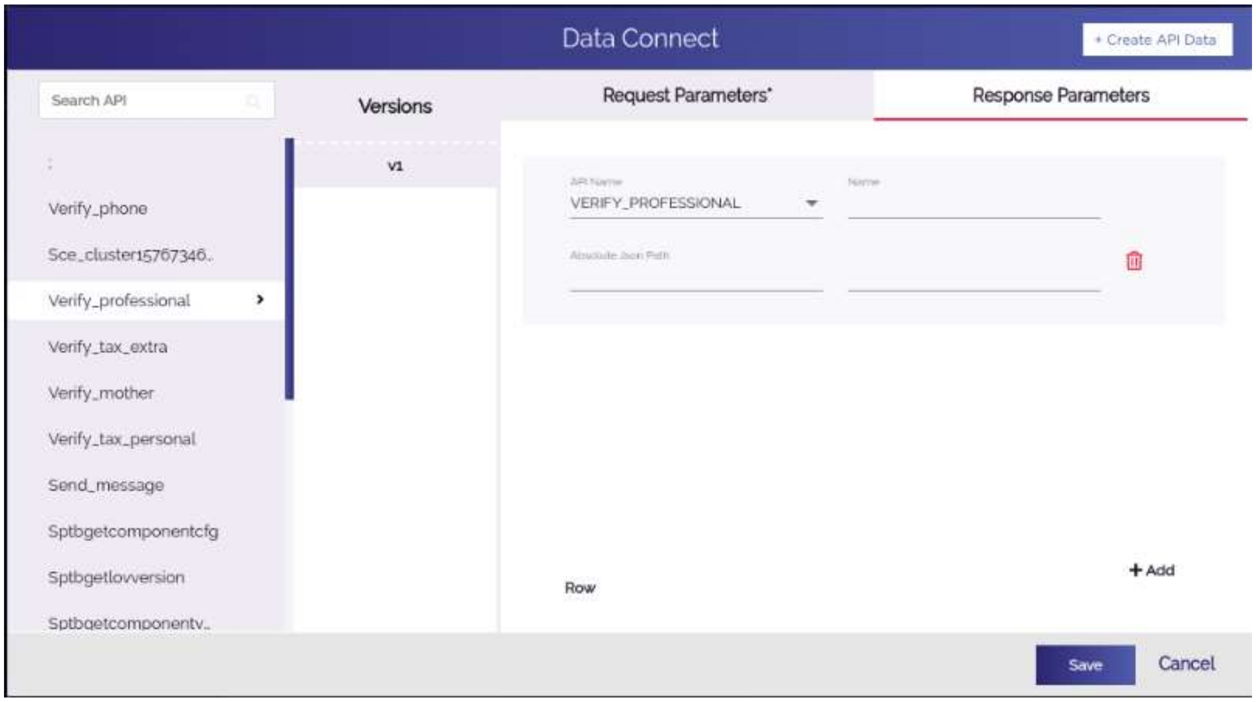
5. Under **Versions**, click the version number (For example: - v1) of API, the **Data Connect** dialog box displays the **Request Parameters** and **Response Parameters** tabs **(Fig 3.3.1.2 (c))**.



Name*	Source Type*	Absolute Json Path
name	JSON	$.POD.name
birthdate	JSON	$.POD.birthdate
birthplace	JSON	$.POD.birthplace
address	JSON	$.POD.address

(Fig 3.3.1.2 (c))

6. Click the **Response Parameters** tab, the **Data Connect** dialog box displays the group of the following fields **(Fig 3.3.1.2 (d))**:
   - **API Name**
   - **Name**
   - **Absolute JSON Path**
   - **Control Id**



(Fig 3.3.1.2 (d))

7. In these parameters, enter values as follows:

Field	**Description**
API Name	This list displays the name of API (For example: - Verify_professional) that you have selected to map the response parameters.
Name	In this box, enter the name of key/attribute (For example: - ID, name, data, and others) that the API receives in the response data that is sent by the third party API.
Absolute JSON Path	In this box, enter the json path (For example: - $.POD.Professionaldata) where API will store the data after it receives the response from third party API.
Control Id	In this box, enter the ID (For example:- 79299) of the control/element (For example:- Text field, label, etc) in which API will display the response data on the mobile app.

8. After you enter values in the respective fields, map other response parameters.
9. To map other parameters, click **Row**, another row of **API Name**, **Name**, and other fields appears.
10. In these fields, enter values to map next response parameter as described in the last table.
11. After you map all response parameters, click **Save (Fig 3.3.1.2 (d))**, the **Data Connect** dialog box closes.
12. In the **API Integration** dialog box, click **Save**, the **API Integration** dialog box closes.
13. In the **Task** dialog box, click **Save**, the **Task** dialog box closes.
14. In the **Action** dialog box, click **Save**, the **Action** dialog box closes.
15. In the **Bottom Button** dialog box, click **Save**, the **Bottom Button** dialog box closes and all response parameters are successfully mapped.

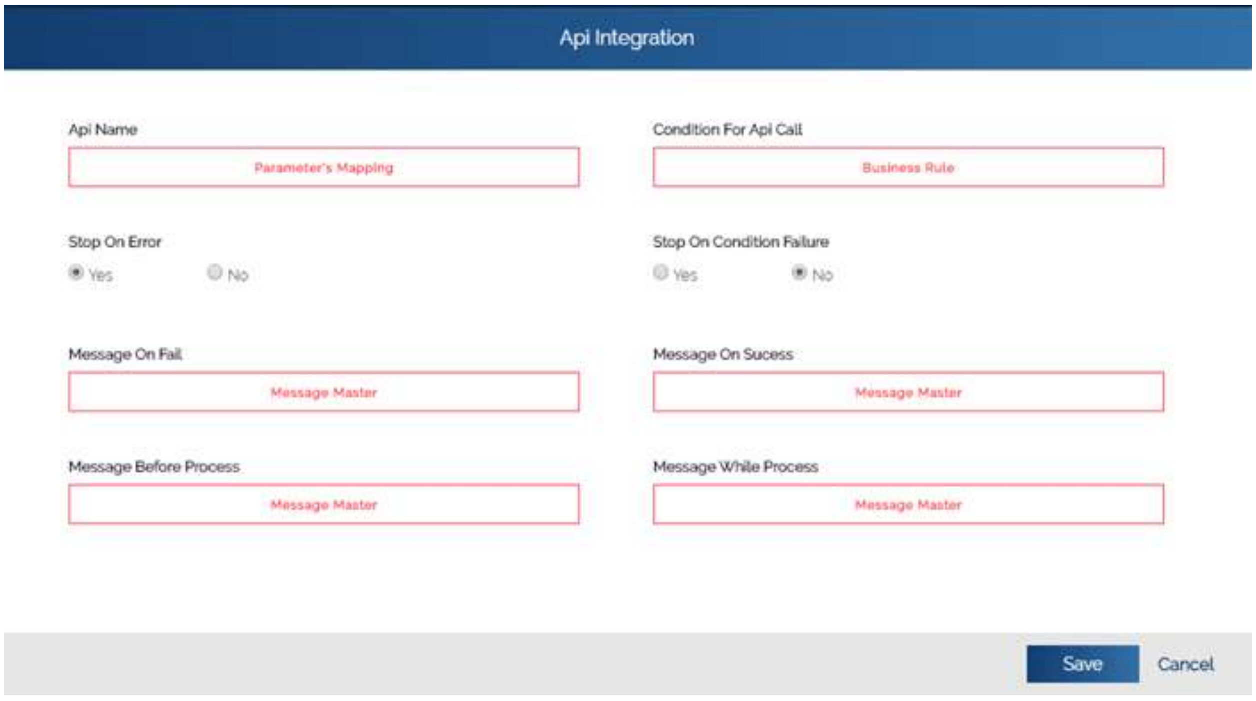
3.3.1.3 Configuring Business Rule for API Call

The vDesigner application allows you configure the business rule for and on different entities that you add to design a mobile app. These entities can include a control/element, form, card, list, and others. You can apply a business rule for several purposes such as disabling or enabling a control, satisfying a condition, displaying a specific set of data, and others.

This section describes how to apply a business rule to stop API call.

To apply a business rule:

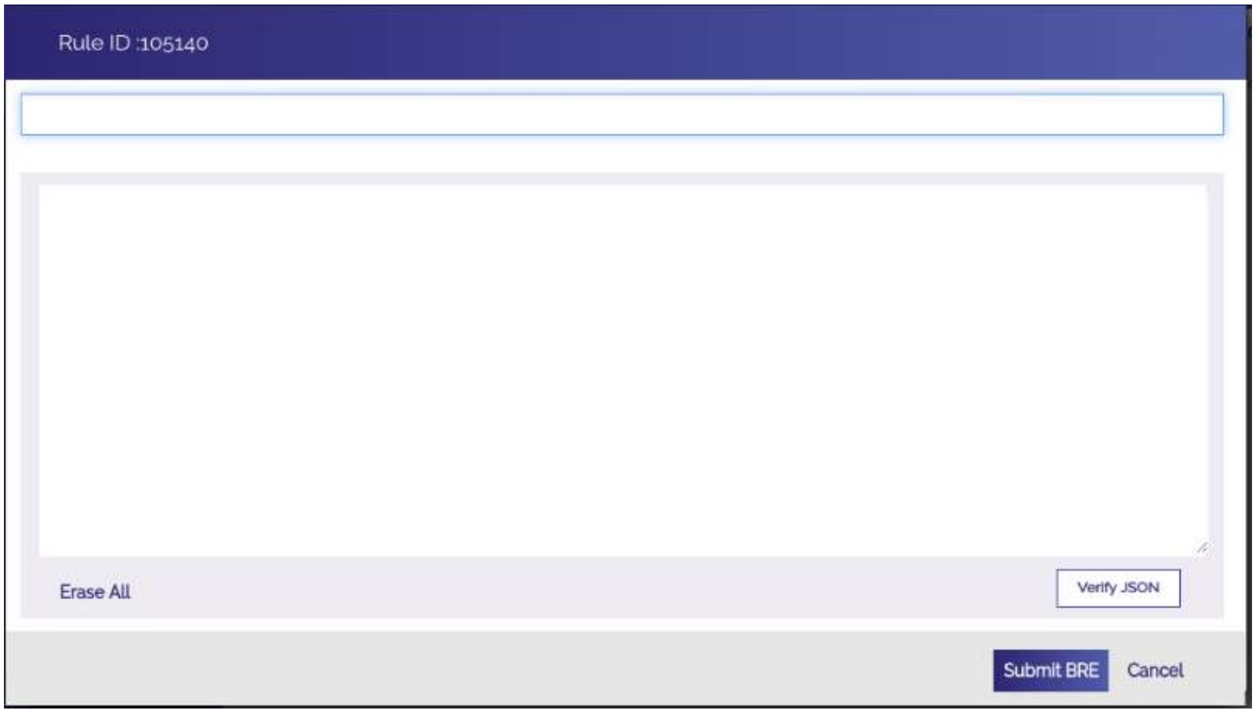
1. On the **API Integration** dialog box, locate **Condition For Api Call (Fig 3.3.1.3 (a))**.


Api Integration

Api Name: Parameter's Mapping
Condition For Api Call: Business Rule
Stop On Error: [✓ Yes] No
Stop On Condition Failure: Yes [✓ No]
Message On Fail: Message Master
Message On Sucess: Message Master
Message Before Process: Message Master
Message While Process: Message Master

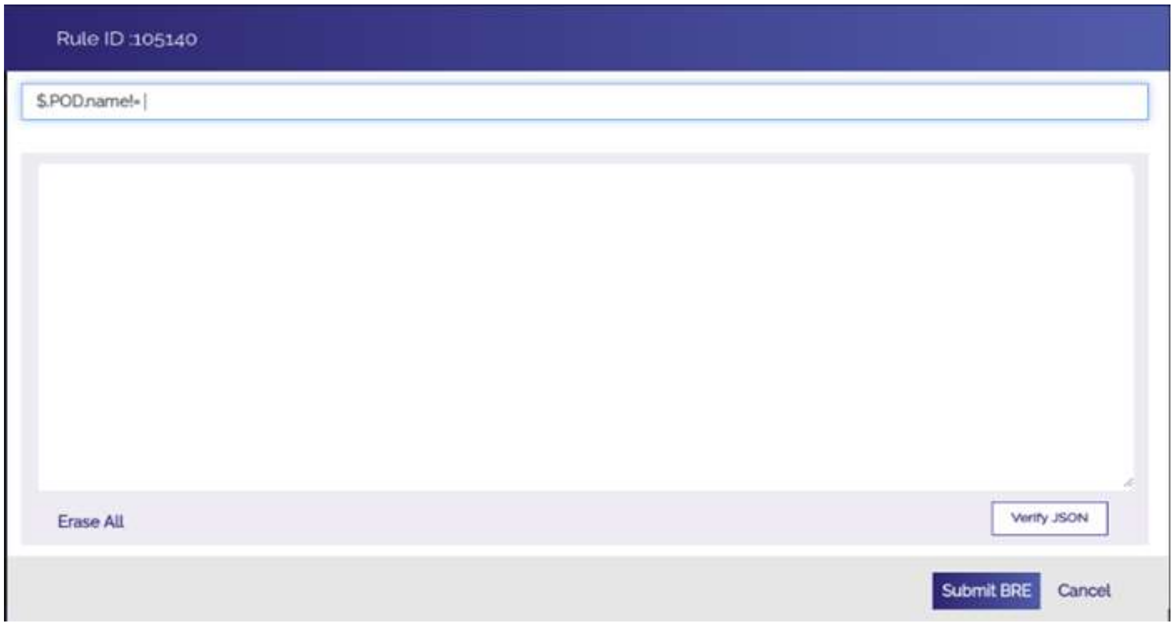
(Fig 3.3.1.3 (a))

2. Under **Condition For Api Call**, click **Business Rule (Fig 3.3.1.3 (a))**, the **Rule ID** dialog box **(Fig 3.3.1.3 (b))** opens.



(Fig 3.3.1.3 (b))

3. In the upper box, enter the business rule (For example: - $.POD.name!=)



(Fig 3.3.1.3 (c))

4. After you enter the business rule, click **Submit BRE**, the business rule is successfully configured.

The written business rule: **$.POD.name!=** specifies that if the user does not enter any value in the **Name** field and then performs the function to submit the data, the business rule will first check the value in the **Name** field. If the business rule:

- Finds the value in the **Name** field, the corresponding API will be called.
- Does not find any value in the **Name** field, the corresponding API will not be called.

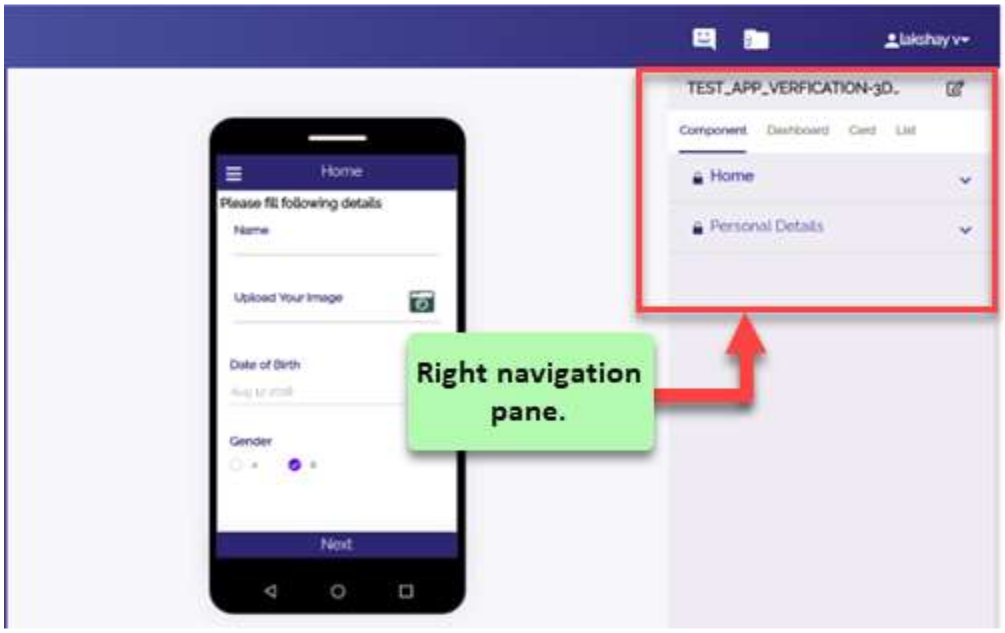
3.3.2 Integrating Sync API(s)

In the vDesigner application, the sync API (s) are integrated to insert the mobile app data into database table. As described earlier, the sync API inserts the data into two tables: ***tb_cop_buss_obj_txn*** and ***tb_image_data_obj***. It inserts numeric and text based data into the ***tb_cop_buss_obj_txn*** table.

To integrate a sync API in the vDesigner application, the API should be deployed on the vConnect platform. The following set of instructions describes how to **SPMBINSERTAOFOBJDATA** sync API, which is already deployed on the vConnect platform.

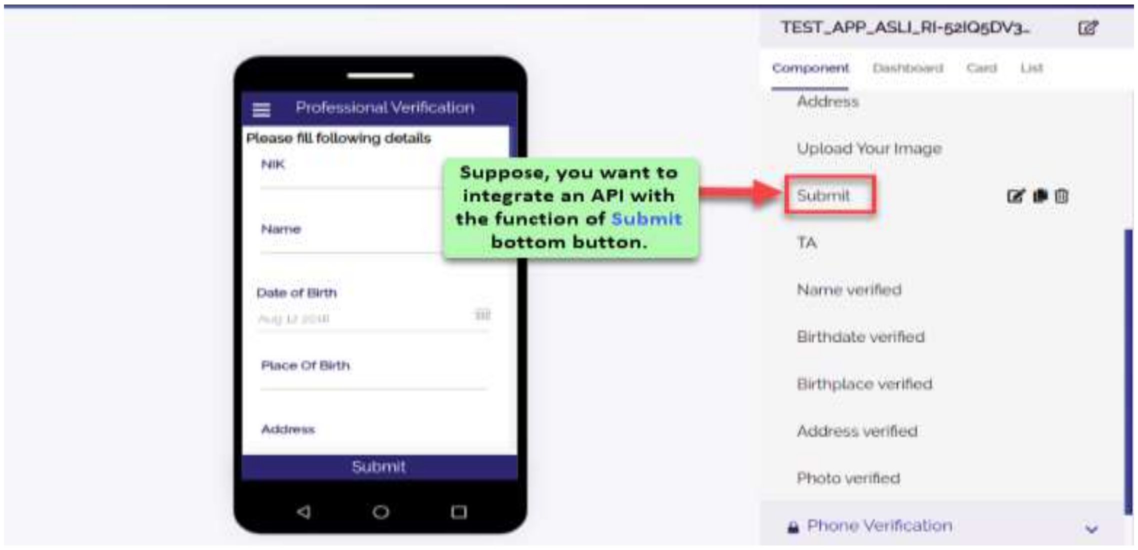
To integrate a sync API:

1. In the vDesigner application, locate the control/element (For example: - Submit (Bottom button)) where you want to integrate the API.
2. To locate the control, locate the right navigation pane **(Fig 3.3.2 (a))**.



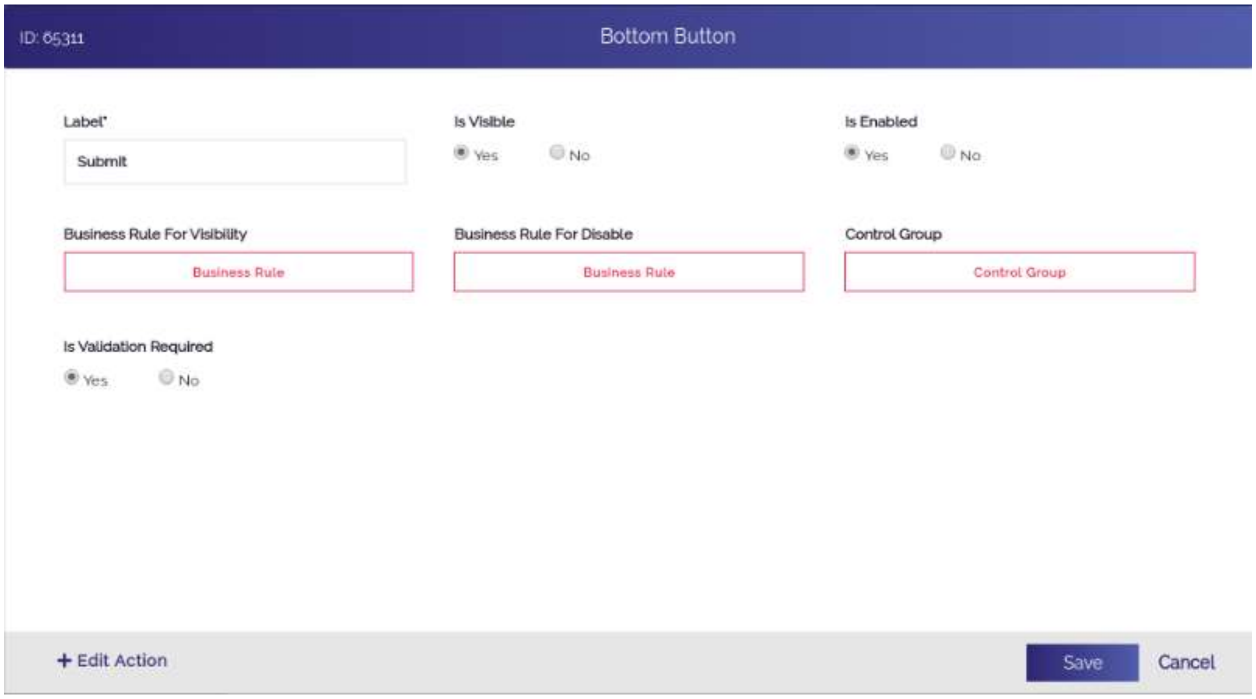
(Fig 3.3.2 (a))

3. In the right navigation pane, click the tab (For example: - Component) to locate the form/control **(Fig 3.3.2 (b))** where you want to integrate the API.



(Fig 3.3.2 (b))

4. After you locate the element/control (For example: - Submit) **(Fig 3.3.1.1 (b))**, double-click the name of element in the right pane, element's dialog box (For example: - Bottom Button dialog box) **(Fig 3.3.2 (c))** opens.


ID: 65311 — Bottom Button

Label*: Submit

Is Visible: [✓ Yes] No

Is Enabled: [✓ Yes] No

Business Rule For Visibility: Business Rule

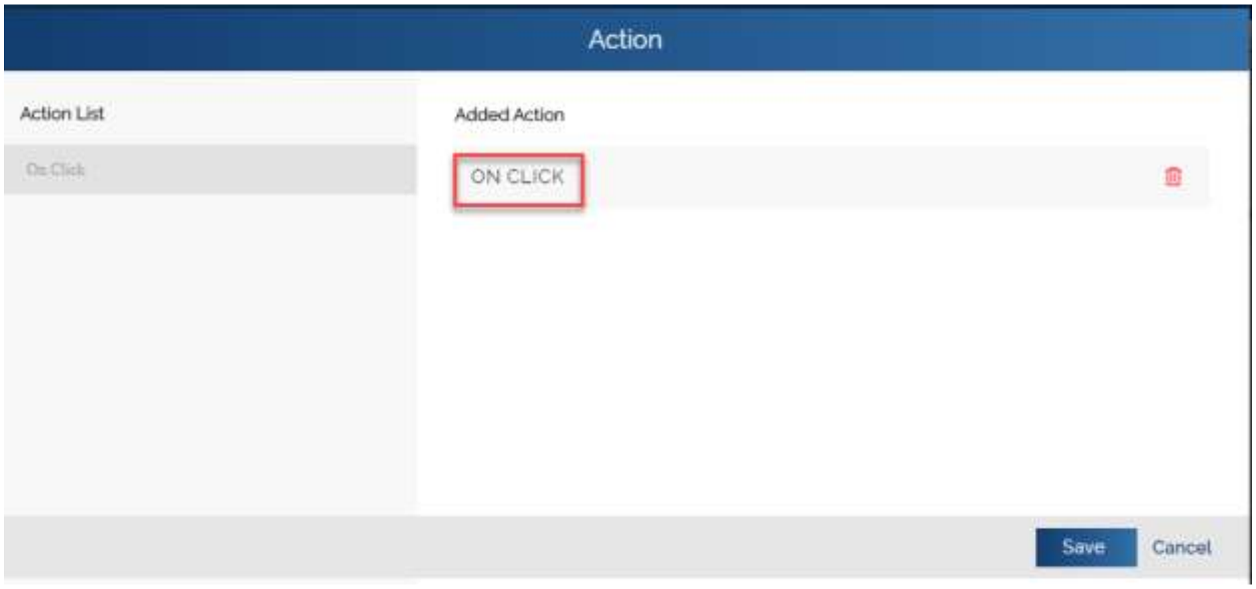
Business Rule For Disable: Business Rule

Control Group: Control Group

Is Validation Required: [✓ Yes] No

(Fig 3.3.2 (c))

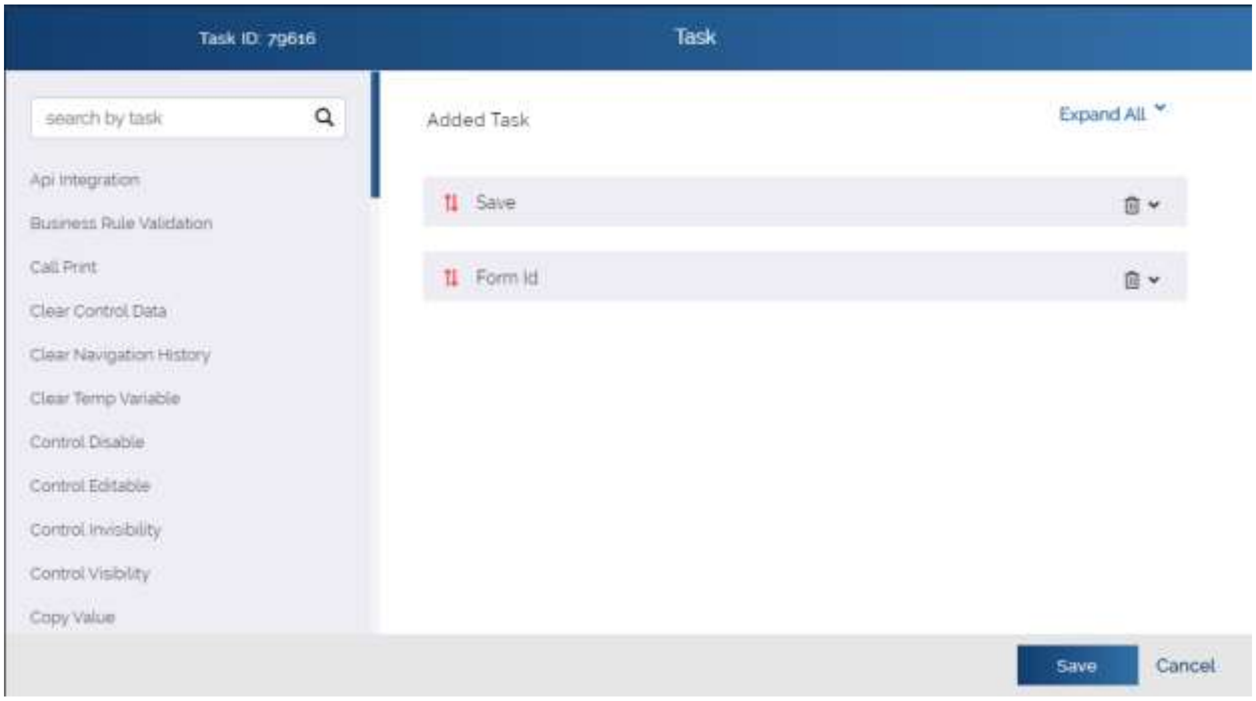
5. In the **Bottom Button** dialog box **(Fig 3.3.2 (c))**, click **Edit Action**, the **Action** dialog box **(Fig 3.3.2 (d))** opens.



(Fig 3.3.2 (d))

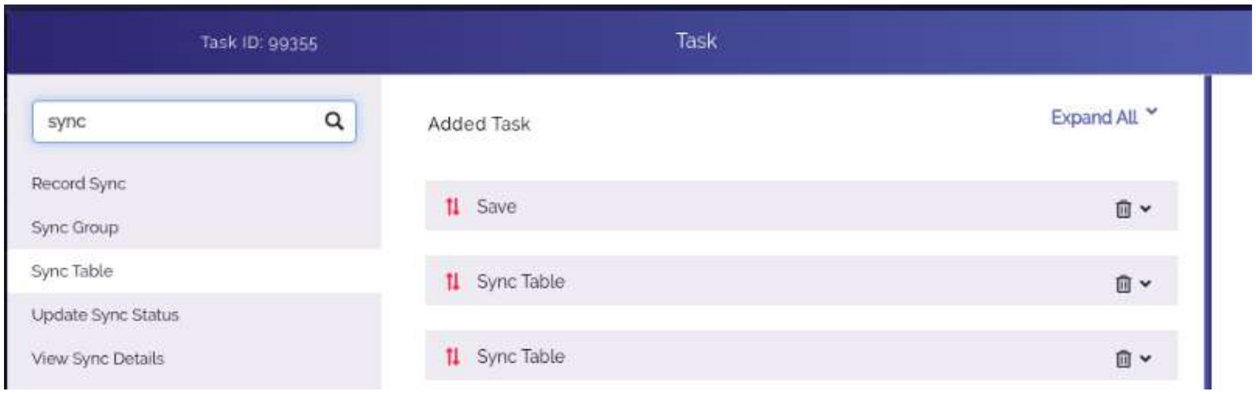
6. In the right pane **(Fig 3.3.2 (d))**, locate the **Added Action** area.
7. Under **Added Action**, click **ON CLICK**, the **Task** dialog box opens.

8. In the **Action** dialog box, locate the **Added Action** area **(Fig 3.3.2 (d))** in the right pane.
9. Under **Added Action**, click **ON CLICK**, the **Task** dialog box **(Fig 3.3.2 (e))** opens.



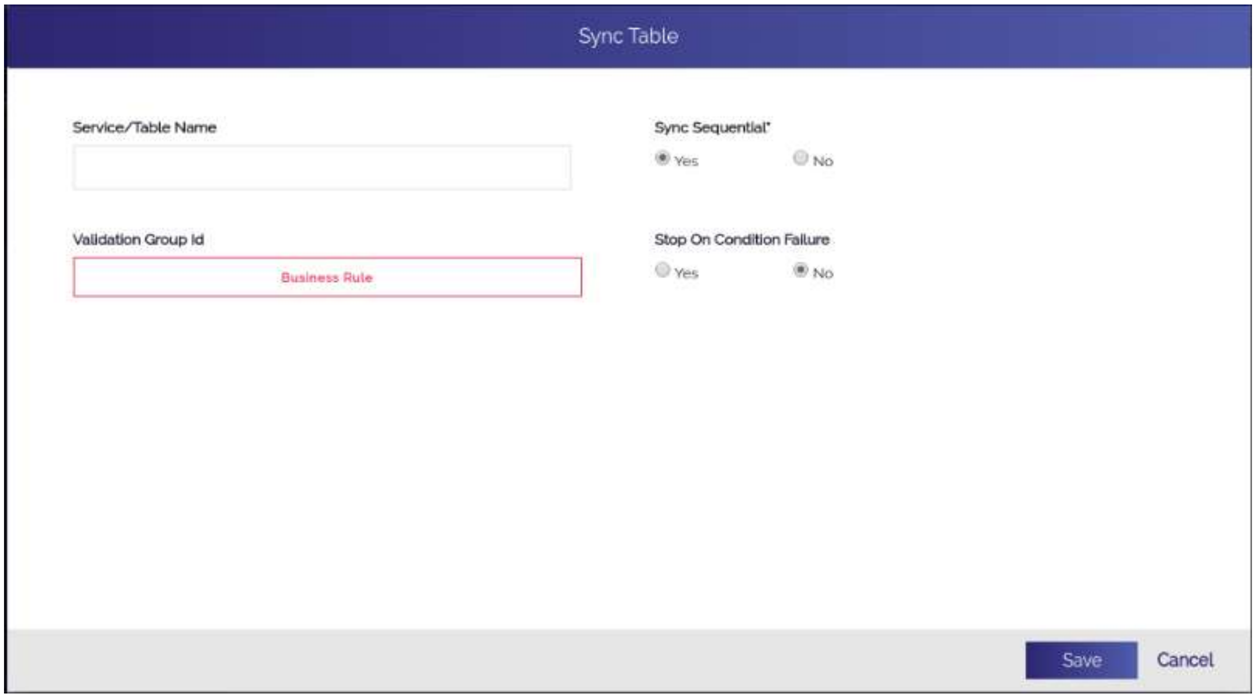
(Fig 3.3.2 (e)

10. In the **Task** dialog box, locate the left navigation pane.
11. In the **Search by task** box **(Fig 3.3.2 (e))**, enter or type **sync**, the left navigation pane displays the results **(Fig 3.3.2 (f))**.



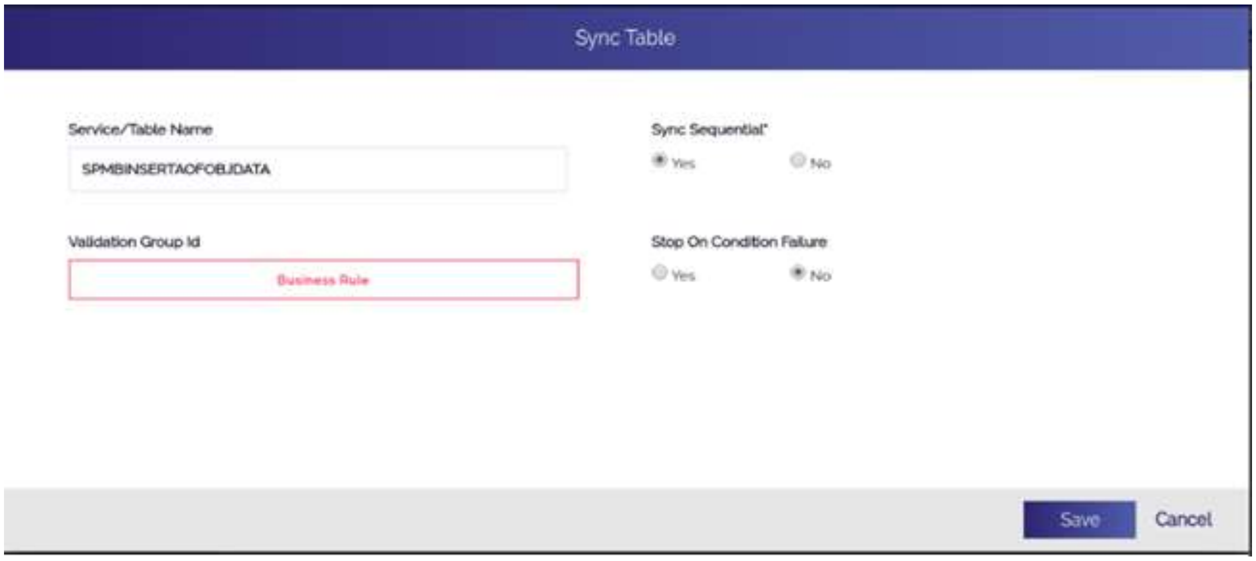
(Fig 3.3.2 (f))

12. In the list of results, click **Sync Table (Fig 3.3.2 (f))**, the **Sync Table** dialog box **(Fig 3.3.2 (g))** opens.



(Fig 3.3.2 (g))

13. In the **Service/Table Name** box **(Fig 3.3.2 (g))**, enter the name of sync API (For example: - SPMBINSERTAOFOBJDATA) that you want to integrate.
14. After you enter the name of sync API, click **Save (Fig 3.3.2 (h))**, the sync API is successfully integrated with the vDesigner application.



Service/Table Name: SPMBINSERTAOFOBJDATA
Validation Group Id
Sync Sequential*: [✓ Yes] No
Stop On Condition Failure: Yes [✓ No]

(Fig 3.3.2 (h))

Similarly, you can integrate other sync API(s).

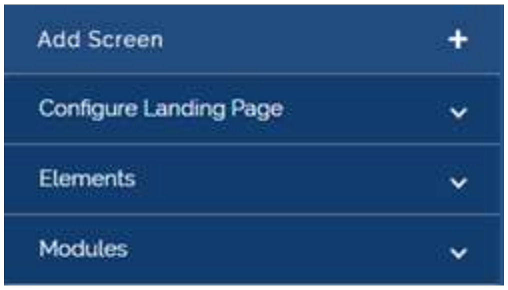
3.4 Configuring a Dashboard

In a mobile app, dashboard is created and configure as landing page. When a user accesses the mobile app, app's dashboard is displayed. Mobile app's dashboard can have several components such as online content, URL, image, list, chart, and others. You can display these components as static items or in carousel format that a mobile app user can slide.

You can design and develop a mobile app dashboard to make it interactive and useful. The vDesigner application provides you a dashboard by default (dashboard with title: Welcome) that you can customize and use it as a dashboard in currently under-development mobile app, or you can create a new dashboard. The following set of instructions/steps describes how to configure a dashboard with image. Apart from image, you can display online data, web URL, charts, and others on the dashboard.

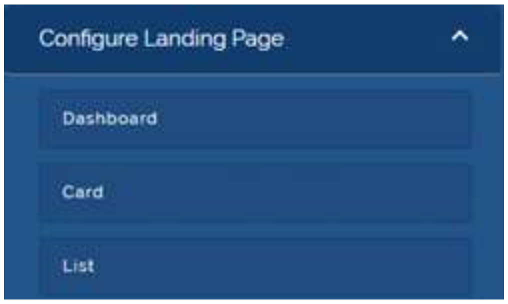
To create/configure a dashboard:

1. On vDesigner application's dashboard, locate the left navigation pane.



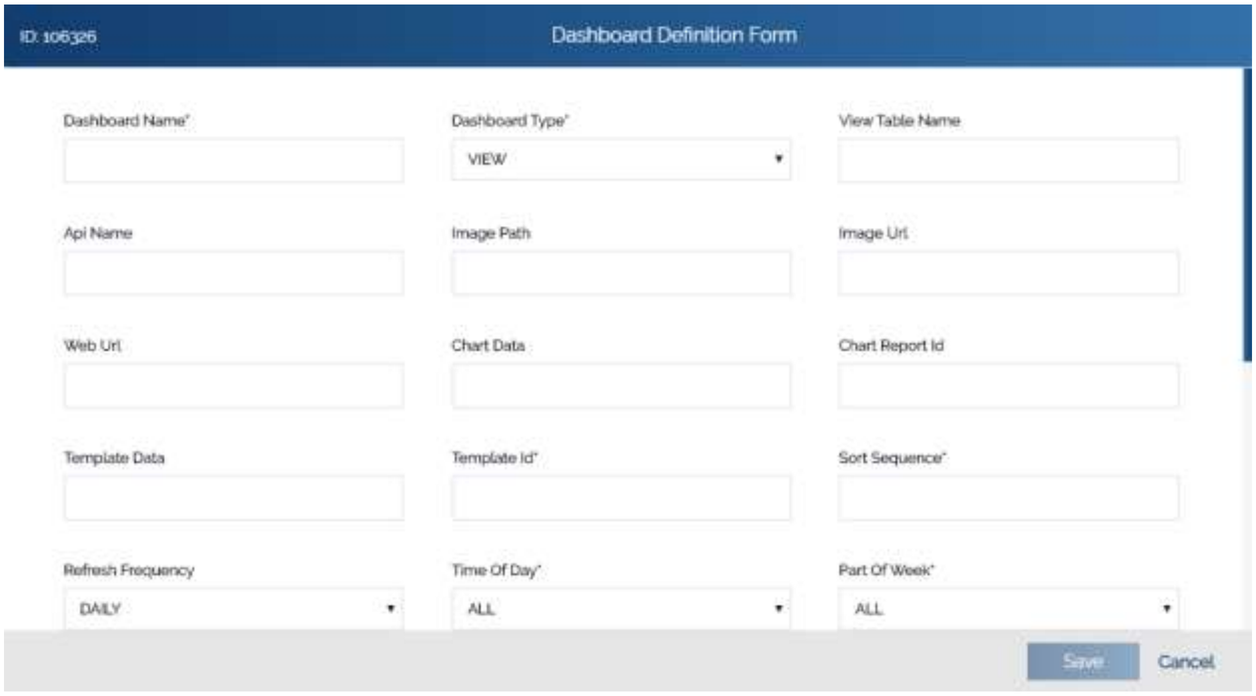
(Fig 3.4 (a))

2. In the left navigation pane, click **Configure Landing Page**, the navigation pane expands.



(Fig 3.4 (b))

3. Under **Configuring Landing Page (Fig 3.4 (b))**, click **Dashboard**, the **Dashboard Definition Form** dialog box opens.



(Fig 3.4 (c))

4. In the **Dashboard Definition Form** dialog box, enter values in the respective boxes/lists as follows:

Box/List	**Description**
Dashboard Name	In this box, enter the name (For example: - TECH5- Mobile App POC) of the dashboard.
Dashboard Type	Click this list to select any of the following options to display on the dashboard ➢ **ONLINE** ➢ **URL** ➢ **CHART** ➢ **IMAGE** ➢ **VIEW** Suppose, you select **Image** in the **Dashboard Type** list to display an image on the dashboard.
View Table Name	In this box, enter the name of the database view if you have selected **VIEW** in the **Dashboard Type** list to display the data from the database view.
Image URL	In this box, you enter the image of URL if you have selected **IMAGE** in the **Dashboard Type** list to display the image on the dashboard. In this box, enter the URL of image (For example: - https://jar-vahana.s3.ap-south-1.amazonaws.com/images/Tech5-web.jpg). **Note:-**

	While configuring a dashboard with **IMAGE** dashboard type, the images are uploaded from Vahana server. So, before entering the URL of an image, you should know that where the image is stored on the Vahana platform.
Template Id	If you choose **IMAGE** in the **Dashboard Type** list, enter 2 in the **Template Id** box. In the **Template Id** box, the value: **2** denotes that you are uploading an image on the dashboard. In the **Dashboard Type** list, if you choose **VIEW**, enter **4** in the **Template ID** box. For every dashboard type such as image, online, view, and others, you will enter unique value in the **Template Id** box.
Sort Sequence	In this box, enter the sequence number (For example: - 1 or 2, or 3) of the dashboard type if you want to display more than one dashboard type on the dashboard.
Refresh Frequency	Click this list to select the any of the following options: ➢ **Daily** ➢ **Weekly** ➢ **Every Login** ➢ **Login Page Load** These options specify when the dashboard will be refreshed. For example: *- In the **Refresh Frequency** box, if you select **Daily**, it means that the dashboard is refreshed on the daily basis*.
Plus Group Name	Click this box to select the plus group that you want to display on the dashboard.
Drawer Name	Click this box to select the hamburger drawer that you want to display on the dashboard.

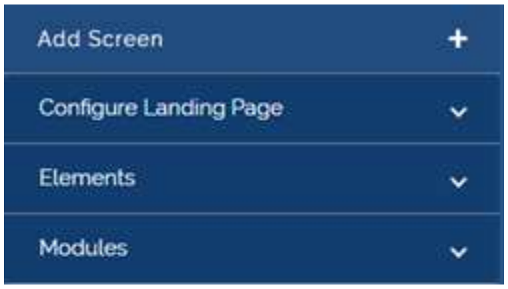
5. After you enter or select values in the respective boxes as described in the last table, click **Save (Fig 3.4 (c))**, the dashboard is configured.

3.4.1 Configuring a Card

In a mobile dashboard, you can configure a card. A dashboard card can consists of header, sub headings, list, textual content, media, icon, button, and other action items/features. . The layout of the card is determined on the basis of content, image, or any other data that you want to show on the card.

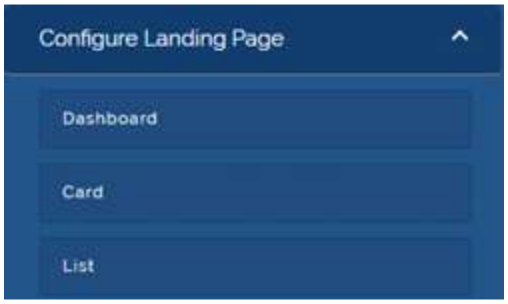
To configure a card:

1. On vDesigner application's dashboard, locate the left navigation pane.



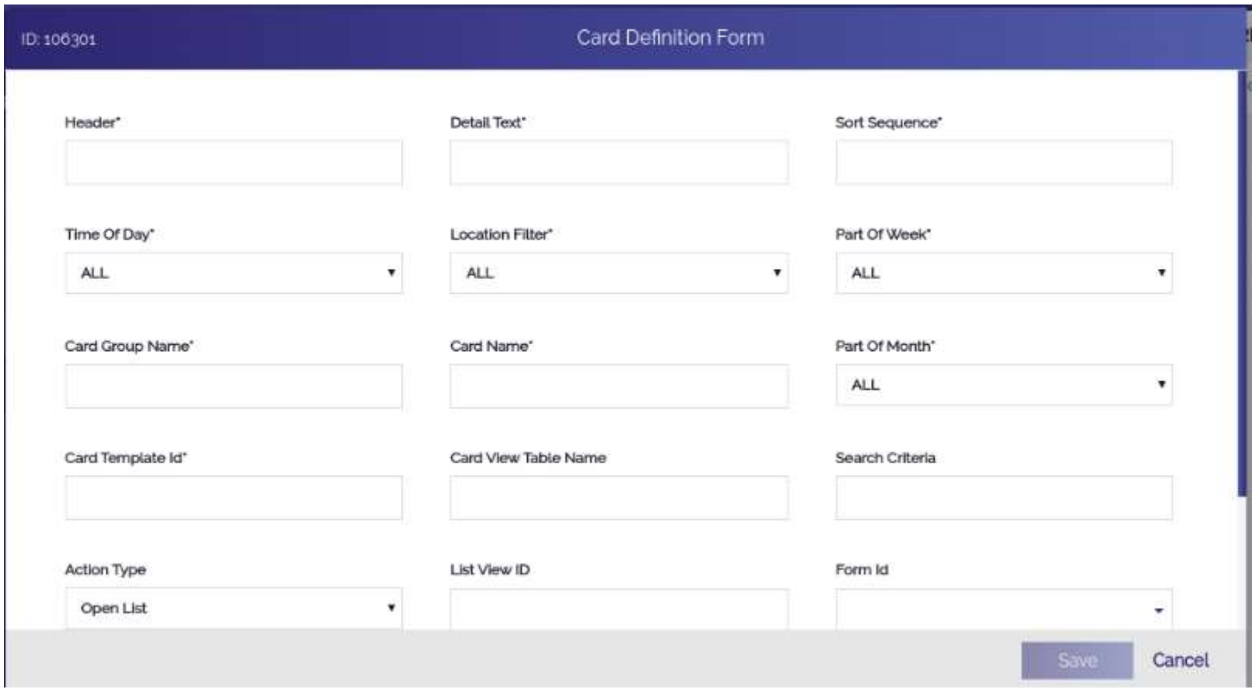
(Fig 3.4.1 (a))

2. In the left navigation pane, click **Configure Landing Page**, the navigation pane expands.



(Fig 3.4.1 (b))

3. Under **Configuring Landing Page (Fig 3.4.1 (b))**, click **Card**, the **Card Definition Form** dialog box **(Fig 3.4.1 (c))** opens.



ID: 106301

Card Definition Form

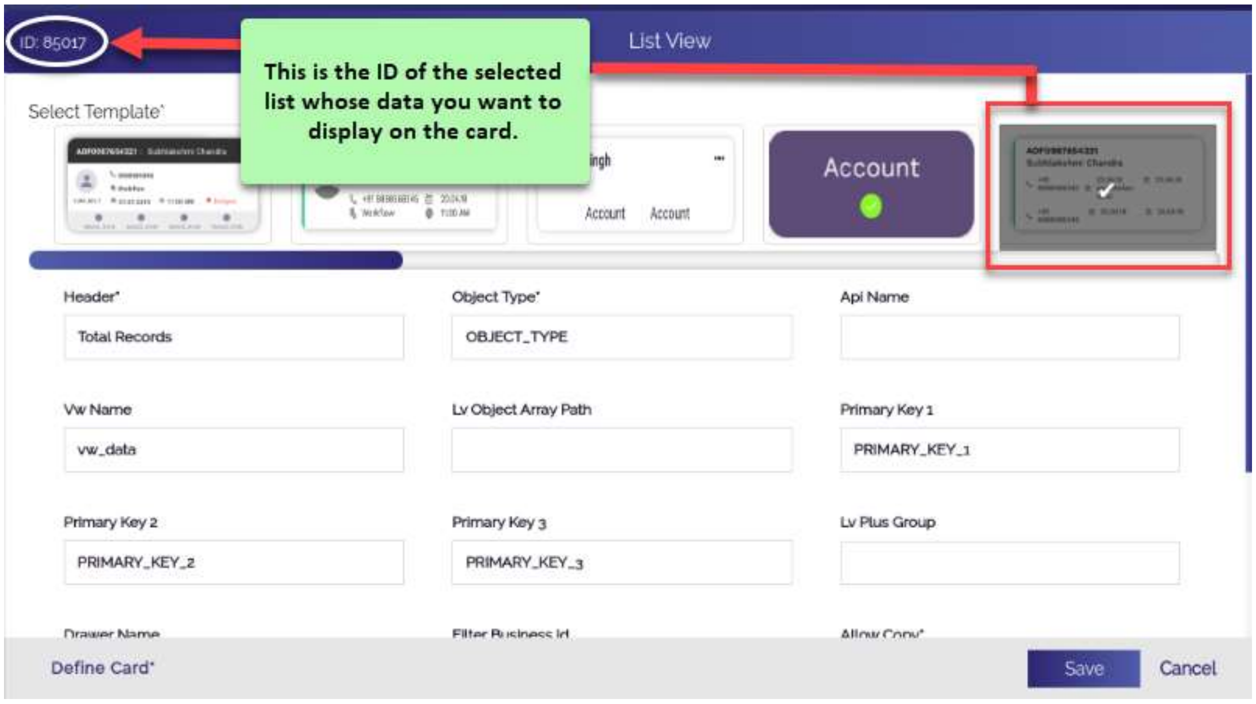
Header*	Detail Text*	Sort Sequence*
Time Of Day*: ALL	Location Filter*: ALL	Part Of Week*: ALL
Card Group Name*	Card Name*	Part Of Month*: ALL
Card Template Id*	Card View Table Name	Search Criteria
Action Type: Open List	List View ID	Form Id

(Fig 3.4.1 (c))

4. In the **Card Definition Form** dialog box, enter or select values in the respective boxes as follows:

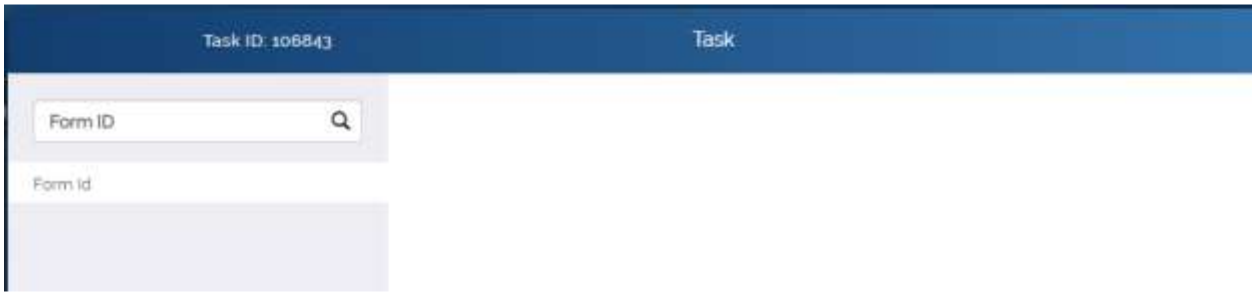
Box/List	**Description**
Header	In this box, enter the title or header name (For example: - Enroll) of the card.
Detail Text	In this box, enter the subheading or brief text (For example: - To start a new Identification process) about card or card's usage. The value that is entered in this box appears under the title or header of the card.
Sort Sequence	In this box, enter the sequence number (For example: - 1 or 2, or 3) of the card if you want to display more than one card on the dashboard. Based on the number that is entered in this box, the card is displayed on the dashboard.
Card Group Name	In this box, enter the value: Dashboard. Entering this value specifies that the card will be displayed on the dashboard.
Card Name	In this box, enter the name of card (For example: - Dashboard Card).
Card Template ID	In this box, enter a predefined template ID (For example: - **1**, **2**, or **3**). This ID is pre-defined at the application level. By using the card template ID, the mobile app displays the card on the screen.
Card View Table Name	In this box, enter the name of database view (For example: - vw_data_count) if you want to display the data from the database view.
Action Type	Click this list to select any of the following values: ➢ **Perform User Defined Action** Select this option if you want to configure the action that the user can perform on the card. ➢ **Open List**

	Select this option if you want to display the data of a list on the card. In this case, the data is fetched from the database view.
List View ID	In this box, enter the ID of list (For example: - 85017) **(Fig 3.4.1 (d))** whose data you want to display on the card. In this box, enter the list ID if you have selected **Open List** in the **Action Type** list. Otherwise, leave this box blank. **Note**:- To know more about list and to co-relate the screen capture: **(Fig 3.4.1 (d))**, visit the heading section: Configuring a List.
User Defined Action	You can use this feature to configure an action. You can configure the action if you have selected **Perform User Defined action** in the **Action Type** list. To configure an action, you can apply several tasks **(Fig 3.4.1 (e))**. The selection of the task to apply it depends on the function behavior that you want to implement on the card. The following set of steps describes to configure a form ID of specific form/screen. To configure an action: 1. On the **Card Definition Form** dialog box, click **Create Action**, the **Task** dialog box **(Fig 3.4.1 (e))** opens. 2. In the **Search by task** box, enter **Form ID**, the **Form id** task is displayed. 3. Click the **Form id** task, the **Form Id** dialog box opens. 4. In the **Form Id** dialog box, locate the **Navigation Form Id** box. 5. Click in the **Navigation Form id** box, a list **(Fig 3.4.1 (f))** of existing forms is displayed. 6. Locate the form by its form ID (For example:- 83882) and then select it. 7. After you select the form, click **Save**, the form id is configured.

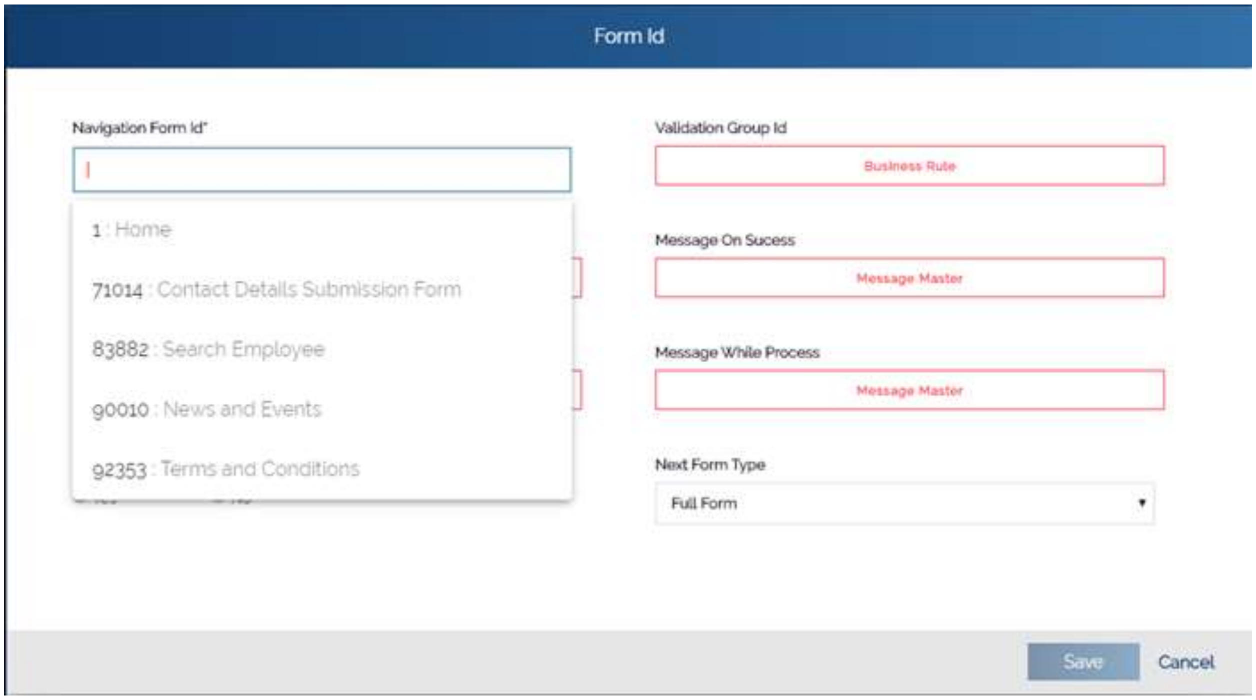

ID: 85017
This is the ID of the selected list whose data you want to display on the card.
List View
Select Template*
Account
Define Card*
Save
Cancel

Header*	Object Type*	Api Name
Total Records	OBJECT_TYPE	
Vw Name	Lv Object Array Path	Primary Key 1
vw_data		PRIMARY_KEY_1
Primary Key 2	Primary Key 3	Lv Plus Group
PRIMARY_KEY_2	PRIMARY_KEY_3	
Drawer Name	Filter Business Id	Allow Copy*

(Fig 3.4.1 (d))



(Fig 3.4.1 (e))



(Fig 3.4.1 (f))

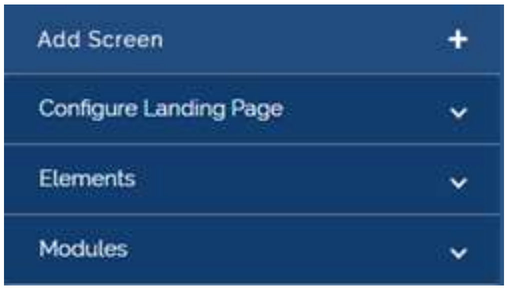
While configuring a card in vDesigner, if you choose **Open List** action, the data is fetched from the database view and therefore is dynamically displayed on the card. If you choose **Perform User defined action**, you can configure an action that allows the user to perform an action on the card.

3.4.2 Configuring a List

In the vDesigner application, you can configure a list to display specific set of data in the list format. To display the data in different layout, the List feature incorporates multiple in-built card templates. When you configure a list, you need to select a template to determine a layout, in which you want to display the data on the card. You can display the data in the list by using **database view** or JSON type **array object**.

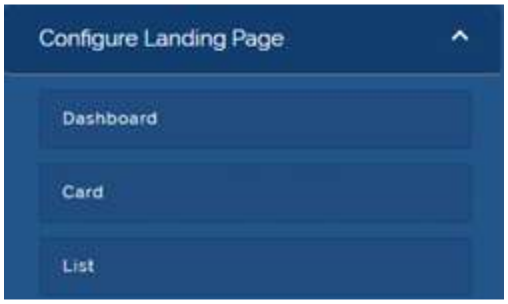
To configure a list:

1. On vDesigner application's dashboard, locate the left navigation pane.



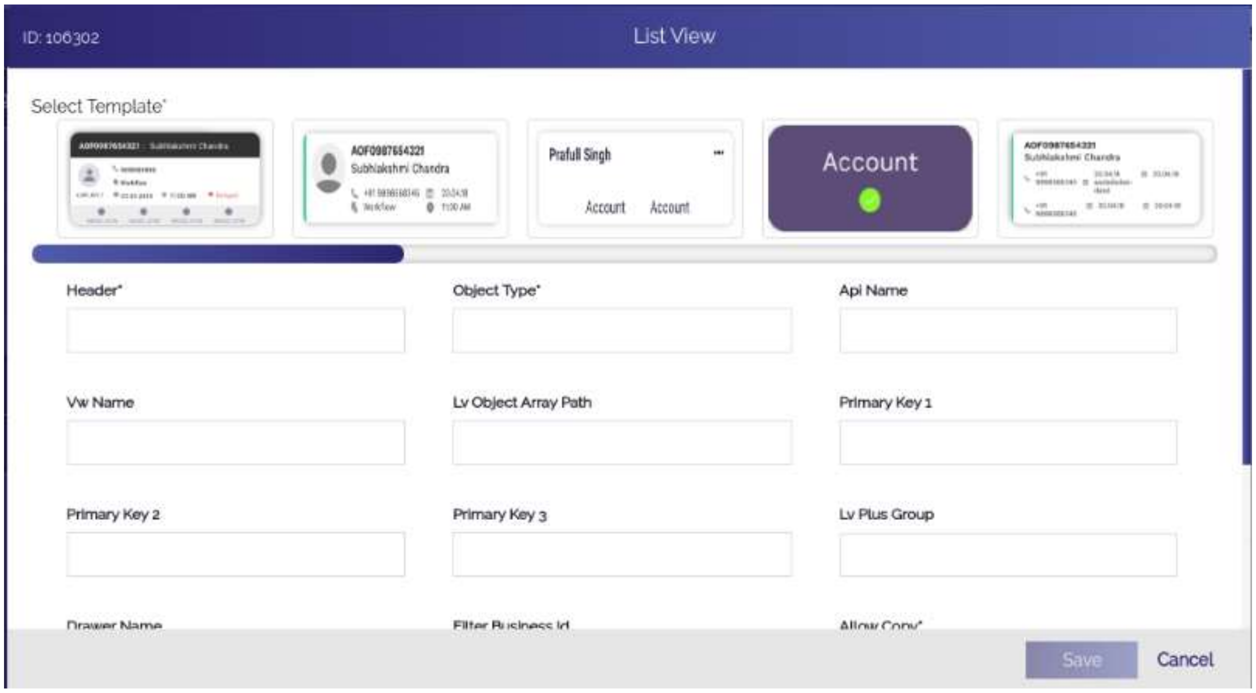
(Fig 3.4.2 (a))

2. In the left navigation pane, click **Configure Landing Page**, the navigation pane expands.



(Fig 3.4.2 (b))

3. Under **Configuring Landing Page (Fig 3.4.2 (b))**, click **List**, the **List View** dialog box **(Fig 3.4.2 (c))** opens.

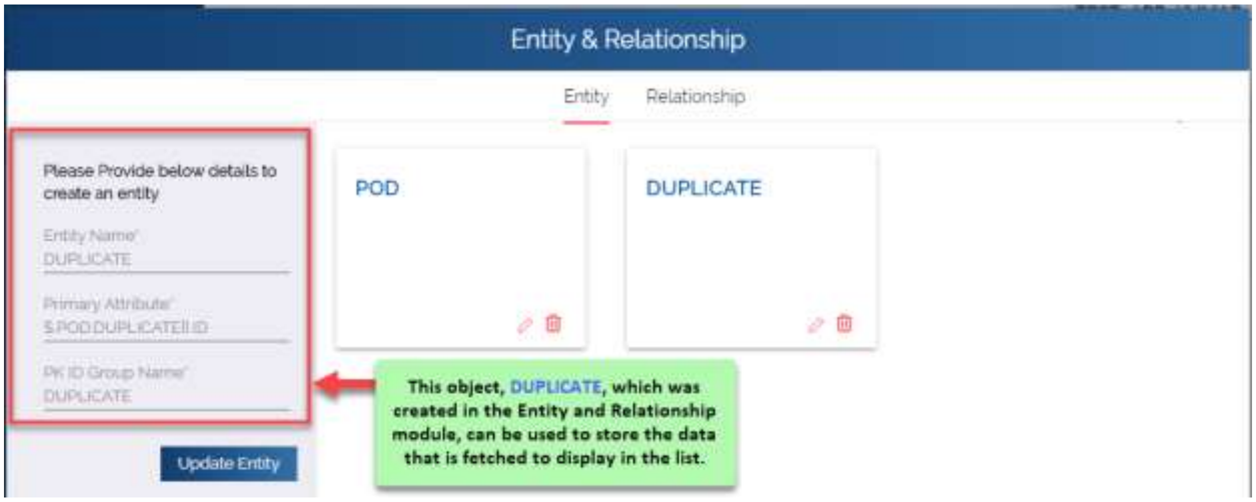


(Fig 3.4.2 (c))

4. In the **List View** dialog box, enter values as follows:

Box	Description
Header	In this box, enter the name/title (For example: - Duplicate List) of the list.
Object Type	In this box, you can enter the name of object as follows: **<u>Case1</u>:-** (If you are using array object to fetch the data) In this box, enter the name object (For example: - DUPLICATE) where the data is stored after it is fetched from the database. After you declare the name of object (For example: - DUPLICATE) in the **Object Type** box, enter the path of array (For example: - $.POD.DUPLICATE[*]) in the **LV Array Object Path** box. **<u>Note</u>:-** In the **Object Type** box, you enter the name of object **(Fig 3.4.2 (d))** that you create in the **Entity & Relationship** module. You can use this object if you are using an API to access the data source and then fetch the data to display it in the list. **<u>Case2</u>:-** (If you are using database view to fetch the data) In this box, enter the name of database object (For example: - OBJECT_TYPE). After the data is fetched, this object holds the entire data in database view format.

Api Name	In this box, enter the name of API if you using the API to fetch the data from the respective data source. After API fetches the data, it stores the data in the **DUPLICATE** object. **Note**:- In the **Api Name** box, you will enter the name of API if you choose **array object** to fetch the data. If you choose **database view** to fetch the data, leave the **API Name** box blank.
LV Object Array Path	In this box, enter the path or array (For example: - $.POD.DUPLICATE[*]) where the data is stored after it is fetched from the data source. In the **LV Object Array Path** box, you enter the path of array if you choose **array object** to fetch the data. **Note**:- If you choose **database view** to fetch the data, leave the **LV Object Array Path** box blank. The path: **$.POD.DUPLICATE[*]** is the path of the **Duplicate** object. In this path of array, the asterisk (*) character that is enclosed by bracket works as a common loop. When the data is fetched, it reads the data from 0th index and picks the entire record set of data and treats it as a single object. It moves through each index consecutively and picks other records of data.
Primary Key1/Primary Key2/ Primary Key3	In this box, you can enter the values as follows: **Case1**:- (If you are using the array object to fetch the data) ➢ In these boxes, enter the variable (For example: - ID) **(Fig 3.4.2 (d))** that you have declared when you created the object in the Entity and Relationship module. You will use this variable if you are fetching the data by using **array object**. **Case2**:- (If you are using database view to fetch the data) ➢ In these boxes, enter the name of primary key (For example: - PRIMARY_KEY_1, PRIMARY_KEY_2, PRIMARY-KEY_3) if you are fetching the data from the **database view**.
Vw Name	In this box, enter the name of database view if you are using the database view to fetch the data from the data source and then display it in the list.



Please Provide below details to create an entity

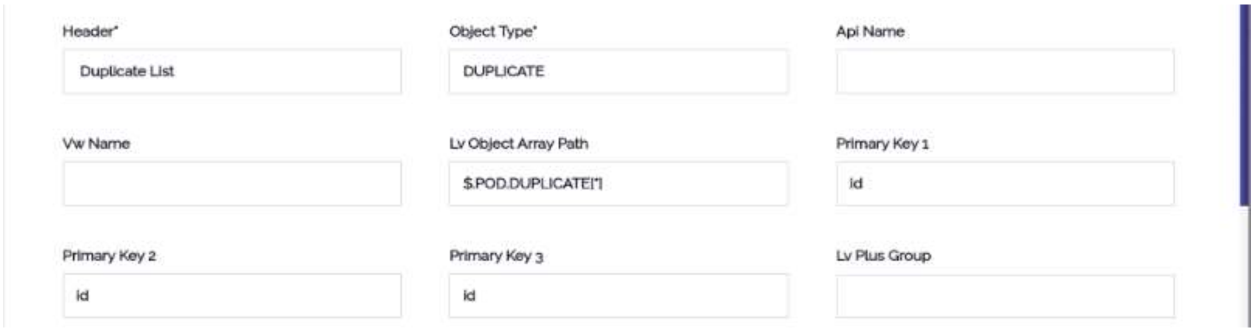
Entity Name*: DUPLICATE

Primary Attribute*: $.POD.DUPLICATE[*].ID

PK ID Group Name*: DUPLICATE

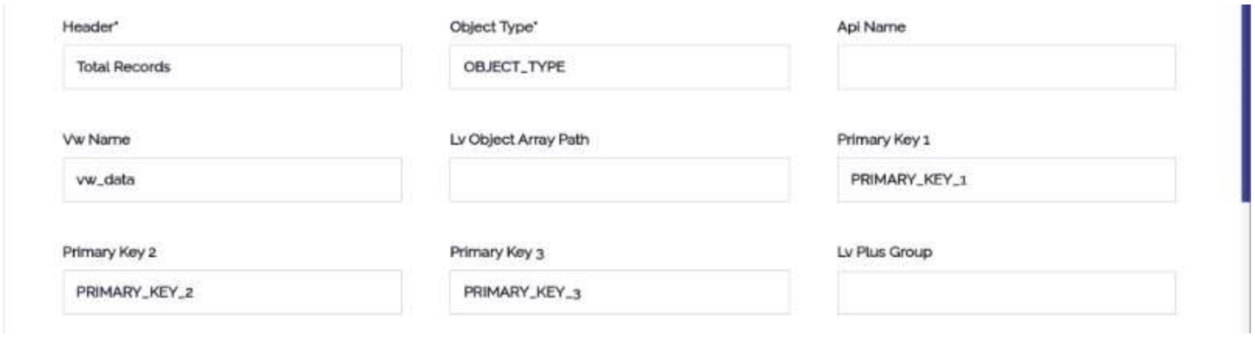
This object, DUPLICATE, which was created in the Entity and Relationship module, can be used to store the data that is fetched to display in the list.

(Fig 3.4.2 (d))



Header*: Duplicate List	Object Type*: DUPLICATE	Api Name:
Vw Name:	Lv Object Array Path: $.POD.DUPLICATE[*]	Primary Key 1: id
Primary Key 2: id	Primary Key 3: id	Lv Plus Group:

(Fig 3.4.2 (e)) (Configuring List with Array Object)



Header*: Total Records	Object Type*: OBJECT_TYPE	Api Name:
Vw Name: vw_data	Lv Object Array Path:	Primary Key 1: PRIMARY_KEY_1
Primary Key 2: PRIMARY_KEY_2	Primary Key 3: PRIMARY_KEY_3	Lv Plus Group:

(Fig 3.4.2 (f)) (Configuring List with Database View)

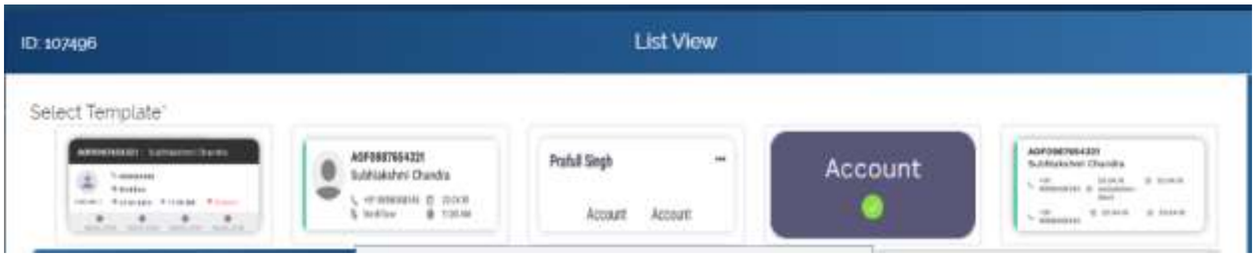
After you enter values in the respective boxes as described in the last table, define the card that is described as follow:

3.4.2.1 Defining Card

You define a card to determine the layout of the list, in which data is displayed on the card.

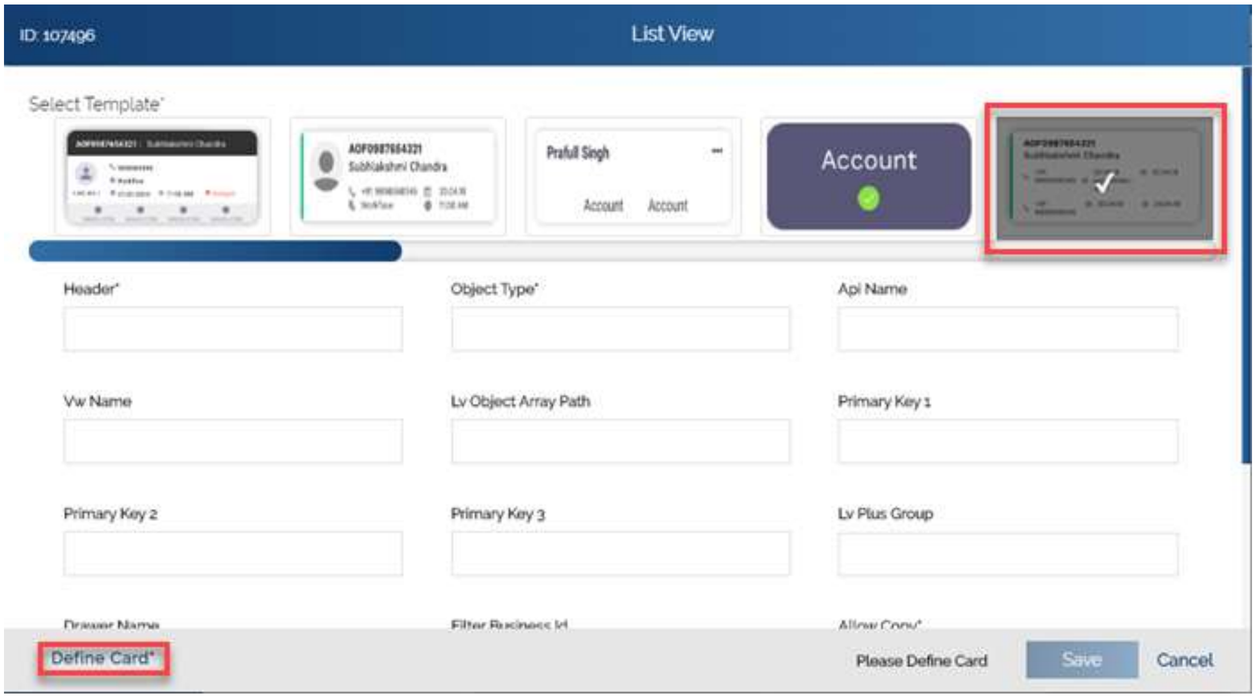
To define the card, perform the function as follows:

1. In the **List View** dialog box **(Fig 3.4.2.1 (a))**, locate upper panel that contains the card templates.



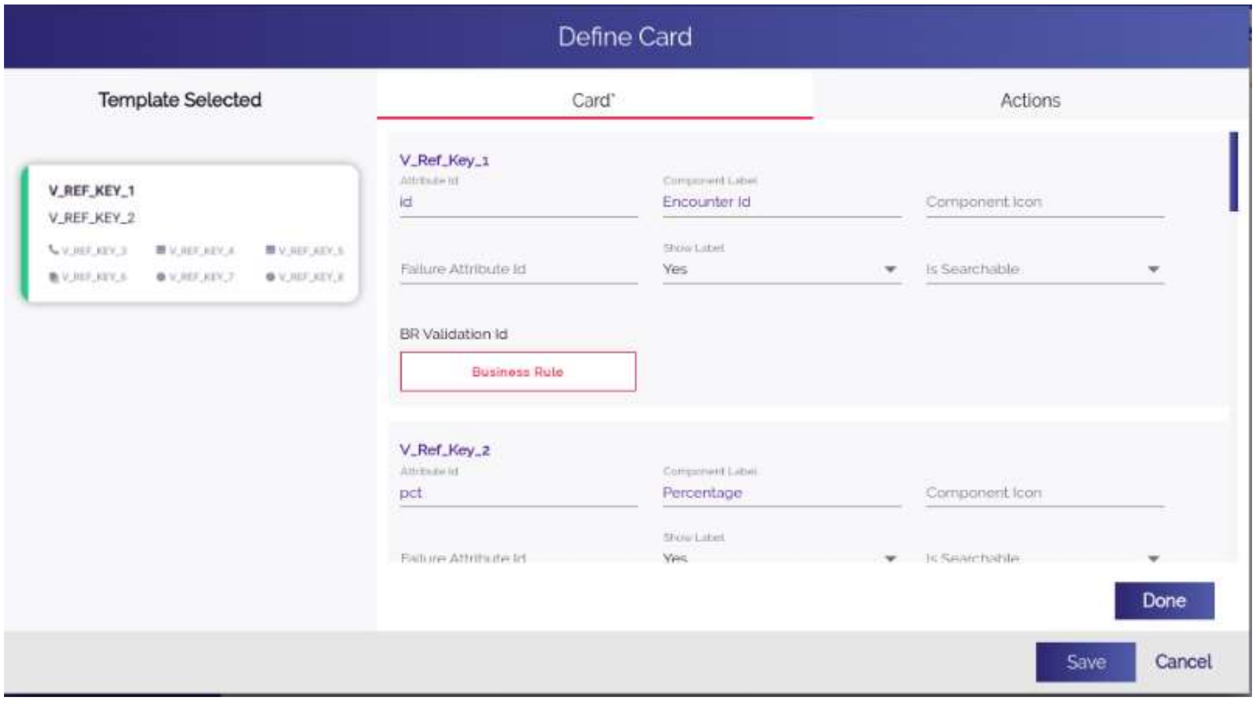
(Fig 3.4.2.1 (a))

2. After you select a card template **(Fig 3.4.2.1 (b))**, the **Define Card** link appears.



(Fig 3.4.2.1 (b))

3. Click the **Define Card** link, the **Define Card** dialog box **(Fig 3.4.2.1 (c))** opens.



V_Ref_Key_1
Attribute Id: id
Component Label: Encounter Id
Component Icon
Failure Attribute Id
Show Label: Yes
Is Searchable
BR Validation Id

V_Ref_Key_2
Attribute Id: pct
Component Label: Percentage
Component Icon
Failure Attribute Id
Show Label: Yes
Is Searchable

(Fig 3.4.2.1 (c))

4. In the **Define Card** dialog box **(Fig 3.4.2.1 (c))**, the **Card** tab displays fields (For example: - **ID**, **Component Label**, etc.) for different variables/key (For example: - **V_Ref_Key_1** or **V_Ref_Key_2**).
5. In the respective fields, enter values as follows:

Box	Description
Attribute ID	In this field, enter the attribute ID (For example: - ID). In the attribute ID, the list holds specific value and then displays it in the corresponding label on the card of mobile app. **Note**:- In the **Attribute ID** field, enter the value that matches the name of key/attribute that you receive in the response after the data is fetched from the respective data source. For instance: - If you are receiving the name of a person in the **Name** key/attribute in the output of the API, enter **Name** in the **Attribute ID** box. If the value in **Attribute ID** field does not match the key name or attribute name in the response of the API, list will not display the data.
Component Label	In this field, enter the name of label (For example:-Employee ID, Count, or Name, etc). The list will display the data along with the label.
Component Icon	This is optional field. In this field, enter server based path of icon, if you want to display the icon of the label.
Show Label	Click the list to:

	➢ Select **Yes** if you want to display the label along with the respective value that the list displays on the card. ➢ Select **No** if you do not want to display the label along with the respective value that the list displays on the card. In this case, the list will only display the value.
Is Searchable	Click the list to: Select **Yes** if you want to incorporate the search option along with the value that the list displays. Otherwise, select **No**.
BR Validation Id	This is the optional feature. While defining a card, if you want to apply the business rule, click Business Rule, the **Rule ID** dialog box opens. In the **Rule ID** dialog box, write the business rule.

6. After you enter values in the respective fields as described in the last table, enter values in these fields for other variables.

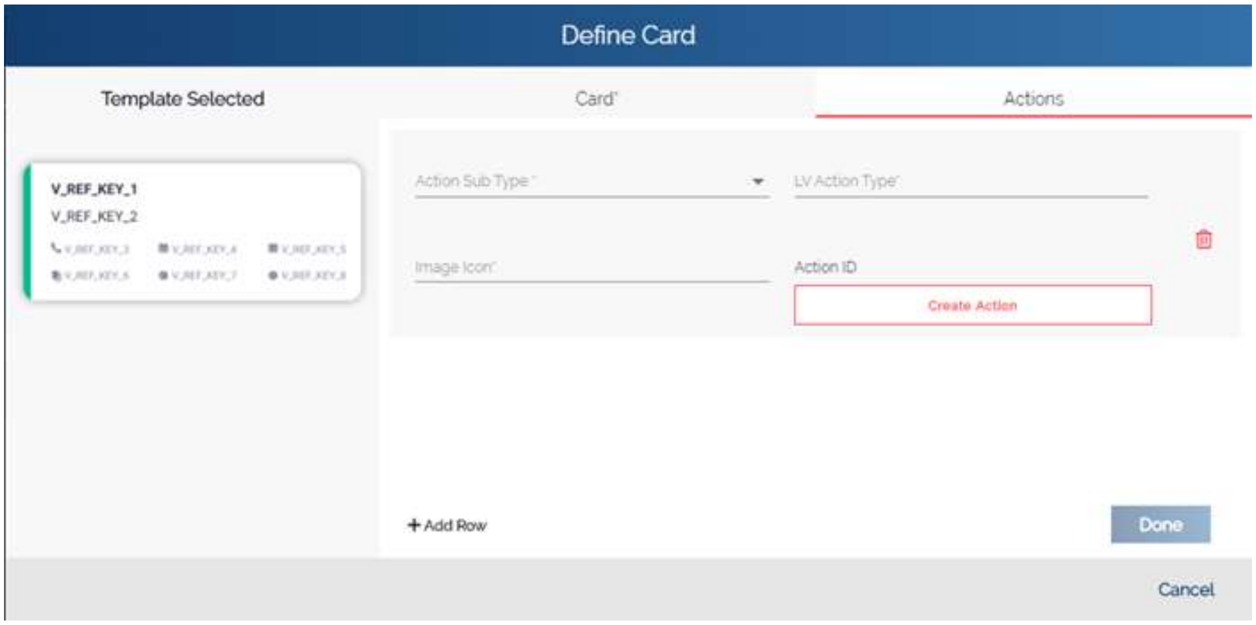
3.4.2.2 Configuring Action on the Card

This function allows you to configure an action on the list of the card. You can configure the action on the list horizontally or vertically. You can incorporate a phone book, delete function, and other usable action items.

To configure an action on the list:

1. In the **Define Card** dialog box, click the **Action** tab **(Fig 3.4.2.2 (a))**, the **Define Card** dialog box displays the group of the following fields:

➢ **Action Sub Type**
➢ **LV Action Type**
➢ **Image Icon**
➢ **Action ID**

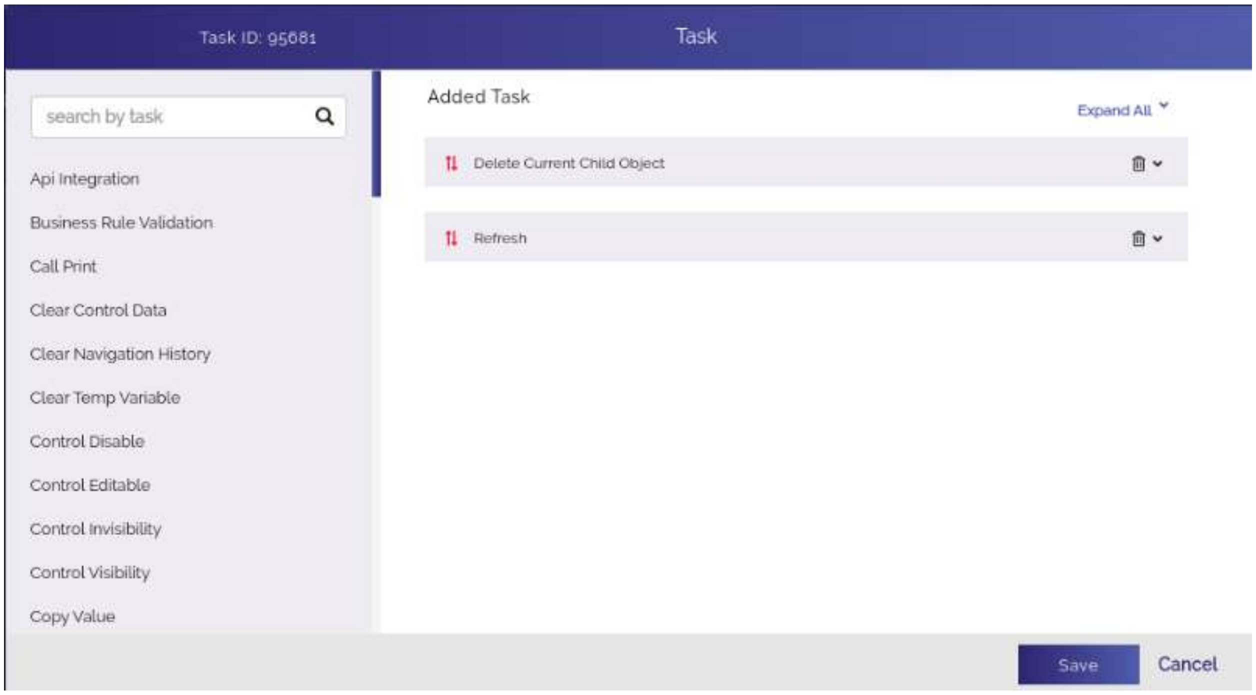


(Fig 3.4.2.2 (a))

2. In these fields, enter values as follows:

Field	Description
Action Sub Type	Click this list to select: ➢ **HAB** if you want to place action items along the horizontal bar. ➢ **VAB** if you want to place action items along the vertical bar.
LV Action Type	In this box, enter the name of action item (For example: - Edit, Delete, Info, etc).
Image Icon	In this box, enter the name of icon that is displayed in the list. The icon denotes the respective action item. For instance: - For the icon of the delete function, enter: **ic_drop**. For the icon of edit function, enter: **ic_appointment**.

	These icons are available on the Vahana server.
Action ID	You can use this function to configure an action for the action item. To configure the action: ➢ Click **Create Action (Fig 3.4.2.2 (a))**, the **Task** dialog box **(Fig 3.4.2.2 (b))** opens. ➢ In the **Search by task** box **(Fig 3.4.2.2 (b))**, enter the task that you want to apply (For example: - Delete Current Child Object), the **Delete Current Child Object (Fig 3.4.2.2 (b))** task appears. ➢ Select the task (For example: - Delete Current Child Object) and then click **Save (Fig 3.4.2.2 (b))**, the action is successfully configured. **Note**:- The function of the **Delete Current Child Object** task is that it deletes currently selected element after the user taps the **Delete** icon in the card template.



(Fig 3.4.2.2 (b))

3. To configure other actions, click **Add Row (Fig 3.4.2.2 (a))**, another row of the following fields: appears.

- **Action Sub Type**
- **LV Action Type**
- **Image Icon**
- **Action ID**

4. In these fields, enter values as described in the last table to configure another action on the list.
5. After you configure the card template and action items that you want to place on the card template, click **Done (Fig 3.4.2.2 (b))**, the **Define Card** dialog closes.
6. In the **List View** dialog box, click **Save**, the list is configured.

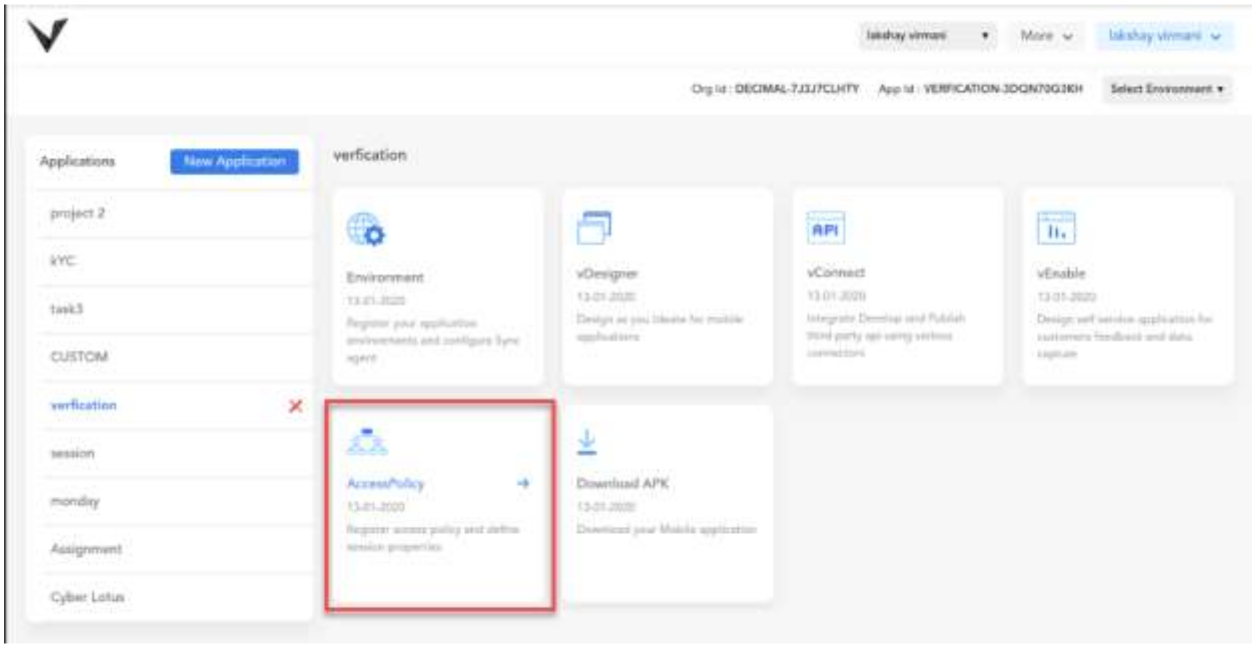
3.5 Defining Access Policy

'Access Policy' is one of the important features of the Vahana platform. It allows you to build the access policy for the mobile app that you design in the vDesigner application. Before publishing a mobile app, you need to define the access policy.

While defining the access policy for the mobile app, you can define several session related timeframes, requirements of app related logs. You can define the access policy as follows:

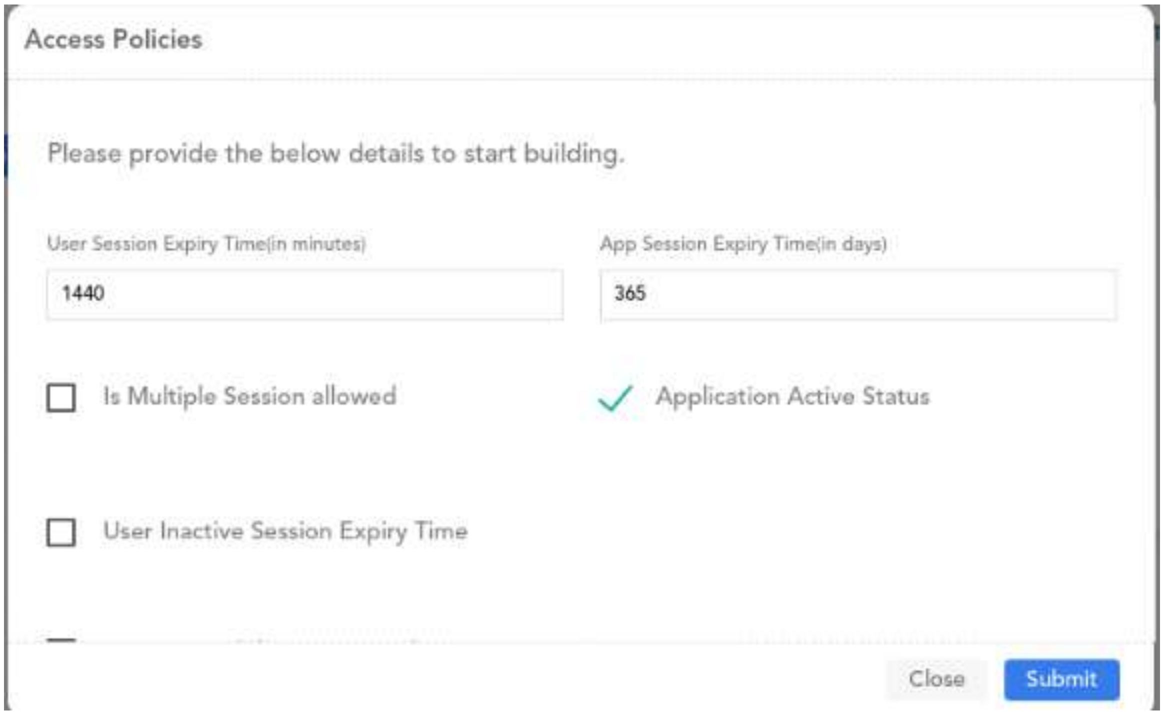
To define the access policy:

1. On the Vahana project management dashboard, locate the **Access Policy** tile **(Fig 3.5 (a))**.


lakshay virmani
More
lakshay virmani
Org Id : DECIMAL-7J3J7CLHTY
App Id : VERIFICATION-3DQN70G3KH
Select Environment
Applications
New Application
project 2
kYC
task3
CUSTOM
verfication
session
monday
Assignment
Cyber Lotus
verfication
Environment
13-01-2020
Register your application environments and configure Sync agent
vDesigner
13-01-2020
Design as you ideate for mobile applications
vConnect
13-01-2020
Integrate Develop and Publish third party api using various connectors
vEnable
13-01-2020
Design self service application for customers feedback and data capture
AccessPolicy
13-01-2020
Register access policy and define session properties
Download APK
13-01-2020
Download your Mobile application

(Fig 3.5 (a))

2. Click the **Access Policy** tile, the **Access Policy** dialog box **(Fig 3.5 (b))** opens.


Access Policies

Please provide the below details to start building.

User Session Expiry Time(in minutes)
1440

App Session Expiry Time(in days)
365

☐ Is Multiple Session allowed

✓ Application Active Status

☐ User Inactive Session Expiry Time

(Fig 3.5 (b))

3. In the **Access Policy** dialog box, enter value or select the option as described in the following table:

Box/Check box	Description
User Session Expiry Time (In Minute)	In this box, enter the value in the minute unit (For example: - 1440). This value specifies the timeframe after which the user's login session in mobile app will automatically expire.
App Session Expiry Time (In Days)	In this box, enter the value in the day unit. This value specifies the number of days after which the mobile app session will automatically expire. If the app session expires, the user needs to restart the mobile app by tapping its icon.
Is Multiple Session Allowed	Click this check box to allow multiple sessions on the mobile app. Allowing multiple sessions on the app specifies that more than one user can access the app by using different devices.
User Inactive Session Expiry Time	Click this check box if you want to define the user inactive session expiry time. The user inactive session time specifies the timeframe, in which the user does perform any function on the mobile app. After the user inactive session time exceeds the configured value of user inactive session expiry time, the user session will automatically expire on the mobile app. Therefore, user will have to log in the mobile app again to access it. To define the user inactive session expiry time: 1. Click the **User Inactive Session Expiry Time** check box, the **User Inactive Session Expiry Time(min)** box opens.

	2. In the **User Inactive Session Expiry Time(min)** box, enter the value in the minute unit (For example:- 40). The value: **40** that is entered in the **User Inactive Session Expiry Time(min)** box specifies that if the user does not perform any function on the mobile app for the last 40 minutes, the user session will automatically expire.
Source IP Validation Required	Click this check box if you want to give access of the mobile on specific IP address. After you click the **Source IP Validation Required** checkbox, the **Provide IP** box opens. In the **Provide IP** box, enter the IP address on which you want to provide the access of the mobile app.
Mobile Application Log Required	Click this check box if you require communication logs of mobile app.
Server Audit Log Required.	Click this check box if you require server audit logs for reporting purposes.

4. After you click the respective check box to select the desired options and enter values in the respective boxes, click **Submit**, access policy is successfully defined.

3.6 Downloading APK

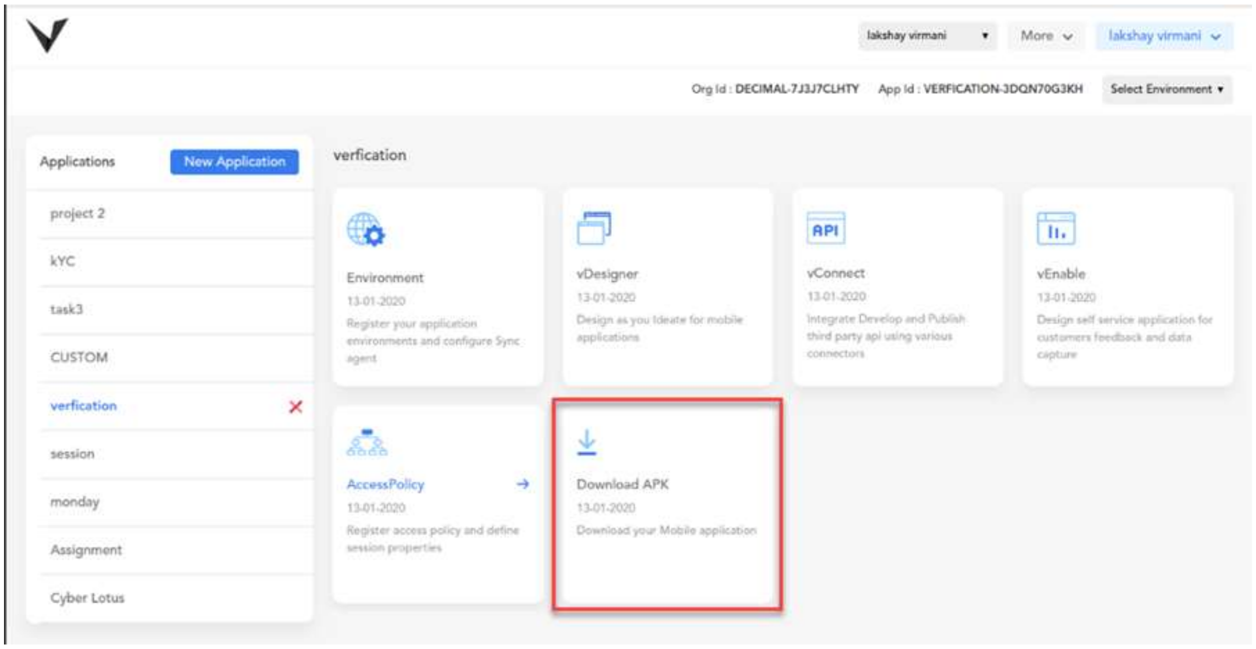
This feature allows you to publish the build of mobile app. Later, this mobile app can be installed on the mobile phone device. Here, it is one important point to note that you cannot publish the build of the mobile app if you do not define access policy. **First you need to define the access policy, only then you can publish the build of the mobile app**.

While publishing the (.apk) file of the mobile app, you can also

- **Upload the logo of the mobile app**
- **Upload the JSON**
- **Define the version name and the version number of the mobile app**
- **Define dashboard header**
- **Using specially designed client-specific SDK(s) such as Roadzen camera and video SDK(s), tata SDK, Tech5 SDK, and others**

To publish the APK file of the mobile app:

1. On the Vahana project management dashboard, locate the **Download APK** tile **(Fig 3.6 (a))**.



Org Id : DECIMAL-7J3J7CLHTY App Id : VERFICATION-3DQN70G3KH

Environment 13-01-2020 Register your application environments and configure Sync agent

vDesigner 13-01-2020 Design as you Ideate for mobile applications

vConnect 13-01-2020 Integrate Develop and Publish third party api using various connectors

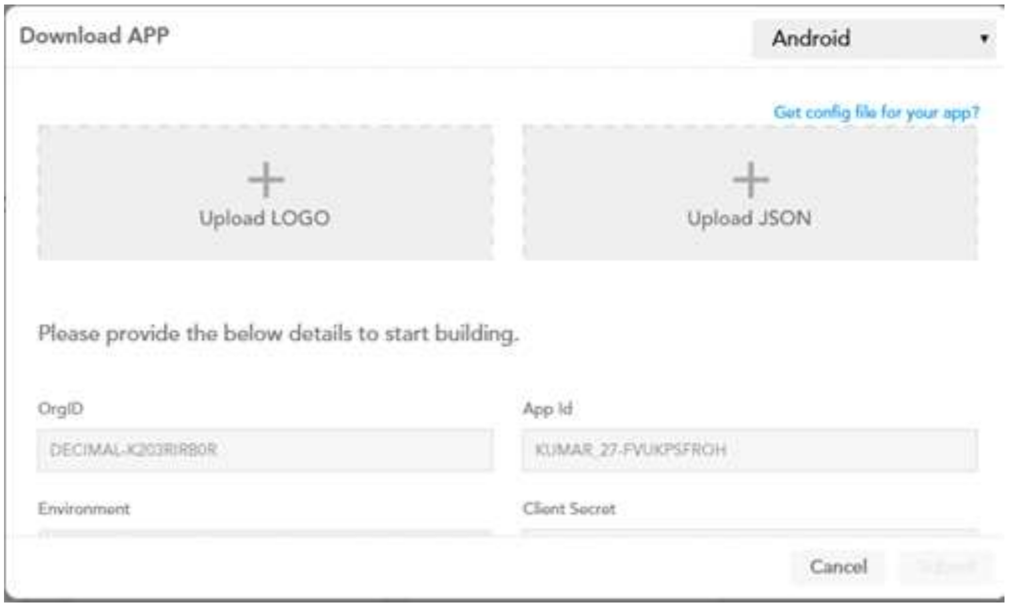
vEnable 13-01-2020 Design self service application for customers feedback and data capture

AccessPolicy 13-01-2020 Register access policy and define session properties

Download APK 13-01-2020 Download your Mobile application

(Fig 3.6 (a))

2. Click the **Download APK** tile **(Fig 3.6 (a))**, the **Download App** dialog box **(Fig 3.6 (b))** opens.


Download APP

Android

Get config file for your app?

Upload LOGO

Upload JSON

Please provide the below details to start building.

OrgID: DECIMAL-K203RIR80R

App Id: KUMAR 27-FVUKPSFROH

Environment

Client Secret

Cancel

(Fig 3.6 (b))

3. In the **Download App** dialog box, perform the function in the respective fields as follows:

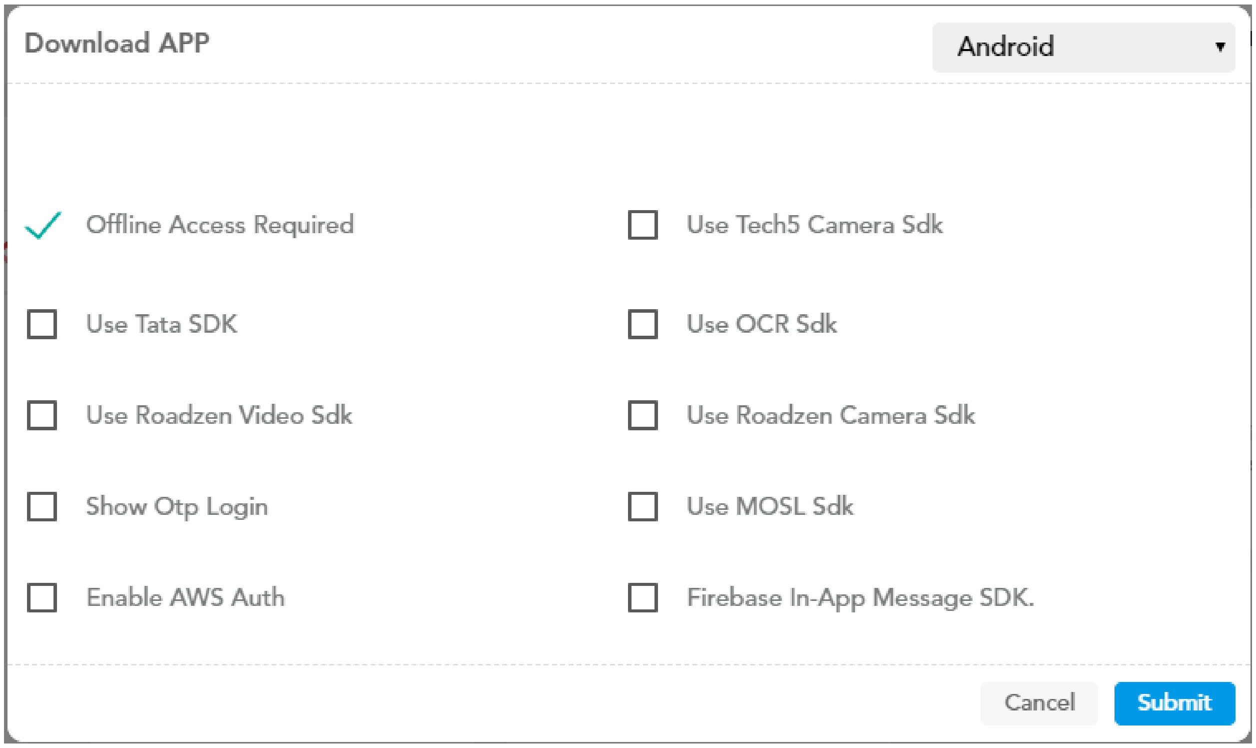
Field	Description
Upload Logo	Click this box and then select the desired image file to upload it as a logo of the mobile app.
Upload Json	Click this box and then select the JSON file to upload.
OrgID and App id	These fields, which are non-editable, display unique org ID and app ID respectively. The values of Org ID and App ID are automatically created at the time, you create new app on the Vahana platform. The instance of specific app is identified by Org ID and App id.
Environment	Click this list to select the environment, in which you have developed the mobile app. If you have used the Sand_box as environment, click the list to select **SAND_BOX**. After you select the environment, the **Submit** button becomes active.
Client Secret	The client secret is commonly used to debug the mobile app.
Platware Client URL	This box displays the URL of the API gateway. This URL is used to handle API(s) that are deployed on the Vahana platform.
Enable Push Notification	Click this check box to enable the push notification from the mobile app.
Version Name	In this box, enter the version name (For example: - 1.0 Or 2.11, etc.) of mobile app. In this box, you can use the decimal to enter or write the version name of the mobile app.
Version Code	In this box, enter the version code (For example: - 1, 2, or 5). The **Version Code** box does not allow decimal when you enter the numeric value to write the version code.

Application Name	In this box, enter the name of the application that you want to publish. After the user install the mobile app on his mobile phone, the name of application appears below the icon/logo of the mobile app.
Dashboard Header	In this box, enter the name of dashboard header (For example: - Welcome or Welcome on CusManagement App). After the user install the mobile app and then access by logging in it, the name of dashboard header appears on the top of the dashboard.
Application Theme Color	Click this list to select the theme color of the mobile app that you want to publish. Suppose, if you select the theme color: ‘blue’, elements, UI, and other objects in the app will be displayed in the blue color.
Login Template	Click this list to select the login template. After you select a login template, you need to implement or code the login mechanism of the selected template. Based on the selected template, the mobile app may ask you to enter: ➢ **Either mobile number with OTP or,** ➢ **Mobile number with password, or** ➢ **Any other mechanism**
Document Manager Client Secret	In this box, enter alphanumeric document manager client secret key to use document manager service.
Document URL	In this box, enter the URL of the document manage service if you want to use the document manager service.
Tech5 License registration key	In this box, enter alphanumeric Tech5 licensed registration key if you want to make the Tech5 SDK available to the mobile app user.
Export as SDK	Click this check box to use the mobile app build as an SDK.
Remove Branding Logo	Click this check box if you want to remove the branding logo.
Enable SSL Bypass	By default, this option is selected. Being selected this option specifies that the you do not want to use the SSL certificate, If you want to include SSL certificate, click to clear the **Enable SSL Bypass** check box and then enter the name of SSL certificate in the **SSL Certificate name** box. .
Enable Register User	Click this check box if you want that only registered user can access the mobile app. After you click this check box, another option: **Open Login After Registration** starts appearing. If you click the **Open Login After Registration** check box, it means that the user can log in the mobile app after s/he becomes registered user.
Offline Access Required	Click this check box if you want to access the mobile app even without the availability of Internet connection.
Show OTP Login	Click this check box if you want to include the OTP feature in the login mechanism. After you click this check box, the server sends a unique OTP to the mobile app user. The user needs to enter the OTP to log in the mobile app.
Enable Forget Password	Click this check box if you want to incorporate the “forget password” feature in the login mechanism. By incorporating this feature, you allow the mobile app user to get the new password if s/he forgets the current password.

Enable AWS Auth	Click this check box if you want to use AWS service.
Enable Video Recording	Click this check box if you want to provide the video recording feature in the mobile app to the user.

4. You can also click the respective check box to use the following SDK (s) that are available on the Vahana platform:

- **Tech5 Camera SDK**
- **Tata SDK**
- **OCR SDK**
- **Roadzen Video SDK**
- **Roadzen Camera SDK**
- **MOSL SDK**
- **Firebase In-App Message SDK**

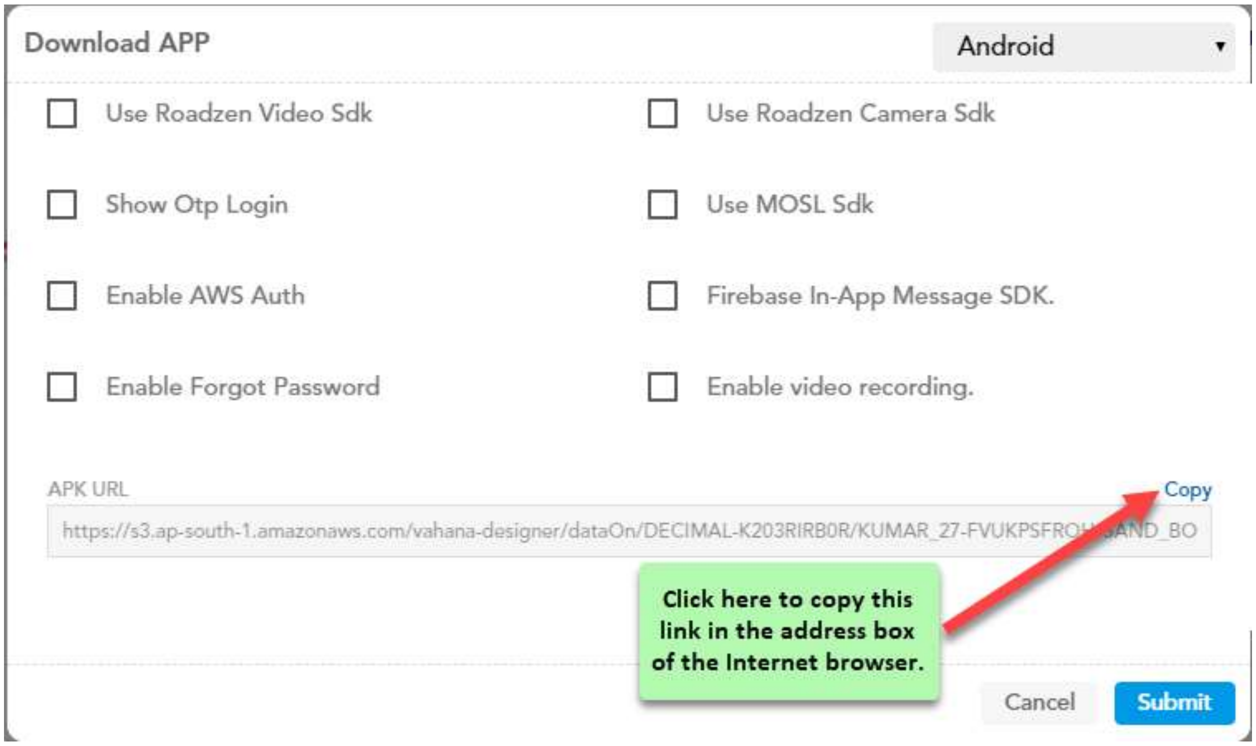

Download APP

Android

[✓] Offline Access Required
[ ] Use Tech5 Camera Sdk
[ ] Use Tata SDK
[ ] Use OCR Sdk
[ ] Use Roadzen Video Sdk
[ ] Use Roadzen Camera Sdk
[ ] Show Otp Login
[ ] Use MOSL Sdk
[ ] Enable AWS Auth
[ ] Firebase In-App Message SDK.

(Fig 3.6 (c))

5. After you enter the values in the respective boxes or click the respective check box to download the APK file, click **Submit (Fig 3.6 (c)).**

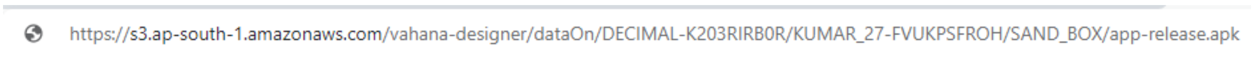


https://s3.ap-south-1.amazonaws.com/vahana-designer/dataOn/DECIMAL-K203RIRB0R/KUMAR_27-FVUKPSFROH/SAND_BO

Click here to copy this link in the address box of the Internet browser.

(Fig 3.6 (d))

6. After you click **Submit**, the Download APK service creates the URL to download the APK of the mobile app.
7. Scroll down the **Download App** dialog box, the **APK URL** box **(Fig 3.6 (d))** displays the URL that you can visit to download the build of mobile app.
8. Click **Copy (Fig 3.6 (d))** and then open the Internet browser on your computer system.
9. In the **Address** box of the Internet browser, place the mouse pointer, right-click and then select **Paste**, the URL **(Fig 3.6 (e))** is copied.



https://s3.ap-south-1.amazonaws.com/vahana-designer/dataOn/DECIMAL-K203RIRB0R/KUMAR_27-FVUKPSFROH/SAND_BOX/app-release.apk

(Fig 3.6 (e))

10. After you paste the URL, press **Enter**, the Internet browser starts downloading the APK file.
11. After the APK file is downloaded, transfer the APK to the mobile phone device and then install it.

3.7 Data Modeling (JSON Creation with Entity and Relationship)

JSON (Java Script Object Notation) is a popular data interchange format that is prevalently used to transmit the data between two entities. The syntax of JSON format is based on Java script and can be easily learnt to design HTTP based web API(s) and services. JSON is light-weight and easily readable. Also JSON can be used as an ideal data model to store object and entity related details that can be subsequently stored and managed on the database level.

In the vDesigner application, the user performs several functions such as creating a form, configuring a list, configuring elements, creating entities, establishing relationship between entities, and others. Each time a user performs the function to create/update these entities, the vDesigner application creates a data model in the JSON format. This data model stores the configuration details of all objects and entities that the user creates to design a mobile app in the vDesigner application. It significantly helps user track the configuration details of several objects in the mobile app and allows user to debug the mobile app during development phase.

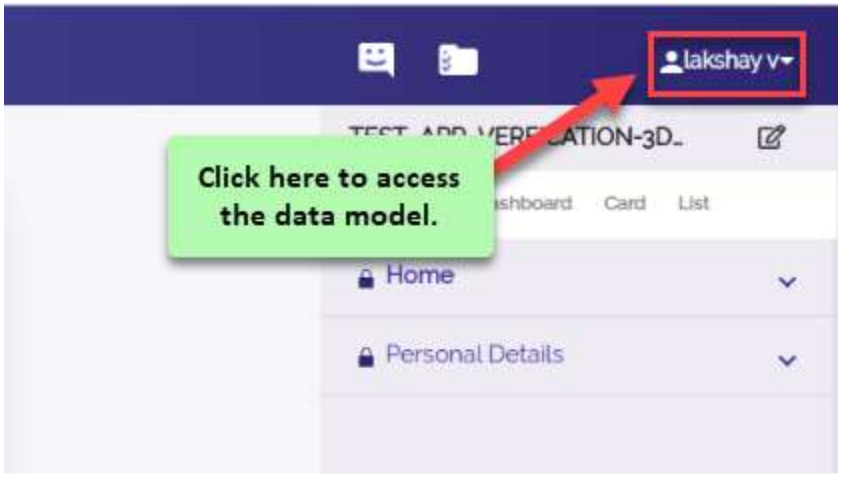
Conclusively, it can be said that the vDesigner application creates JSON data model on the mobile app level. This data model contains the configuration details of object/entity that can be tracked to debug the mobile app. **To track and monitor JSON data model in vDesigner, you must have proficiency to understand JSON syntax on the programming level**.

The advantage of vDesigner application is that it creates the JSON data model by using **de-normalized** approach. In the de-normalized approach, the data model manages the details of objects/entities across different nested blocks in a single document. To access the data of specific object, **these blocks can be expanded and collapsed**.

The following section explains how to access the JSON data model in the vDesigner application.

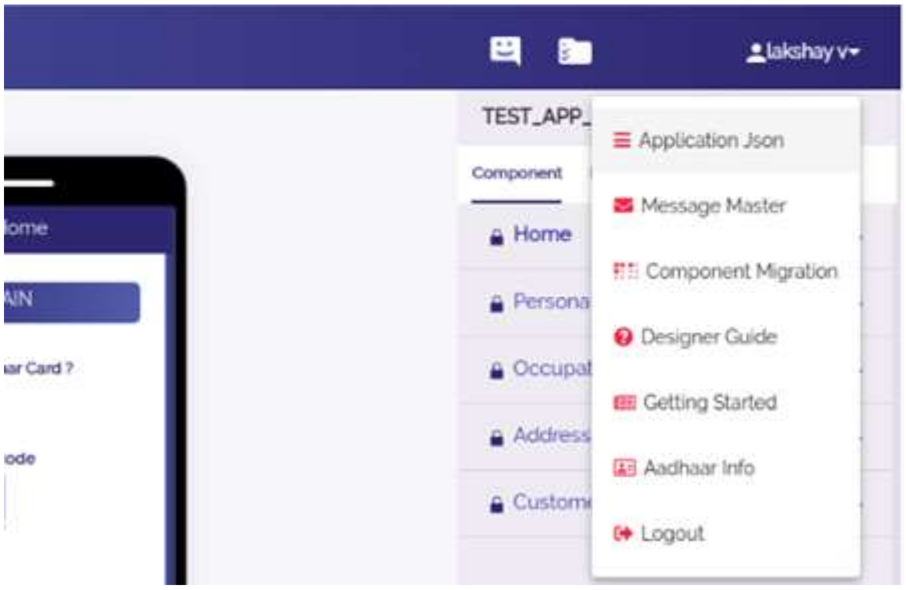
To access JSON data model in vDesigner:

1. In the top-right corner of the vDesigner dashboard, locate the user name **(Fig 3.7 (a))** of the application.


lakshay v
TEST_APP_VERIFICATION-3D_
Click here to access the data model.
Dashboard
Card
List
Home
Personal Details

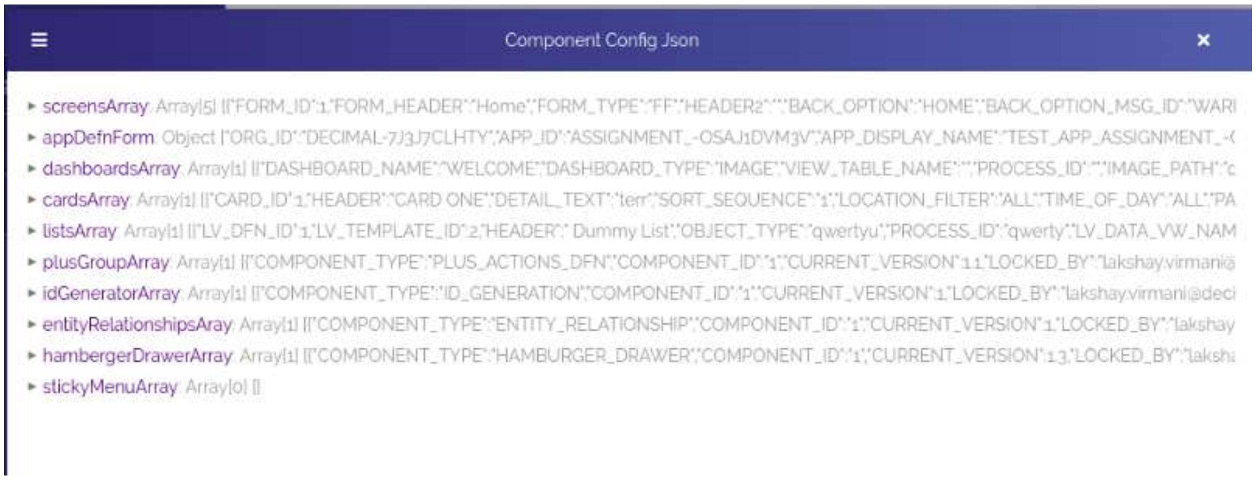
(Fig 3.7 (a))

2. Click the user name in the top-right corner, a toolbar **(Fig 3.7 (b))** opens.



(Fig 3.7 (b))

3. In the toolbar, select **Application Json**, the **Component Config Json** box **(Fig 3.7 (c))** opens.

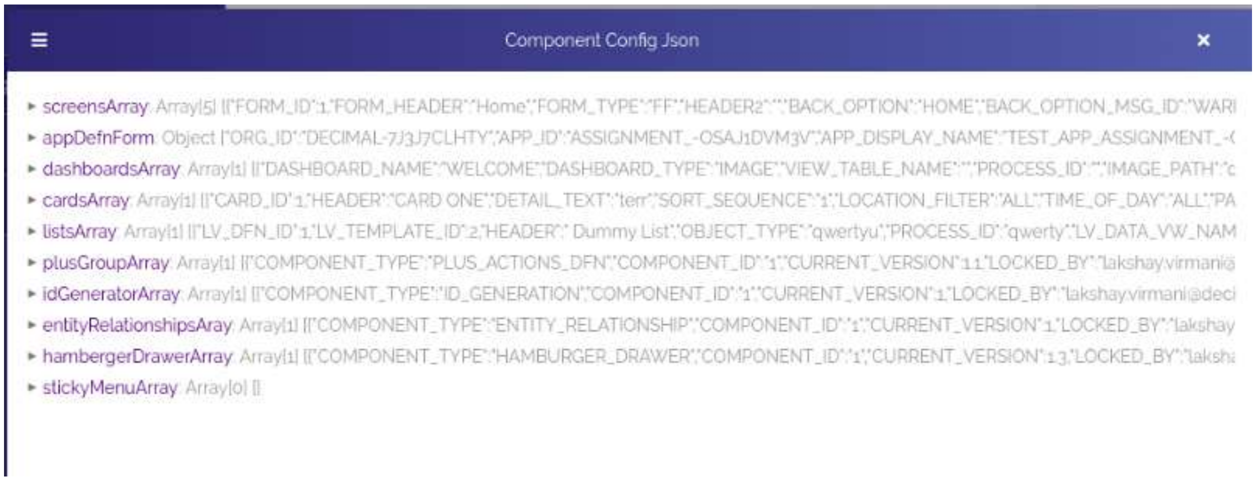


```
▸ screensArray: Array[5] [{"FORM_ID":1,"FORM_HEADER":"Home","FORM_TYPE":"FF","HEADER2":"","BACK_OPTION":"HOME","BACK_OPTION_MSG_ID":"WARI
▸ appDefnForm: Object {"ORG_ID":"DECIMAL-7J3J7CLHTY","APP_ID":"ASSIGNMENT_-OSAJ1DVM3V","APP_DISPLAY_NAME":"TEST_APP_ASSIGNMENT_-(
▸ dashboardsArray: Array[1] [{"DASHBOARD_NAME":"WELCOME","DASHBOARD_TYPE":"IMAGE","VIEW_TABLE_NAME":"","PROCESS_ID":"","IMAGE_PATH":"c
▸ cardsArray: Array[1] [{"CARD_ID":1,"HEADER":"CARD ONE","DETAIL_TEXT":"terr","SORT_SEQUENCE":"1","LOCATION_FILTER":"ALL","TIME_OF_DAY":"ALL","PA
▸ listsArray: Array[1] [{"LV_DFN_ID":1,"LV_TEMPLATE_ID":2,"HEADER":" Dummy List","OBJECT_TYPE":"qwertyu","PROCESS_ID":"qwerty","LV_DATA_VW_NAM
▸ plusGroupArray: Array[1] [{"COMPONENT_TYPE":"PLUS_ACTIONS_DFN","COMPONENT_ID":"1","CURRENT_VERSION":1.1,"LOCKED_BY":"lakshay.virmani@
▸ idGeneratorArray: Array[1] [{"COMPONENT_TYPE":"ID_GENERATION","COMPONENT_ID":"1","CURRENT_VERSION":1,"LOCKED_BY":"lakshay.virmani@deci
▸ entityRelationshipsAray: Array[1] [{"COMPONENT_TYPE":"ENTITY_RELATIONSHIP","COMPONENT_ID":"1","CURRENT_VERSION":1,"LOCKED_BY":"lakshay
▸ hambergerDrawerArray: Array[1] [{"COMPONENT_TYPE":"HAMBURGER_DRAWER","COMPONENT_ID":"1","CURRENT_VERSION":1.3,"LOCKED_BY":"laksh
▸ stickyMenuArray: Array[0] []
```

(Fig 3.7 (c))

4. The **Component Config Json** box displays the json data model that is created on the app level.
5. In the vDesigner application, the app level JSON data model can be interpreted as follows:

3.7.1 App Level Data Model



```
▸ screensArray: Array[5] [{"FORM_ID":1,"FORM_HEADER":"Home","FORM_TYPE":"FF","HEADER2":"","BACK_OPTION":"HOME","BACK_OPTION_MSG_ID":"WARI
▸ appDefnForm: Object {"ORG_ID":"DECIMAL-7J3J7CLHTY","APP_ID":"ASSIGNMENT_-OSAJ1DVM3V","APP_DISPLAY_NAME":"TEST_APP_ASSIGNMENT_-(
▸ dashboardsArray: Array[1] [{"DASHBOARD_NAME":"WELCOME","DASHBOARD_TYPE":"IMAGE","VIEW_TABLE_NAME":"","PROCESS_ID":"","IMAGE_PATH":"c
▸ cardsArray: Array[1] [{"CARD_ID":1,"HEADER":"CARD ONE","DETAIL_TEXT":"terr","SORT_SEQUENCE":"1","LOCATION_FILTER":"ALL","TIME_OF_DAY":"ALL","PA
▸ listsArray: Array[1] [{"LV_DFN_ID":1,"LV_TEMPLATE_ID":2,"HEADER":" Dummy List","OBJECT_TYPE":"qwertyu","PROCESS_ID":"qwerty","LV_DATA_VW_NAM
▸ plusGroupArray: Array[1] [{"COMPONENT_TYPE":"PLUS_ACTIONS_DFN","COMPONENT_ID":"1","CURRENT_VERSION":1.1,"LOCKED_BY":"lakshay.virmani@
▸ idGeneratorArray: Array[1] [{"COMPONENT_TYPE":"ID_GENERATION","COMPONENT_ID":"1","CURRENT_VERSION":1,"LOCKED_BY":"lakshay.virmani@deci
▸ entityRelationshipsAray: Array[1] [{"COMPONENT_TYPE":"ENTITY_RELATIONSHIP","COMPONENT_ID":"1","CURRENT_VERSION":1,"LOCKED_BY":"lakshay
▸ hambergerDrawerArray: Array[1] [{"COMPONENT_TYPE":"HAMBURGER_DRAWER","COMPONENT_ID":"1","CURRENT_VERSION":1.3,"LOCKED_BY":"laksh
▸ stickyMenuArray: Array[0] []
```

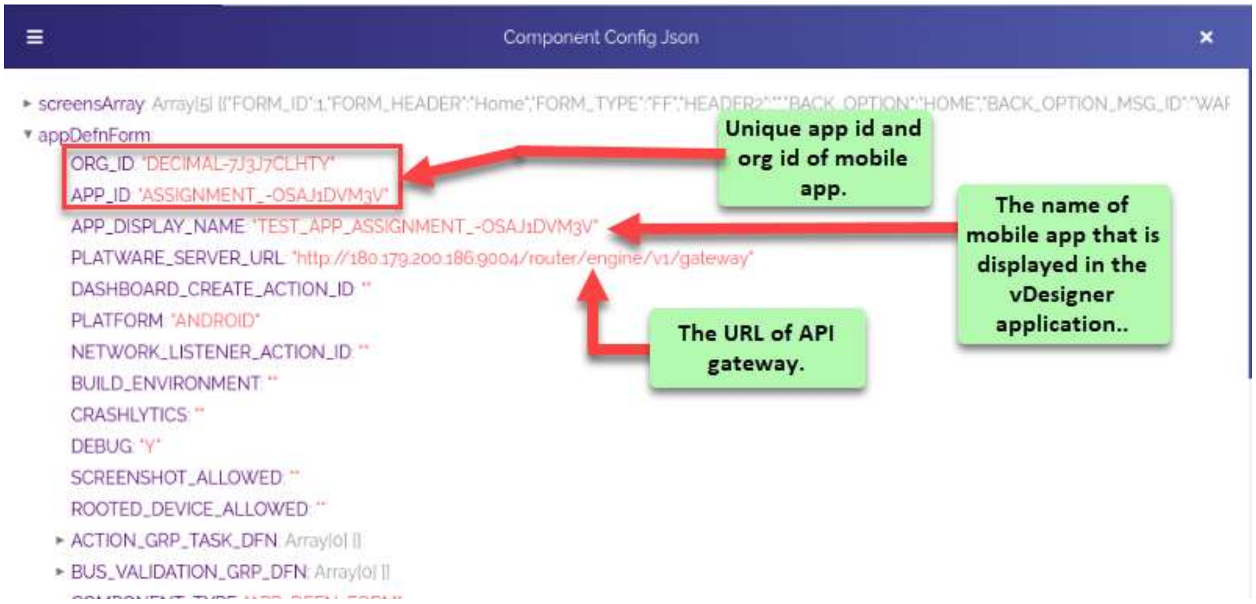
(Fig 3.7.1 (a))

If you look at the screen capture: **(Fig 3.7.1 (a))**, this is JSON based data model that the vDesigner application automatically creates as soon as application user performs the functions in the application. The screen displays the JSON data model at the app (mobile app) level and stores the details of different objects and entities in the array type JSON object.

The application user has created these objects/entities earlier in the vDesigner application. When user creates these entities such as screens/forms, dashboard, card, list, element, entities and others, the vDesigner application automatically creates the JSON data model and stores every details, including configuration details, property values of these entities in the JSON data model, and others.

For instance: - In the screen capture: the **screensArray** object contains exhaustive configuration details of all the forms/screens that the user has added in the mobile app.

Let's focus on app level data model first. To access mobile app level details in the screen: **(Fig 3.7.1 (a))**, click the **appDefnForm** array, it expands and displays the details **(Fig 3.7.1 (b))** of mobile app that you have created in the vDesigner application.


Component Config Json
Unique app id and org id of mobile app.
The name of mobile app that is displayed in the vDesigner application..
The URL of API gateway.

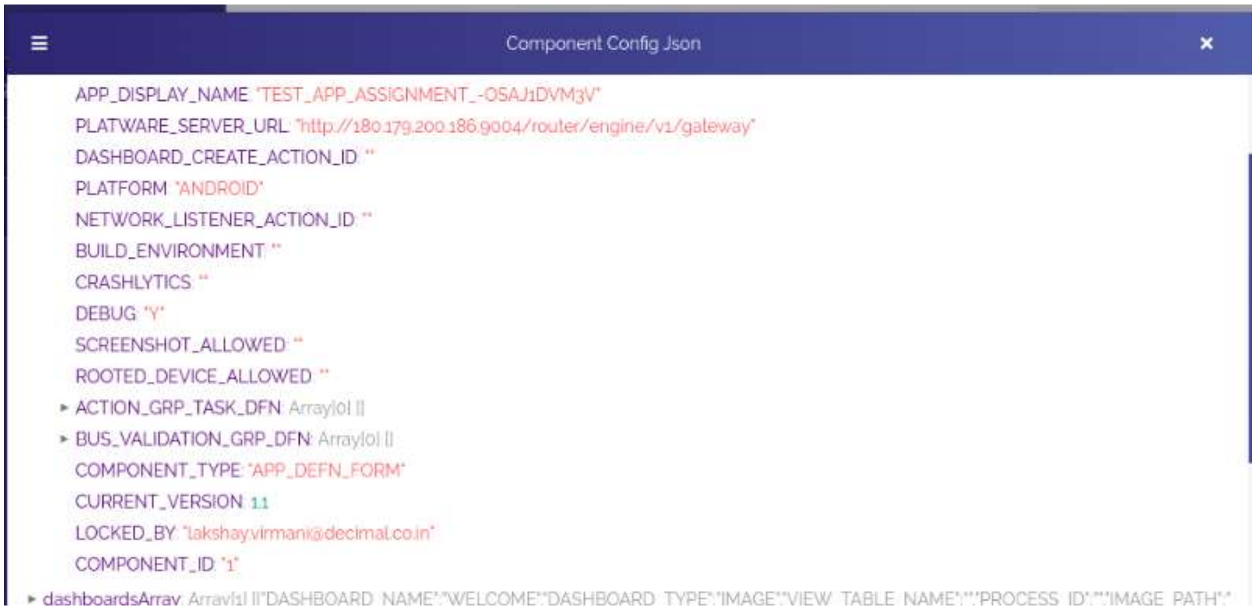
```
▸ screensArray: Array[5] [{"FORM_ID":1,"FORM_HEADER":"Home","FORM_TYPE":"FF","HEADER2":"","BACK_OPTION":"HOME","BACK_OPTION_MSG_ID":"WAI
▾ appDefnForm:
    ORG_ID: "DECIMAL-7J3J7CLHTY"
    APP_ID: "ASSIGNMENT_-OSAJ1DVM3V"
    APP_DISPLAY_NAME: "TEST_APP_ASSIGNMENT_-OSAJ1DVM3V"
    PLATWARE_SERVER_URL: "http://180.179.200.186:9004/router/engine/v1/gateway"
    DASHBOARD_CREATE_ACTION_ID: ""
    PLATFORM: "ANDROID"
    NETWORK_LISTENER_ACTION_ID: ""
    BUILD_ENVIRONMENT: ""
    CRASHLYTICS: ""
    DEBUG: "Y"
    SCREENSHOT_ALLOWED: ""
    ROOTED_DEVICE_ALLOWED: ""
  ▸ ACTION_GRP_TASK_DFN: Array[0] []
  ▸ BUS_VALIDATION_GRP_DFN: Array[0] []
```

(Fig 3.7.1 (b))

As the screen capture **(Fig 3.7.1 (b))** clearly depicts that the **appDefnForm** is the array type JSON object that stores the configuration details of the mobile app. To access the configuration details of the app, click the name of **appDefnForm** array. The array expands and displays the details of the mobile app. The **ORG_ID** and **APP_ID** attributes store the unique app id and org id respectively, which are created at the time you create a new agent on the Vahana platform.

The **PLATWARE_SERVER_URL** attribute contains the URL of the API gateway. As described earlier in the document, the API gateway is used to handle API(s) that are deployed on the Vahana platform. If you look at the **DEBUG** attribute, you will find that it stores the value: Y. The 'Y' value specifies that the debug feature for this app is active. With active debugging mode, you can debug the mobile app by using its JSON data model.

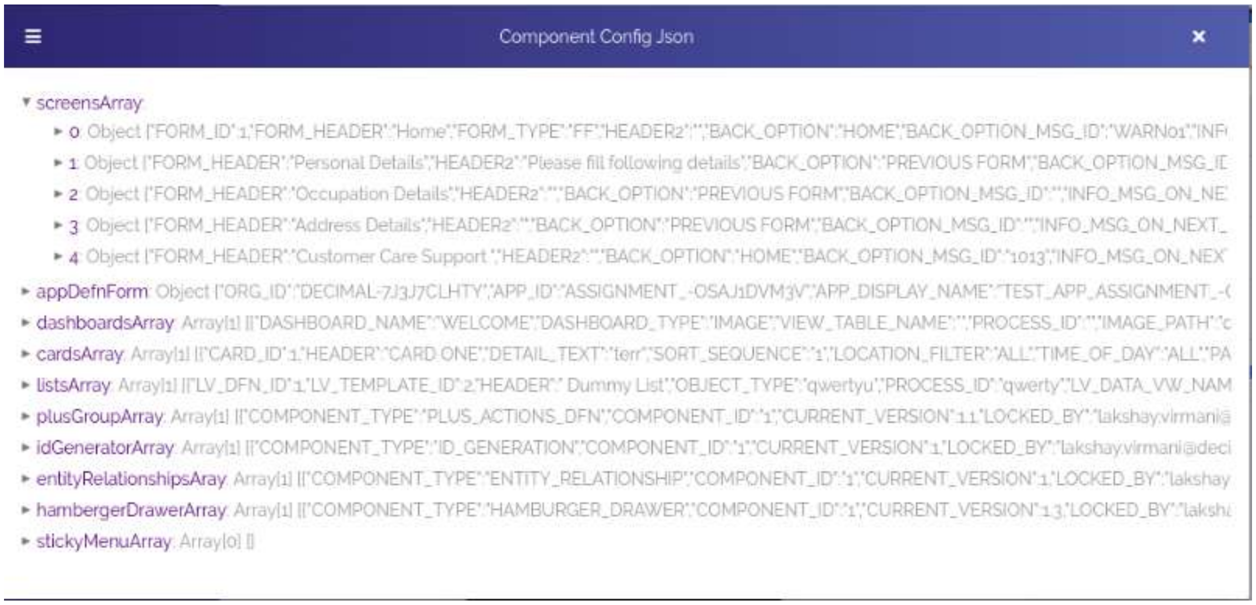
The following screen capture: **(Fig 3.7.1 (c))** also displays the details of mobile app. In the screen, the **COMPONENT_TYPE** attribute contains the type of entity. For each entity/component that you add or create in the vDesigner application, the **COMPONENT_TYPE** attribute stores the type of entity. In the following screen: **(Fig 3.7.1 (c))**, **COMPONENT_TYPE** attribute stores the value: APP_DEFN_FORM. This value specifies that the component type of the entity is the form where you can add dashboard, card, list, elements, and others.



```
    APP_DISPLAY_NAME: "TEST_APP_ASSIGNMENT_-O5AJ1DVM3V"
    PLATWARE_SERVER_URL: "http://180.179.200.186:9004/router/engine/v1/gateway"
    DASHBOARD_CREATE_ACTION_ID: ""
    PLATFORM: "ANDROID"
    NETWORK_LISTENER_ACTION_ID: ""
    BUILD_ENVIRONMENT: ""
    CRASHLYTICS: ""
    DEBUG: "Y"
    SCREENSHOT_ALLOWED: ""
    ROOTED_DEVICE_ALLOWED: ""
  ▸ ACTION_GRP_TASK_DFN: Array[0] []
  ▸ BUS_VALIDATION_GRP_DFN: Array[0] []
    COMPONENT_TYPE: "APP_DEFN_FORM"
    CURRENT_VERSION: 1.1
    LOCKED_BY: "lakshay.virmani@decimal.co.in"
    COMPONENT_ID: "1"
▸ dashboardsArray: Array[1] [{"DASHBOARD_NAME":"WELCOME","DASHBOARD_TYPE":"IMAGE","VIEW_TABLE_NAME":"","PROCESS_ID":"","IMAGE_PATH":"
```

(Fig 3.7.1 (c))

3.7.2 Screen Level Data Model



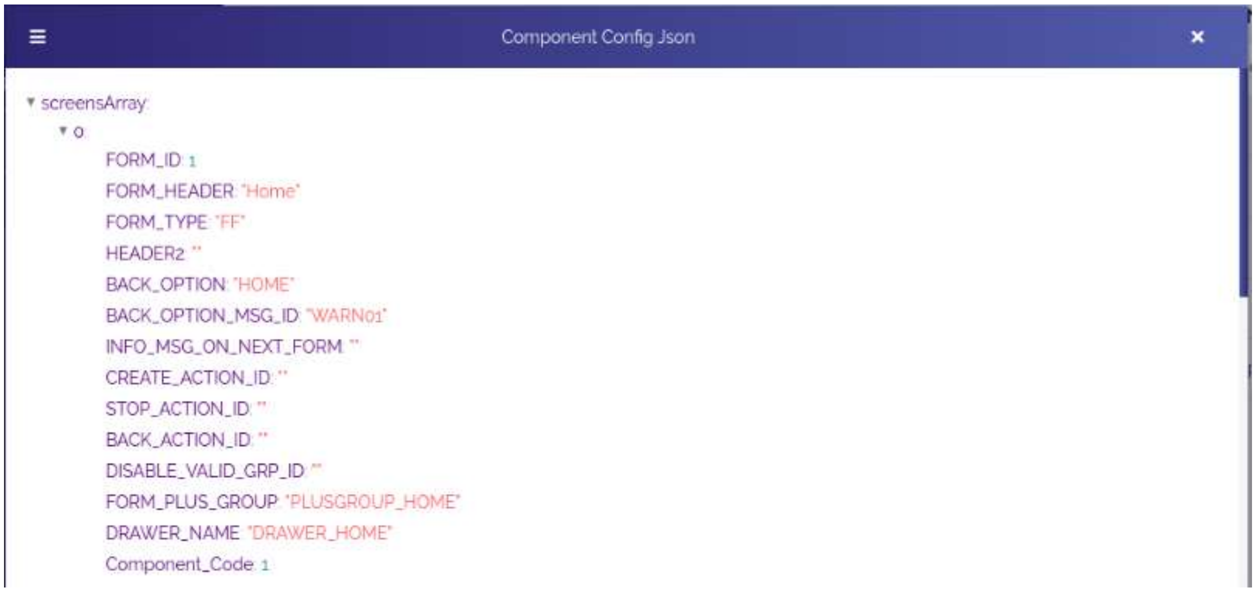
```
▾ screensArray
    ▸ 0 Object {"FORM_ID":1,"FORM_HEADER":"Home","FORM_TYPE":"FF","HEADER2":"","BACK_OPTION":"HOME","BACK_OPTION_MSG_ID":"WARN01","INF
    ▸ 1 Object {"FORM_HEADER":"Personal Details","HEADER2":"Please fill following details","BACK_OPTION":"PREVIOUS FORM","BACK_OPTION_MSG_ID
    ▸ 2 Object {"FORM_HEADER":"Occupation Details","HEADER2":"","BACK_OPTION":"PREVIOUS FORM","BACK_OPTION_MSG_ID":"","INFO_MSG_ON_NE
    ▸ 3 Object {"FORM_HEADER":"Address Details","HEADER2":"","BACK_OPTION":"PREVIOUS FORM","BACK_OPTION_MSG_ID":"","INFO_MSG_ON_NEXT_
    ▸ 4 Object {"FORM_HEADER":"Customer Care Support ","HEADER2":"","BACK_OPTION":"HOME","BACK_OPTION_MSG_ID":"1013","INFO_MSG_ON_NEX
▸ appDefnForm: Object {"ORG_ID":"DECIMAL-7J3J7CLHTY","APP_ID":"ASSIGNMENT_-OSAJ1DVM3V","APP_DISPLAY_NAME":"TEST_APP_ASSIGNMENT_-(
▸ dashboardsArray: Array[1] [{"DASHBOARD_NAME":"WELCOME","DASHBOARD_TYPE":"IMAGE","VIEW_TABLE_NAME":"","PROCESS_ID":"","IMAGE_PATH":"c
▸ cardsArray: Array[1] [{"CARD_ID":1,"HEADER":"CARD ONE","DETAIL_TEXT":"terr","SORT_SEQUENCE":"1","LOCATION_FILTER":"ALL","TIME_OF_DAY":"ALL","PA
▸ listsArray: Array[1] [{"LV_DFN_ID":1,"LV_TEMPLATE_ID":2,"HEADER":" Dummy List","OBJECT_TYPE":"qwertyu","PROCESS_ID":"qwerty","LV_DATA_VW_NAM
▸ plusGroupArray: Array[1] [{"COMPONENT_TYPE":"PLUS_ACTIONS_DFN","COMPONENT_ID":"1","CURRENT_VERSION":11,"LOCKED_BY":"lakshay.virmani@
▸ idGeneratorArray: Array[1] [{"COMPONENT_TYPE":"ID_GENERATION","COMPONENT_ID":"1","CURRENT_VERSION":1,"LOCKED_BY":"lakshay.virmani@deci
▸ entityRelationshipsAray: Array[1] [{"COMPONENT_TYPE":"ENTITY_RELATIONSHIP","COMPONENT_ID":"1","CURRENT_VERSION":1,"LOCKED_BY":"lakshay
▸ hambergerDrawerArray: Array[1] [{"COMPONENT_TYPE":"HAMBURGER_DRAWER","COMPONENT_ID":"1","CURRENT_VERSION":13,"LOCKED_BY":"laksh
▸ stickyMenuArray: Array[0] []
```

(Fig 3.7.2 (a))

The screen capture: **(Fig 3.7.2 (a))** displays the JSON data model at the screen level. If you click the name of the **screensArray** object, it expands and displays the configuration details of each screen that you have added in the mobile app. The objects: ***0 Object***, ***1 Object***, ***2 Object***…are array type JSON objects that store configuration details of the respective screen. The **FORM_HEADER** attribute contains the title name of screen, while if you look at the **HEADER2** attribute, you will find that it contains blank value, which means that the respective screen does not have any sub-title. The **BACK_OPTION** attribute contains the value that specifies the screen or page to which the user will navigate after s/he taps the back option. For instance: - The value: **PREVIOUS FORM** specifies that the user will navigate to the previous form or screen if s/he taps the back option.

Subsequently, to view screen-wise configuration details, you can click the name of the object array of specific screen (For example: - 0 Object), the 0 Object array expands and displays the configuration details **(Fig 3.7.2 (b))** of the respective form/screen.

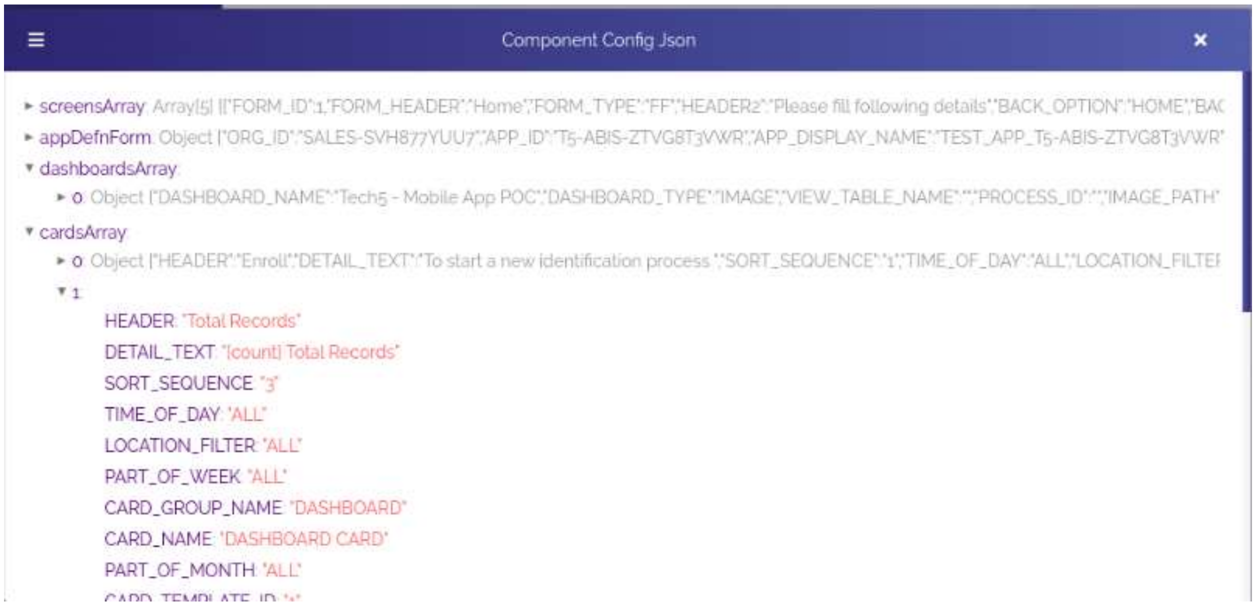
In the configuration details **(Fig 3.7.2 (b))** of the screen, the **FORM_HEADER** attribute contains the title of the screen. Again, the **BACK_OPTION** attribute contains the header name (title) of the screen where the user will navigate if s/he taps the back option on the mobile app. The **FORM_PLUS_GROUP** and **DRAWER_NAME** attributes respective store the name plus group and hamburger drawer that will be displayed on the current screen. Whereas, the **LOCKED_BY** attribute stores the user ID of the application user who has added/created the current screen.



```
▾ screensArray:
  ▾ 0:
      FORM_ID: 1
      FORM_HEADER: "Home"
      FORM_TYPE: "FF"
      HEADER2: ""
      BACK_OPTION: "HOME"
      BACK_OPTION_MSG_ID: "WARN01"
      INFO_MSG_ON_NEXT_FORM: ""
      CREATE_ACTION_ID: ""
      STOP_ACTION_ID: ""
      BACK_ACTION_ID: ""
      DISABLE_VALID_GRP_ID: ""
      FORM_PLUS_GROUP: "PLUSGROUP_HOME"
      DRAWER_NAME: "DRAWER_HOME"
      Component_Code: 1
```

(Fig 3.7.2 (b))

3.7.3 Dashboard Level Data Model



```
▸ screensArray: Array[5] [{"FORM_ID":1,"FORM_HEADER":"Home","FORM_TYPE":"FF","HEADER2":"Please fill following details","BACK_OPTION":"HOME","BA(
▸ appDefnForm: Object {"ORG_ID":"SALES-SVH877YUU7","APP_ID":"T5-ABIS-ZTVG8T3VWR","APP_DISPLAY_NAME":"TEST_APP_T5-ABIS-ZTVG8T3VWR"
▾ dashboardsArray:
    ▸ 0: Object {"DASHBOARD_NAME":"Tech5 - Mobile App POC","DASHBOARD_TYPE":"IMAGE","VIEW_TABLE_NAME":"","PROCESS_ID":"","IMAGE_PATH"
▾ cardsArray:
    ▸ 0: Object {"HEADER":"Enroll","DETAIL_TEXT":"To start a new identification process.","SORT_SEQUENCE":"1","TIME_OF_DAY":"ALL","LOCATION_FILTEI
    ▾ 1:
          HEADER: "Total Records"
          DETAIL_TEXT: "{count} Total Records"
          SORT_SEQUENCE: "3"
          TIME_OF_DAY: "ALL"
          LOCATION_FILTER: "ALL"
          PART_OF_WEEK: "ALL"
          CARD_GROUP_NAME: "DASHBOARD"
          CARD_NAME: "DASHBOARD CARD"
          PART_OF_MONTH: "ALL"
```

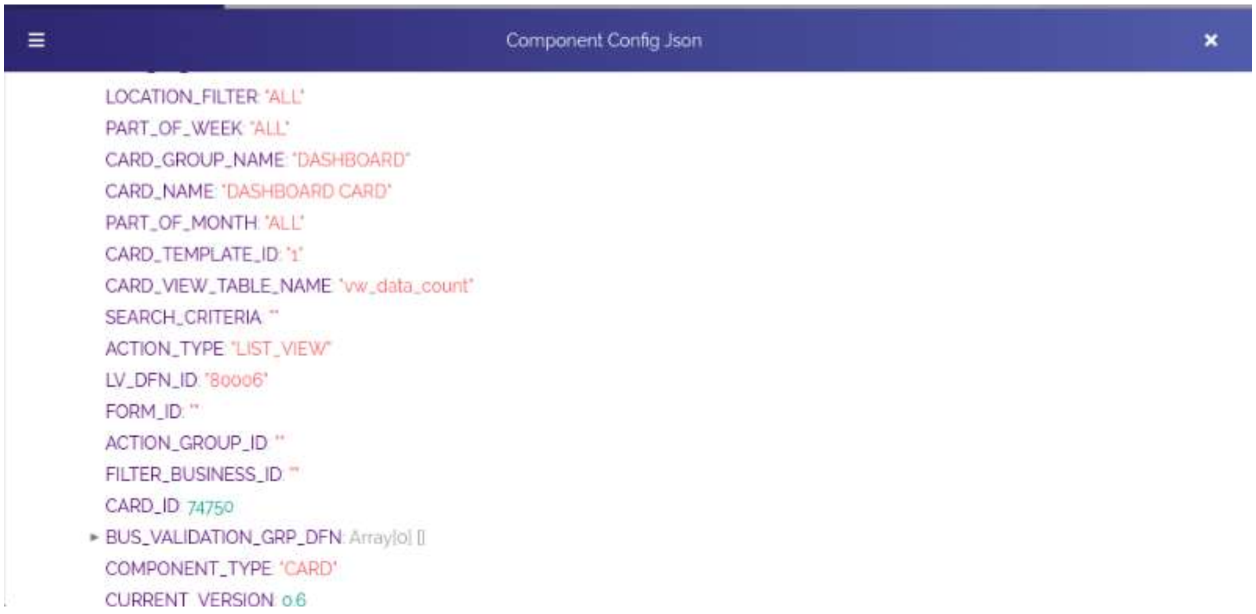
(Fig 3.7.3 (a))

In the data model document, as depicted in the screen capture: **(Fig 3.7.3 (a))**, the **dashboardsArray** object stores the configuration details of the mobile app dashboard. After you click the **dashboardsArray** object to expand it, it displays the configuration details of the currently added dashboard. Under dashboardsArray, only one array type JSON object: **0 Object** is available. It means that the mobile app has only single dashboard. The **0 Object** stores the details of the dashboard.

In the configuration details of the dashboard, the **DASHBOARD_NAME** attribute contains the header/title of the dashboard, while the value: **IMAGE** in the **DASHBOARD_TYPE** attribute specifies that the dashboard displays the image on the mobile app. To view all the details of dashboard, you can click the **0 Object** to expand it.

Another array type JSON object: **cardsArray** contains the configuration details of dashboard card. Under **cardsArray (Fig 3.7.3 (a))**, you can view two JSON type array objects: ***0 Object*** and ***1 Object***. Under the **cardsArray** object, the availability of these two objects specifies that the dashboard contains two different cards. As shown in the screen: **(Fig 3.7.3 (a))**, if you click the **1 Object**, it expands and then displays the configuration details of the respective card. In the details of card, the **HEADER** attribute stores the name/title of the card. The **SORT_SEQUENCE** attribute stores the sequence number/order number of the card. The sequence number of the card specifies the number on which the dashboard displays the respective.

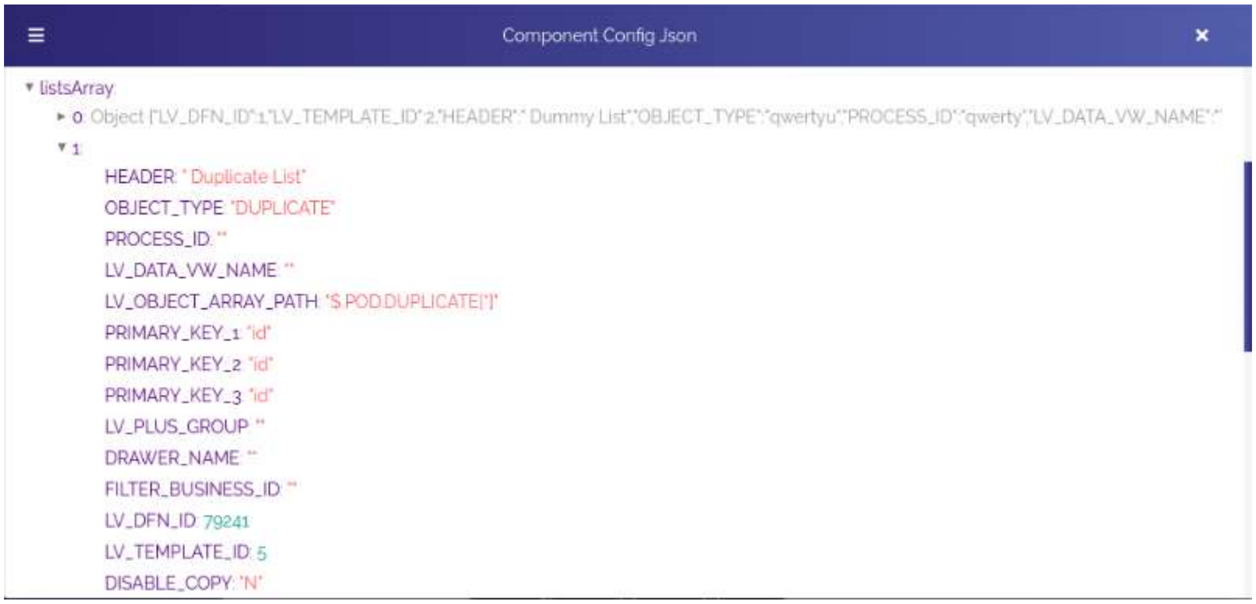
In the following screen: **(Fig 3.7.3 (b))**, the **CARD_GROUP_NAME** attribute stores the value: **DASHBOARD**, which means that the card is placed in the dashboard.



```
LOCATION_FILTER: 'ALL'
PART_OF_WEEK: 'ALL'
CARD_GROUP_NAME: "DASHBOARD"
CARD_NAME: 'DASHBOARD CARD'
PART_OF_MONTH: 'ALL'
CARD_TEMPLATE_ID: '1'
CARD_VIEW_TABLE_NAME: 'vw_data_count'
SEARCH_CRITERIA: ''
ACTION_TYPE: 'LIST_VIEW'
LV_DFN_ID: '80006'
FORM_ID: ''
ACTION_GROUP_ID: ''
FILTER_BUSINESS_ID: ''
CARD_ID: 74750
▸ BUS_VALIDATION_GRP_DFN: Array[0] []
COMPONENT_TYPE: 'CARD'
CURRENT_VERSION: 0.6
```

(Fig 3.7.3 (b))

In the **ACTION_TYPE** attribute **(Fig 3.7.3 (b))**, the value: **LIST_VIEW** specifies that the card displays the list after the user accesses the mobile app dashboard. The **CARD_ID** attribute **(Fig 3.7.3 (b))** stores unique ID of the card, while the **COMPONENT_TYPE** attribute stores the value: **CARD**; this value specifies that the entity/component is "card".



```
▾ listsArray
  ▸ 0: Object {"LV_DFN_ID":1,"LV_TEMPLATE_ID":2,"HEADER":" Dummy List","OBJECT_TYPE":"qwertyu","PROCESS_ID":"qwerty","LV_DATA_VW_NAME":"
  ▾ 1:
      HEADER: " Duplicate List"
      OBJECT_TYPE: 'DUPLICATE'
      PROCESS_ID: ''
      LV_DATA_VW_NAME: ''
      LV_OBJECT_ARRAY_PATH: '$.POD.DUPLICATE[*]'
      PRIMARY_KEY_1: 'id'
      PRIMARY_KEY_2: 'id'
      PRIMARY_KEY_3: 'id'
      LV_PLUS_GROUP: ''
      DRAWER_NAME: ''
      FILTER_BUSINESS_ID: ''
      LV_DFN_ID: 79241
      LV_TEMPLATE_ID: 5
      DISABLE_COPY: 'N'
```

(Fig 3.7.3 (c))

In the screen capture: **(Fig 3.7.3 (c))**, **listsArray** is array type JSON object. If you click the **listsArray** object, it expands and displays the configuration details of all lists that the user has applied on the dashboard or elsewhere. In the screen **(Fig 3.7.3 (c))**, you can view two other array type JSON objects: ***0 Object*** and ***1 Object***. These objects store the configuration details of two different lists. Therefore, if you click the **1 Object** array, it expands and then displays the configuration details of the respective list.

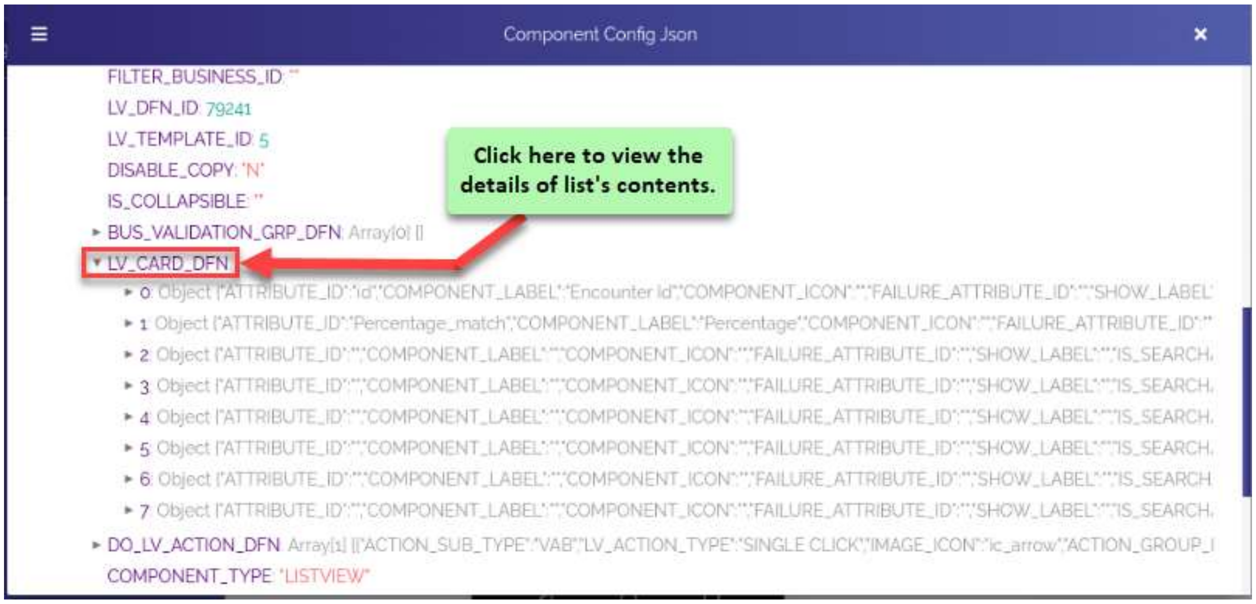
In the details of list, the **HEADER** attribute stores the name or title of the list. The **OBJECT_TYPE** attribute stores the name of object where the data is stored. This object can be a **JSON object** or **database object**. In the **OBJECT_TYPE** attribute **(Fig 3.7.3 (c))**, the value: **DUPLICATE** specifies that the object is the JSON type object because **LV_OBJECT_ARRAY_PATH** attribute contains the path of JSON type array. The path of JSON type array is: **$.POD.DUPLICATE[*]**.

If the **LV_OBJECT_ARRAY_PATH** attribute contains the blank value, it means that the user chooses the database view to display the data in the list. If the application user chooses the database view, the **OBJECT_TYPE** attribute will store the database object. The **PRIMARY_KEY_1** and **PRIMARY_KEY_2**, and **PRIMARY_KEY_3** objects contain any of the following values:

- ➢ **The name of variable that is declared when the application user creates an entity to hold the data.**
- ➢ **The name of primary kay of the database view.**

The database view is used to fetch the data from the database source and then display it in the list.

Subsequently, while exploring JSON data model, you can access and view configuration details of list's content.

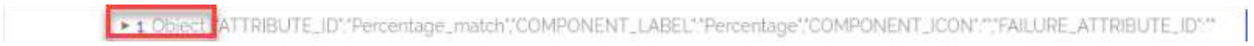


```
    FILTER_BUSINESS_ID: ""
    LV_DFN_ID: 79241
    LV_TEMPLATE_ID: 5
    DISABLE_COPY: 'N'
    IS_COLLAPSIBLE: ""
  ▸ BUS_VALIDATION_GRP_DFN: Array[0] []
  ▾ LV_CARD_DFN
    ▸ 0: Object {"ATTRIBUTE_ID":"id","COMPONENT_LABEL":"Encounter Id","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"","SHOW_LABEL"
    ▸ 1: Object {"ATTRIBUTE_ID":"Percentage_match","COMPONENT_LABEL":"Percentage","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"
    ▸ 2: Object {"ATTRIBUTE_ID":"","COMPONENT_LABEL":"","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"","SHOW_LABEL":"","IS_SEARCH
    ▸ 3: Object {"ATTRIBUTE_ID":"","COMPONENT_LABEL":"","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"","SHOW_LABEL":"","IS_SEARCH
    ▸ 4: Object {"ATTRIBUTE_ID":"","COMPONENT_LABEL":"","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"","SHOW_LABEL":"","IS_SEARCH
    ▸ 5: Object {"ATTRIBUTE_ID":"","COMPONENT_LABEL":"","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"","SHOW_LABEL":"","IS_SEARCH
    ▸ 6: Object {"ATTRIBUTE_ID":"","COMPONENT_LABEL":"","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"","SHOW_LABEL":"","IS_SEARCH
    ▸ 7: Object {"ATTRIBUTE_ID":"","COMPONENT_LABEL":"","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":"","SHOW_LABEL":"","IS_SEARCH
  ▸ DO_LV_ACTION_DFN: Array[1] [{"ACTION_SUB_TYPE":"VAB","LV_ACTION_TYPE":"SINGLE CLICK","IMAGE_ICON":"ic_arrow","ACTION_GROUP_I
    COMPONENT_TYPE: "LISTVIEW"
```

(Fig 3.7.3 (d))

Under the **listsArray** object, locate the **LV_CARD_DFN** object **(Fig 3.7.3 (d))**. After you click the **LV_CARD_DFN** object, it expands and then displays configuration details of list's content. Under **LV_CARD_DFN** object, **0 Object**, **1 Object**, and such other objects are array type JSON objects that store the details of different attributes. VDesigner application user defines these attributes when s/he configures the respective list. These attributes hold data to display it in the list.

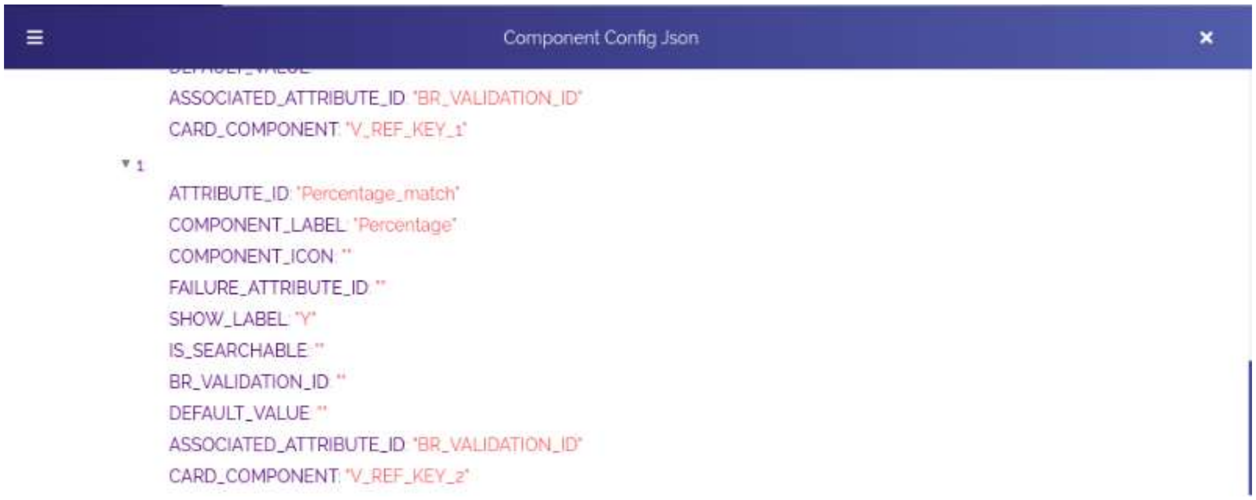
To access and view the details of specific list's attribute, click the array (For example: - 1 Object) **(Fig 3.7.3 (e))** of the attribute.



▸ 1 Object ATTRIBUTE_ID:"Percentage_match","COMPONENT_LABEL":"Percentage","COMPONENT_ICON":"","FAILURE_ATTRIBUTE_ID":""

(Fig 3.7.3 (e))

After you click the name of array (For example: - 1 Object), the array expands and then displays the details **(Fig 3.7.3 (f))** of attribute.



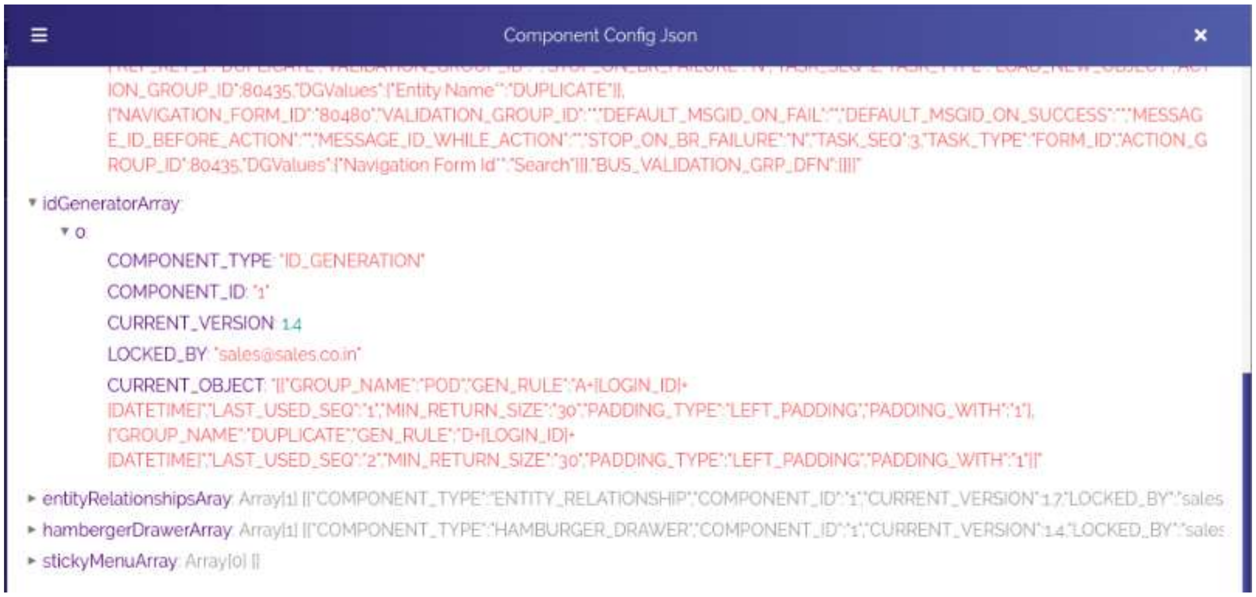
```
     ASSOCIATED_ATTRIBUTE_ID: "BR_VALIDATION_ID"
     CARD_COMPONENT: "V_REF_KEY_1"
▾ 1
     ATTRIBUTE_ID: "Percentage_match"
     COMPONENT_LABEL: "Percentage"
     COMPONENT_ICON: ""
     FAILURE_ATTRIBUTE_ID: ""
     SHOW_LABEL: "Y"
     IS_SEARCHABLE: ""
     BR_VALIDATION_ID: ""
     DEFAULT_VALUE: ""
     ASSOCIATED_ATTRIBUTE_ID: "BR_VALIDATION_ID"
     CARD_COMPONENT: "V_REF_KEY_2"
```

(Fig 3.7.3 (f))

The screen capture: **(Fig 3.7.3 (f))** displays the configuration details of the **Percentage_match** attribute. In the details of attribute, the **ATTRIBUTE_ID** key/attribute contains the name of attribute, **Percentage_match**. The **COMPONENT_LABEL** attribute stores the name of label under which the value is displayed.

After the data is fetched, the **Percentage_match** attribute holds the respective data value and then displays it either right to or under the **Percentage** label.

3.7.4 Entity Level Data Model



```
ION_GROUP_ID":80435,"DGValues":{"Entity Name":"DUPLICATE"}},
{"NAVIGATION_FORM_ID":"80480","VALIDATION_GROUP_ID":"","DEFAULT_MSGID_ON_FAIL":"","DEFAULT_MSGID_ON_SUCCESS":"","MESSAG
E_ID_BEFORE_ACTION":"","MESSAGE_ID_WHILE_ACTION":"","STOP_ON_BR_FAILURE":"N","TASK_SEQ":3,"TASK_TYPE":"FORM_ID","ACTION_G
ROUP_ID":80435,"DGValues":{"Navigation Form Id":"Search"}}],"BUS_VALIDATION_GRP_DFN":[]}]"
▾ idGeneratorArray:
  ▾ 0:
      COMPONENT_TYPE: "ID_GENERATION"
      COMPONENT_ID: "1"
      CURRENT_VERSION: 14
      LOCKED_BY: "sales@sales.co.in"
      CURRENT_OBJECT: "[{"GROUP_NAME":"POD","GEN_RULE":"A+[LOGIN_ID]+
      [DATETIME]","LAST_USED_SEQ":"1","MIN_RETURN_SIZE":"30","PADDING_TYPE":"LEFT_PADDING","PADDING_WITH":"1"},
      {"GROUP_NAME":"DUPLICATE","GEN_RULE":"D+[LOGIN_ID]+
      [DATETIME]","LAST_USED_SEQ":"2","MIN_RETURN_SIZE":"30","PADDING_TYPE":"LEFT_PADDING","PADDING_WITH":"1"}]"
▸ entityRelationshipsAray: Array[1] [{"COMPONENT_TYPE":"ENTITY_RELATIONSHIP","COMPONENT_ID":"1","CURRENT_VERSION":17,"LOCKED_BY":"sales
▸ hambergerDrawerArray: Array[1] [{"COMPONENT_TYPE":"HAMBURGER_DRAWER","COMPONENT_ID":"1","CURRENT_VERSION":14,"LOCKED_BY":"sales
▸ stickyMenuArray: Array[0] []
```

(Fig 3.7.4 (a))

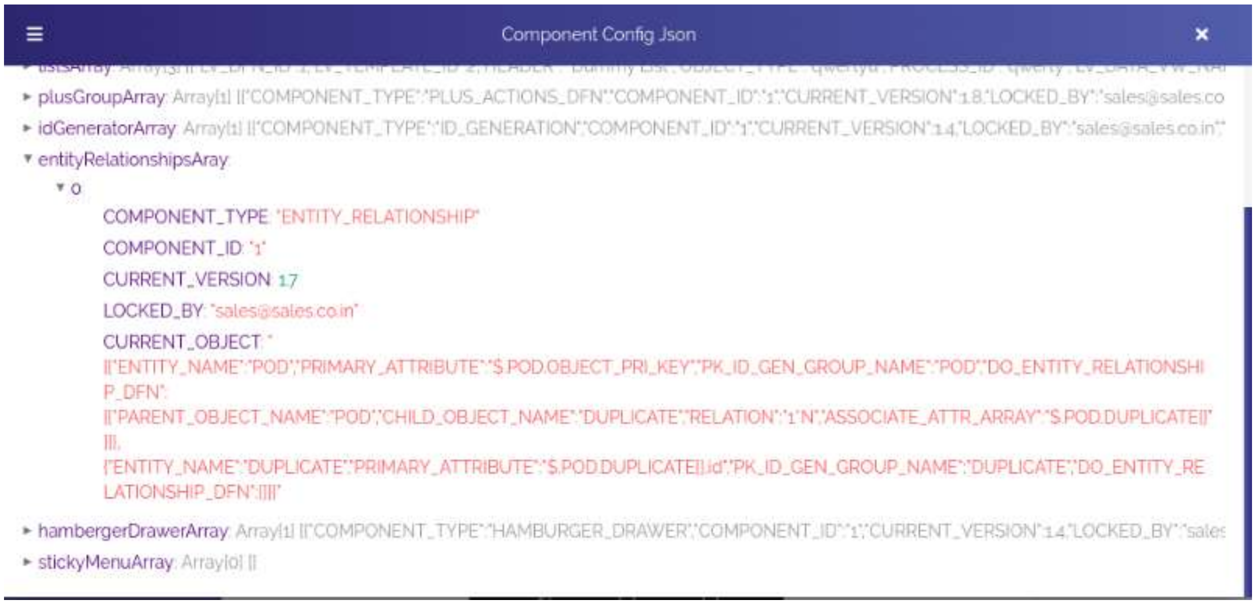
The screen **(Fig 3.7.4 (a))** displays the data model of the ID Generator. In vDesigner, the ID Generator module allows you to configure the new rule that creates a unique ID after the **Load New Object** task executes. This ID is commonly used to track the journey of lead, view the communication logs that are created as a result of lead's journey, etc.

To access and view the data model of the ID Generator module, you need to click the **idGeneratorArray** object, it expands and then displays the details of ID generator rule that the vDesigner application user configures to create unique ID. In the details of data model, the **COMPONENT_TYPE** attribute stores the type of component. In the screen **(Fig 3.7.4 (a))**, the value that is stored in the **COMPONENT_TYPE** attribute specifies that the type of component is ID Generator.

In the data model **(Fig 3.7.4 (a))**, the **CURRENT_OBJECT** attribute stores the configuration details of ID generator rule. In the configuration details, the **GROUP_NAME** attribute (For example: - POD) contains the name of entity for which the ID rule is configured. The **GEN_RULE** attribute stores the formula that is configured to create the unique ID. In the **GEN_RULE** attribute, the formula: A+[LOGIN_ID]+[DATETIME] creates a unique ID after the **LOAD NEW OBJECT** task executes as a result of user action. The **PADDING_TYPE** attribute stores the hardcoded value: LEFT, which means that the ID generator rule will start to create the unique ID from the left side.

Another array type JSON object: **entityRelationshipsAray** contains the details of different entities and the relationships between them. After you click the **entityRelationshipsAray** object, it expands and then displays the configuration details of entities and the relationships between them. The screen capture **(Fig 3.7.4 (b))** displays:

- **The details of different entities, including parent and child entities**
- **The details of relationships that are established between parent entity and child entity**



```
▸ plusGroupArray: Array[1] [{"COMPONENT_TYPE":"PLUS_ACTIONS_DFN","COMPONENT_ID":"1","CURRENT_VERSION":1.8,"LOCKED_BY":"sales@sales.co
▸ idGeneratorArray: Array[1] [{"COMPONENT_TYPE":"ID_GENERATION","COMPONENT_ID":"1","CURRENT_VERSION":1.4,"LOCKED_BY":"sales@sales.co.in",
▾ entityRelationshipsAray:
    ▾ 0:
        COMPONENT_TYPE: "ENTITY_RELATIONSHIP"
        COMPONENT_ID: "1"
        CURRENT_VERSION: 1.7
        LOCKED_BY: "sales@sales.co.in"
        CURRENT_OBJECT: "
        [{"ENTITY_NAME":"POD","PRIMARY_ATTRIBUTE":"$.POD.OBJECT_PRI_KEY","PK_ID_GEN_GROUP_NAME":"POD","DO_ENTITY_RELATIONSHI
        P_DFN":
        [{"PARENT_OBJECT_NAME":"POD","CHILD_OBJECT_NAME":"DUPLICATE","RELATION":"1*N","ASSOCIATE_ATTR_ARRAY":"$.POD.DUPLICATE[]"
        }]},
        {"ENTITY_NAME":"DUPLICATE","PRIMARY_ATTRIBUTE":"$.POD.DUPLICATE[].id","PK_ID_GEN_GROUP_NAME":"DUPLICATE","DO_ENTITY_RE
        LATIONSHIP_DFN":[]}]"
▸ hambergerDrawerArray: Array[1] [{"COMPONENT_TYPE":"HAMBURGER_DRAWER","COMPONENT_ID":"1","CURRENT_VERSION":1.4,"LOCKED_BY":"sales
▸ stickyMenuArray: Array[0] []
```

(Fig 3.7.4 (b))

The **CURRENT_OBJECT** attribute stores the configuration details of ***parent and child entities*** and ***relationship between parent and child entities***. In the configuration details, the **Entity_name** attribute stores the name of entity. The **PRIMARY_ATTRIBUTE** attribute/key stores the path of array where the data is stored. The path of array (For example: - $.POD or $.POD.DUPLICATE[[].ID) contains the name of JSON object (For example: - POD, DUPLICATE). In the path of array, **POD** and **DUPLICATE** are declared as array type JSON objects that store data record consisting different data values. While defining the path of array, you can declare the "ID" variable, which stores the value of unique ID that is created when the **LOAD NEW OBJECT** task executes as a result of user action.

Under **CURRENT_OBJECT**, the **PARENT_OBJECT_NAME** attribute stores the name of the parent object, while the **CHILD_OBJECT_NAME** attribute stores the name of the child object. The **RELATION** attribute stores the value: 1*N. This value denotes one-to-many relationship between parent and child object.

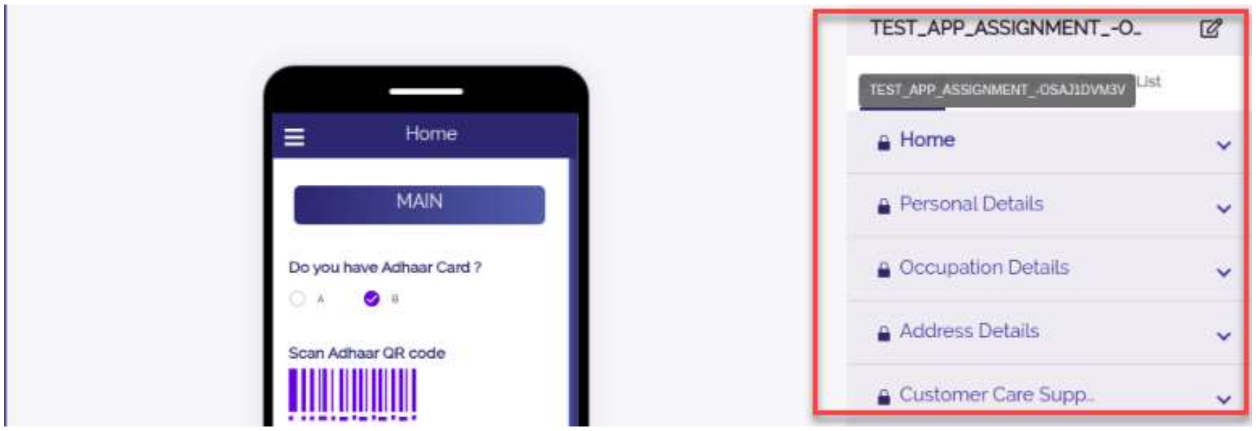
In the configuration details of objects' relationship, the **ASSOCIATE_ATTR_ARRAY** attribute stores the path of array: **$.POD.DUPLICATE[]**. In the **$.POD.DUPLICATE[]** array path, the **POD** object refers to another object: **DUPLICATE[]**. This reference scheme specifies that **POD** has been defined as a parent object to the **DUPLICATE[]** array.

3.7.5 Turning On Debug Mode

The vDesigner application allows you to use the debug feature to debug the mobile app on the functional level. To use the debug feature, you need to make it active. You can make the debug mode active as follow:

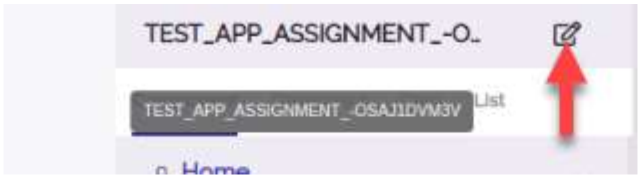
To make the debug feature active:

1. On the vDesigner dashboard, locate the right-most navigation pane **(Fig 3.7.5 (a))**.



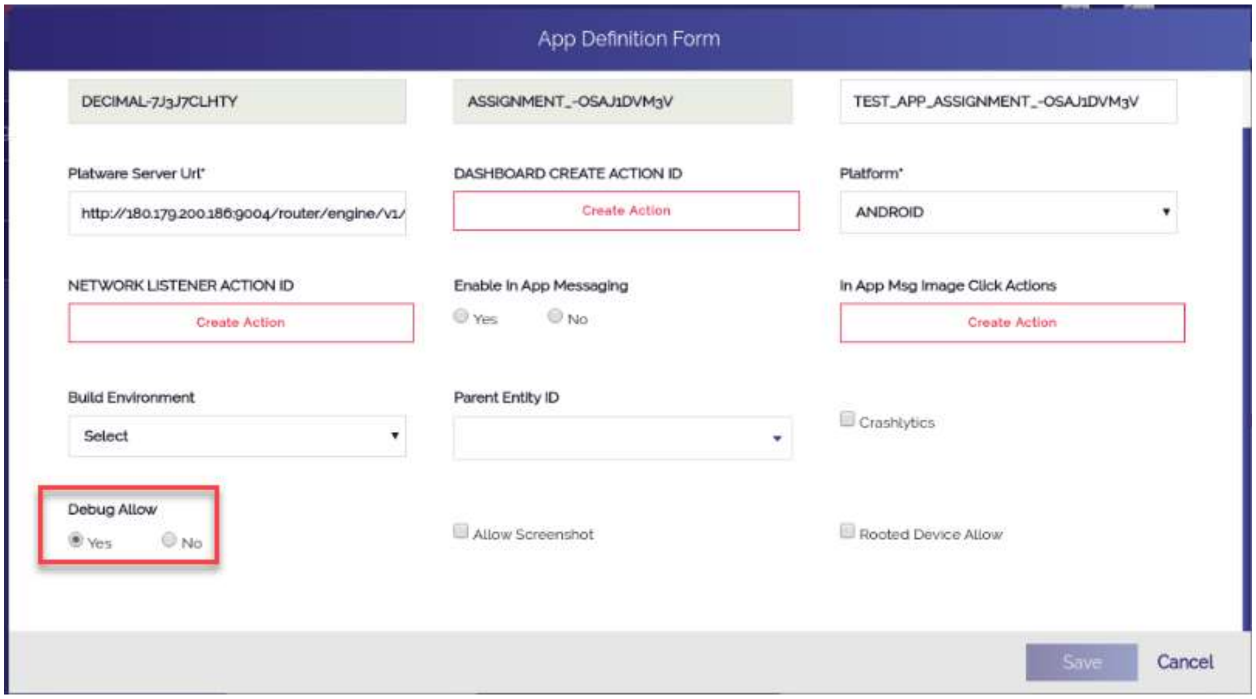
(Fig 3.7.5 (a))

2. In the right-most navigation pane, locate the name (For example: - Test_app_assignment_-O) of mobile app.
3. After you locate the name of mobile app, locate the edit icon (✎) **(Fig 3.7.5 (b))** right to the name of the mobile app.



(Fig 3.7.5 (b))

4. Click the edit icon (✎), the **App Definition Form** dialog box **(Fig 3.7.5 (c))** opens.



DECIMAL-7J3J7CLHTY
ASSIGNMENT_-OSAJ1DVM3V
TEST_APP_ASSIGNMENT_-OSAJ1DVM3V

Platware Server Url*: http://180.179.200.186:9004/router/engine/v1/
DASHBOARD CREATE ACTION ID: Create Action
Platform*: ANDROID
NETWORK LISTENER ACTION ID: Create Action
Enable In App Messaging: Yes No
In App Msg Image Click Actions: Create Action
Build Environment: Select
Parent Entity ID:
Crashlytics
Debug Allow: [✓ Yes] No
Allow Screenshot
Rooted Device Allow

(Fig 3.7.5 (c))

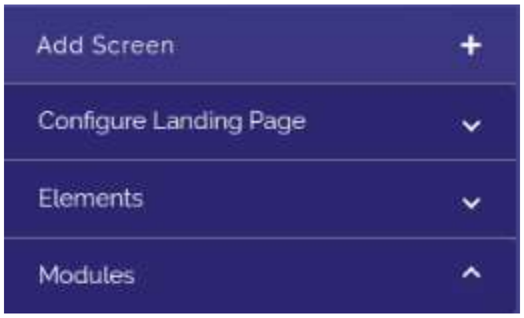
5. In the **App Definition Form** dialog box, locate the **Debug Allow** mutually exclusive options **(Fig 3.7.5 (c))**.
6. Under **Debug Allow**, click **Yes** and then click **Save**, the debug feature becomes active.

3.8 Configuring Plus Group

In the vDesigner application, the Plus Group module allows you to collectively place group of features/functions in single menu bar. This menu bar expands when you tap the plus group icon. Plus group can be placed on different screens/pages. Also you can design more than one plus group if you want to consolidate features module-wise. Commonly plus group includes most frequently used features that the user can easily access by expanding the menu bar of plus group.

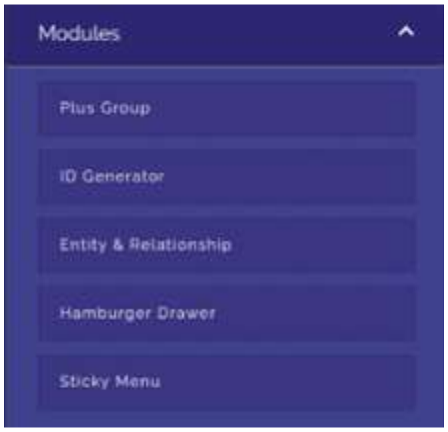
To configure the plus group:

1. On the vDesigner dashboard, locate the left-most navigation pane **(Fig 3.8 (b))**.



(Fig 3.8 (a))

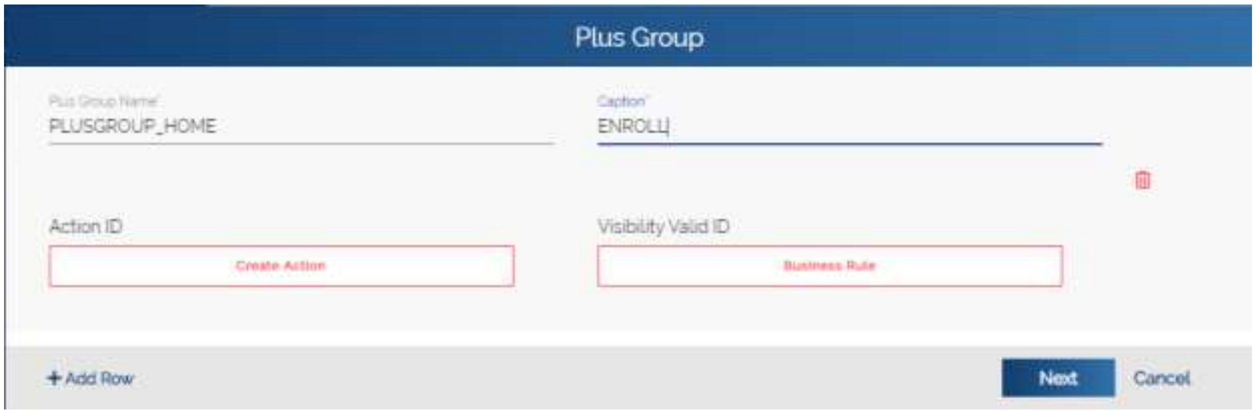
2. In the left navigation pane, click **Module**, the navigation pane **(Fig 3.8 (b))** expands.



(Fig 3.8 (b))

3. Under **Module**, click **Plus Group (Fig 3.8 (b))**, the **Plus Group** dialog box **(Fig 3.8 (c))** opens.

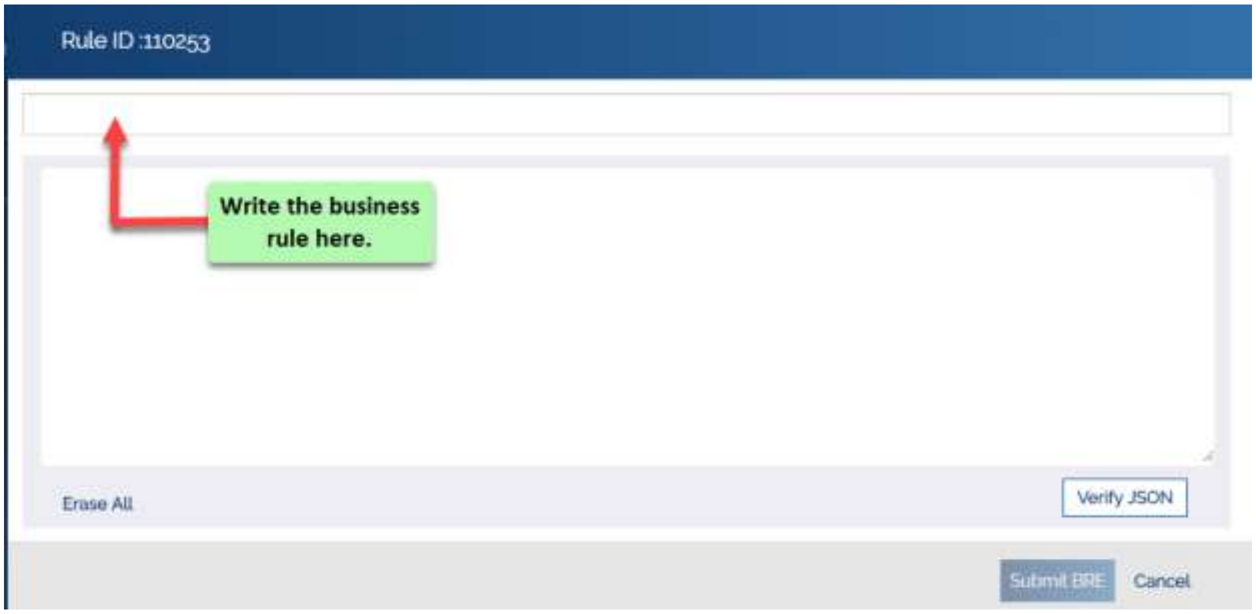
Note:-
vDesigner by default provides you a plus group with the title/name: **PLUSGROUP_HOME**. You can decisively perform any of two tasks. Either you can use the in-built plus group and add new feature to it, or you can create a new plus group and then add features to it.



(Fig 3.8 (c))

4. In the **Plus Group** dialog box, add new features/menus as described in the following steps:
5. To add new features/menus in the plus group, perform the function as follows:

Field/Box	**Description**
Plus Group Name	In this field, you can perform any of the following functions: **Case1**:- (If you want to use the existing plus group) If you want to use the existing plus group, do not enter any value in this box. Let it be the name of existing plus group: **PLUSGROUP_HOME**. **Case2**:- (If you want to create a new plus group) To create a new plus group, enter the name of new plus group (For example: - PLUSGROUP_INSURANCESCHEME).
Caption	In the box, enter the name of feature or menu (For example: - Enroll) that you want to add to the plus group.
Action ID	In the **Plus Group** dialog box, this function is used to assign an action to the feature/menu. The action is assigned to the feature/menu based on the functional requirement of the respective menu. To assign the action to a plus group item, visit the heading section: Assigning Action to Plus Group.
Validity Business Rule	You can use this function to apply the business rule for the visibility of specific plus group menu item. To apply the business rule: 1. Under **Visibility Valid ID**, click **Business Rule**, the **Rule ID** dialog box opens. 2. In the **Rule ID** dialog box **(Fig 3.8 (d))**, you can write the business rule in the upper box. 3. After you write the business rule, click **Submit BRE**, the business rule is applied to the plus group menu item.

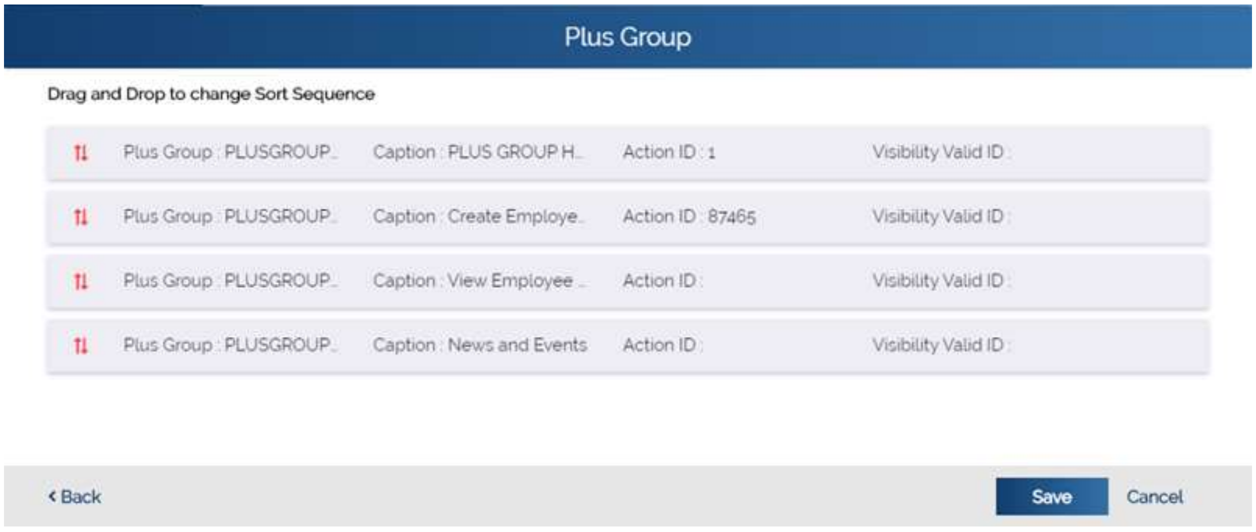


(Fig 3.8 (d))

6. After you enter the details of first plus group item as described in the last table, click **Add Row (Fig 3.8 (c))**, the following fields reappear in the next row.

➢ **Plus Group Name**
➢ **Caption**
➢ **Action ID**
➢ **Visibility Valid ID**

7. In these fields, enter the details of new plus group item as described in the last table.
8. After you add all plus group items, click **Next (Fig 3.8 (c))**, the **Plus Group** dialog box **(Fig 3.8 (e))** displays the details of all added plus group items.



Plus Group : PLUSGROUP...	Caption : PLUS GROUP H...	Action ID : 1	Visibility Valid ID :
Plus Group : PLUSGROUP...	Caption : Create Employe...	Action ID : 87465	Visibility Valid ID :
Plus Group : PLUSGROUP...	Caption : View Employee ...	Action ID :	Visibility Valid ID :
Plus Group : PLUSGROUP...	Caption : News and Events	Action ID :	Visibility Valid ID :

(Fig 3.8 (e))

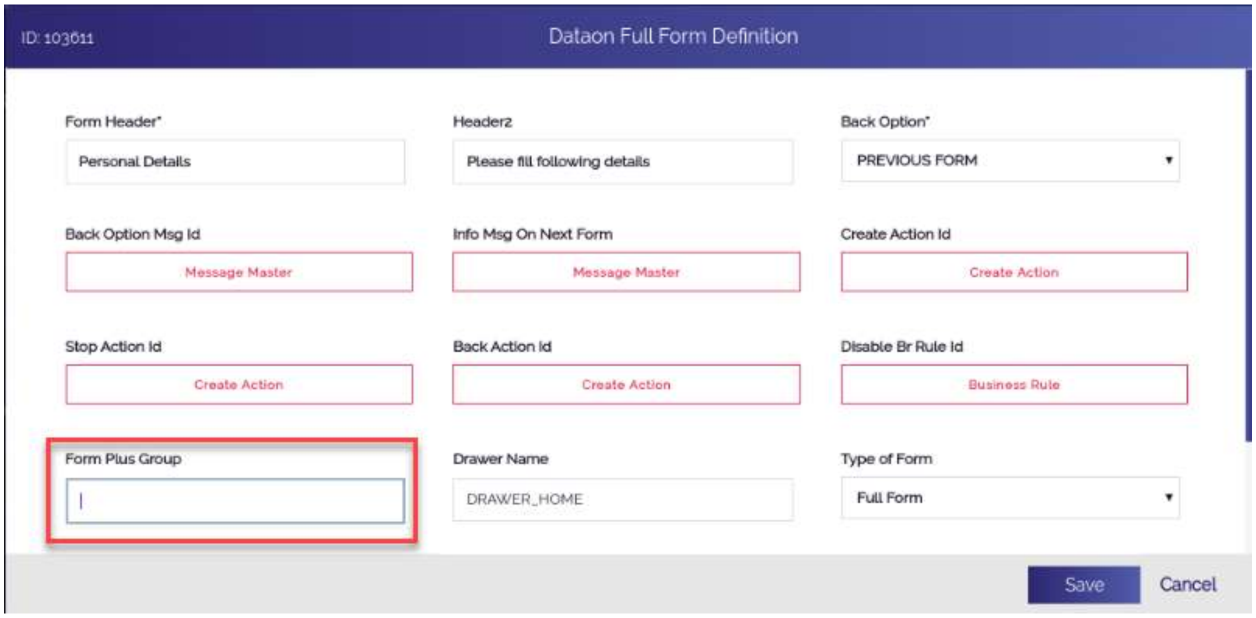
9. After you ensure that you have added and configured all plus group items **(Fig 3.8 (e))**, click **Save**, the plus group is successfully configured.

After you configure the plus group, you can select the respective plus group while configuring a mobile app screen.

3.8.1 Applying a plus group

To apply a plus group on specific screen:

1. In the **Dataon Full Form Definition** dialog box **(Fig 3.8.1 (a))**, locate the **Form Plus Group** box.



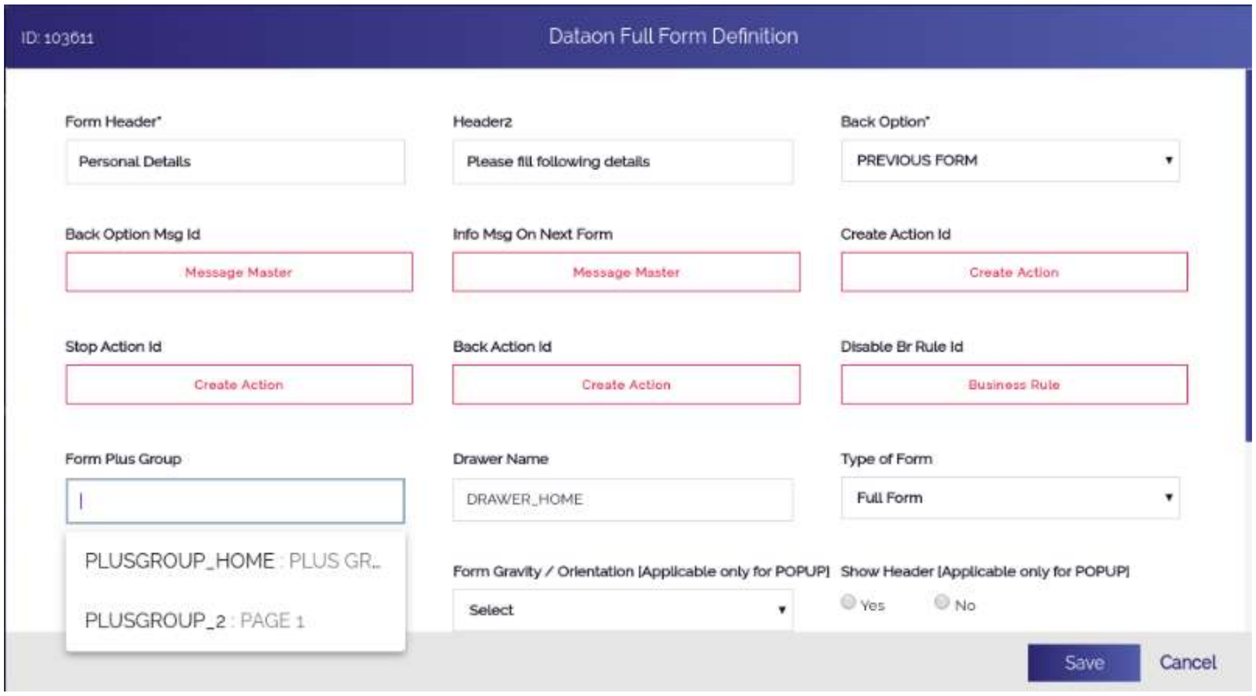
ID: 103611

Dataon Full Form Definition

Form Header*: Personal Details	Header2: Please fill following details	Back Option*: PREVIOUS FORM
Back Option Msg Id: Message Master	Info Msg On Next Form: Message Master	Create Action Id: Create Action
Stop Action Id: Create Action	Back Action Id: Create Action	Disable Br Rule Id: Business Rule
Form Plus Group:	Drawer Name: DRAWER_HOME	Type of Form: Full Form

(Fig 3.8.1 (a))

2. Click inside the **Form Plus Group** box, a list **(Fig 3.8.1 (b))** of available plus groups appears.



ID: 103611

Dataon Full Form Definition

Form Header*: Personal Details	Header2: Please fill following details	Back Option*: PREVIOUS FORM
Back Option Msg Id: Message Master	Info Msg On Next Form: Message Master	Create Action Id: Create Action
Stop Action Id: Create Action	Back Action Id: Create Action	Disable Br Rule Id: Business Rule
Form Plus Group:	Drawer Name: DRAWER_HOME	Type of Form: Full Form
	Form Gravity / Orientation [Applicable only for POPUP]: Select	Show Header [Applicable only for POPUP]: Yes No

PLUSGROUP_HOME : PLUS GR...

PLUSGROUP_2 : PAGE 1

(Fig 3.8.1 (b))

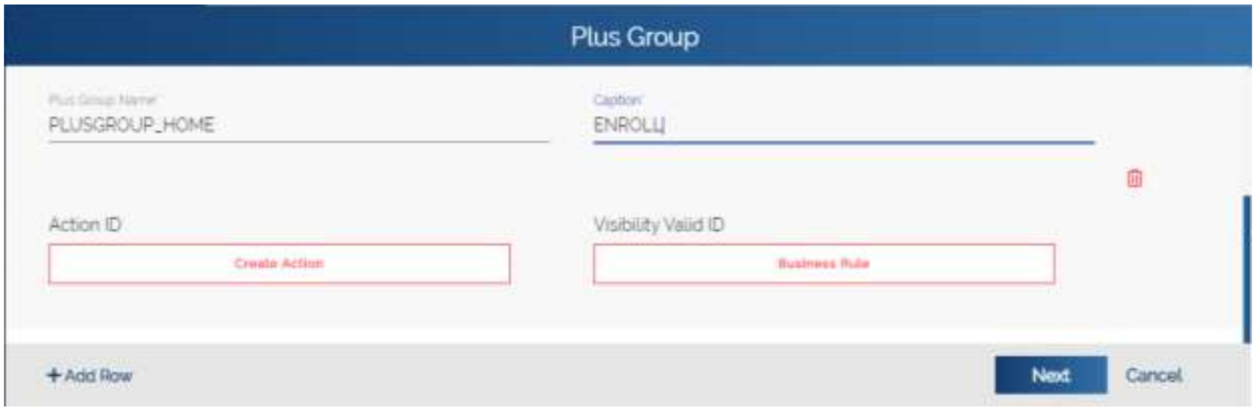
3. In the list of available plus groups **(Fig 3.8.1 (b))**, select the plus group (For example: - PLUSGROUP_HOME: PLUSGROUP) that you want to apply, and then click **Save (Fig 3.8.1 (b))**, the plus group is selected for the respective screen.

3.8.2 Assigning Actions to a Plus Group

This function allows you to define an action to a plus group item. After you define the action to a plus group item, you can perform the function on the respective plus group item. The functional behavior of specific plus group item depends on the action that you assign to the respective plus group.

To assign the action to the plus group item:

1. On the **Plus Group** dialog box, locate the **Action ID** field.

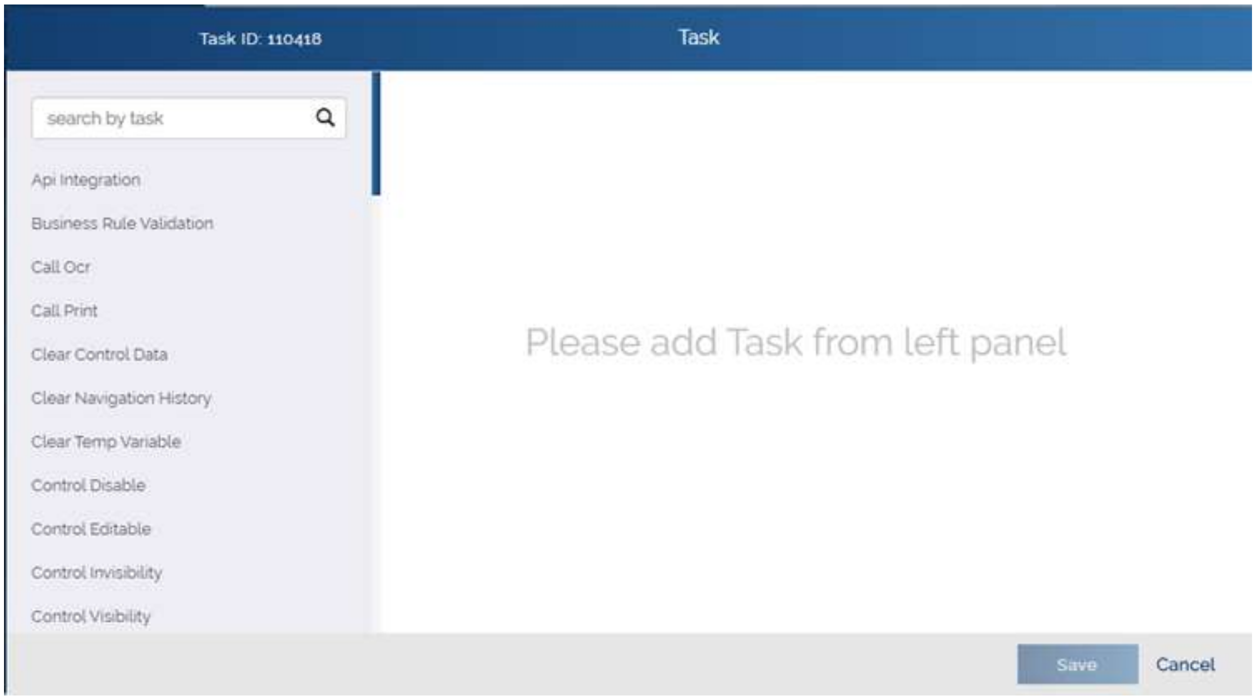


PLUSGROUP_HOME

ENROLL

(Fig 3.8.2 (a))

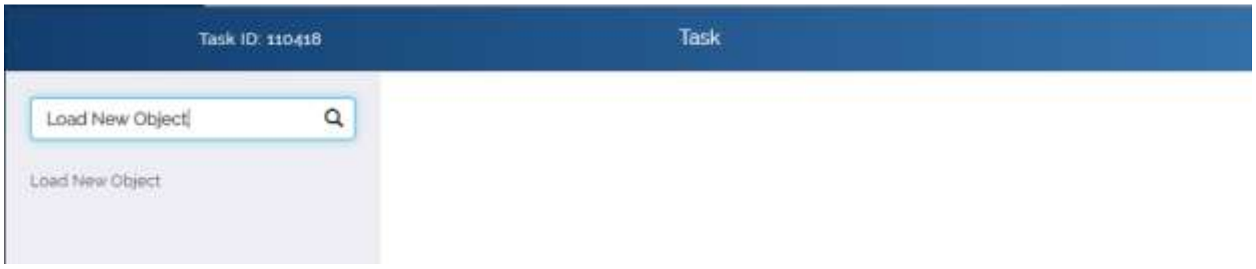
2. Under **Action ID**, click **Create Action (Fig 3.8.2)**, the **Task** dialog box **(Fig 3.8.2 (b))** opens.



Please add Task from left panel

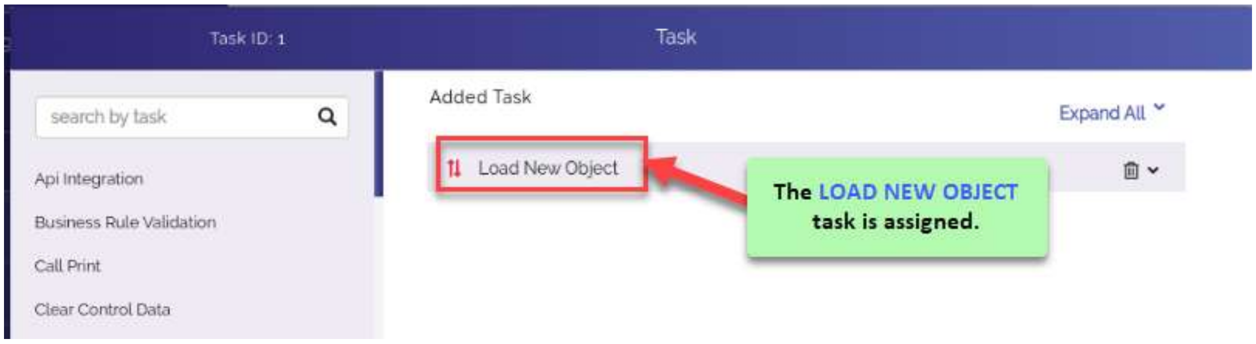
(Fig 3.8.2 (b))

3. In the **Task** dialog box, locate left navigation pane.
4. In the left navigation pane, locate the **search by task** box **(Fig 3.8.2 (b))**.
5. In the **search by task** box, enter the name of task (For example: - Load New Object) that you want to assign, the left navigation pane displays the name of the task **(Fig 3.8.2 (c))**.



(Fig 3.8.2 (c))

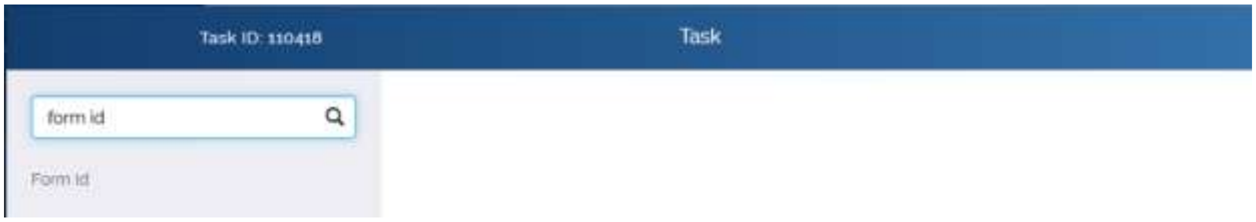
6. Select that task and then click **Save (Fig 3.8.2 (b))**, the task is successfully assigned to the respective plus group item.



(Fig 3.8.2 (d))

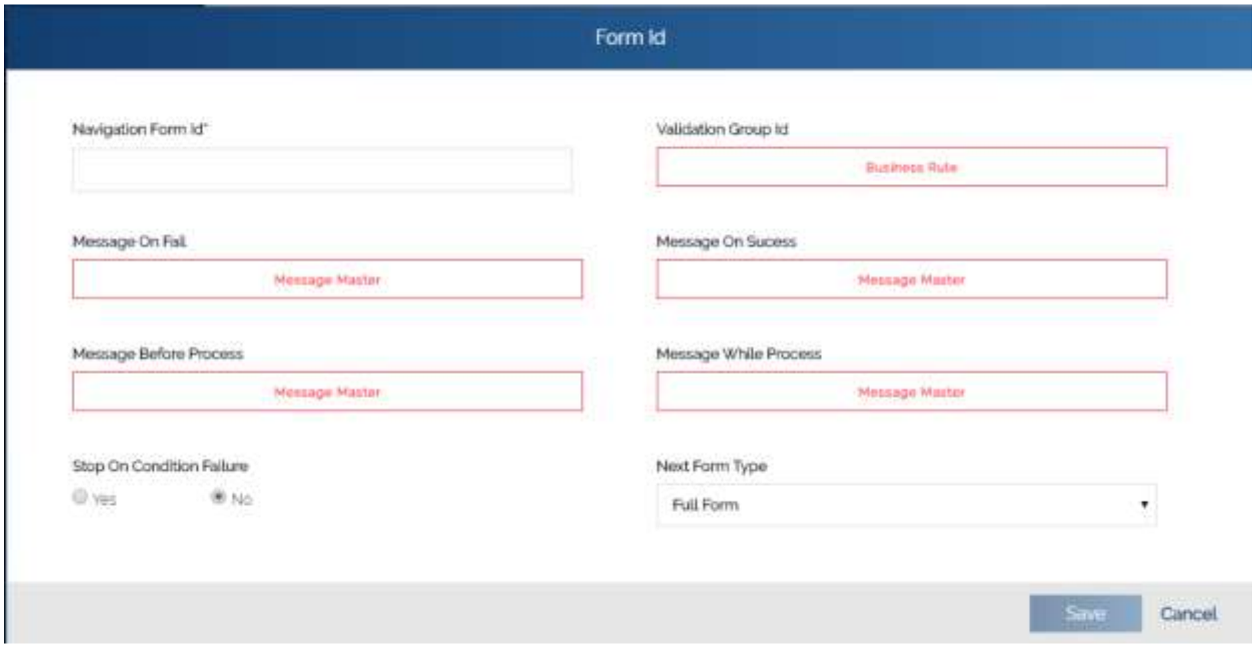
The following steps describe how to assign the Form Id task to the plus group item.

7. In the **search by task** box, enter **form id**, the left pane **(Fig 3.8.2 (d)** displays the **Form Id** task.



(Fig 3.8.2 (e))

8. In the left pane, click the **Form Id** task **(Fig 3.8.2 (e))**, the **Form Id** dialog box **(Fig 3.8.2 (f))** opens.



Form Id

Navigation Form Id*

Validation Group Id: Business Rule

Message On Fail: Message Master

Message On Sucess: Message Master

Message Before Process: Message Master

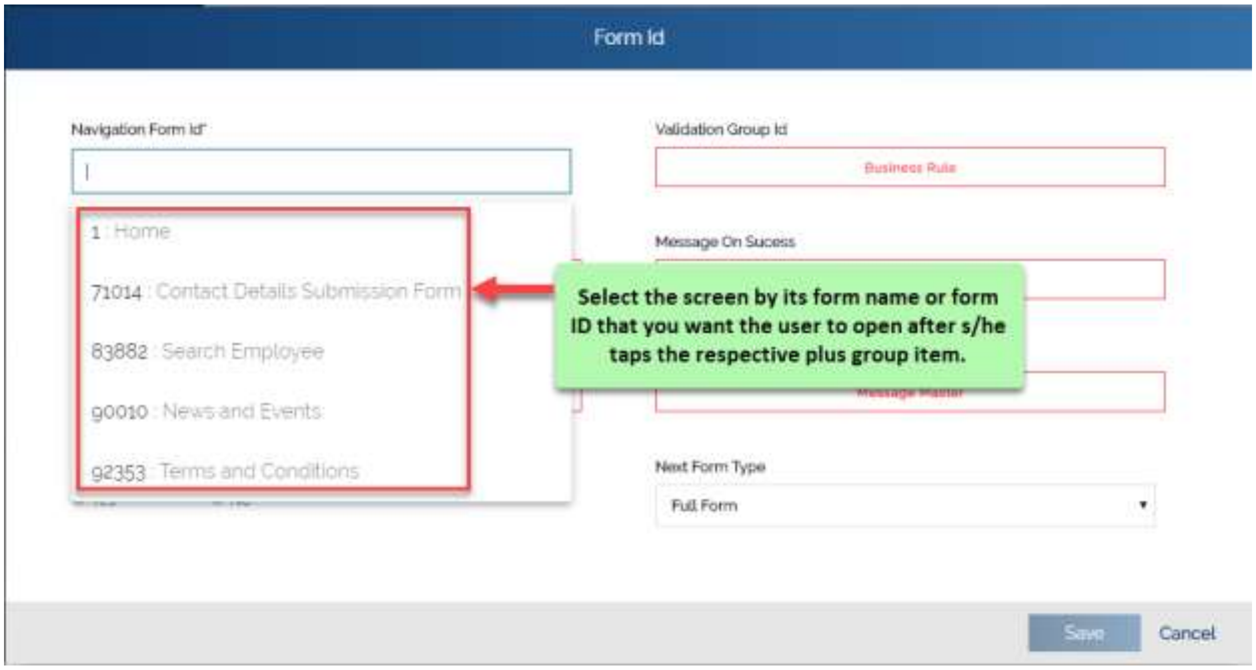
Message While Process: Message Master

Stop On Condition Failure: Yes [✓ No]

Next Form Type: Full Form

(Fig 3.8.2 (f))

9. In the **Form Id** dialog box, locate the **Navigation Form Id** box **(Fig 3.8.2 (f))**.
10. Click inside the **Navigation Form Id** box, it displays the list **(Fig 3.8.2 (g))** of the available forms along with their form ID.



Form Id

Navigation Form Id*

1 : Home
71014 : Contact Details Submission Form
83882 : Search Employee
90010 : News and Events
92353 : Terms and Conditions

Select the screen by its form name or form ID that you want the user to open after s/he taps the respective plus group item.

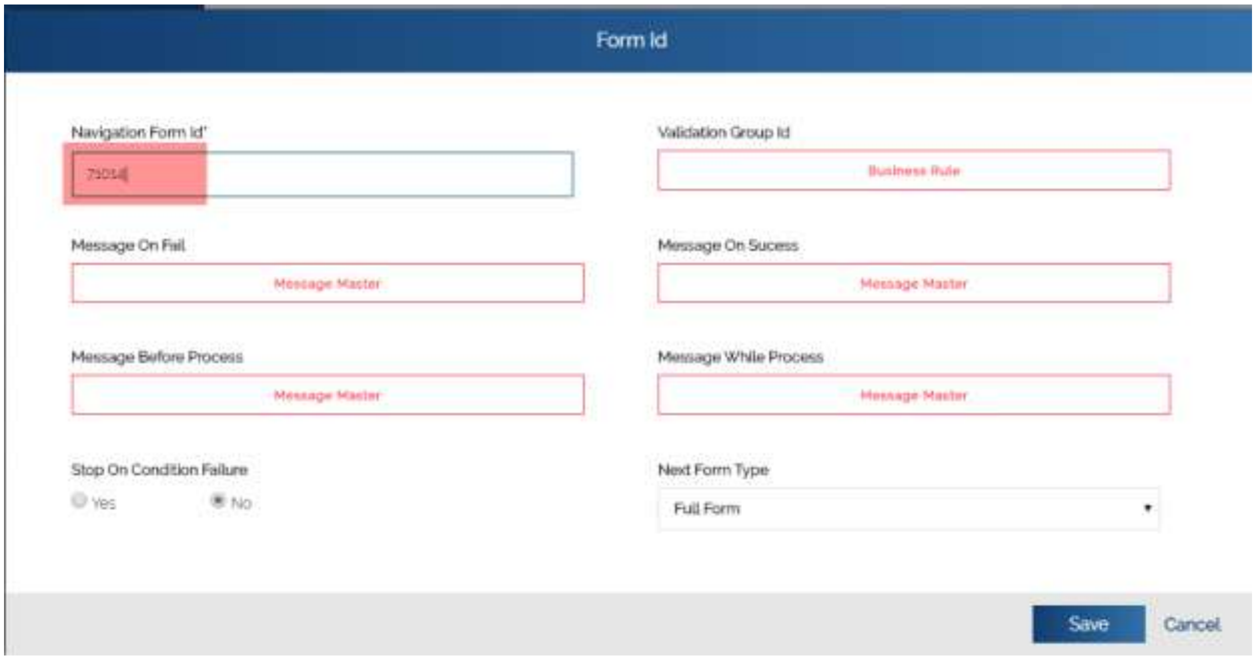
Validation Group Id: Business Rule

Message On Sucess

Next Form Type: Full Form

(Fig 3.8.2 (g))

11. In the list of forms, select the desired form, the form ID of the selected form starts appearing.



Form Id

Navigation Form Id*	71014
Validation Group Id	Business Rule
Message On Fail	Message Master
Message On Sucess	Message Master
Message Before Process	Message Master
Message While Process	Message Master
Stop On Condition Failure	Yes [✓ No]
Next Form Type	Full Form

(Fig 3.8.2 (h))

12. After you select the form, click **Save (Fig 3.8.2 (h))**, the **Form Id** task is successfully assigned to the respective plus group item.

3.9 Displaying List from Database

This section describes how to display the list of database. Displaying list from database means that the corresponding API or database view will execute to fetch the required data from the respective database. Therefore, API or database view will display the data on the screen/page of the mobile app.

It has been extensively discussed in the heading section: Configuring a List that you can display the data in the mobile app list by using two components: **API** or **database view**. Apart from designing the layout of a list in the vDesigner application, technical team implements the code of the API or database view externally. Therefore, you can use these two components to display the list in the mobile app as follows:

Case1:- (API call or invocation)
If you choose to display the list by using REST API, you need bind the API to the UI object (For example: - Submit or View button) that the mobile app user clicks to view the list. On the click of the respective button, you need to configure the API call as a result of load new object. After you configure the API call, the API will execute to fetch the data from database and then store it in the array object.

When user clicks the **Submit** or **View** button, the array type JSON object will display the data in the respective fields in the mobile app.

Case2:- (Using database view)
If you choose to display the list by database view, you need to enter the details of database view when you configure a list in the vDesigner application. In the details of the database view, you need to enter the name of the database view and the name of primary keys that are used to fetch data from the respective data source.

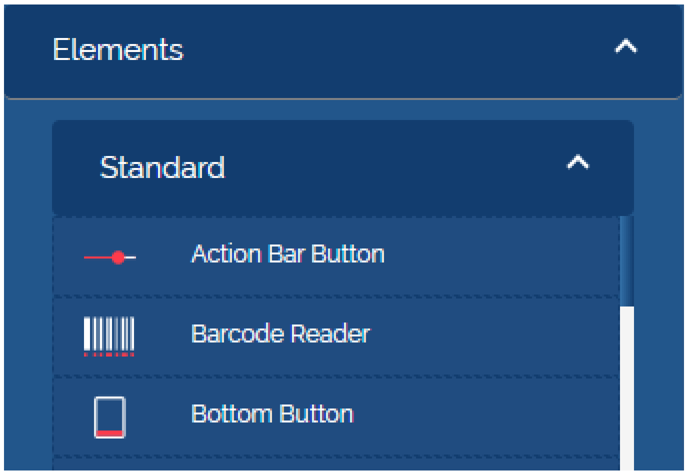
Before discussing these two cases, **let's discuss how to bind an API call to a bottom button** (For example: - Submit or View) so that when mobile app user clicks the respective button, the API executes to fetch the data and then display it in the list.

3.9.1 Configuring API Call on Button

You will configure an API call on the "On Click" event of a button if you choose to display list by using REST API. If you choose the database view to display list on the mobile app, you will not configure the API call on the "On Click" event of a button.

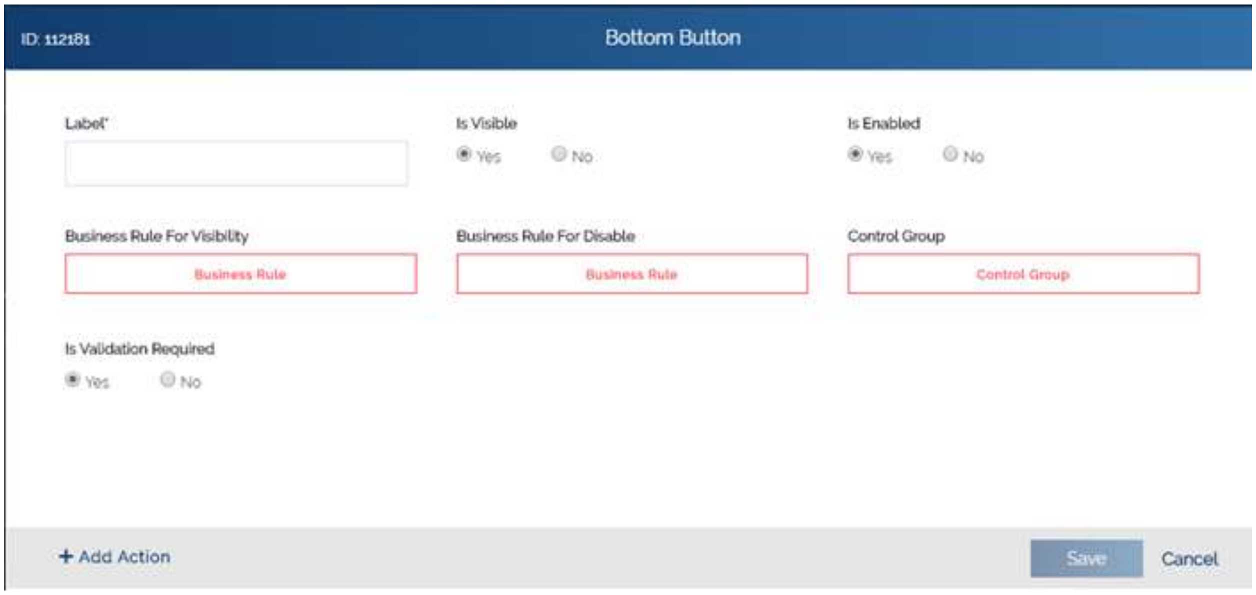
To configure an API call:

1. In the vDesigner application, locate the left navigation pane **(Fig 3.9.1 (a))**.



(Fig 3.9.1 (a))

2. In the left navigation pane, click **Elements** and then click **Standard (Fig 3.9.1 (a))**, the navigation pane expands.
3. Under **Standard**, click **Bottom Button (Fig 3.9.1 (a))**, the **Bottom Button** dialog box **(Fig 3.9.1 (b))** opens.


ID: 112181

Bottom Button

Label*

Is Visible: [✓ Yes] No

Is Enabled: [✓ Yes] No

Business Rule For Visibility: Business Rule

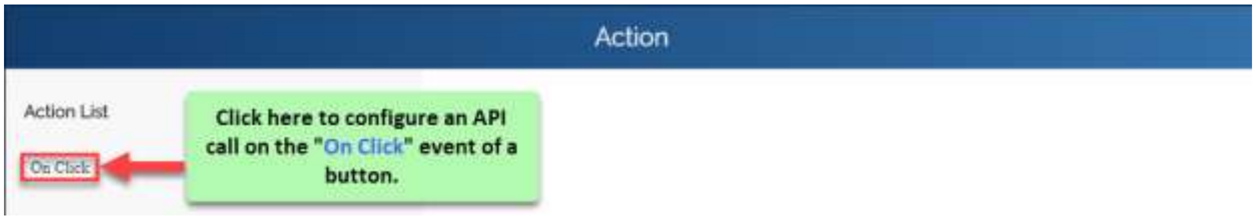
Business Rule For Disable: Business Rule

Control Group: Control Group

Is Validation Required: [✓ Yes] No

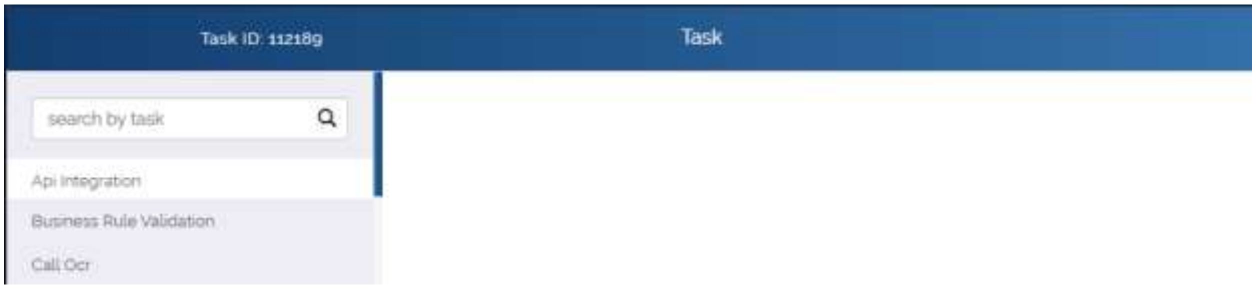
(Fig 3.9.1 (b))

4. In the **Label** box **(Fig 3.9.1 (b))**, enter the name of button (For example: - View).
5. In the **Bottom Button** dialog box, click **Add Action (Fig 3.9.1 (b))**, the **Action** dialog box **(Fig 3.9.1 (c))** opens.


Action
Action List
On Click
Click here to configure an API call on the "On Click" event of a button.

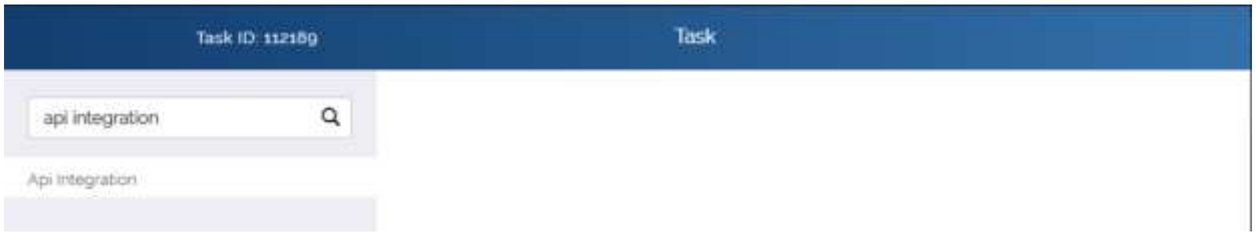
(Fig 3.9.1 (c))

6. In the **Action** dialog box, click **On Click (Fig 3.9.1 (c))**; the **Task** dialog box **(Fig 3.9.1 (d))** opens.



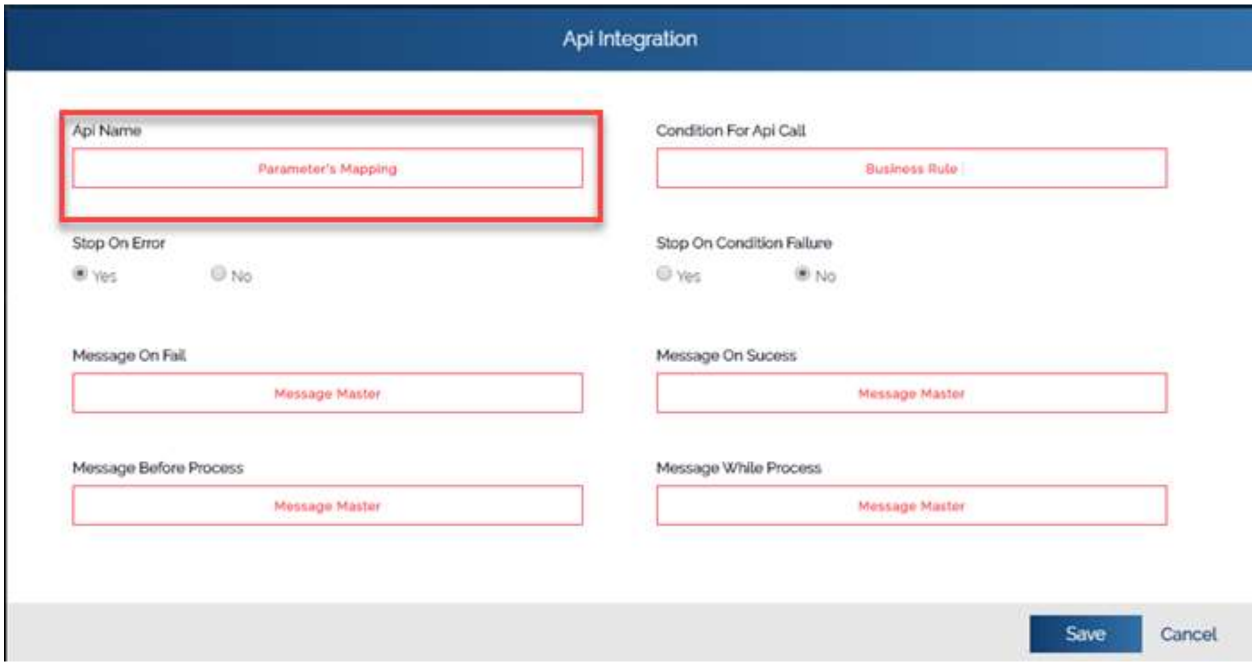
(Fig 3.9.1 (d))

7. In the **Task** dialog box, locate the **Search by task** box **(Fig 3.9.1 (d))**.
8. In the **Search by task** box, enter **API integration**, the **API Integration** task **(Fig 3.9.1 (e))** starts appearing.



(Fig 3.9.1 (e))

9. Click the **API Integration** task **(Fig 3.9.1 (e))**, the **Api Integration** dialog box **(Fig 3.9.1 (f))** opens.



Api Integration

Api Name: Parameter's Mapping

Condition For Api Call: Business Rule

Stop On Error: [✓ Yes] No

Stop On Condition Failure: Yes [✓ No]

Message On Fail: Message Master

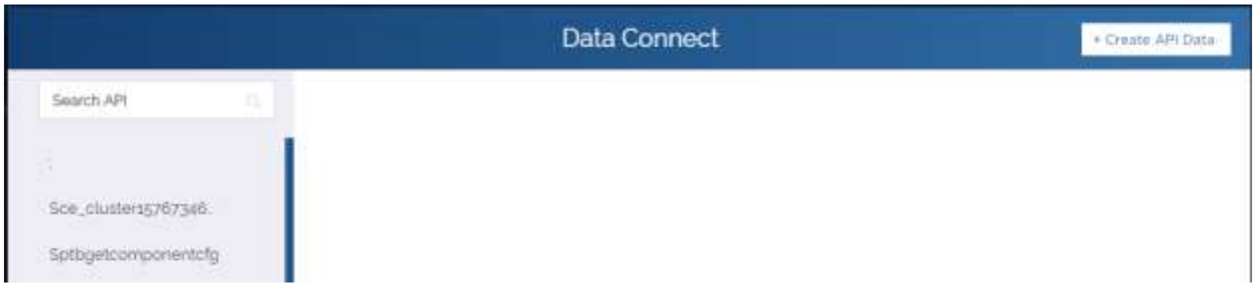
Message On Sucess: Message Master

Message Before Process: Message Master

Message While Process: Message Master

(Fig 3.9.1 (f))

10. In the **Api Integration** dialog box, click **Parameter's Mapping (Fig 3.9.1 (f))**, the **Data Connect** dialog box **(Fig 3.9.1 (g))** opens.

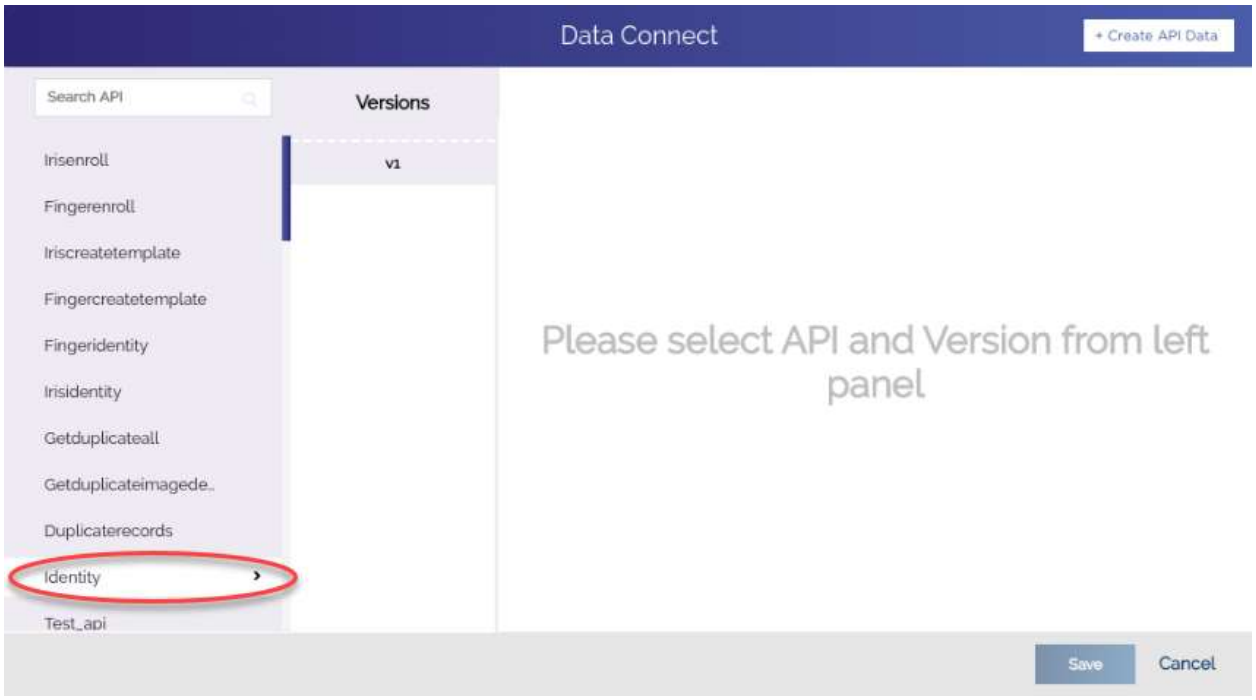


Data Connect

Sce_cluster15767346
Sptbgetcomponentcfg

(Fig 3.9.1 (g))

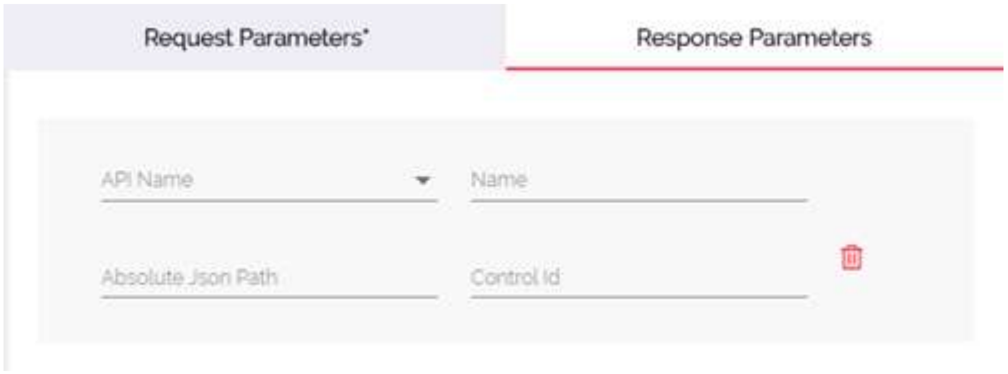
11. In the **Data Connect** dialog box **(Fig 3.9.1 (g))**, the left navigation pane displays the list of existing API(s).
12. In the **Search API** box **(Fig 3.9.1 (g))**, enter the name of API (For example: - Identity) that you want to configure.



(Fig 3.9.1 (h))

13. After you locate the API (For example: - Identity), click it, the **Data Connect** dialog box displays the version number (For example: - v1) of API.
14. In the middle pane, click the version number (For example: - v1) of the API, the right pane displays two tabs: **Request Parameters** and **Response Parameters**.
15. In the right pane, click the **Response Parameters** tab **(Fig 3.9.1 (i))**, it displays the following fields:

- **API Name**
- **Name**
- **Absolute JSON Path**
- **Control Id**

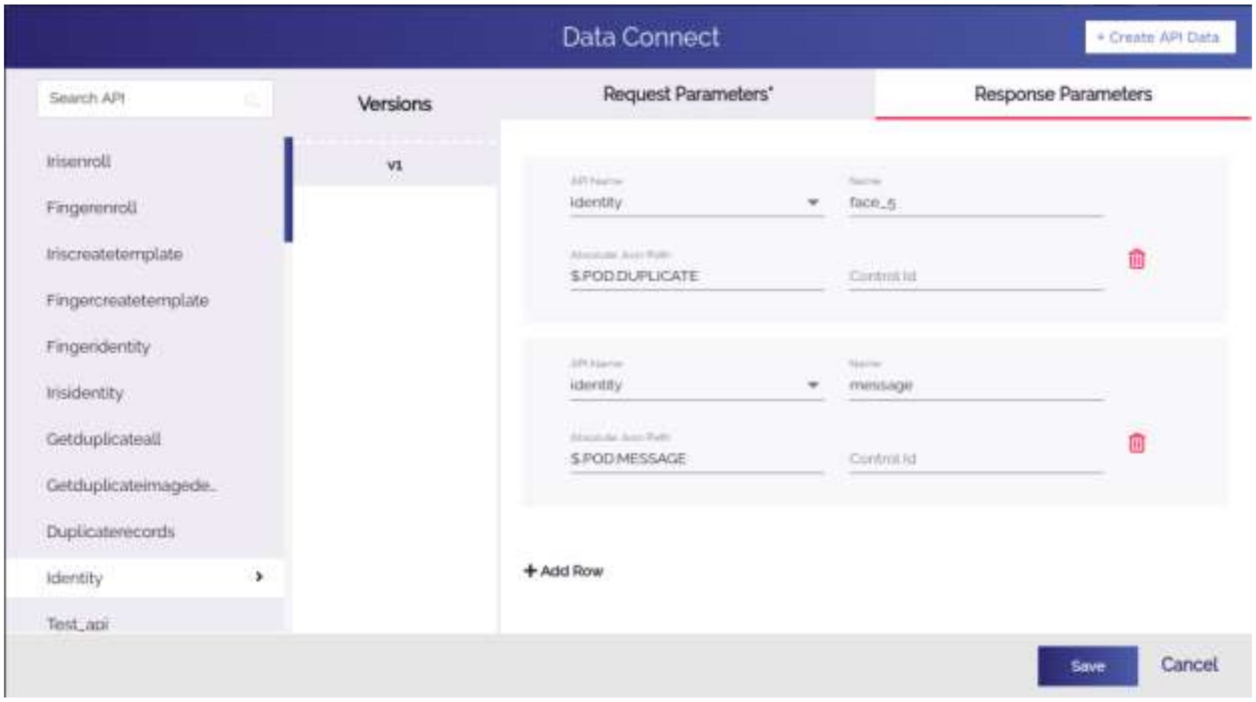


(Fig 3.9.1 (i))

16. In these fields, enter the values as follows:

Field	Description
API Name	Click this list and then select the API (For example: - Identity) **(Fig 3.9.1 (j))** that you want to bind with the "**On Click**" action of the bottom button.
Name	In this field, enter the name of response parameter of the API that holds specific value that the API fetches from the database.
Absolute JSON Path	In this field, define the JSON path of entity where the data will be stored. You can define the JSON path as follows: $.POD.DUPLICATE In this JSON path, the keyword: **DUPLICATE** is the entity that you define by using the Entity And Relationship module. In the JSON path, **DUPLICATE** is treated as the array type JSON object where entire dataset is stored. To know more about entity and the relationship between two entities, visit the heading section: Creating New Entity and Defining Relationship between Entities.

17. After you enter the details of first response parameters of the API, enter the details of next response parameter as described in the heading section.



API Name	Name	Absolute Json Path	Control Id
identity	face_5	$.POD.DUPLICATE	
identity	message	$.POD.MESSAGE	

(Fig 3.9.1 (j))

<u>Note</u>:-
In the screen capture, if you look at the details of the second response parameter: message, you will find that the JSON path is defined as **$.POD.MESSAGE**. As you can notice that this JSON path does not reference the **DUPLICATE** entity. It references the **MESSAGE** object.

In the **$.POD.MESSAGE** JSON path, the **MESSAGE** keyword is treated as a simple object, which is declared in the JSON path to store the message string that the **message** response parameter fetches from the data source. The **MESSAGE** keyword is not treated as array type JSON object If you want, you can change the name of **MESSAGE** object to **MESSAGE_1**. It is a simple object, which is used to store random message string.

That is how you define the entity or simple object in the JSON path based on the type of data that you receive in the response from API.

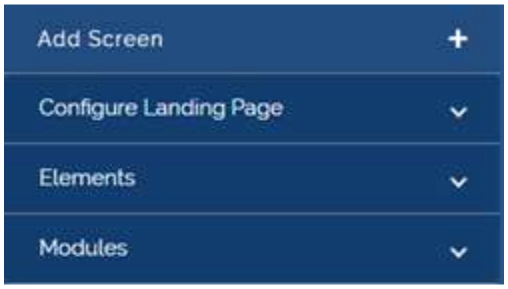
18. To enter the details of next response parameters, click **Add Row (Fig 3.9.1 (j))** and then enter the details of parameter as described in the last table.
19. After you enter the details of all parameters, click **Save (Fig 3.9.1 (j))**, the **Data Connect** dialog box is closed.
20. In the **API Integration** dialog box, click **Save**, the **API Integration** dialog box is closed.
21. In the **Task** dialog box, click **Save**, the **Task** dialog box is closed.
22. In the **Action** dialog box, click **Save**, the **Action** dialog box is closed.
23. In the **Bottom Button** dialog box, click **Save**, the API is successfully configured with the “On Click” action of the button.

3.9.2 Configuring List with API

If you want to configure a list with an API call, you need to define the **name of object** where the data is stored after respective API fetches the data from the database and the array type JSON object that holds the data to display it in the list.

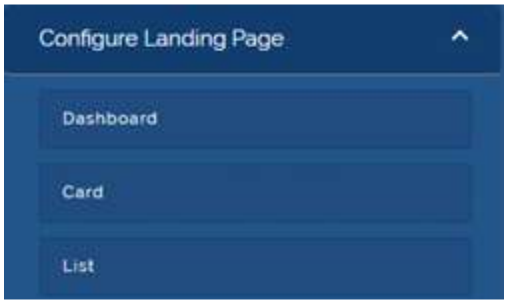
To configure the list:

1. On vDesigner application's dashboard, locate the left navigation pane.



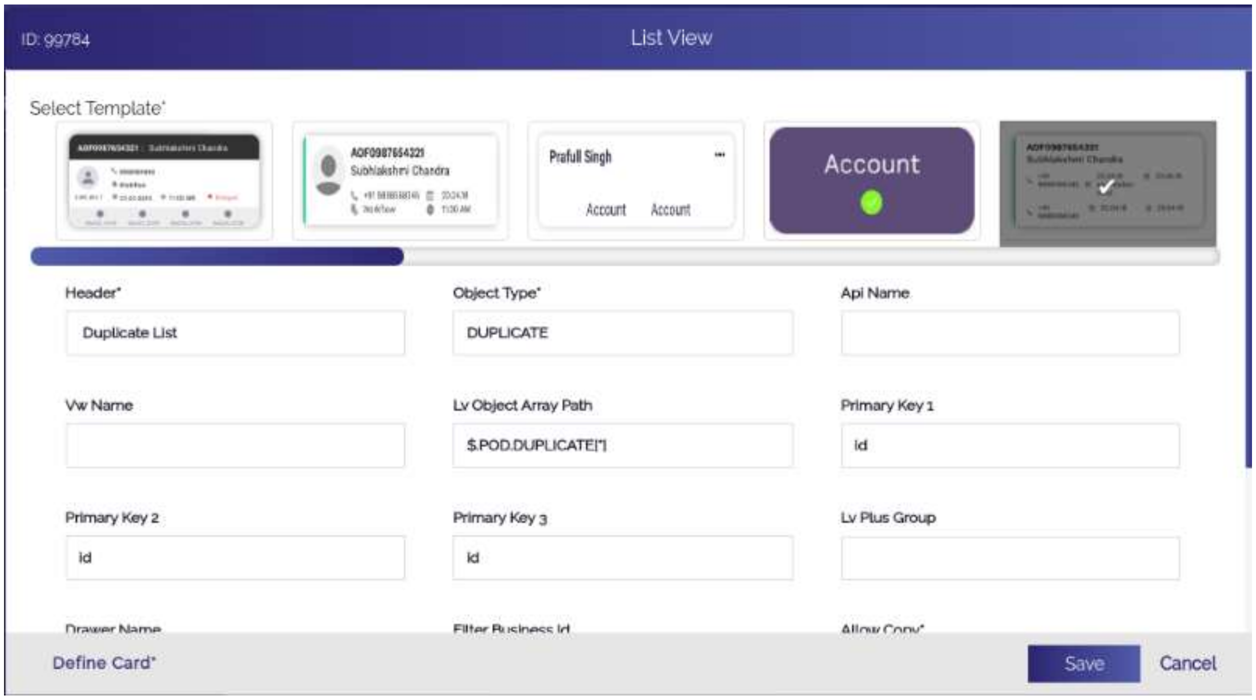
(Fig 3.9.2 (a))

2. In the left navigation pane, click **Configure Landing Page**, the navigation pane expands.



(Fig 3.9.2 (b))

3. Under **Configuring Landing Page (Fig 3.9.2 (b))**, click **List**, the **List View** dialog box **(Fig 3.9.2 (c))** opens.


ID: 99784 List View

Select Template*

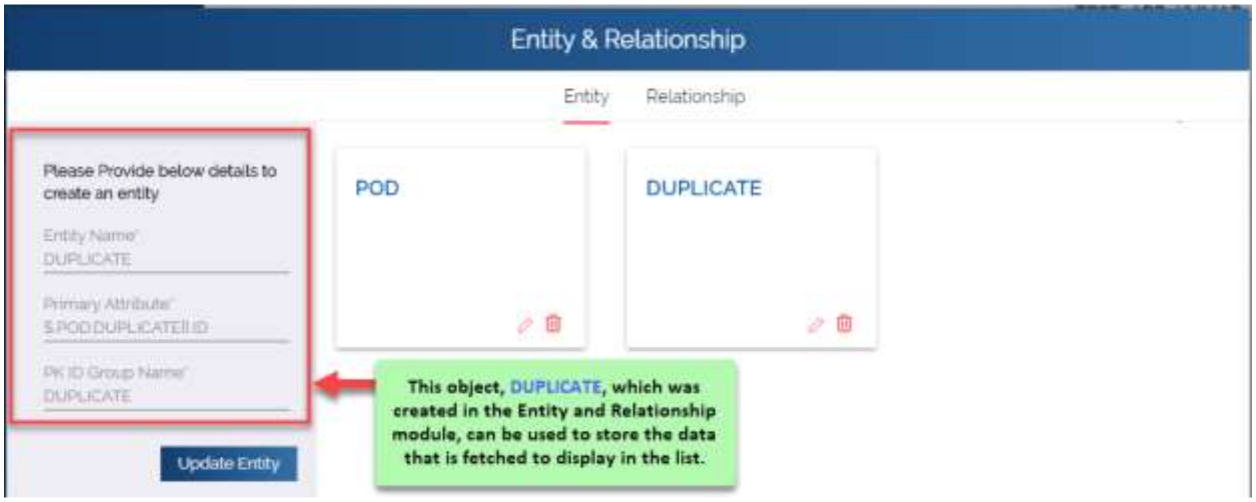
Header*: Duplicate List
Object Type*: DUPLICATE
Api Name:
Vw Name:
Lv Object Array Path: $.POD.DUPLICATE[*]
Primary Key 1: id
Primary Key 2: id
Primary Key 3: id
Lv Plus Group:
Drawer Name
Filter Business Id
Allow Copy*

(Fig 3.9.2 (c))

4. In the **List View** dialog box, enter values in the respective boxes as follows:

Box	Description
Header	In this box, enter the name/title (For example: - Duplicate List) of the list.
Object Type	In this box, enter the name object (For example: - DUPLICATE) where the data is stored after it is fetched from the database. **Note**:- In the **Object Type** box, you enter the name of object **(Fig 3.4.2 (d))** that you create in the **Entity & Relationship** module. If you enter any random name in the **Object Type** box, the list will not display the data. First you need to create the object by using **Entity and Relation** module and then use this object when you configure the list.
LV Object Array Path	In this box, define the array of the **DUPLICATE** object that you have defined in the **Object Type** box. You can define the array as follows: $.POD.DUPLICATE[*] The path: **$.POD.DUPLICATE[*]** is the path of the **Duplicate** object. In this path of array, the asterisk (*) character that is enclosed by bracket works as a common loop. When the data is fetched, it reads the data from 0th index and picks the entire record set of data and treats it as a single object. It moves through each index consecutively and picks other records of data.

Primary Key1/Primary Key2/ Primary Key3	In these boxes, enter the variable (For example: - ID) **(Fig 3.9.2 (d))** that you have declared when you created the object in the Entity and Relationship module. You will use this variable if you are fetching the data by using **array object**.



This object, DUPLICATE, which was created in the Entity and Relationship module, can be used to store the data that is fetched to display in the list.

(Fig 3.9.2 (d))

After you configure the list, you need to define the card where the data of the list is displayed.

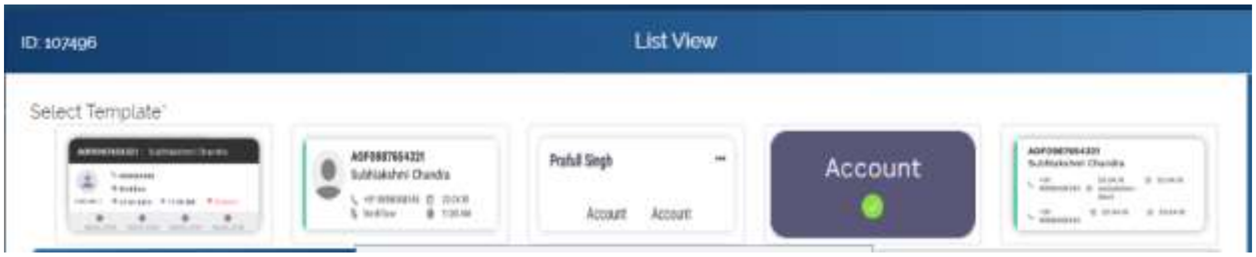
3.9.2.1 Define Card

You define the card to determine the layout in which the list displays the data.

To define the card:

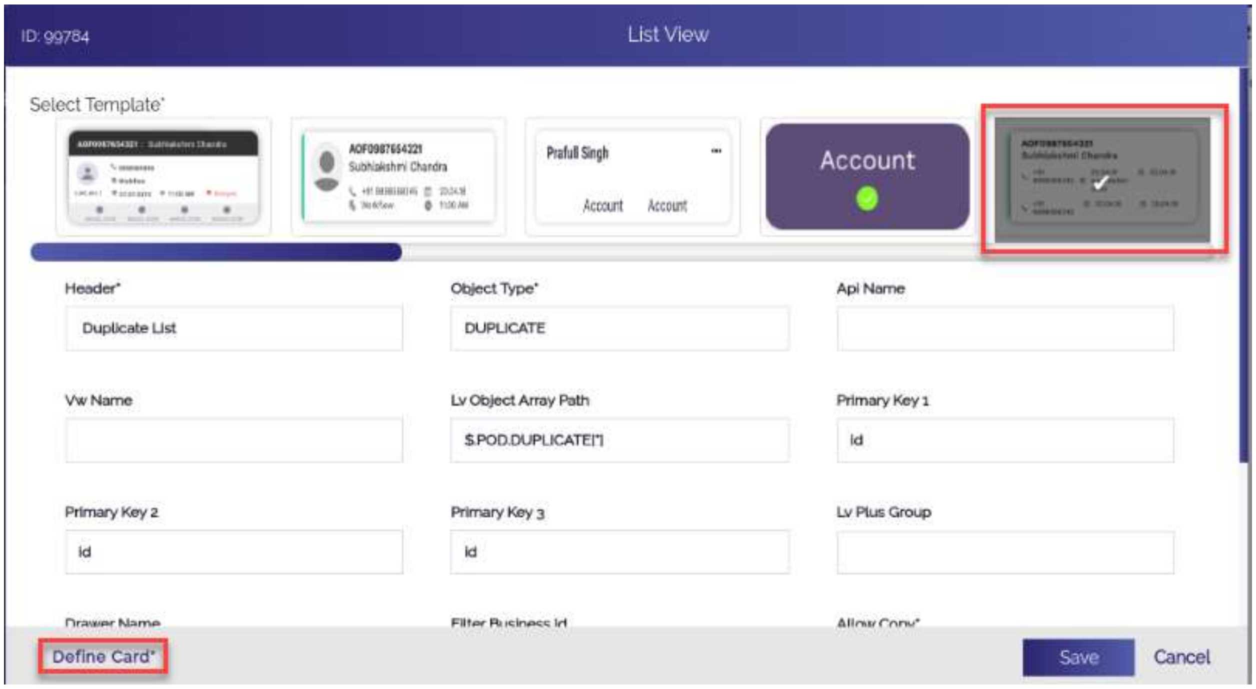
To define the card, perform the function as follows:

1. In the **List View** dialog box **(Fig 3.9.2.1 (a))**, locate upper panel that contains the card templates.



(Fig 3.9.2.1 (a))

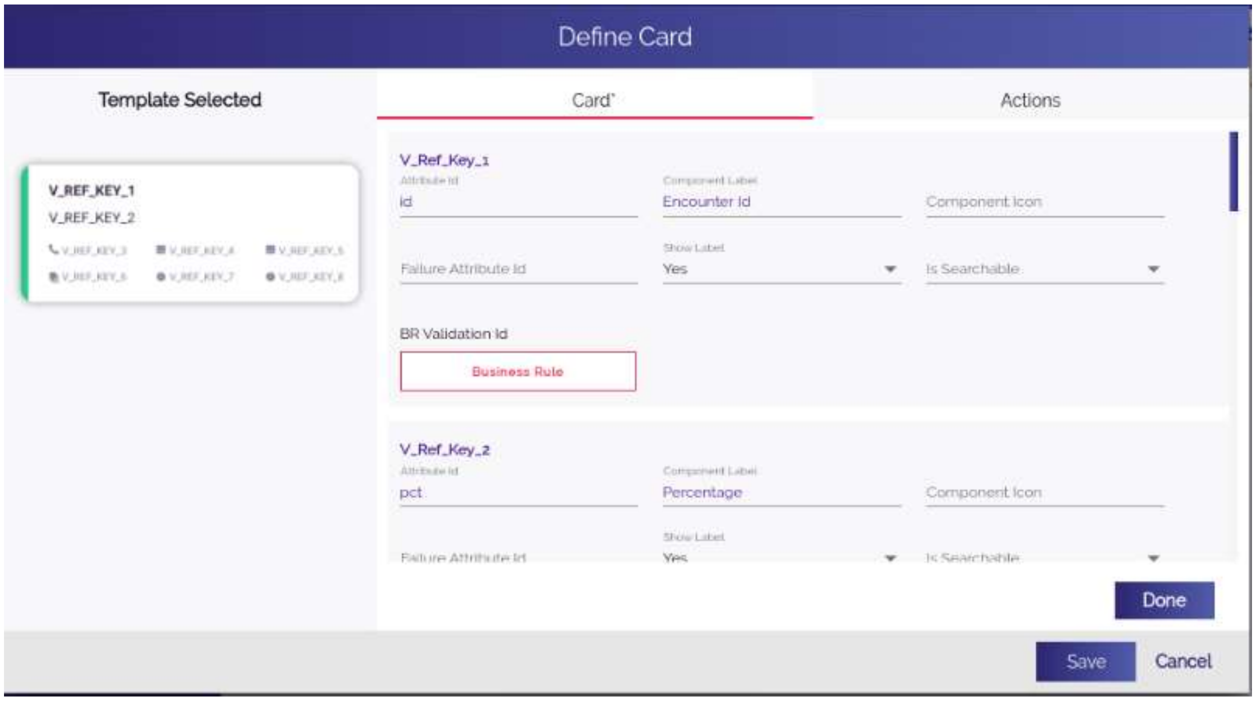
2. After you select a card template **(Fig 3.9.2.1 (b))**, the **Define Card** link appears.



Header*: Duplicate List
Object Type*: DUPLICATE
Api Name:
Vw Name:
Lv Object Array Path: $.POD.DUPLICATE[*]
Primary Key 1: id
Primary Key 2: id
Primary Key 3: id
Lv Plus Group:
Drawer Name
Filter Business Id
Allow Copy*
Define Card*

(Fig 3.9.2.1 (b))

3. Click the **Define Card** link, the **Define Card** dialog box **(Fig 3.9.2.1 (c))** opens.



V_Ref_Key_1
Attribute Id: id
Component Label: Encounter Id
Component Icon
Failure Attribute Id
Show Label: Yes
Is Searchable
BR Validation Id
Business Rule

V_Ref_Key_2
Attribute Id: pct
Component Label: Percentage
Component Icon
Failure Attribute Id
Show Label: Yes
Is Searchable

(Fig 3.9.2.1 (c))

4. In the **Define Card** dialog box **(Fig 3.9.2.1 (c))**, the **Card** tab displays fields (For example: - **ID**, **Component Label**, etc.) for different variables/key (For example: - **V_Ref_Key_1** or **V_Ref_Key_2**).
5. In the respective fields, enter values as follows:

Box/List	Description
Attribute ID	In this field, enter the attribute ID **(Fig 3.9.2.1 (c))** (For example: - ID or PCT). In the attribute ID, the list holds specific value and then displays it in the corresponding label on the card of mobile app. **Note:-** In the **Attribute ID** field, enter the value that matches the name of key/attribute that you receive in the response after the data is fetched from the respective data source. For instance: - If you are receiving the name of a person in the **Name** key/attribute in the output of the API, enter **Name** in the **Attribute ID** box. If the value in **Attribute ID** field does not match the key name or attribute name in the response of the API, list will not display the data.
Component Label	In this box, enter the name of label (For example: - Encounter ID or Percentage). The list will display the data along with the label.
Show Label	Click this list to select **Yes** so that the list displays the name of label in addition to label's value.

6. For the next variable **V_Ref_Key_2**, enter the details of **Attribute ID**, **Component Label** and the **Show Label** fields as described in the last table.
7. After you enter the details of all required variables, click the **Action** tab.
8. In the **Action** tab, configure action as follow:

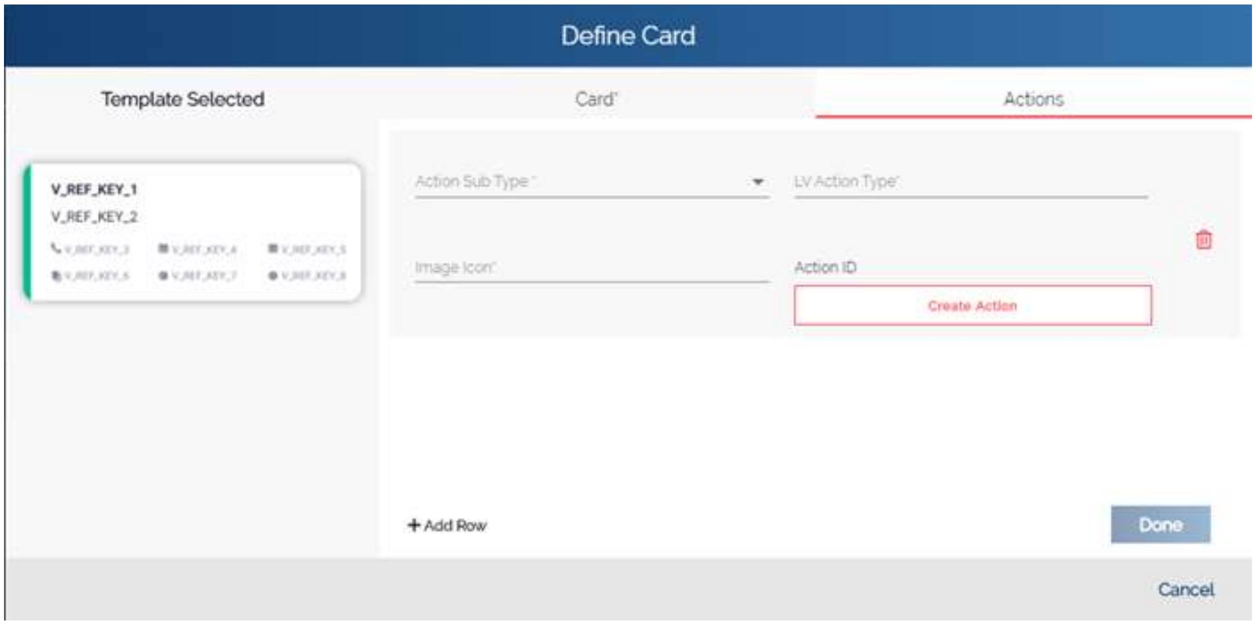
3.9.2.2 Configuring Action on Card

This function allows you to configure an action on the list of the card. You can configure the action on the list horizontally or vertically. You can incorporate a phone book, delete function, and other usable action items.

To configure an action on the list:

7. In the **Define Card** dialog box, click the **Action** tab **(Fig 3.4.2.2 (a))**, the **Define Card** dialog box displays the group of the following fields:

- **Action Sub Type**
- **LV Action Type**
- **Image Icon**
- **Action ID**



(Fig 3.9.2.2 (a))

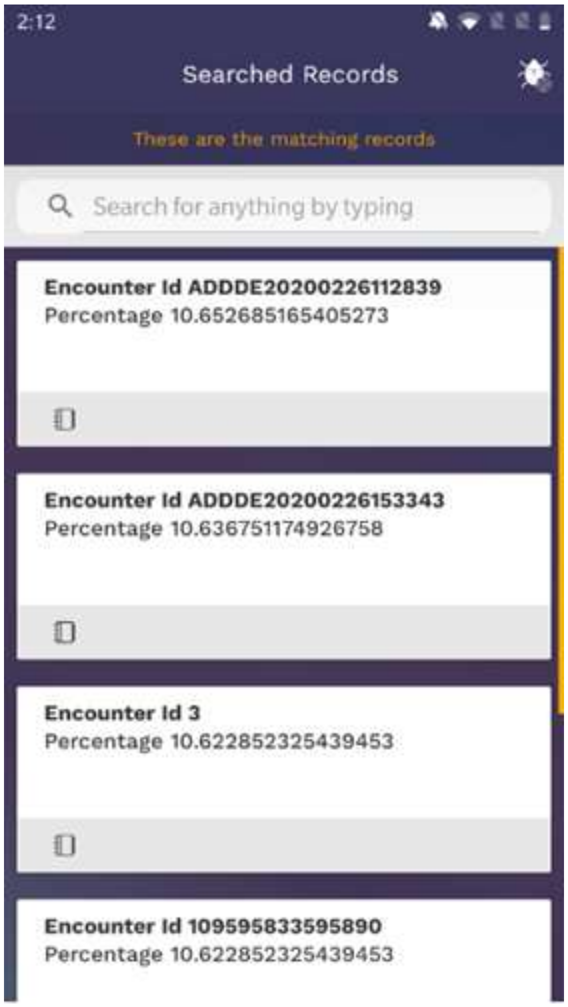
8. In these fields, enter values as follows:

Field	Description
Action Sub Type	Click this list to select: ➢ **HAB** if you want to place action items along the horizontal bar. ➢ **VAB** if you want to place action items along the vertical bar.
LV Action Type	In this box, enter the name of action item (For example: - Edit, Delete, Info, etc).
Image Icon	In this box, enter the name of icon that is displayed in the list. The icon denotes the respective action item. For instance: - For the icon of the delete function, enter: **ic_drop**. For the icon of edit function, enter: **ic_appointment**. These icons are available on the Vahana server.
Action	You can click **Create Action** to apply the task on the icon of specific action item. For example:- You can apply the the **Delete Current Child Object** task that deletes currently selected element after the user taps the **Delete** icon in the card template.

9. After you configure one action item, click **Add Row** to configure the next action item as described in the last table.
10. After you enter configure the card template and necessary action items that you want to display on the card, click **Done (Fig 3.9.2.2 (a))**, the **Define Card** dialog box is closed.
11. In the List View dialog box, click Save, the list is successfully configured.

When you configure the list with API, define card lay out to display the list's data, you need to migrate the component on the vDesigner application and then sync the mobile app by accessing the Environment module on the Vahana platform.

When user clicks the corresponding button (For example: - View or Submit), the mobile app display the list in the card layout **(Fig 3.9.2.2 (b))**.


2:12

Searched Records

These are the matching records

Search for anything by typing

Encounter Id ADDDE20200226112839
Percentage 10.652685165405273

Encounter Id ADDDE20200226153343
Percentage 10.636751174926758

Encounter Id 3
Percentage 10.622852325439453

Encounter Id 109595833595890
Percentage 10.622852325439453

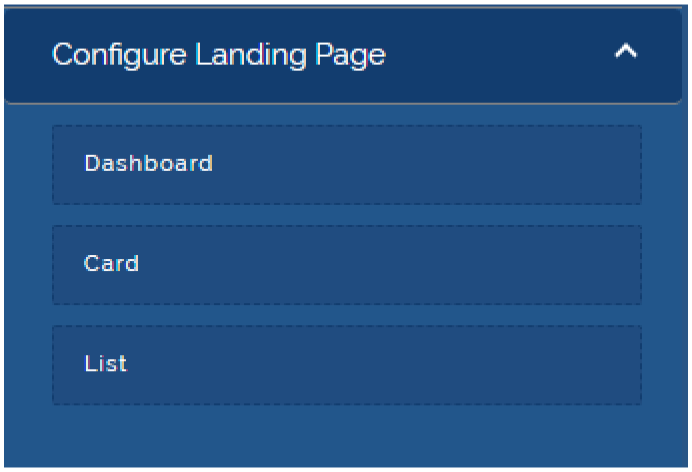
(Fig 3.9.2.2 (b))

3.9.3 Configuring List with Database View

If you choose to display a list with database view, you will provide the details of the database view while configuring a list. In the configuration details of the list, you need to provide the name of the database, name of object/object type that stores list data, and primary keys of database view. These primary keys are used to fetch the data from the data source.

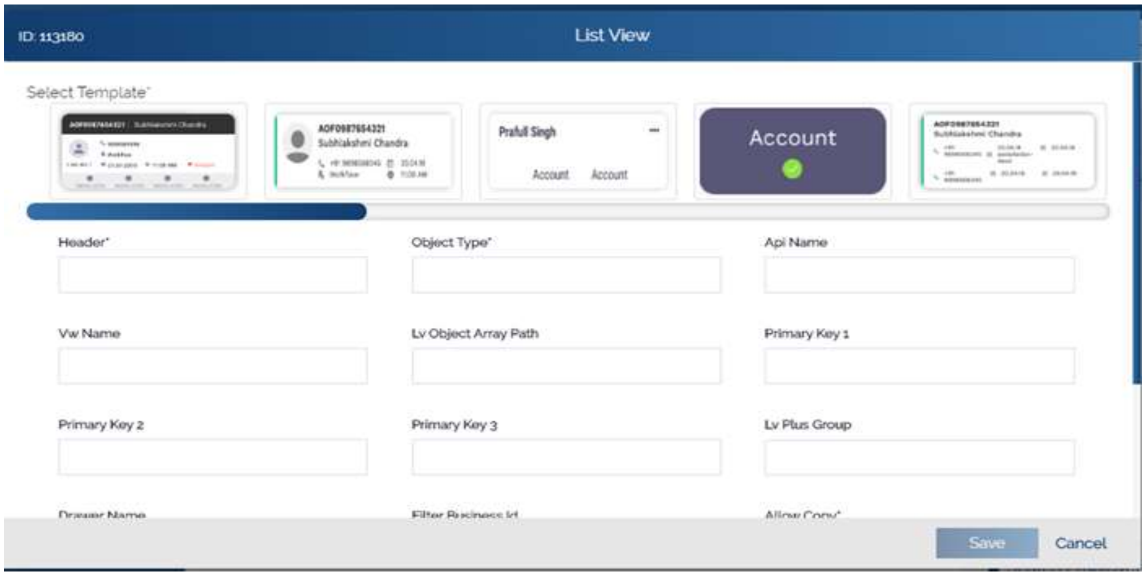
To configure the list with database view:

1. On vDesigner application's dashboard, locate the left navigation pane.



(Fig 3.9.3 (a))

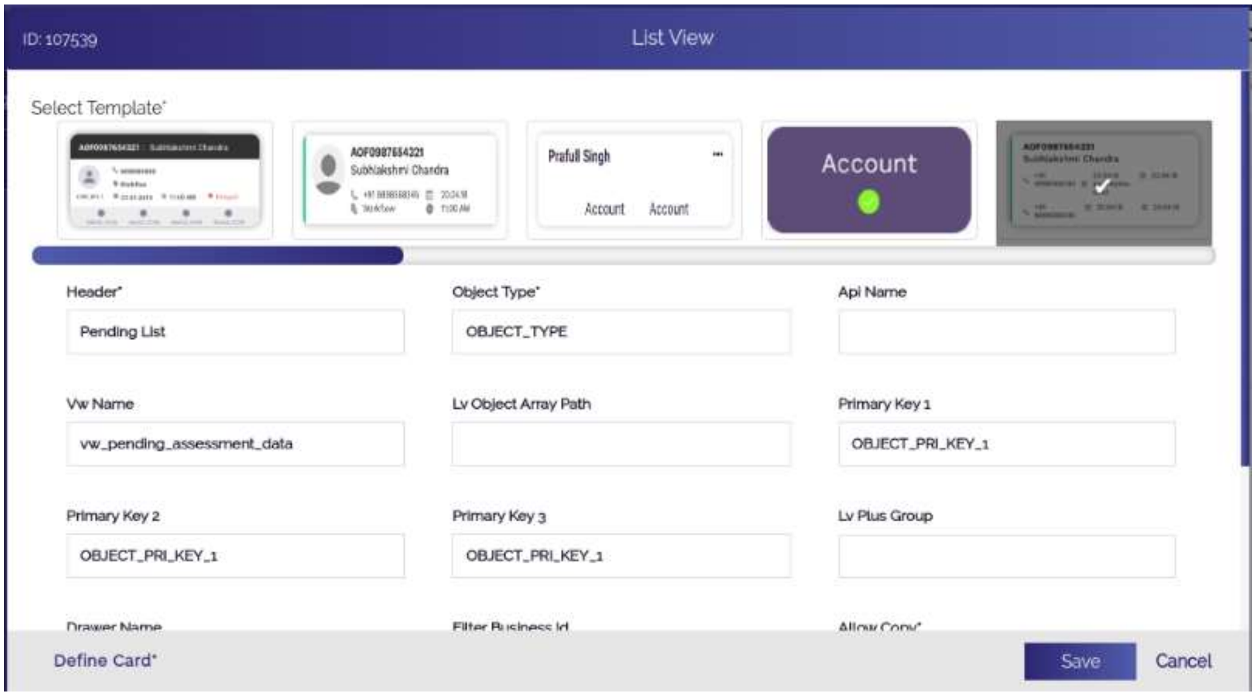
2. In the left navigation pane, click **Configure Landing Page** and then click **List**, the **List View** dialog box **(Fig 3.9.3 (b))** opens.



(Fig 3.9.3 (b))

3. **In the List View** dialog box, enter the configuration details **(Fig 3.9.3 (c))** as follows:

Box	Description
Header	In this box, enter the name of the list (For example: - Pending List) **(Fig 3.9.3 (c))**.
Object Type	In this box, enter the name of database object (For example: - OBJECT_TYPE) **(Fig 3.9.3 (c))**. It stores the data that the database view fetches from the data source. **Note:-** If you do not know the name of object that you want to enter in the **Object Type** box, you need to enquire the name of object of the database view from the database team.
Vw Name	In this box, enter the name of database view (For example: - **vw_pending_assessment_data**). The database view holds the data that it displays in the list.
Primary Key1/Primary Key2/Primary Key3	In these boxes, enter the name of primary key (For example: - PRIMARY_KEY_1, PRIMARY_KEY_2, PRIMARY-KEY_3). These primary keys are used to fetch the data from the data source where the data is stored.



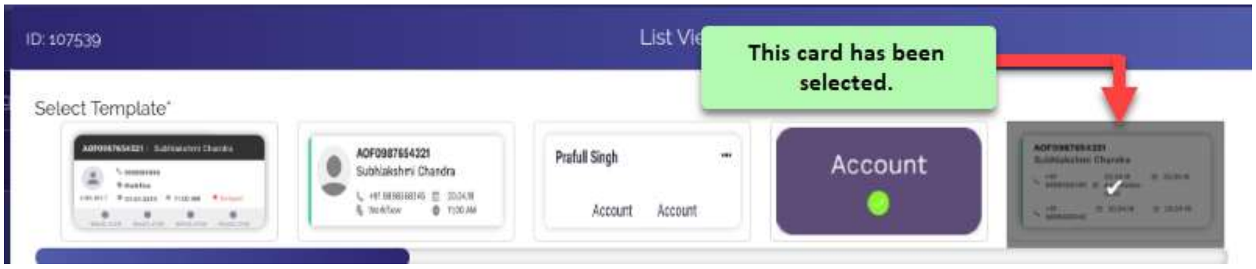
Header*	Object Type*	Api Name
Pending List	OBJECT_TYPE	
Vw Name	Lv Object Array Path	Primary Key 1
vw_pending_assessment_data		OBJECT_PRI_KEY_1
Primary Key 2	Primary Key 3	Lv Plus Group
OBJECT_PRI_KEY_1	OBJECT_PRI_KEY_1	
Drawer Name	Filter Business Id	Allow Copy*

(Fig 3.9.3 (c))

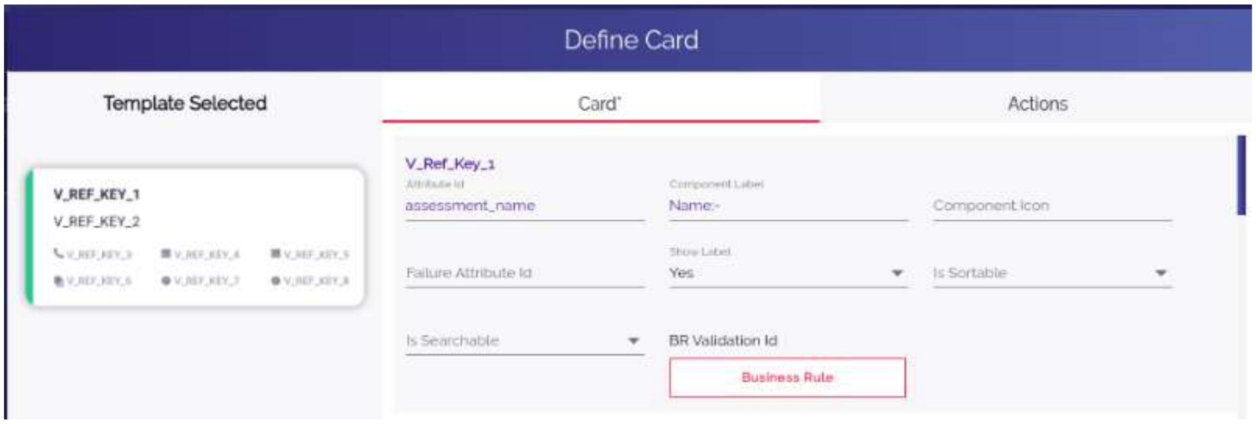
After you configure the list, you need to define the card. To define the card, you select specific card template that determines the card lay out in which the list displays the data.

3.9.3.1 Define Card

To define the card and then define action items in the card, visit the heading section: Define Card. You can use the values of the following screen captures while defining a new card:



(Fig 3.9.3.1 (a))



V_Ref_Key_1

Attribute Id: assessment_name

Component Label: Name:-

Component Icon

Failure Attribute Id

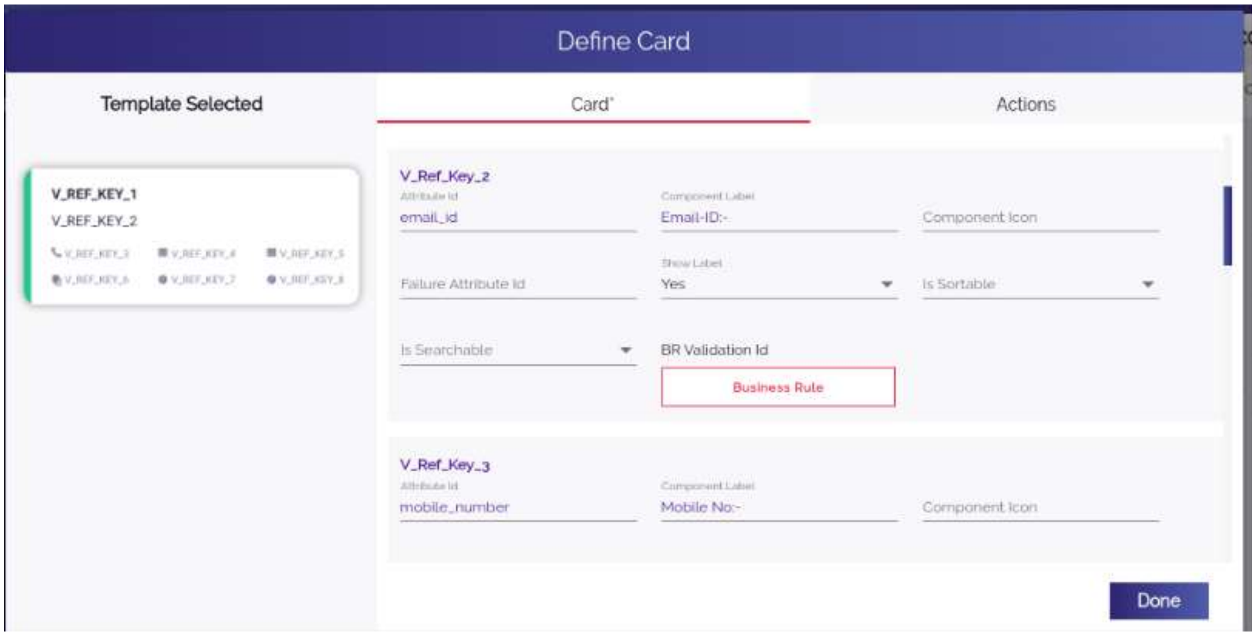
Show Label: Yes

Is Sortable

Is Searchable

BR Validation Id

(Fig 3.9.3.1 (b))



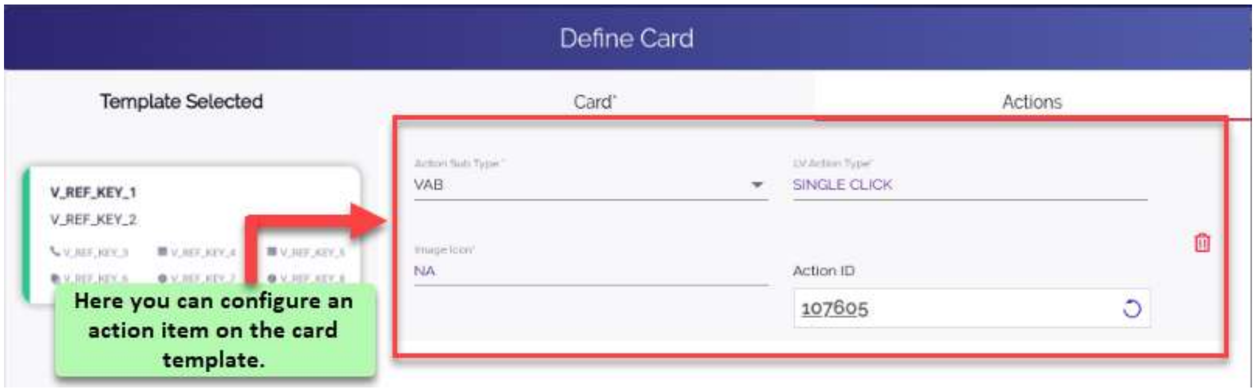
V_Ref_Key_2

Attribute Id: email_id
Component Label: Email-ID:-
Component Icon
Failure Attribute Id
Show Label: Yes
Is Sortable
Is Searchable
BR Validation Id

V_Ref_Key_3

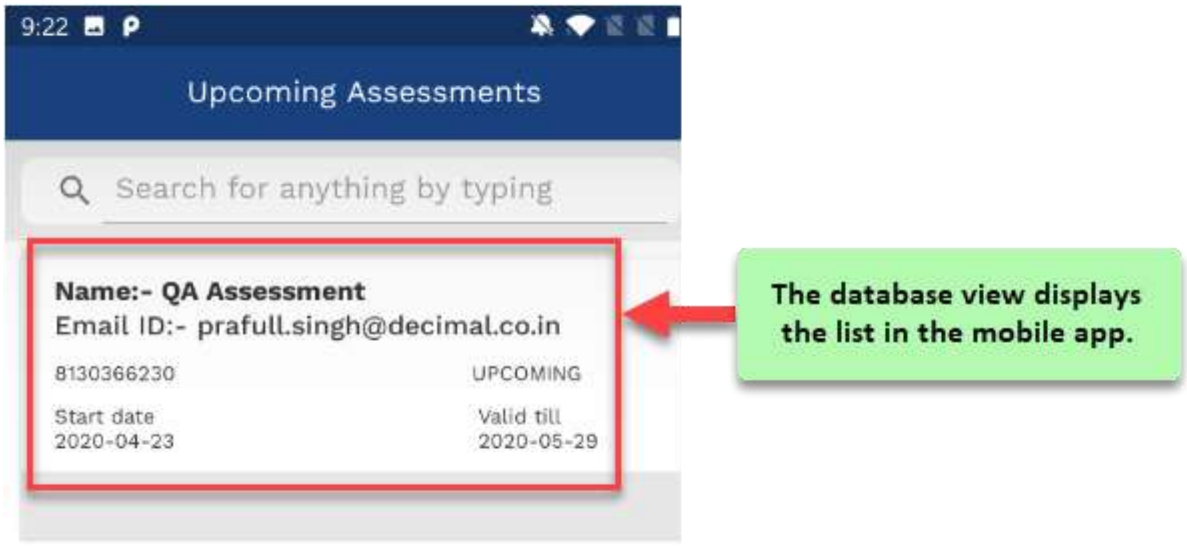
Attribute Id: mobile_number
Component Label: Mobile No:-
Component Icon

(Fig 3.9.3.1 (c))



Action Sub Type*: VAB
LV Action Type*: SINGLE CLICK
Image Icon*: NA
Action ID: 107605

(Fig 3.9.3.1 (d))


9:22
Upcoming Assessments
Search for anything by typing
The database view displays the list in the mobile app.

Name:- QA Assessment
Email ID:- prafull.singh@decimal.co.in
8130366230 UPCOMING
Start date 2020-04-23
Valid till 2020-05-29

(Fig 3.9.3.1 (e))

3.10 Configuring Hamburger Drawer

In mobile apps, hamburger drawer is a well-known feature that is prevalently used to add multiple features and functionality in orderly manner. Hamburger drawer is also referred to as Hamburger menu. In mobile apps, three horizontal parallel lines (☰) are used as the icon of the Hamburger menu.

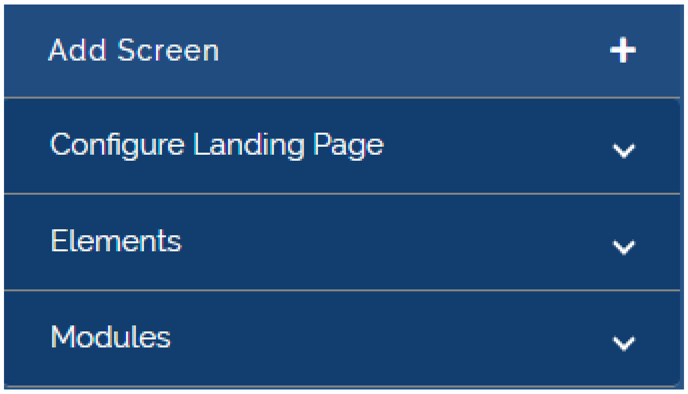
Under Hamburger menu, you can add multiple features and functions. A few functions can be very common among enterprise mobile apps such as My Profile, 24X7 Help, Terms and Conditions, Change Password, Log out, and others. Apart from these common features, you can add other functions based on the type of mobile app and the segment in which mobile app serves to the end-user.

Adding functions to Hamburger drawer depends on the navigation model that the mobile app developers design and develop to make the mobile app more interactive and easily accessible. Hence, the hamburger menu can includes from very frequently accessed features to other unique features that the mobile app user hardly finds to search and use.

When you tap the icon (☰) of the Hamburger menu, it expands and displays all added features in specific order.

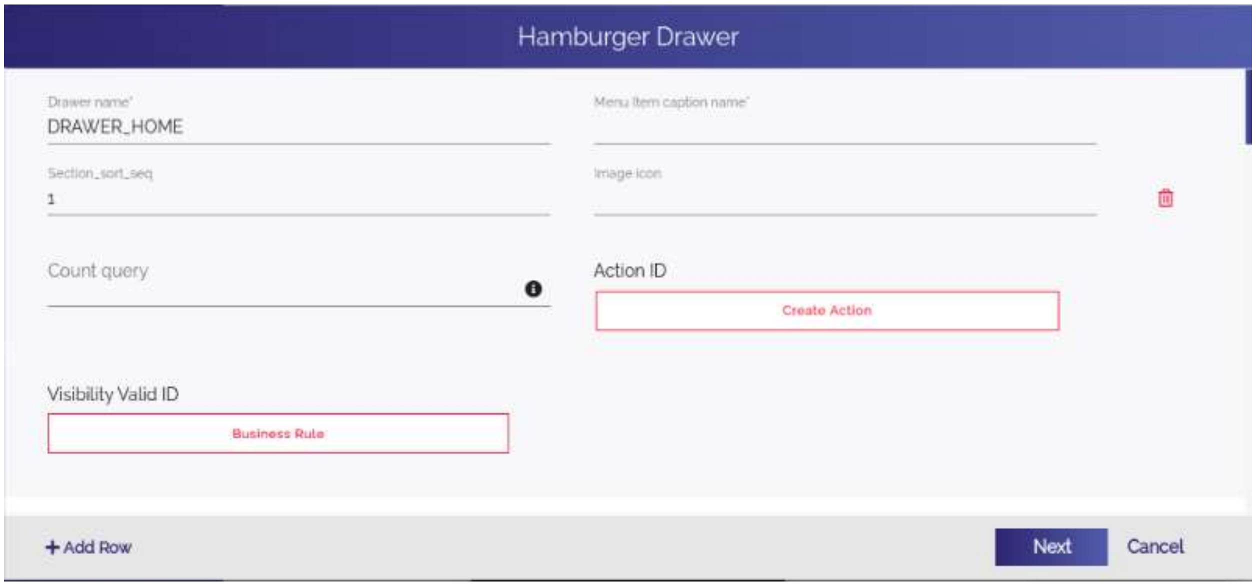
To configure Hamburger menu:

1. In the vDesigner application, locate the left navigation pane.



(Fig 3.10 (a))

2. In the left navigation pane, click **Modules** and then click **Hamburger Drawer**, the **Hamburger Drawer** dialog box **(Fig 3.10 (a))** opens.



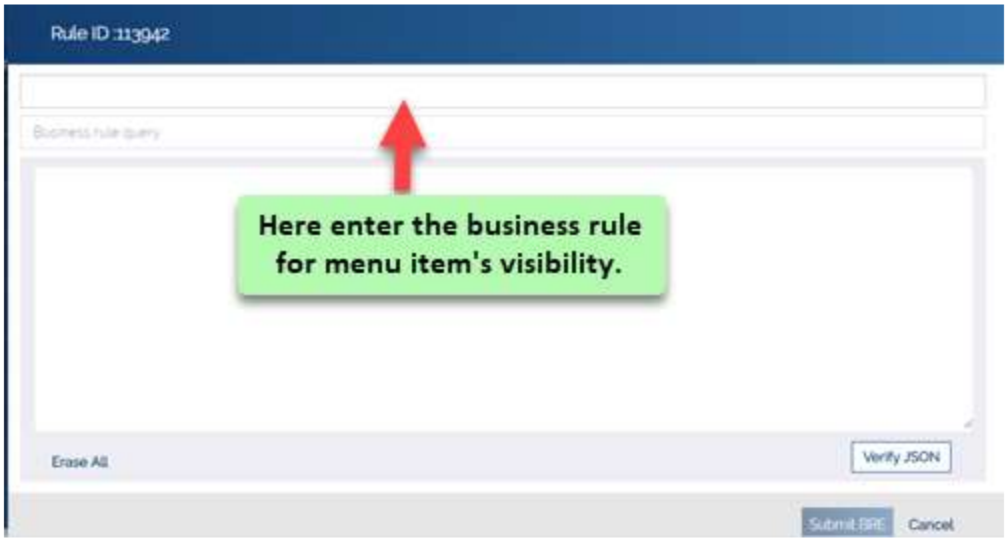
(Fig 3.10 (b))

Note:-
The vDesigner application provides DRAWER_HOME as a default inbuilt drawer. You can either desirably use the **DRAWER_HOME** hamburger menu on different pages of the mobile app, or you can create a new hamburger menu.

3. On the **Hamburger Drawer** dialog box, enter values in the respective boxes as follows:

Field	Description
Drawer name	In this field, you can perform any of the following functions: ➢ If this field contains the value: **DRAWER_HOME** and you want to use the **DRAWER_HOME** hamburger. You need not to change value. If you want to create a new hamburger drawer, enter the name of new hamburger menu ((For example: - DRAWER_LOANAPP). ➢ If you add second or any other menu item, this field displays blank value. To use inbuilt drawer, enter **DRAWER_HOME**. To use new hamburger drawer, enter the name of new hamburger menu (For example: - DRAWER_LOANAPP).
Menu Item Caption Name	In this field, enter the name of the menu item (For example: - Change Password or My Profile). The name of the menu item is displayed under the hamburger menu.
Section_sort_seq	In this field, enter the order number of menu item. For Example:- In the **Section_sort_seq** box, if you enter **4**, the hamburger menu will display the respective menu item as 4th menu item from the top of the hamburger menu.

Image	In this box, enter the name of icon that you want to use as the icon of the respective menu. **<u>Note</u>:-** Different icons that you can use for menu items are stored on the Vahana server.
Action ID	You can use this function to assign an action to the menu item. After you assign the action to the menu item, you can perform the desired function based on the action that you have assigned. How to assign the action to the hamburger menu, visit the heading section: Assigning Action to Hamburger Menu.
Visibility Valid ID	You can use this feature to apply the visibility rule to the menu item. If the visibility business rule satisfies, only then the menu item will become visible. To apply the business rule: 1. Click **Business Rule**, the **Rule ID** dialog box **(Fig 3.10 (c))** opens. 2. In the upper box, enter the business rule. 3. After you enter the business rule, click **Submit BRE**, the business rule is successfully applied. Though, this feature is optional. If you do not want to apply the business rule, you need not to click **Business Rule**.

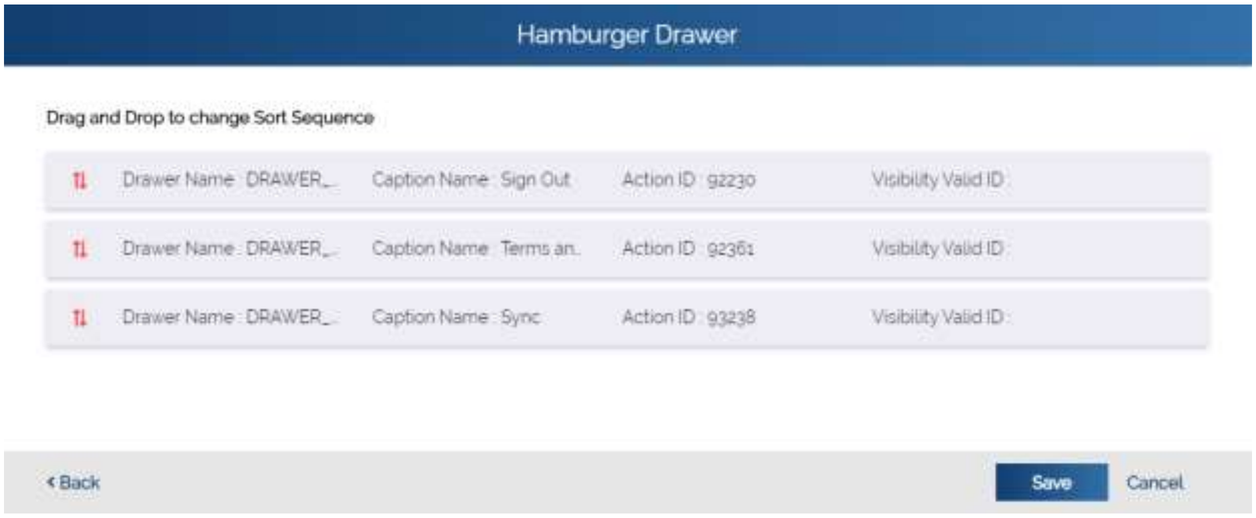


(Fig 3.10 (c))

4. After you enter the values in the respective boxes and assign action to the respective menu item, click **Add Row (Fig 3.10 (b))**, the Hamburger Drawer dialog box displays another row of the following fields:

- **Drawer name**
- **Menu Item Caption Name**
- **Section_sort_seq**
- **Image icon**
- **Count query**
- **Action ID**
- **Visibility Valid ID**

5. In these fields, enter the details of another menu item as described in the last table.
6. After you add all menu items to the hamburger menu, click **Next**, the **Hamburger Drawer** dialog box displays the list of added menu items, in addition to the details of each menu item.



Drawer Name : DRAWER_	Caption Name : Sign Out	Action ID : 92230	Visibility Valid ID :
Drawer Name : DRAWER_	Caption Name : Terms an..	Action ID : 92361	Visibility Valid ID :
Drawer Name : DRAWER_	Caption Name : Sync	Action ID : 93238	Visibility Valid ID :

(Fig 3.10 (d))

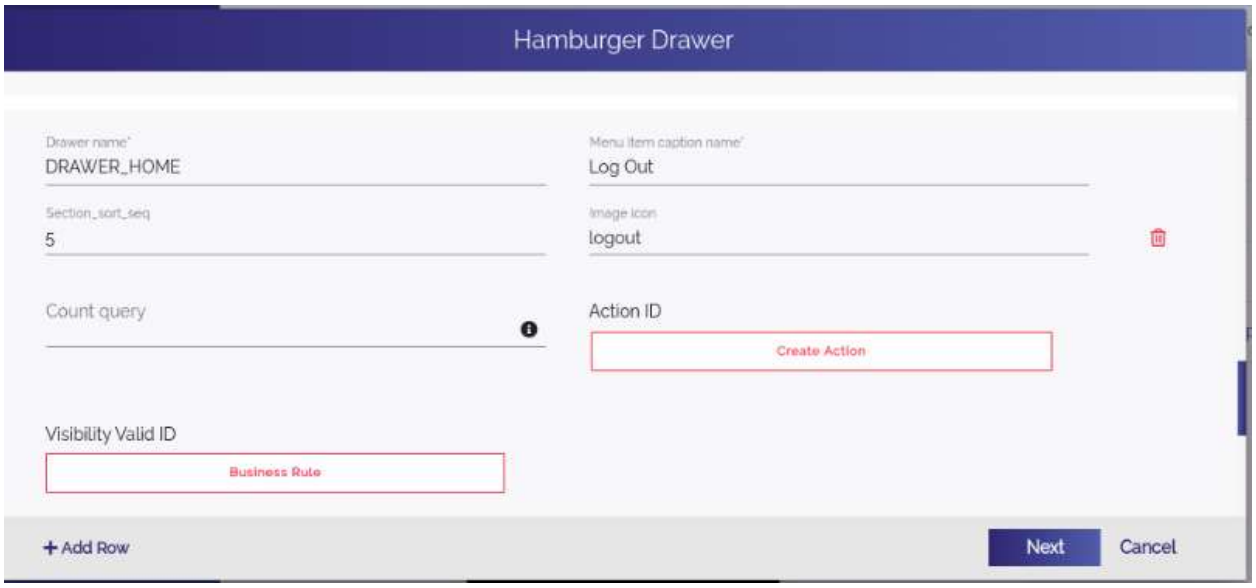
7. After you make sure that you have added menu items correctly, click **Save (Fig 3.10 (d))**, hamburger drawer is successfully configured.

3.10.1 Adding Action to Hamburger Drawer

Under hamburger drawer, you assign an action to the hamburger menu item to impart functional behavior to it. After an action is assigned to the hamburger menu item, it performs the function based on the action that is assigned to it. The action is assigned to the menu item based on the functional requirement of that menu item. You can assign an action to the menu item as follows:

To assign the action:

1. On the **Hamburger Drawer** dialog box **(Fig 3.10.1 (a))**, locate the menu item to assign an action to it.

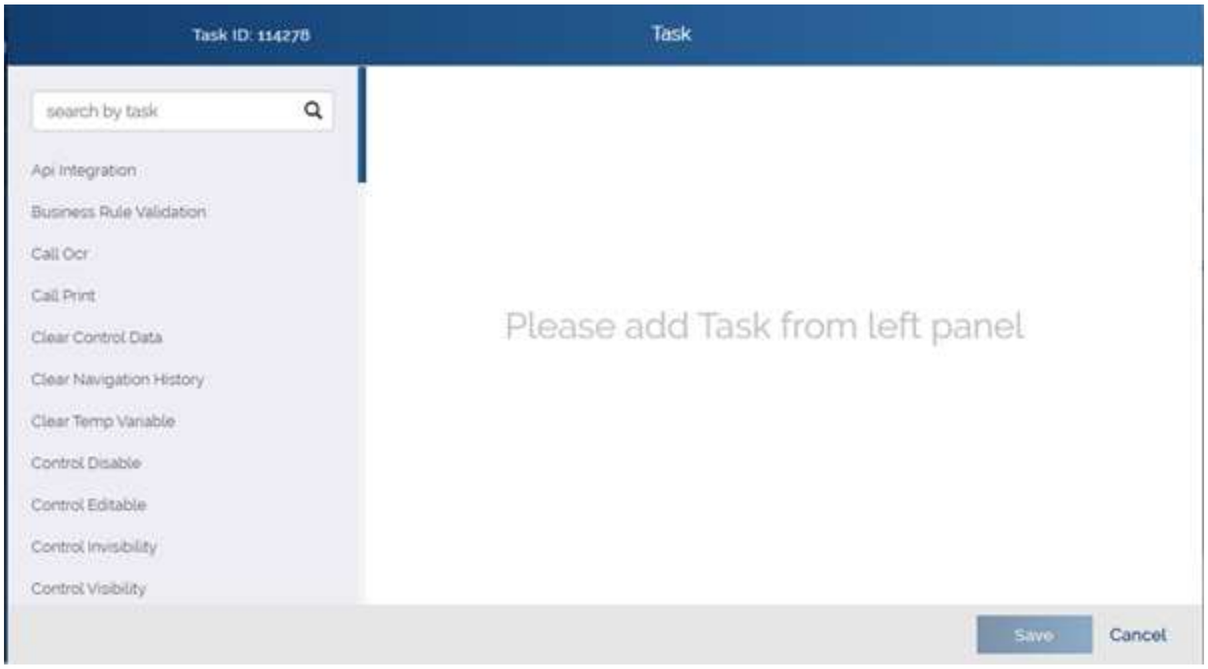


Hamburger Drawer

Drawer name*: DRAWER_HOME
Menu item caption name*: Log Out
Section_sort_seq: 5
Image icon: logout
Count query
Action ID: Create Action
Visibility Valid ID: Business Rule

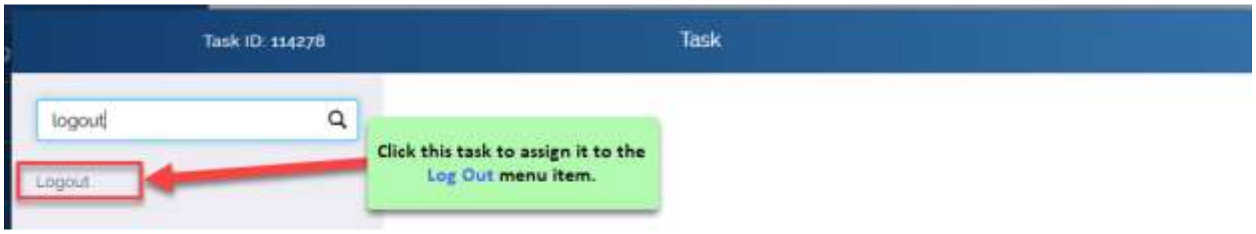
(Fig 3.10.1 (a))

2. After you locate the menu item (For example: - Log Out), click **Create Action (Fig 3.10.1 (a))**, the **Task** dialog box **(Fig 3.10.1 (b))** opens.



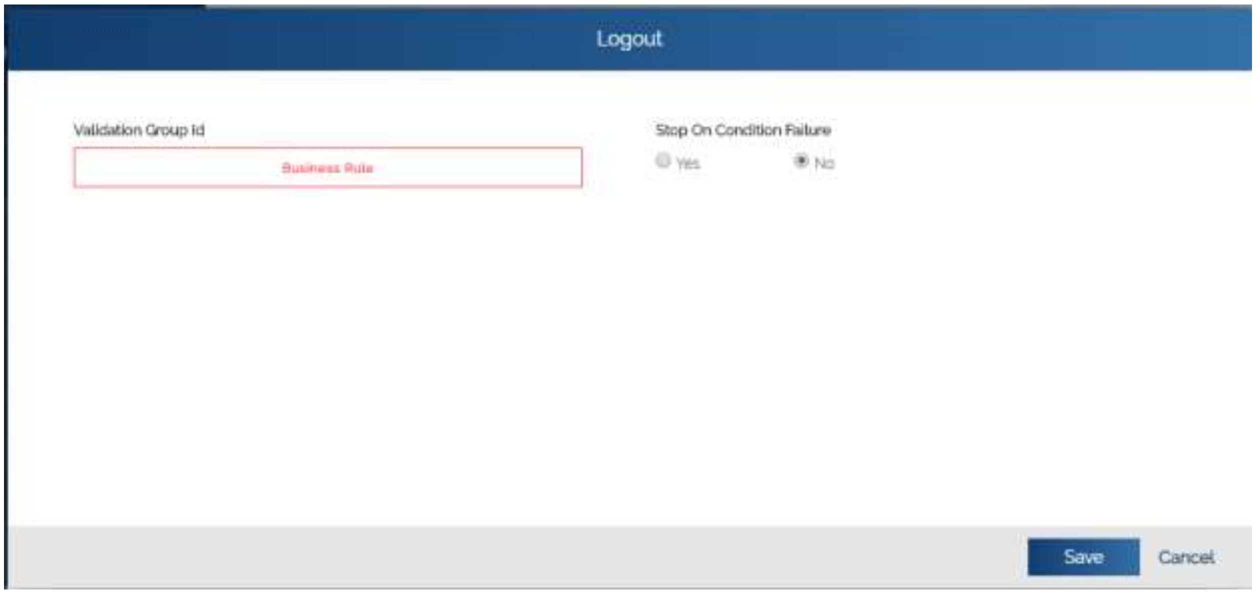
(Fig 3.10.1 (b))

3. In the **search by task** box **(Fig 3.10.1 (b))**, enter the name of task (For example: - Logout) that you want to assign, the left pane displays the task.



(Fig 3.10.1 (c))

4. In the left pane, click the task to select it, the **Logout** dialog box **(Fig 3.10.1 (d))** opens.

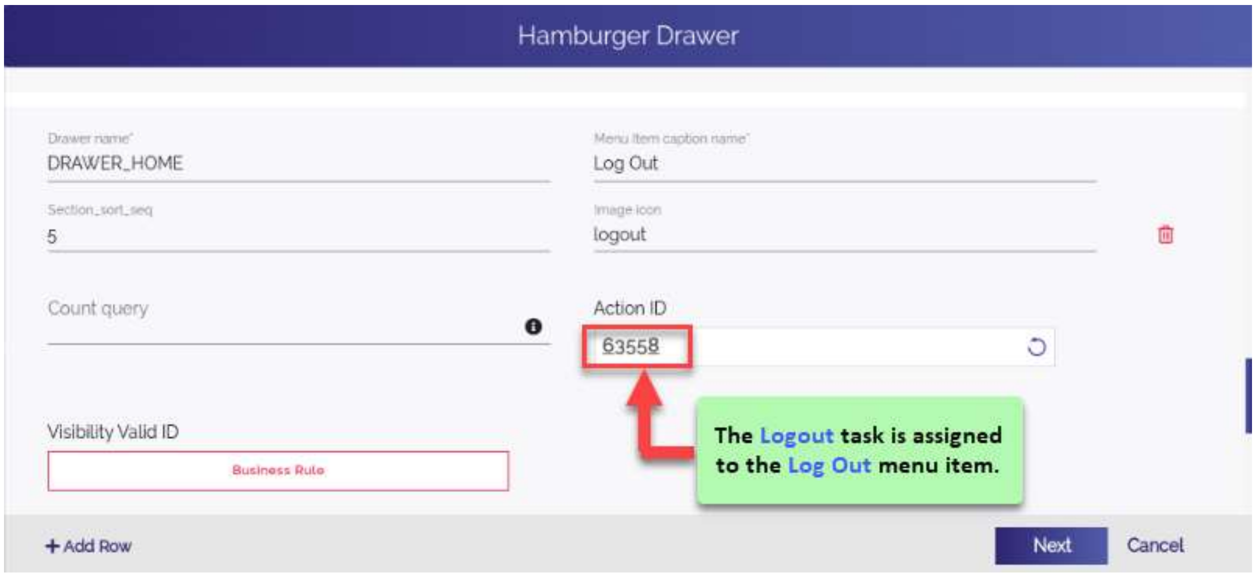


Validation Group Id: Business Rule

Stop On Condition Failure: Yes [✓ No]

(Fig 3.10.1 (d))

5. On the **Logout** dialog box, click **Save (Fig 3.10.1 (d))**, the **Logout** task is successfully assigned to the **Log Out** menu item.



Drawer name*: DRAWER_HOME

Menu Item caption name*: Log Out

Section_sort_seq: 5

Image icon: logout

Count query:

Action ID: 63558

Visibility Valid ID: Business Rule

(Fig 3.10.1 (e))

Likewise, you can apply other tasks based on the functional requirement of specific menu item. A few tasks are briefly described as below:

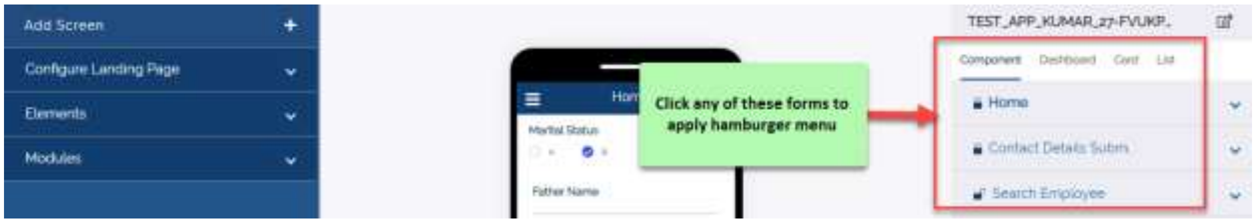
Task	Description
Exit	You can apply this task to exit the application. To apply the **Exit** task: 1. In the **search by task** box, enter **exit** to search the **Exit** task. 2. In the left pane, click **Exit**, the **Exit** dialog box opens. 3. In the **Exit** dialog box, click **Save**, the Exit task is successfully assigned.
Form ID	This task is frequently assigned to the hamburger menu item if you want the user to navigate to specific screen or page of the mobile app. In this case, the form ID of the respective page is mapped to the hamburger menu item. To apply the **Form Id** task: 1. In the **search by task** box, enter **form id** to search the **Form Id** task. 2. In the left pane, click **Form Id,** the **Form Id** dialog box opens. 3. In the **Form Id** dialog box, click in the **Navigation Form Id** box, it displays the list of existing forms, in addition to the form Id. 4. Locate the name of the form that you want to map to the hamburger menu item. 5. After you locate the form in the list, select it, the name of form is displayed in the **Navigation Form Id** box. **6.** Click **Save**, the form is successfully mapped to the menu item.
Reload App Config	Though this task is assigned to reload configuration settings of an application, you can assign this task to a hamburger menu item if you want to provide this feature to the user. After user clicks a menu item to reload the configuration of the app, the **Reload App Config** task executes to load the latest configuration of the mobile app. To apply this task: In the **search by task** box, enter reload app config to search the **Reload App Config** task. In the left pane, click **Reload App Config**, the **Reload App Config** dialog box opens. On the **Reload App Config** dialog box, click Save, the **Reload App Config** task is assigned to the menu item.

3.10.2 *Applying Hamburger Drawer on Form*

While adding multiple menu items to the hamburger drawer, you can desirably have two choices: either you can use default DRAWER_HOME hamburger, or you can create a new hamburger. After you design a hamburger menu by adding new menu items to it and then assigning actions to each of the menu items, you can use the hamburger menu on different pages of the mobile app. To display a hamburger menu on specific page of the mobile app, you need to apply the hamburger menu on the form of the respective screen/page:

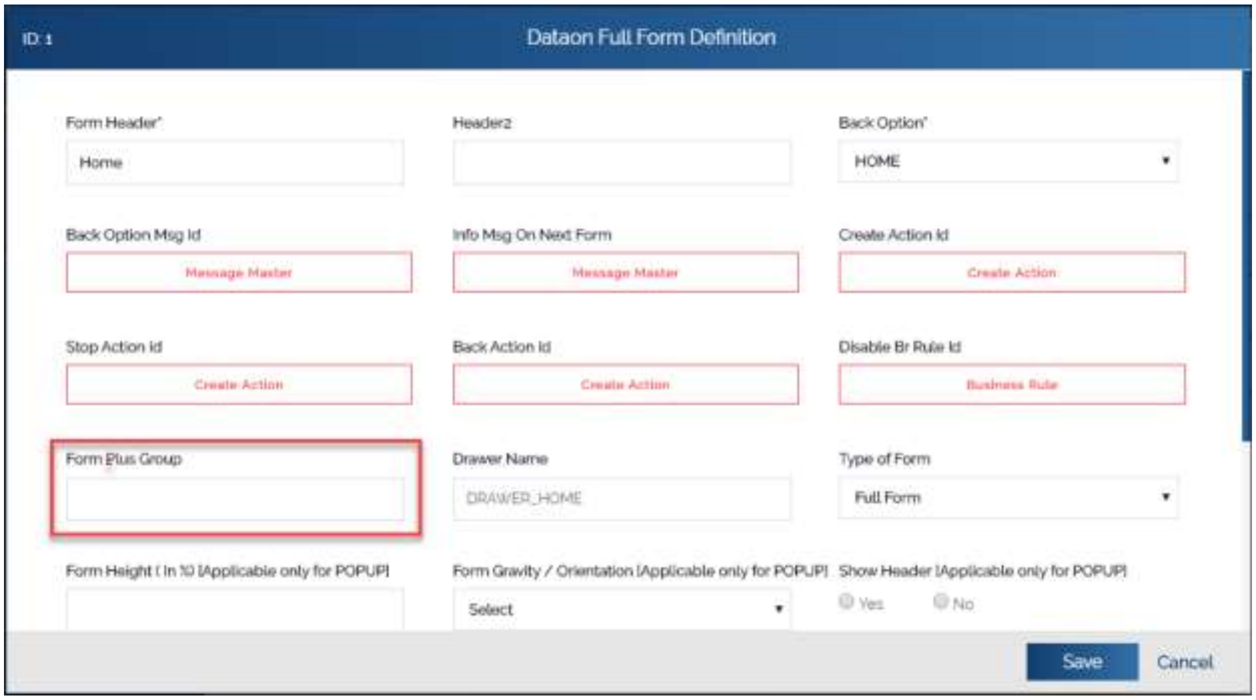
You can apply the hamburger menu on the form as follows:

1. In the vDesigner application's dashboard, locate the right pane.



(Fig 3.10.2 (a))

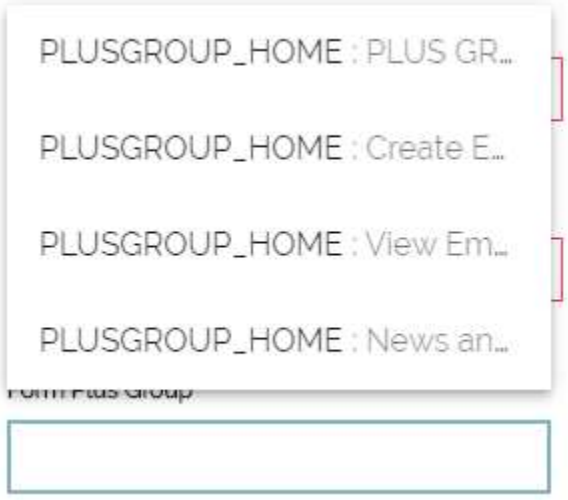
2. In right pane, Move the mouse pointer on the name of the form (For example: - Home) where you want to apply the hamburger menu, three icons appear.
3. Click the edit icon ( ), the **Dataon Full Form Definition** dialog box opens.


ID: 1 Dataon Full Form Definition

Form Header*: Home	Header2:	Back Option*: HOME
Back Option Msg Id: Message Master	Info Msg On Next Form: Message Master	Create Action Id: Create Action
Stop Action Id: Create Action	Back Action Id: Create Action	Disable Br Rule Id: Business Rule
Form Plus Group:	Drawer Name: DRAWER_HOME	Type of Form: Full Form
Form Height (In %) (Applicable only for POPUP)	Form Gravity / Orientation (Applicable only for POPUP): Select	Show Header (Applicable only for POPUP): Yes No

(Fig 3.10.2 (b))

4. On the **Dataon Full Form Definition** dialog box, locate the **Form Plus Group** box **(Fig 3.10.2 (b))**.
5. Click in the **Form Plus Group** box, a list displays existing plus groups **(Fig 3.10.2 (c))**.


PLUSGROUP_HOME : PLUS GR...
PLUSGROUP_HOME : Create E...
PLUSGROUP_HOME : View Em...
PLUSGROUP_HOME : News an...
Form Plus Group

(Fig 3.10.2 (c))

6. In the list, click to select the plus group that you want to apply on the respective form.
7. After you select the plus group, click **Save (Fig 3.10.2 (b))**, the plus group is applied on the form.

****