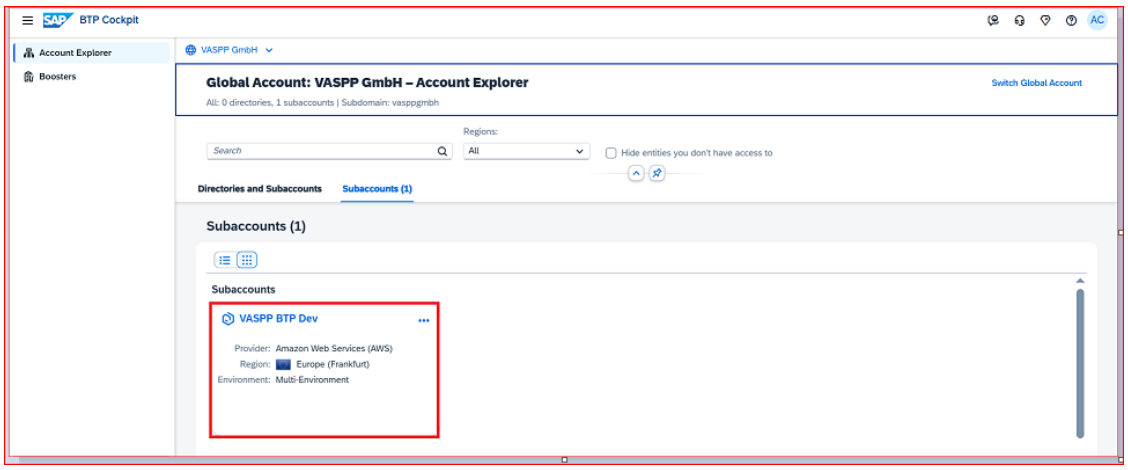
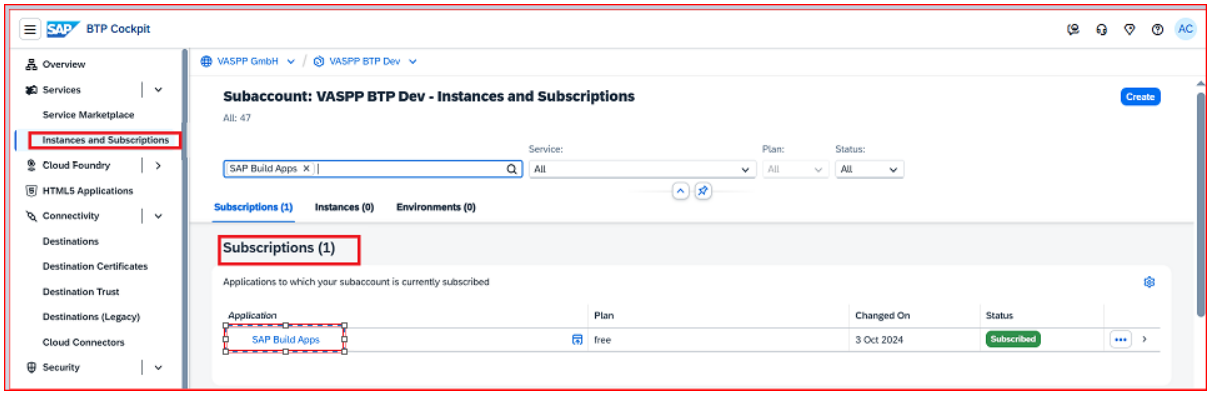
Exercise 1 : Creating Project in SAP Build Apps

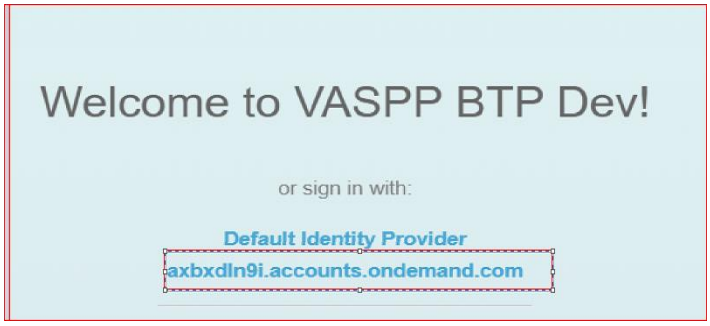
1. Login to your **SAP BTP Global Account** and select the **sub-account**



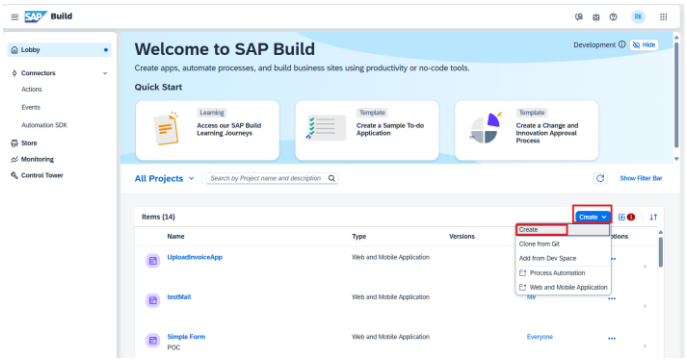
2. Within the sub-account you have selected under **Instances and Subscriptions** select **SAP Build APPS**



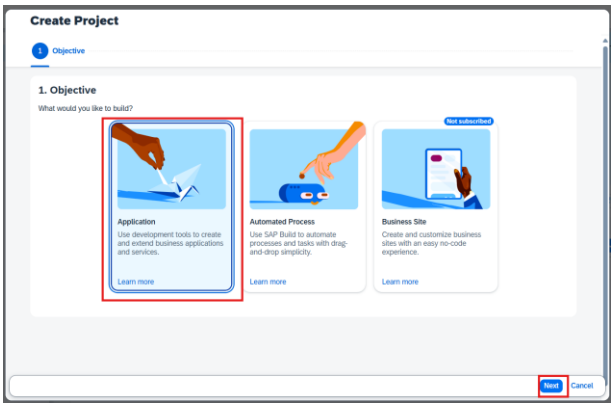
3. You will be redirected to login page of SAP Build apps and there select the **custom IAS account**



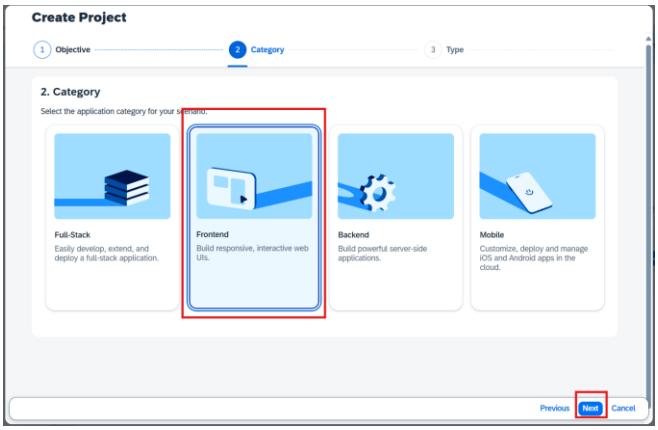
4. Once you login to SAP Build Apps lobby click on **create** option to create the project



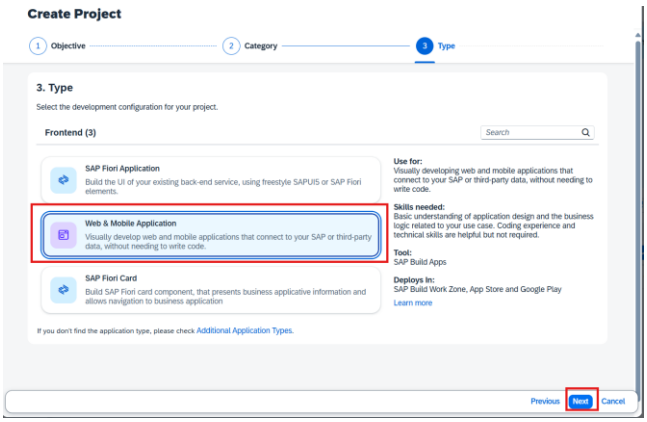
5. Click on **Application** and **Next**



6. Click on **Front end** and **Next**

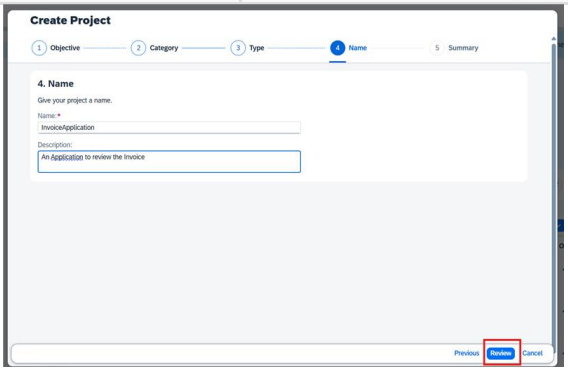


7. Click **Web & Mobile Application** and click **Next**.

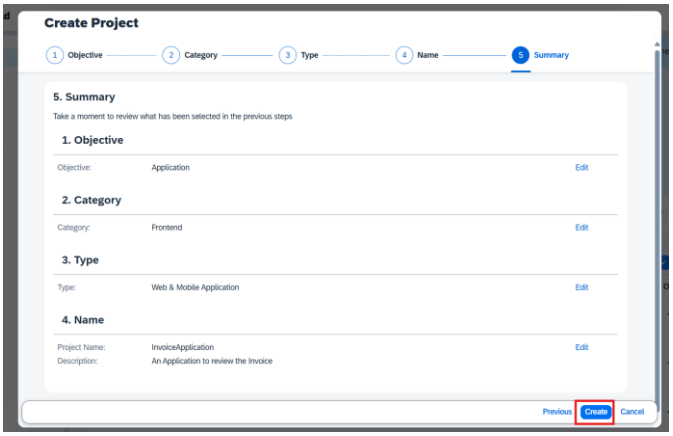


8. Name the application and provide the description and click on review

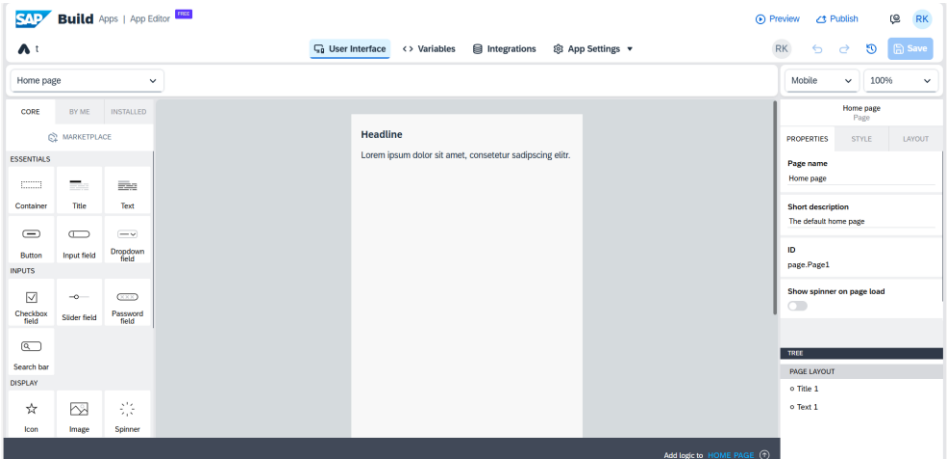
Field	Value
Name	**InvoiceApplication**
Description	**An Application to review the invoice**



On the **summary page** click on **create**



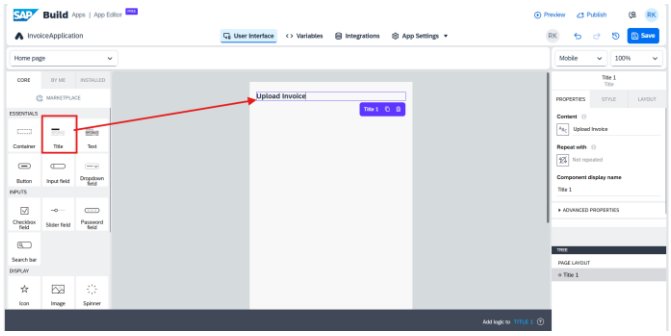
9. Now you have created the project, and you will be able to see the home Page



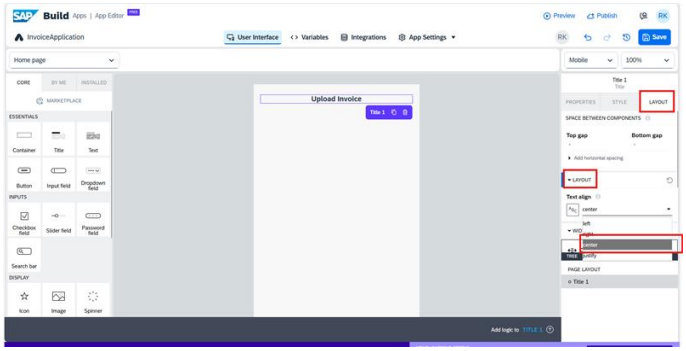
Exercise 2: Build Project in SAP Build Apps

Creating the Upload Invoice Page(Default Home Page)

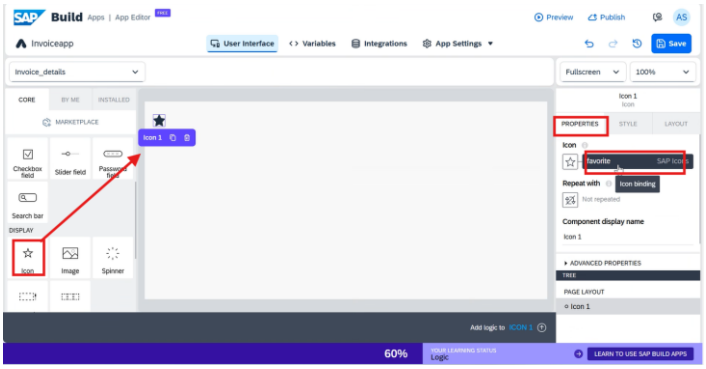
1. Under core -> Essentials Drag and drop the **Title,** double click to edit the name as **Upload Invoice**



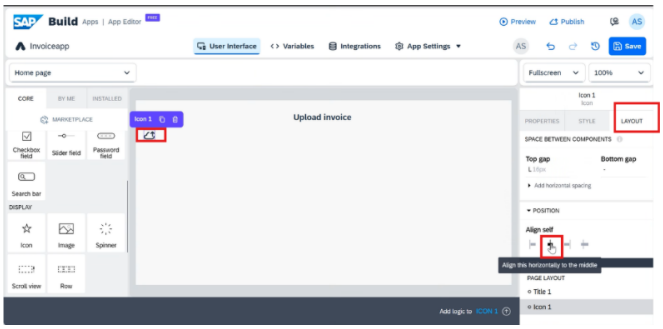
2. Towards the right-hand side under layout text align it to centre



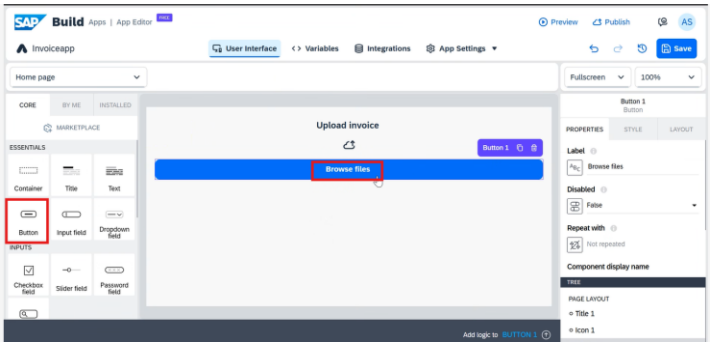
3. Under core -> Display Drag and drop the icon, click on properties, under favourites search for upload icon.



Once you select, the upload icon under layout -> position -> align self -> centre

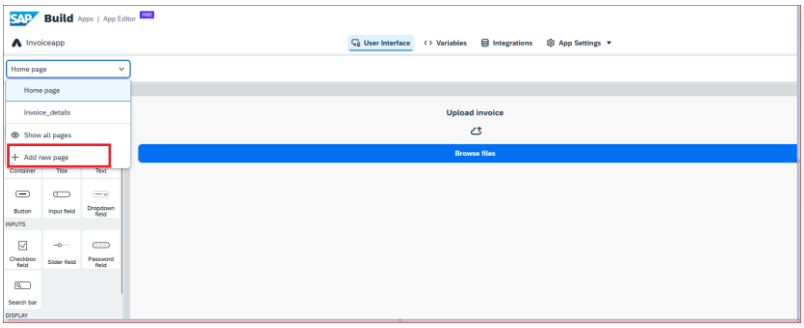


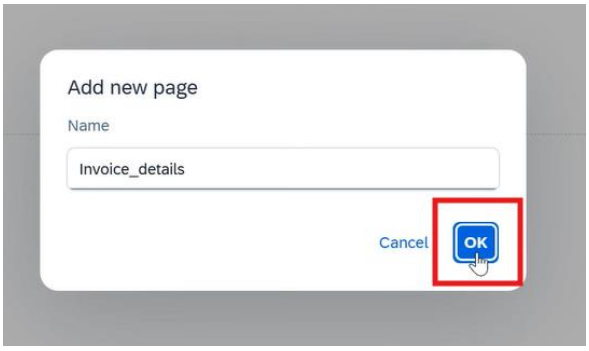
4. Under core -> Essentials Drag and drop the **Button**, double click to edit the name as **Browse File**



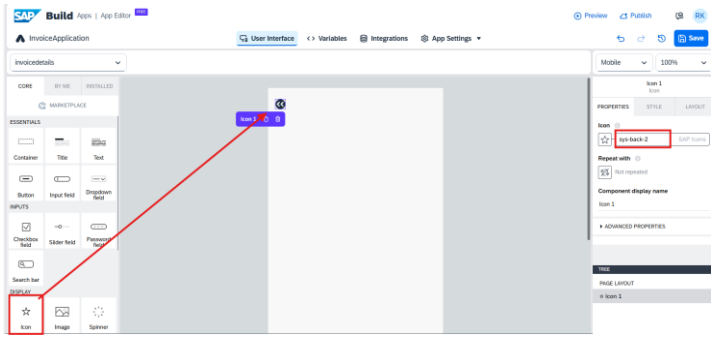
Creating Invoice Review Page

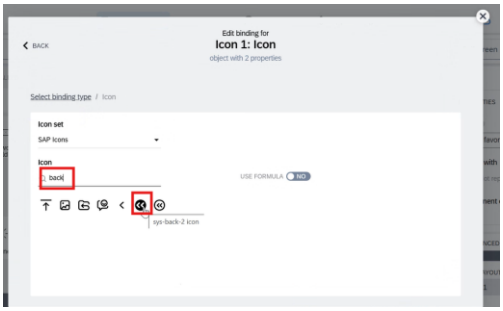
1. Towards the top left-hand side there is drop down displaying **Home page** click on it.
2. Select **add new page** and provide the **Invoice_details**



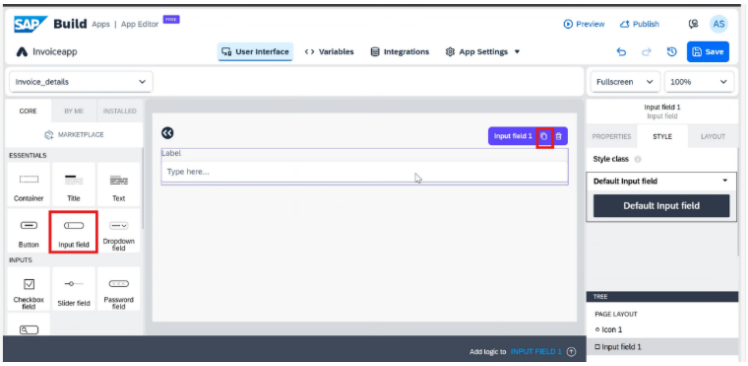


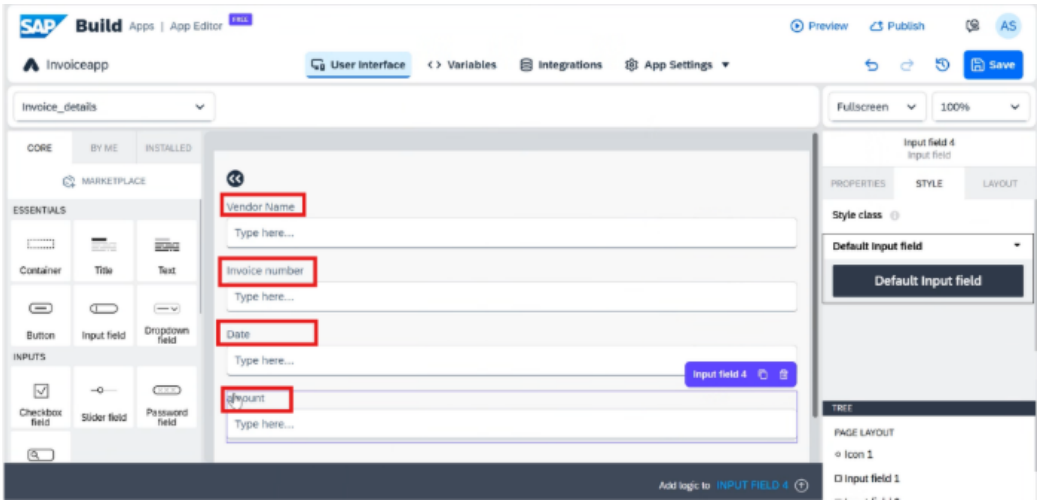
3. Under core -> Display Drag and drop the **icon**, click on properties, under favourites search for **back icon**.



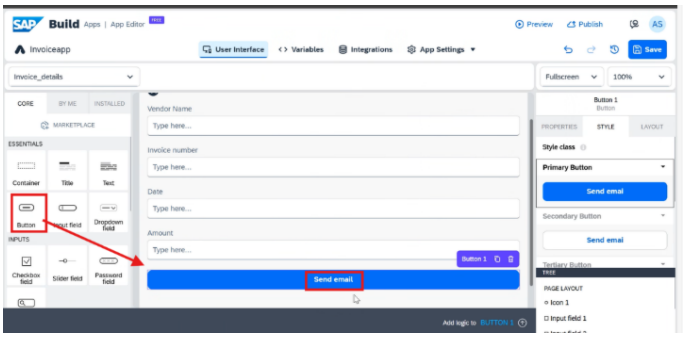


4. Under core -> Essentials Drag and drop the **Input Filed**, double click to edit the name as and provide the input labels and similarly copy/ duplicate for other input fields
The field names are: Vendor, Invoice Number, Invoice Date, Total Amount



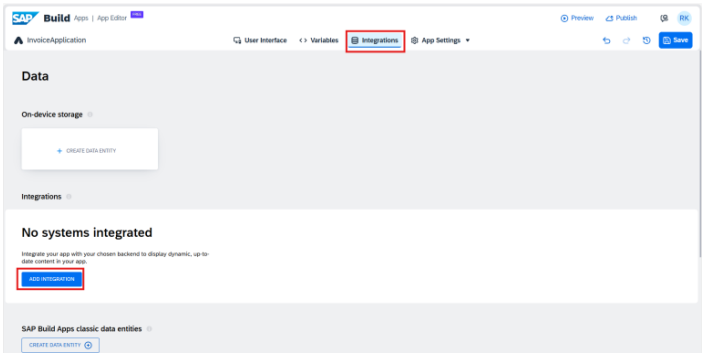


5. Under core -> Essentials Drag and drop the **Button**, double click to edit the name as **Send Mail**

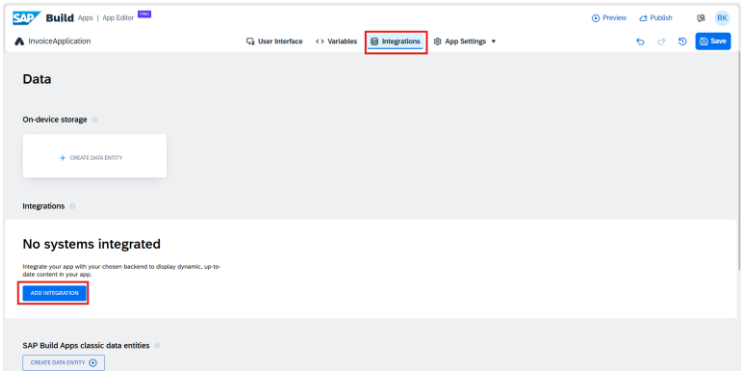


Exercise 3: Integration of SAP Document AI

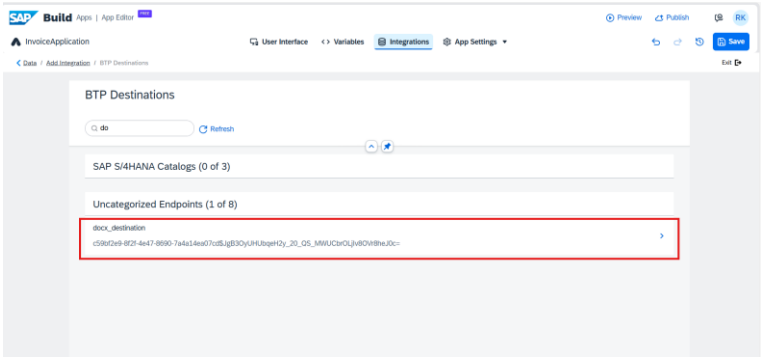
1. Select **Integration tab** and click on **Add Integration**



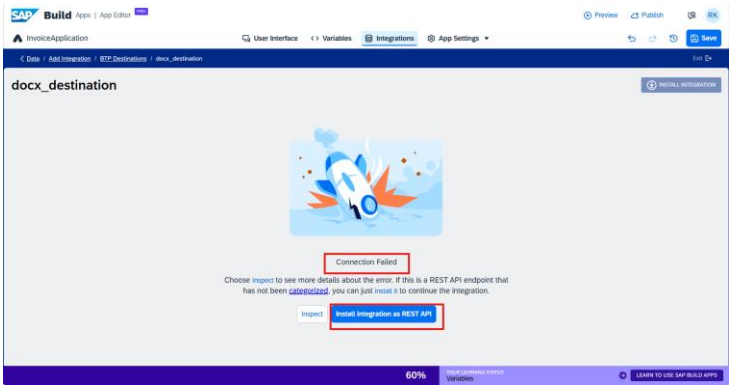
2. Under SAP System select SAP BTP Destinations



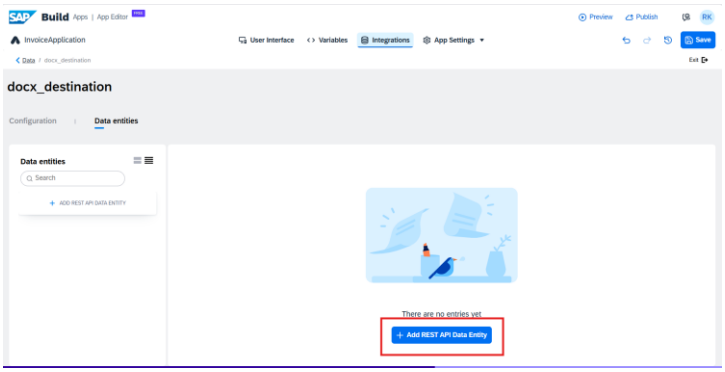
3. We can see many destinations that our created in out BTP Sub-account. Search for **Docx_Destination** select it and save



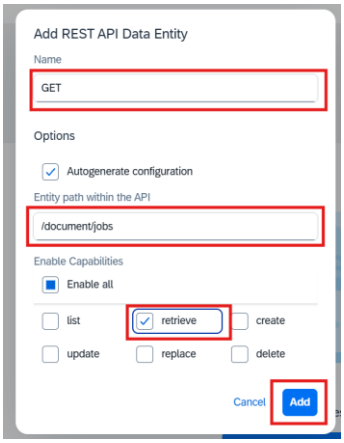
4. As soon as you click it below screen will be displayed, the connection failed is not an error, minimise the screen so you will be able to see the **Install integration a REST API** option click on it



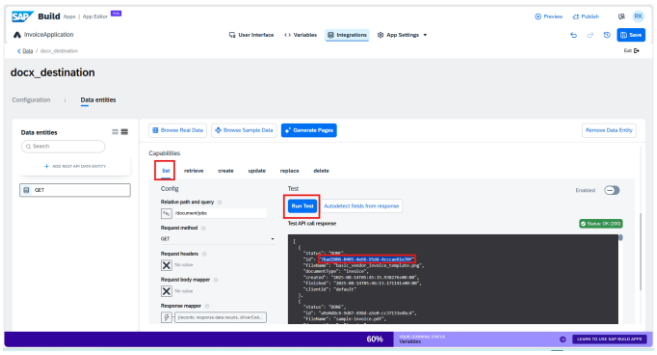
5. Click on **+ Add REST API Data Entity**



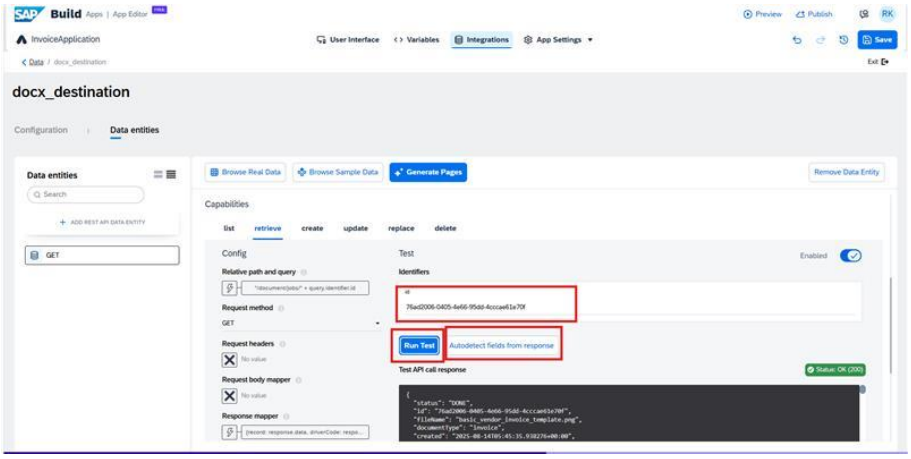
6. We will get a pop up to file the below details
   - Name: GET
   - Check the Autogenerated configuration option
   - Entity path within the API: /document/jobs
   - Enale Capabilites: Check only retreive option
   - And finally flick on ADD button



7. Once you save the details you will be able to see the updated screen, scroll down to test the API.
   Under **list** click on **Run Test** to check the status and you will be able to see **Status Ok(200)** and the list of posted document's data.
   Now copy the ID of one of the document data so we use it next step.

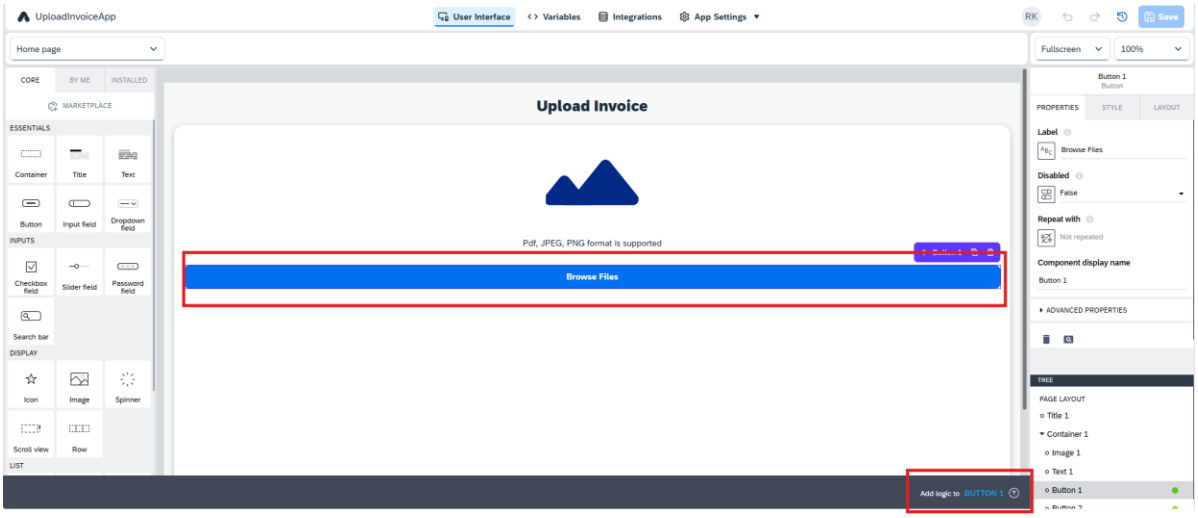


8. Now go to **retrieve tab** paste the ID that you have copied previously under ID and click on Run test.
   Once you get the run test result as **status Ok (200)** click on **Autodetect fields from response** and finally save it

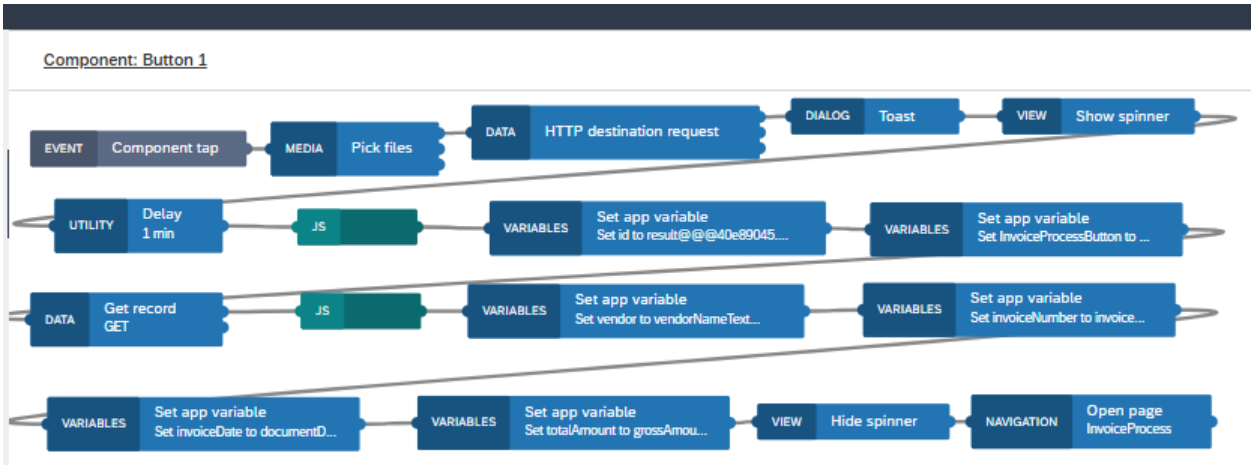


Exercise 4 - Integration of UI

Step 1: Click on Browse Files Button to add custom logic.

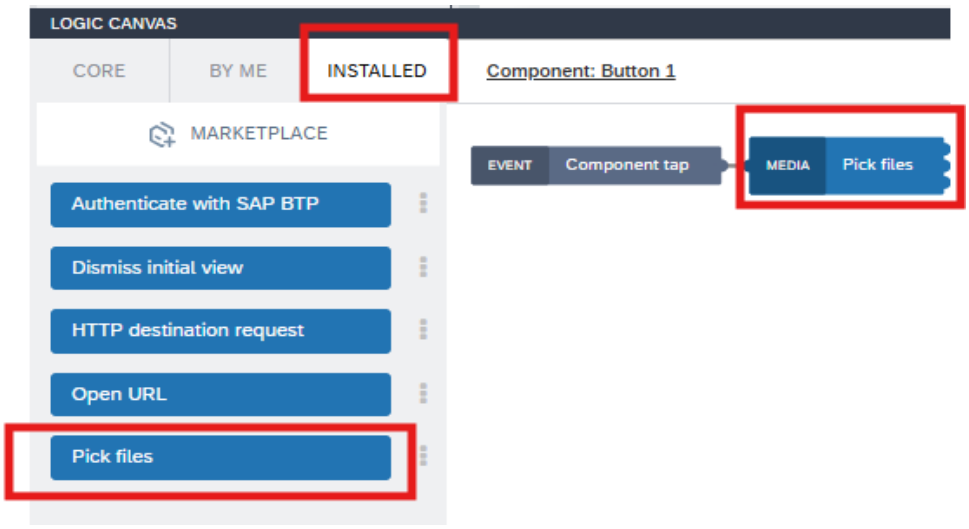


Step2: To Add logic please click on Logic editor with name “Button 1”

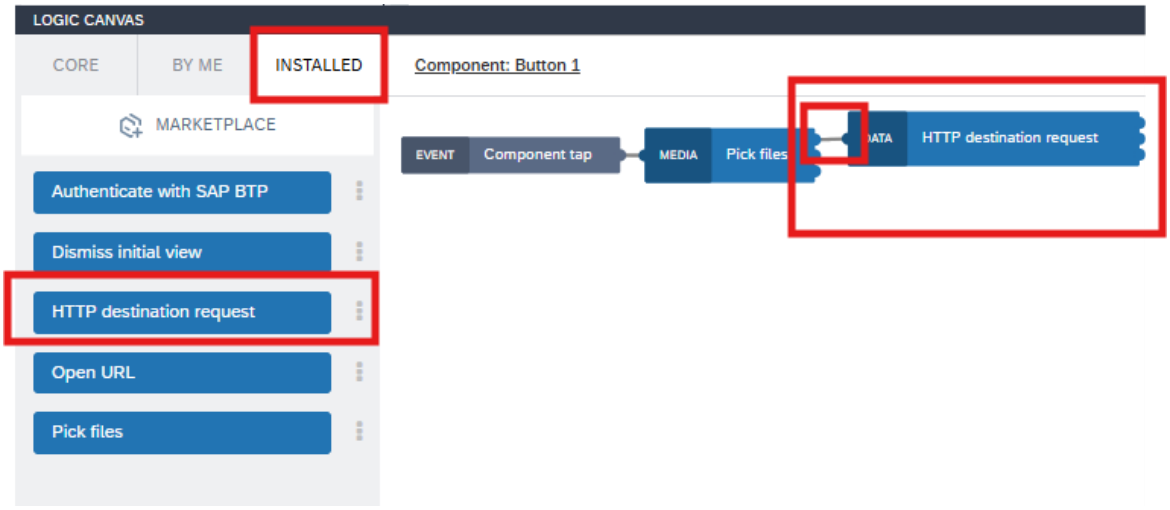


The above are the components I added for browsing File. Now Let us add one by one

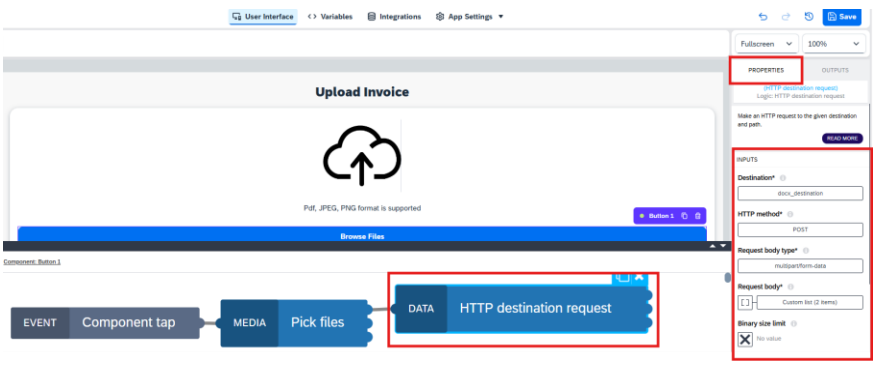
Step3: Under Logic Canvas, Go to marketplace and search for Pick Files install it and similarly install HTTP destination request and Open URL .In Logic Canvas (left side-bottom), go to Installed tab and select **“Pick Files**” and drop it in Logic editor.



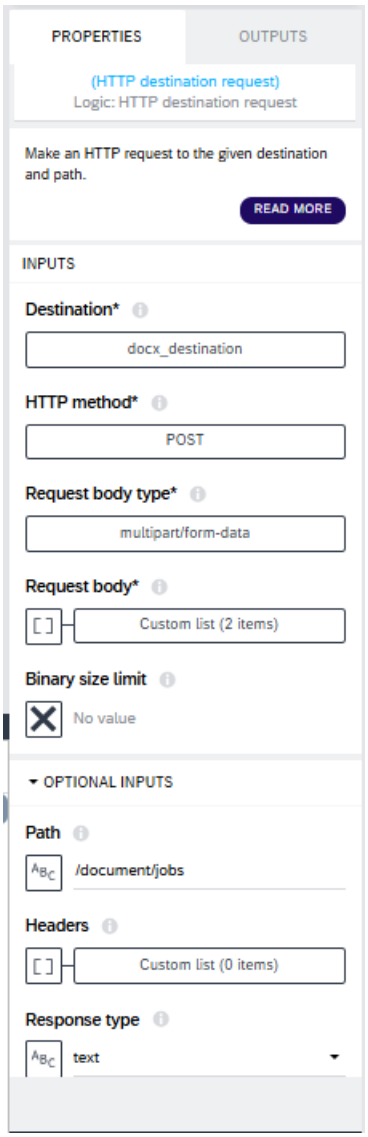
Step4: Next drag and drop “HTTP destination request from Installed tab and connect it to the previous node “Pick Files”



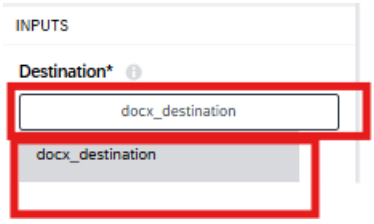
Step5: Click on Component “HTTP destination request” and on right side you can view the Properties Panel.



Step6: Now we need to fill in the details, one by one.

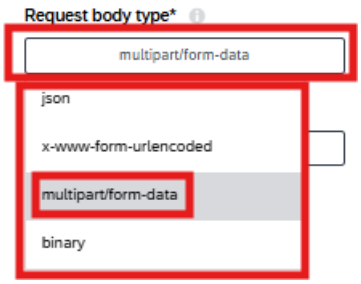


Step7: Destination: On click of destination drop down you see the list already available which we added via integrations. Select the **DocX_destination from the drop down** which we created in SAP BTP Cockpit destination and add in integration in SAP Build Apps.

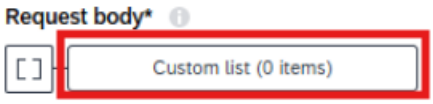


Step8: HTTP Method: Select the method **“POST”** from drop down as we upload the invoice to Document AI Service.

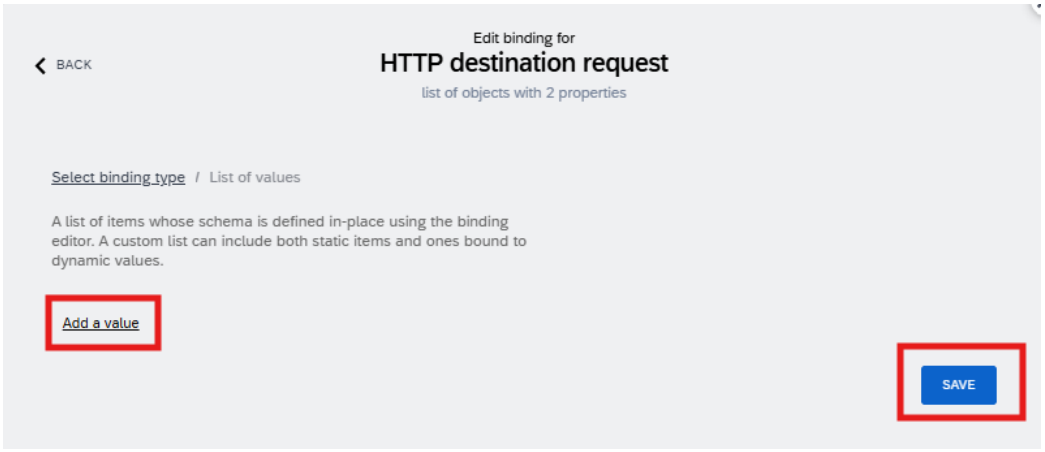
Step9: Request Body Type: select type **“multipart/form-data”.**



Step10: Request Body: Click on Custom List a window opens.



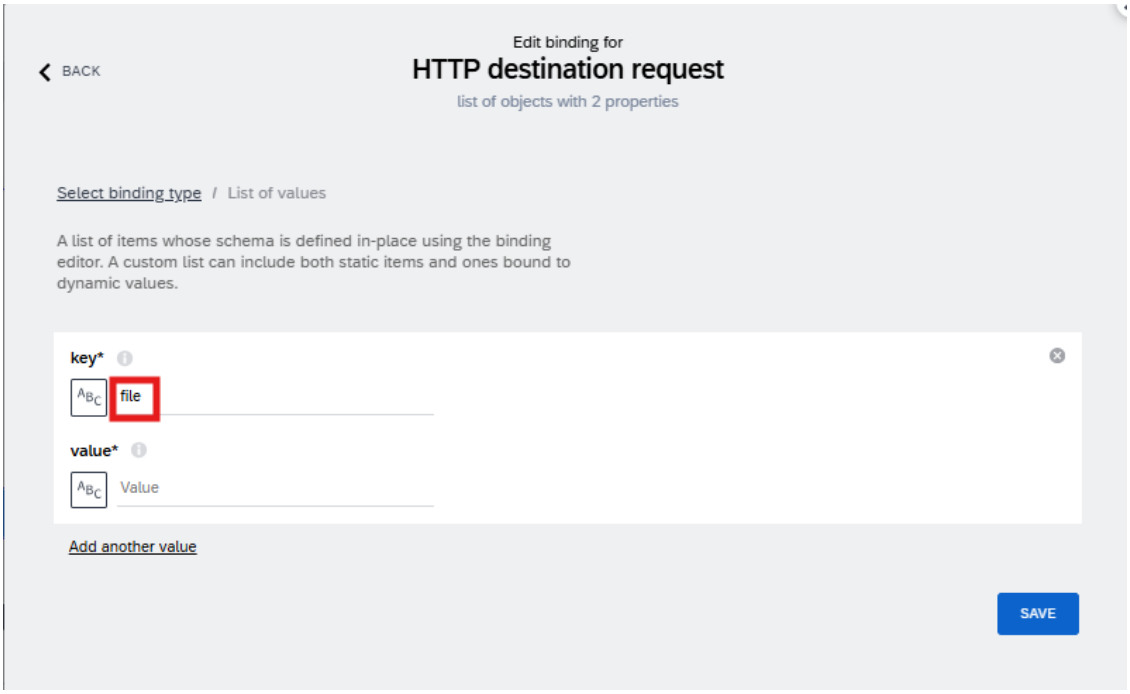
Step11: Click on Add Value



Step12: Please add below values:

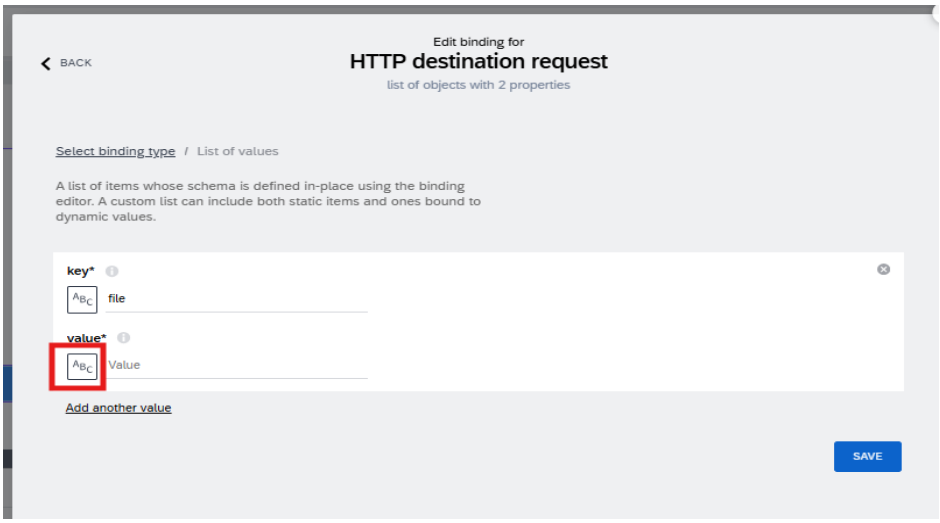
First Value:

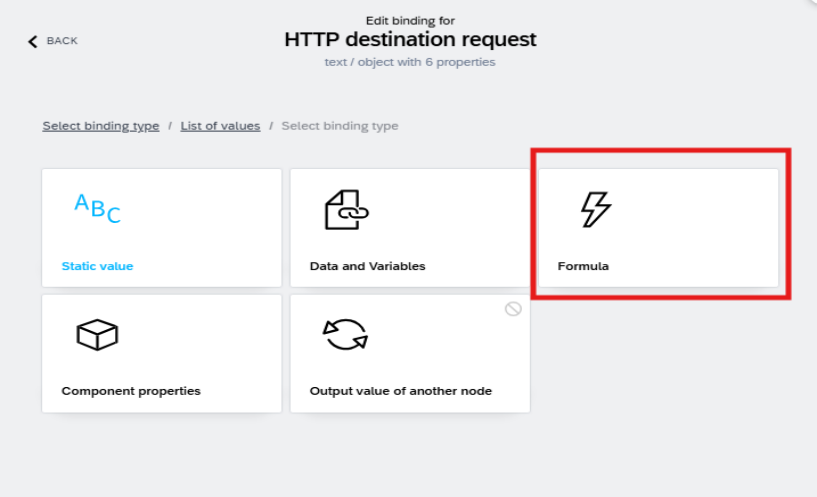
Key: file

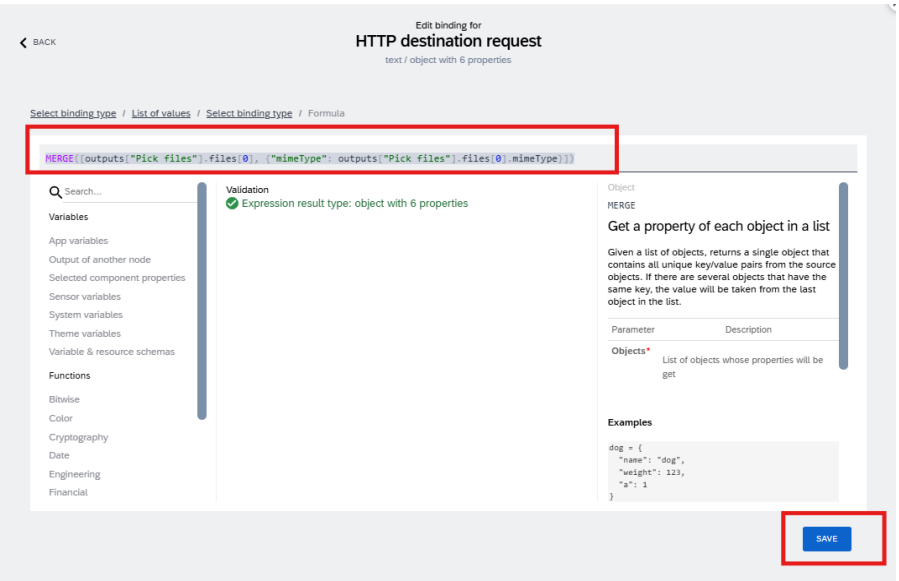


Step13: First Click on Value --> Select Formula --> Paste given Value

Code Snippet 1 **Value**:
MERGE([outputs["Pick files"].files[0], {"mimeType": outputs["Pick files"].files[0].mimeType}])







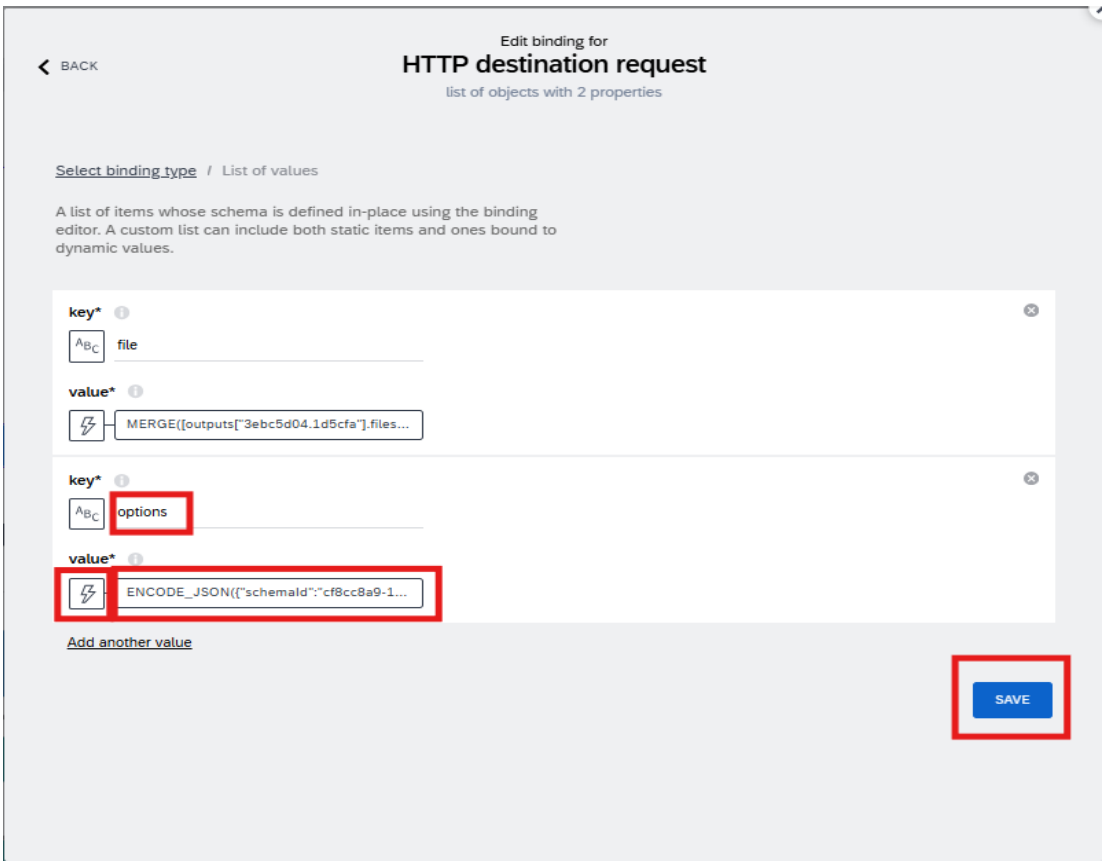
Step 14: Now add another value just like the previous Step (First Click on Value --> Select Formula --> Paste given Value).

Second Value:

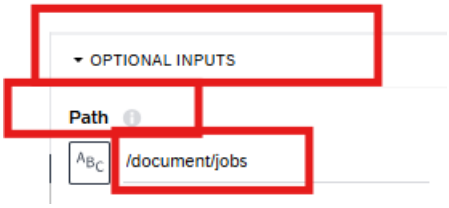
Key: options

Code Snippet 2 Value:

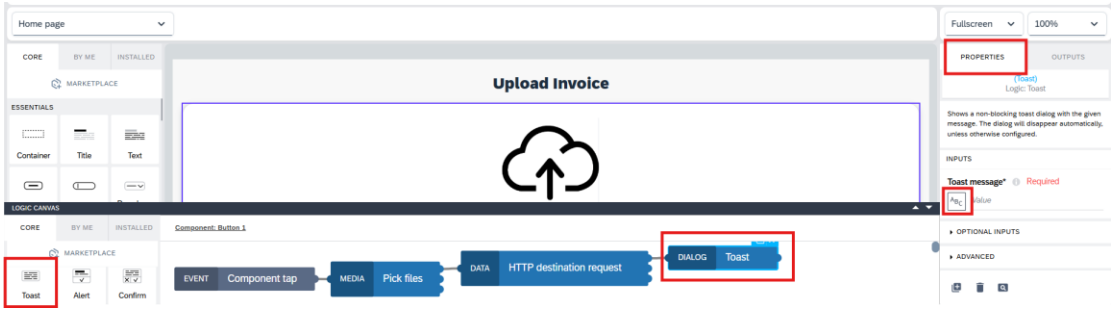
ENCODE_JSON({"schemaId":"cf8cc8a9-1eee-42d9-9a3e-507a61baac23","schemaVersion":"1","clientId":"default","documentType":"invoice","enrichment":{}})



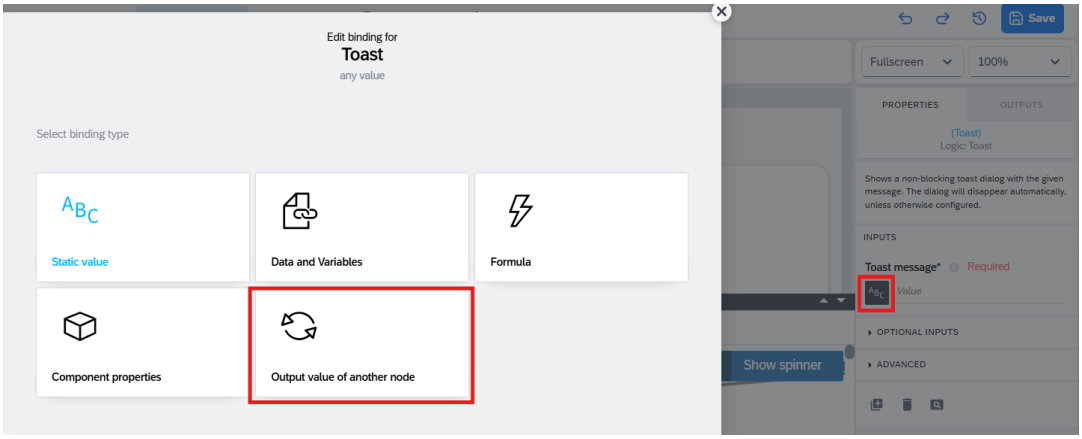
Step15: In Optional Inputs, give the **relative path --> /document/jobs**



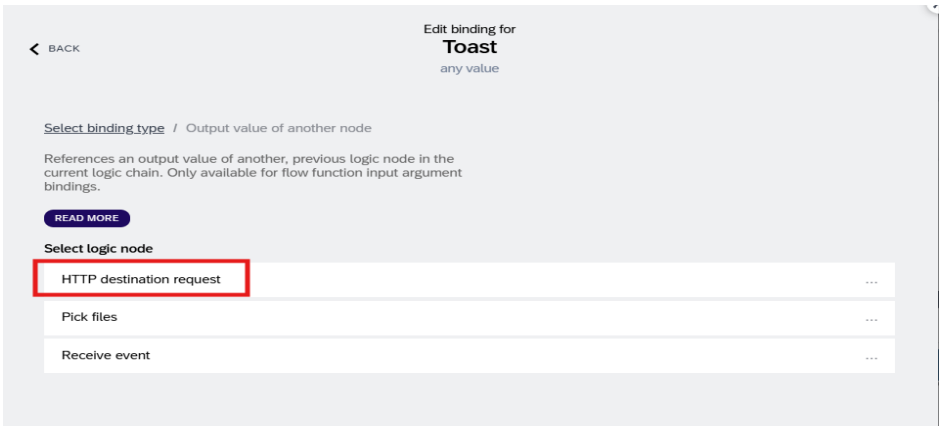
Step16: Next drag and drop toast button from the core and click on ABC option in Properties.



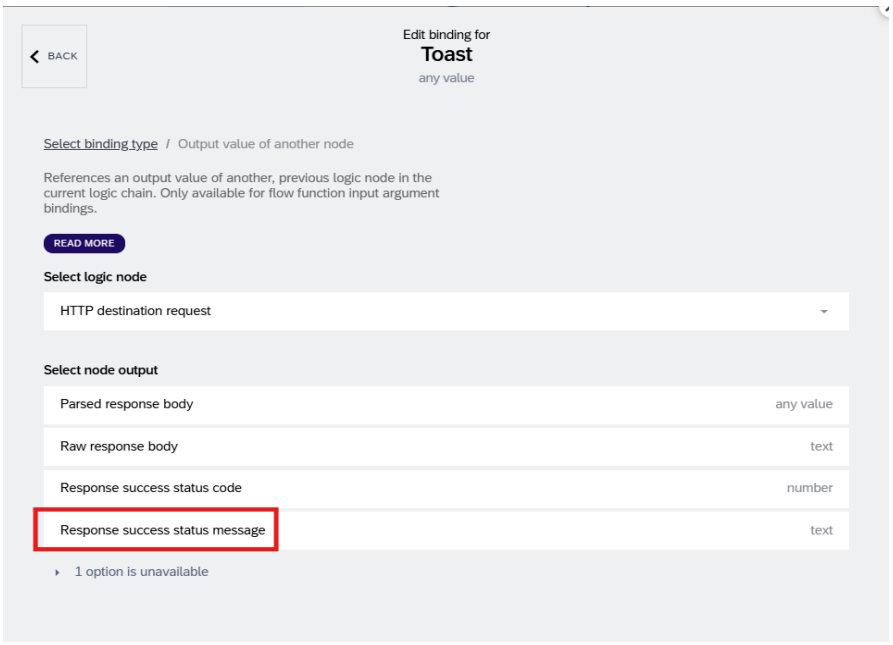
Step17: Click on Output Value of Another node



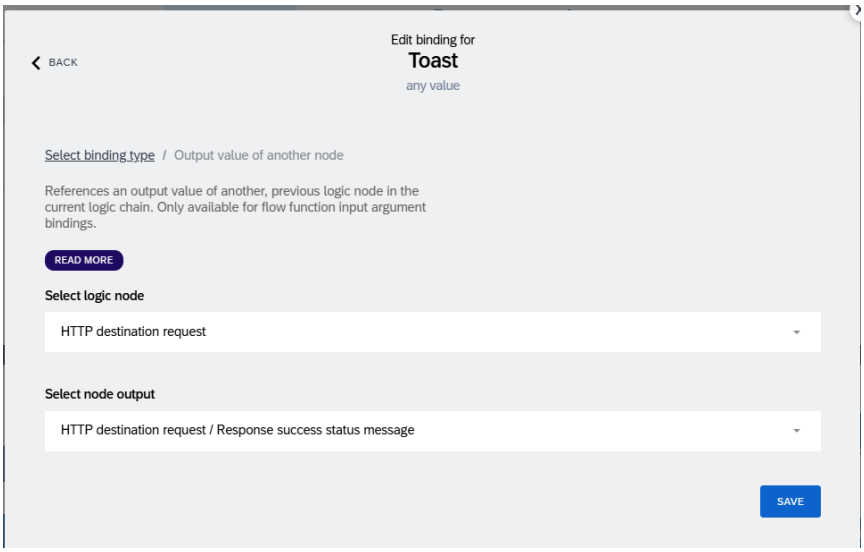
Step18: Click on HTTP destination request



Step19: Click on Response Status Messages. This will toast the success message in successful upload of Document to Document AI.

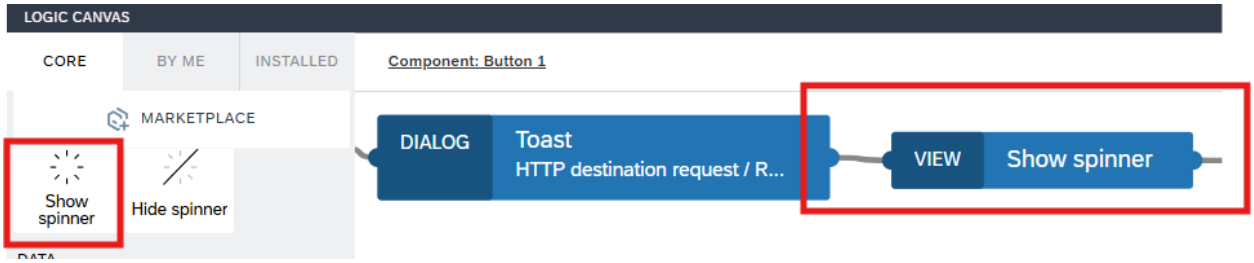


Step 20: Click Save



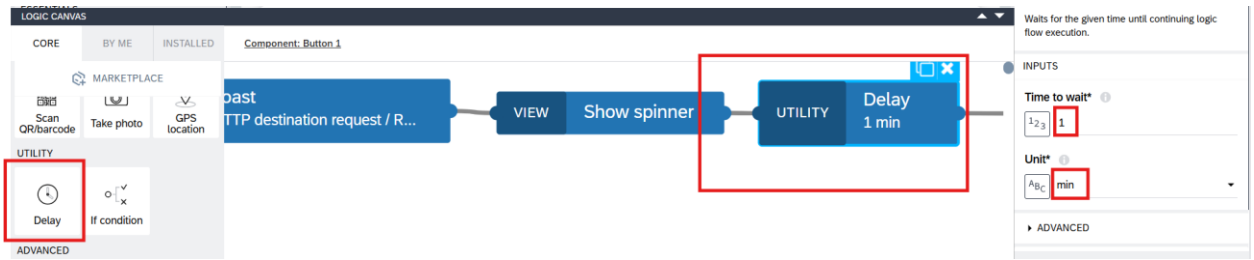
Step 21: In Core, select **show spinner** and connect it to the previous component:

Note: Since we're posting file content, to show process has started we're adding a spinner

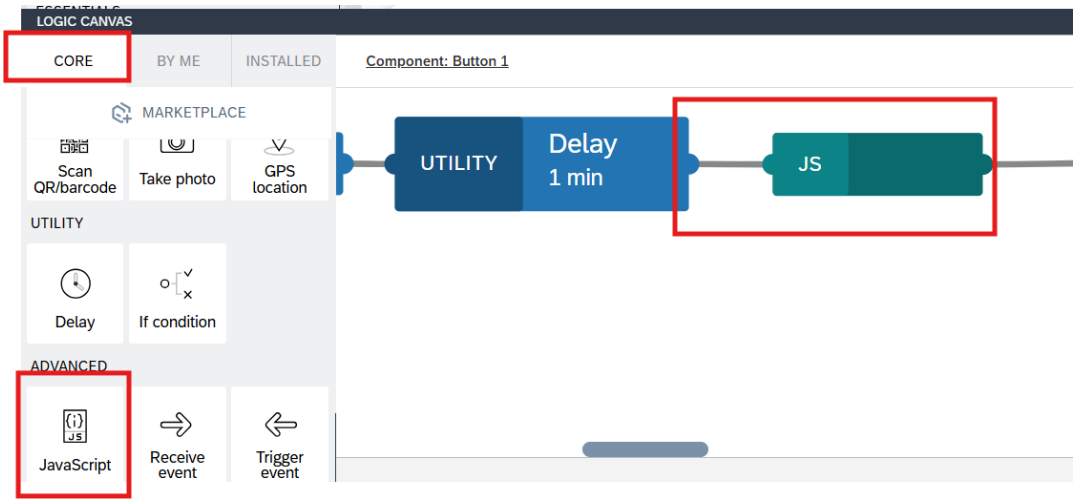


Step22: Now we add a Delay component connect to previous component. Now click on **delay Component** and add Properties for it.

Note: The reason why we need to add delay is , when we upload the file , the DocX service tries to get the content of it and during this time the status of file is "Pending" even though it is posted , so if we immediately look for file content we will not get the full content so this delay helps to get full file content as response.



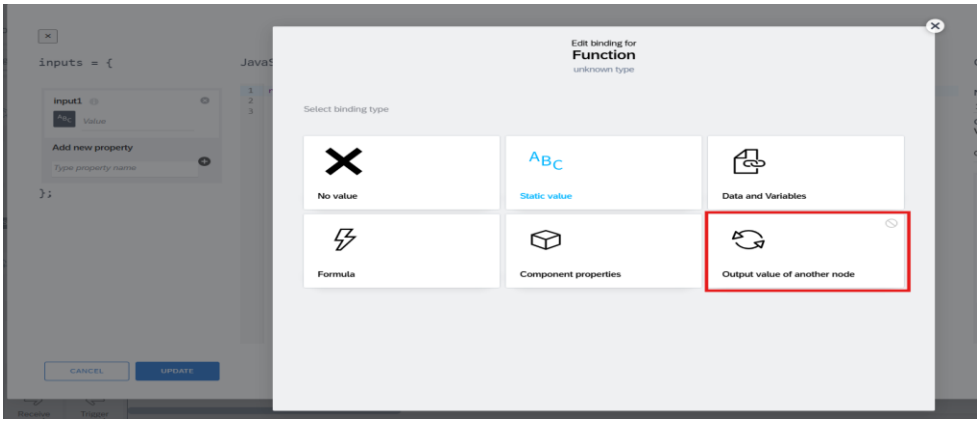
Step 23: Now from Logic Canvas, select the **JavaScript component**. Drag it and drop it to the logic editor and connect it to the previous component.



Step24: **Double click on JS component.**

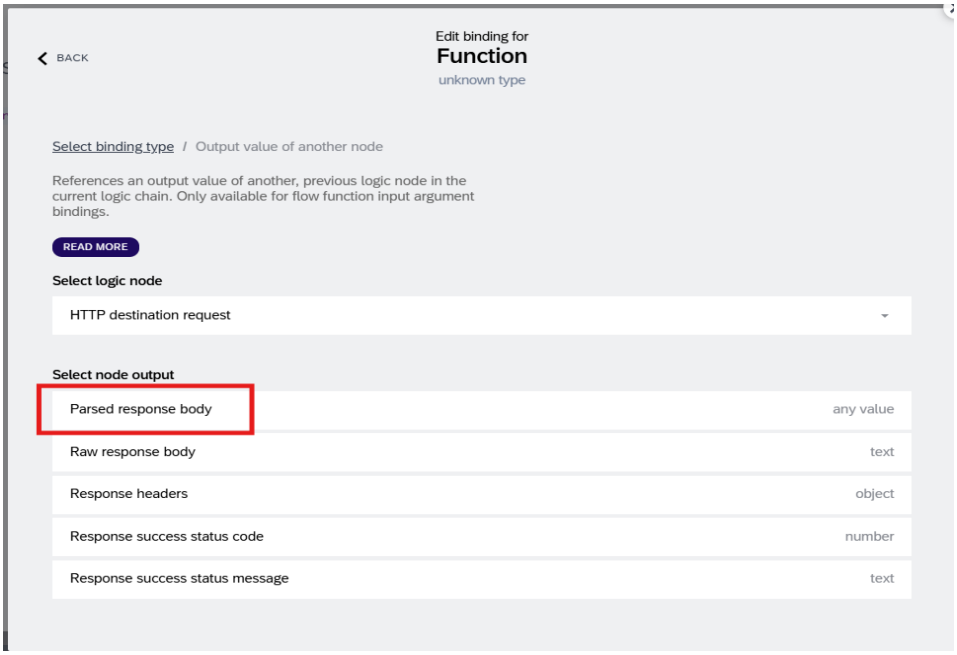
Step25: Now this dialog will open, and we must write the script task in it.

Step26: Select the output value of another node



Step27: Select the HTTP destination request from drop down.

Step28: After selecting it, you will get another node output option to select. Please select the Parsed response body and Save it.



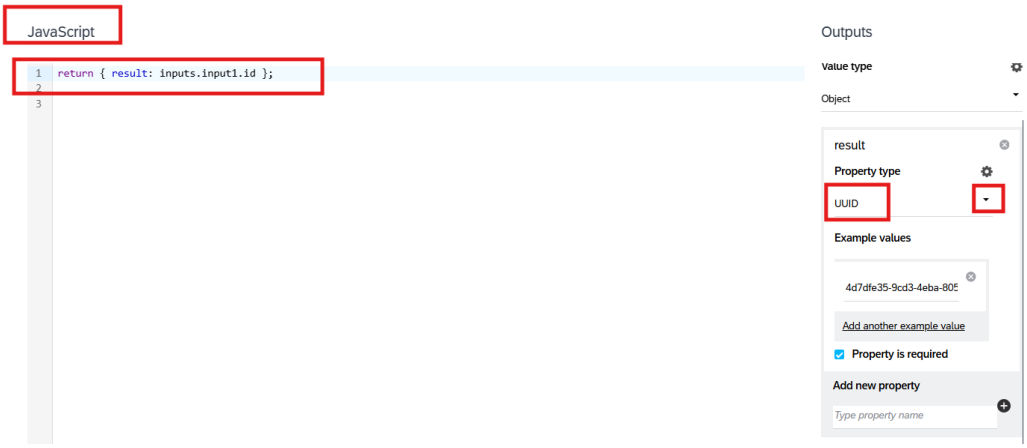
Step29: Under the JavaScript editor remove the default code and add the below code snippet.

Code Snippet 3 Value:

return { result: inputs.input1.id };

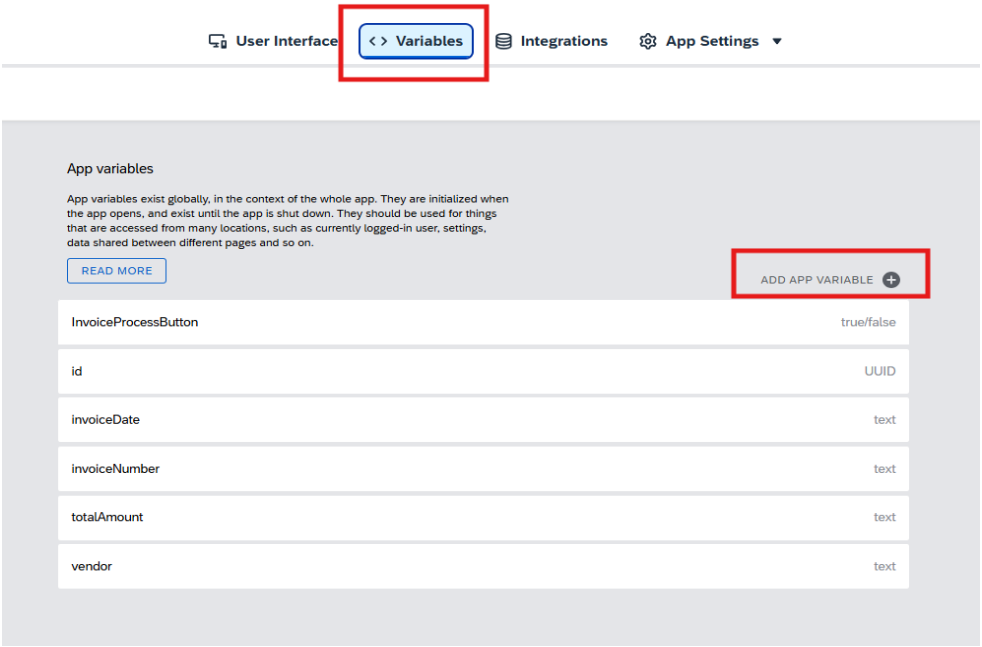
As we're trying to read the Id from response, this id is nothing but UUID type, which is unique key, so set the output type as UUID in right side Options panel

Please Select Property type drop down--> Select UUID and click + icon.

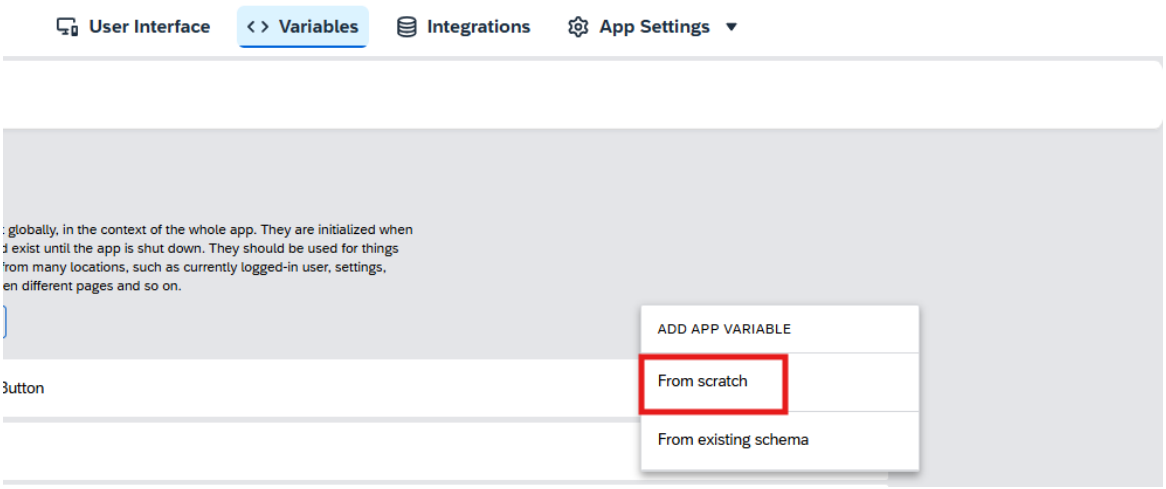


Step 30: Creation of Variable.

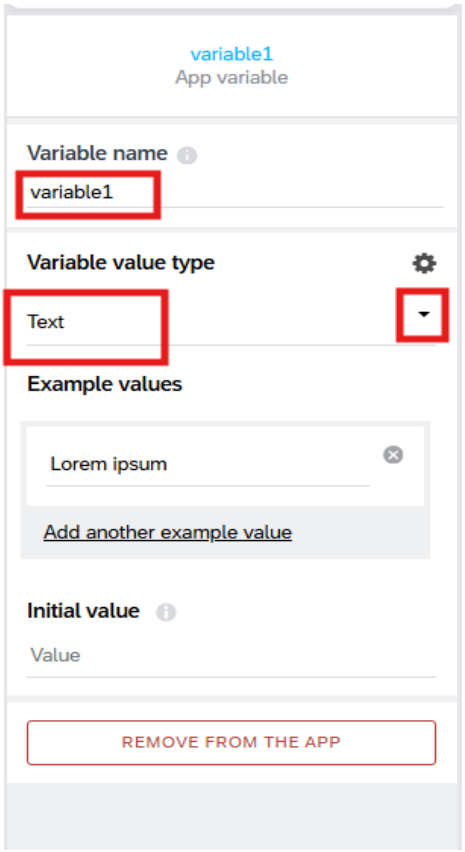
Now let's create few app variables: We need to create the below app variables, and I will explain why we need it in coming steps



Step31: Click on Add App Variable -> Select from Scratch.



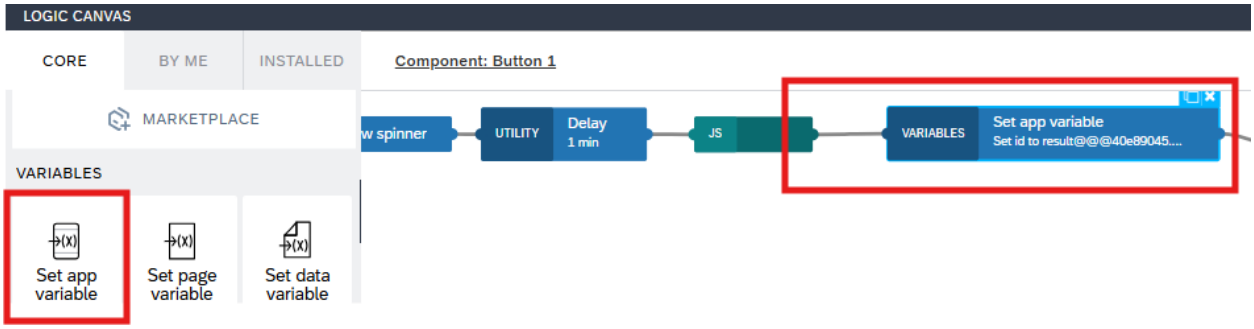
Step32: We need to give variable name, Value type from drop down. Please refer to the table below for the Variable names and Value Types that need to be created.



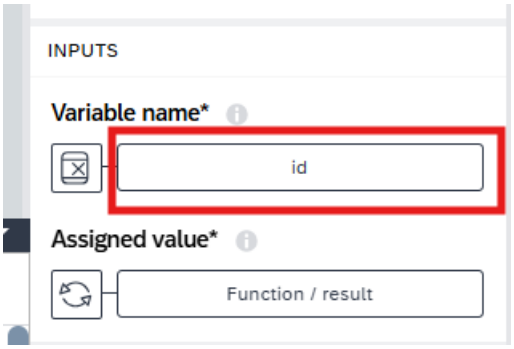
Key	Initial Value	Value Type

id		UUID
invoiceNumber		Text
grossAmount		Text
documentDate		Text

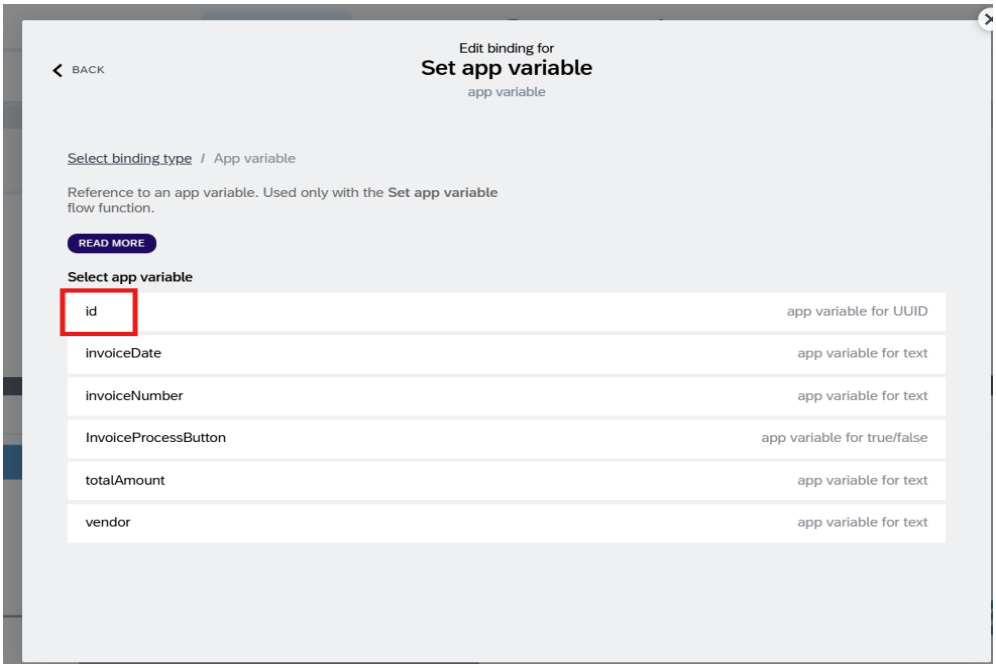
Step 33: Now from Canvas choose **set app Variable**, drag it and drop it. Connect it to JS component.



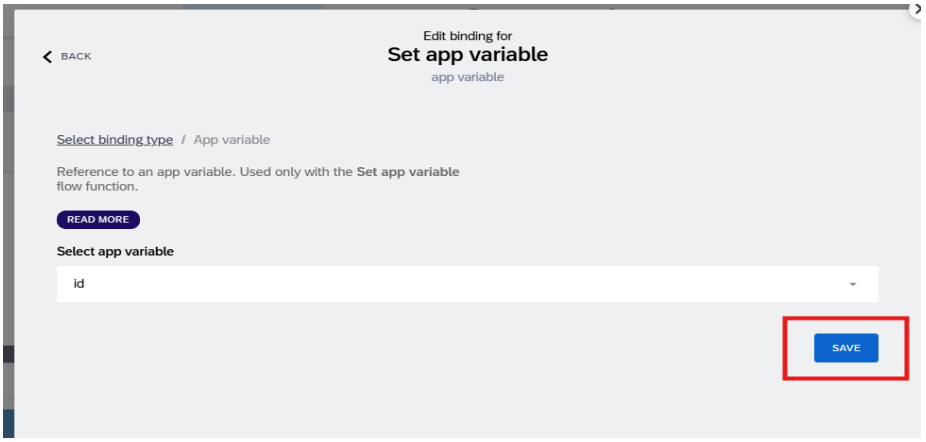
Step34: In properties Panel, by default in variable name it will show the first variable in the list, but you can click on list and choose which variable you want to set now.



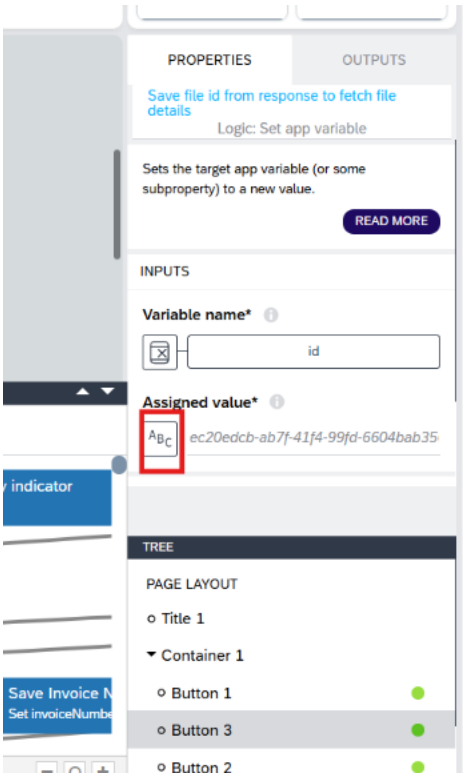
Step35: Click on Drop down and select the id from drop down List.



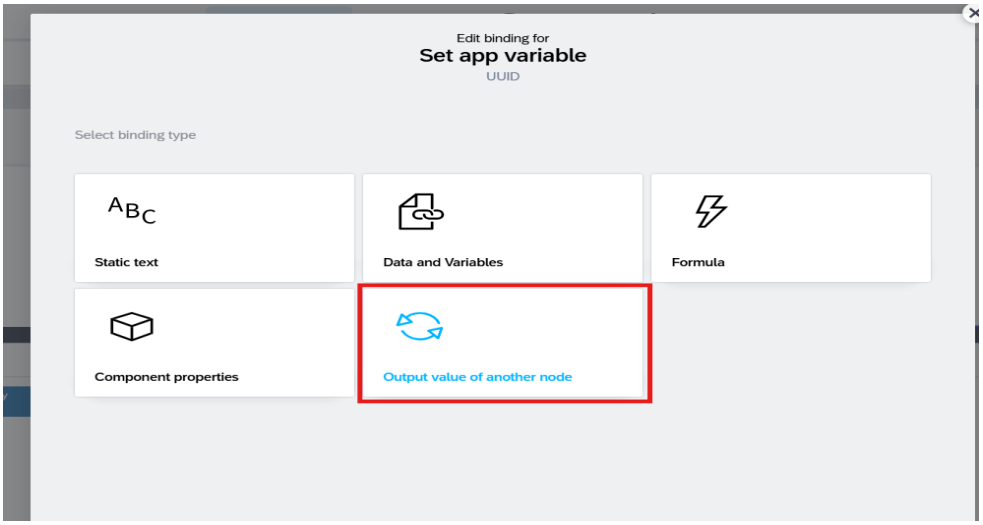
Step36: Save it



Step 37: Let's assign actual ID value to this variable, please select the ”ABC” box in assigned value.

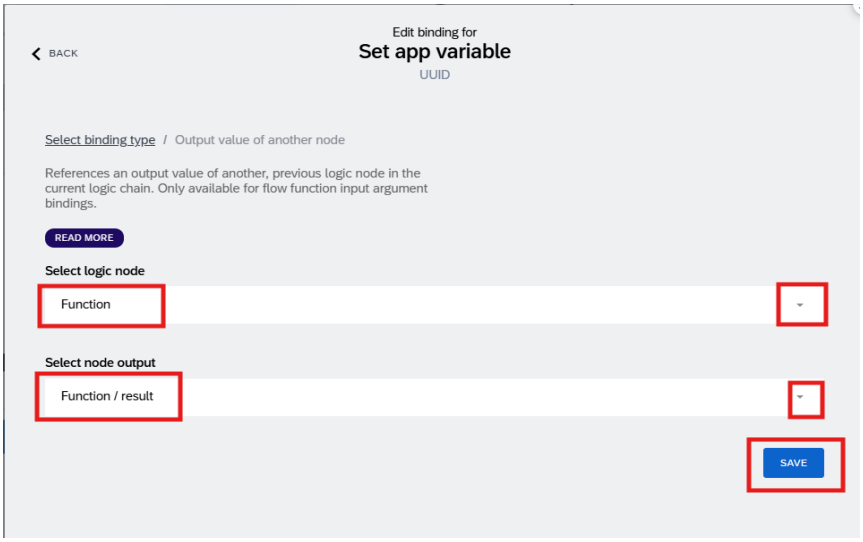


Step 38: Select the Output Value of another node.



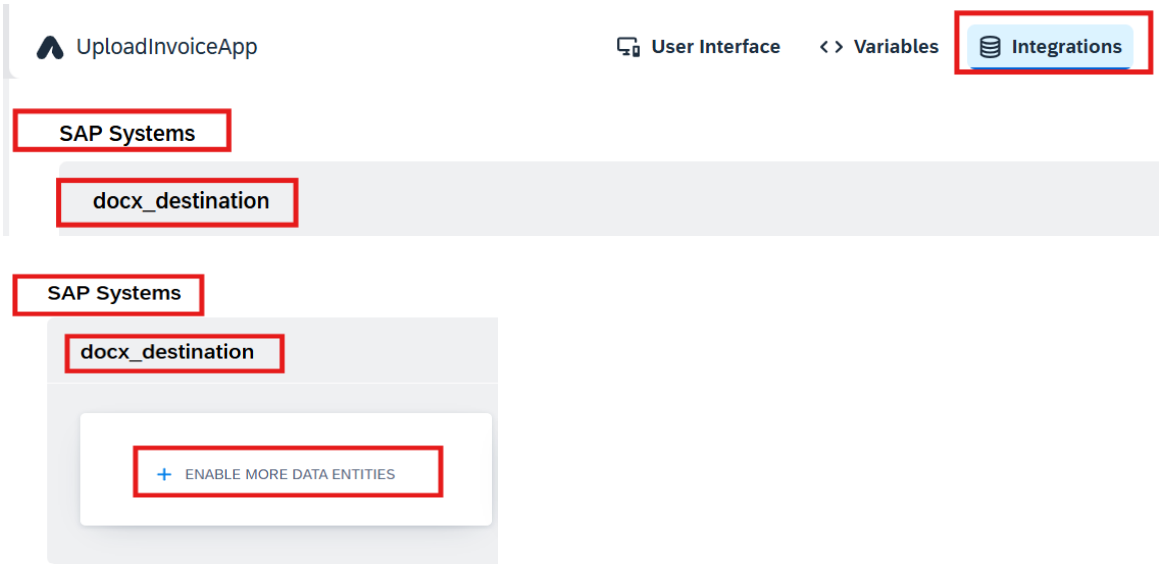
Step 39: Select the Logic Node Function (previous node JS function we're referring here), after selecting the logic node automatically the node output will pop up and select the result from function from drop down. Save it.

Note: From the POST service response, we're able to save the ID into variable which we use later to fetch the invoice details.

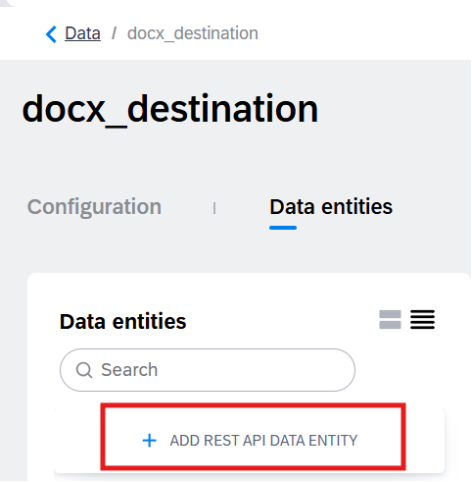


Step41: Now let's get the file content and store main details like Vendor, Invoice Number, Invoice Date, Total Amount in variables. Before that lets do one small step in Integration.

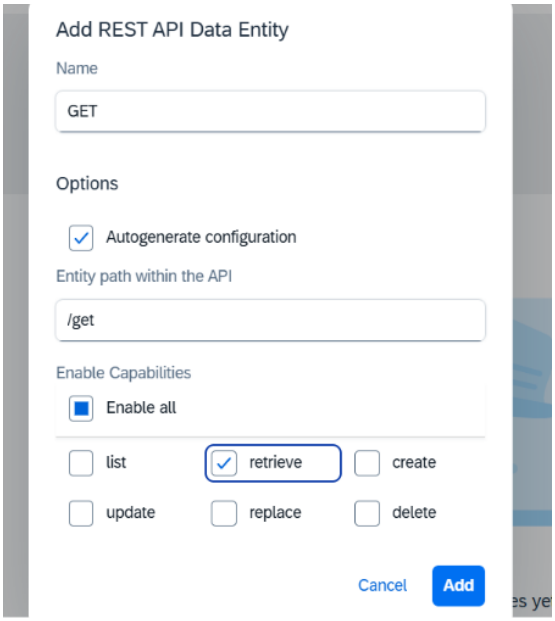
Go to Integration and select the destination we added -->docX_destination --> Click + Icon



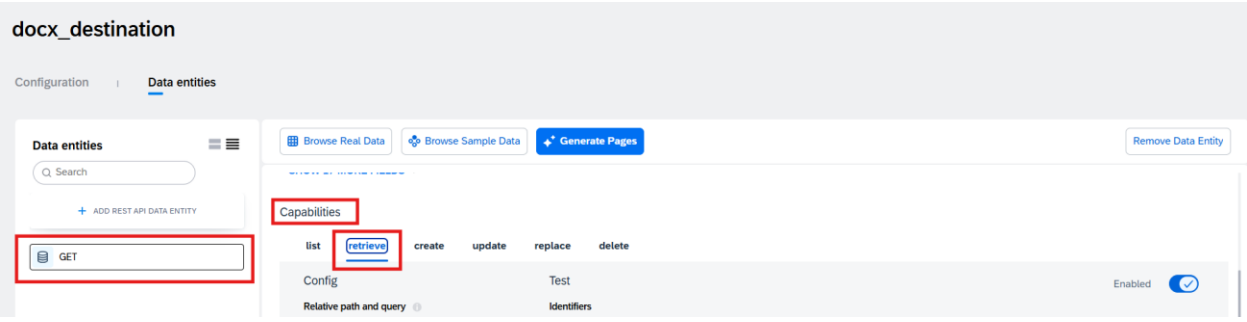
Step 42: You will see the below screen and in the left pane select ”+”



Step 43: Now fill in the details in the pop up: Give Service name as GET and select only retrieve

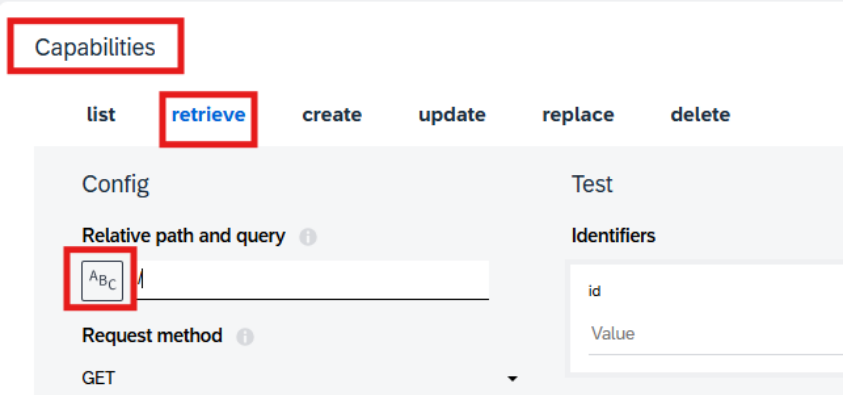


Step 44: Now you will see the capabilities for this service, as we selected retrieve only retrieve is enabled for now.

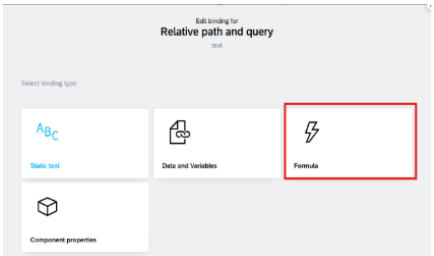


Step 45: Now we need to fill the Config details: In Relative path we need to give the path so Click on

“ABC” option and **select the formula**

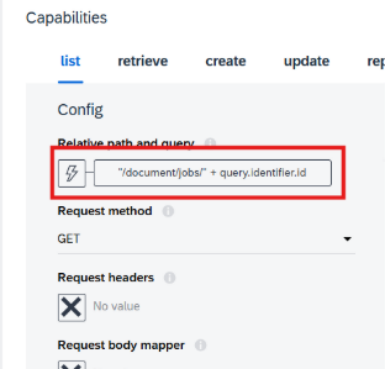


Step46: Select formula



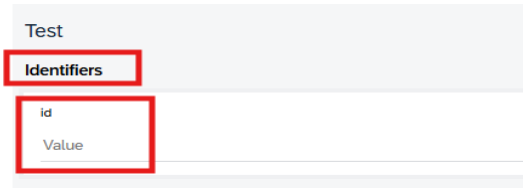
Step47: Now we need to give the relative path + uniquely generated id which comes as response from DocX service when we “Post” Invoice file.

ADD the below snippet and save: **"/document/jobs/" + query.identifier.id**

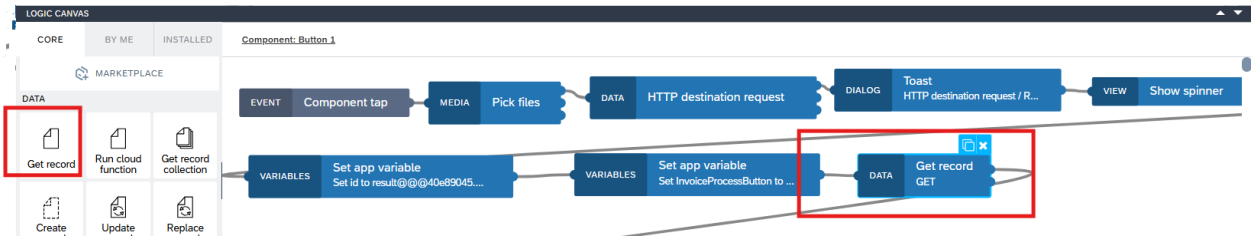


Step 48: Now inside identifiers id we must enter id and when we click the **“Run test”** we can see the response from DocX service.

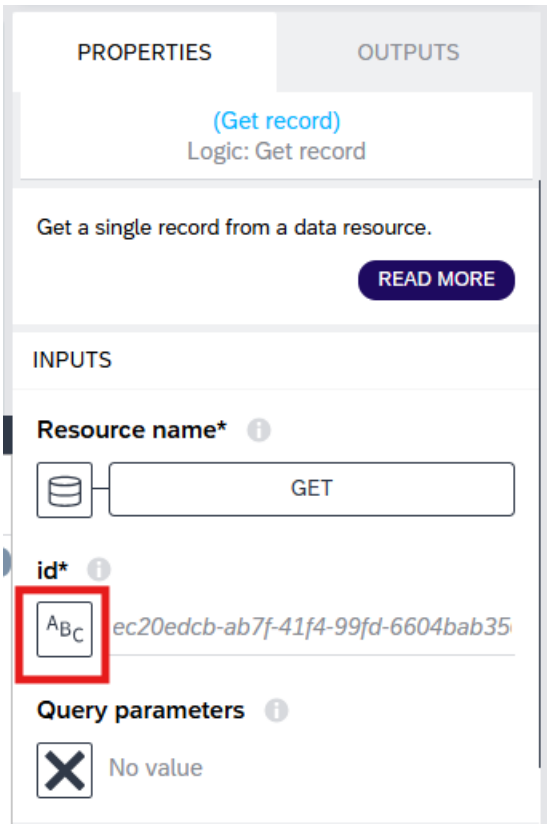
Note: You will get the id when you upload file to Document AI, so you can directly open Document AI application and upload fie then you will get the id as response

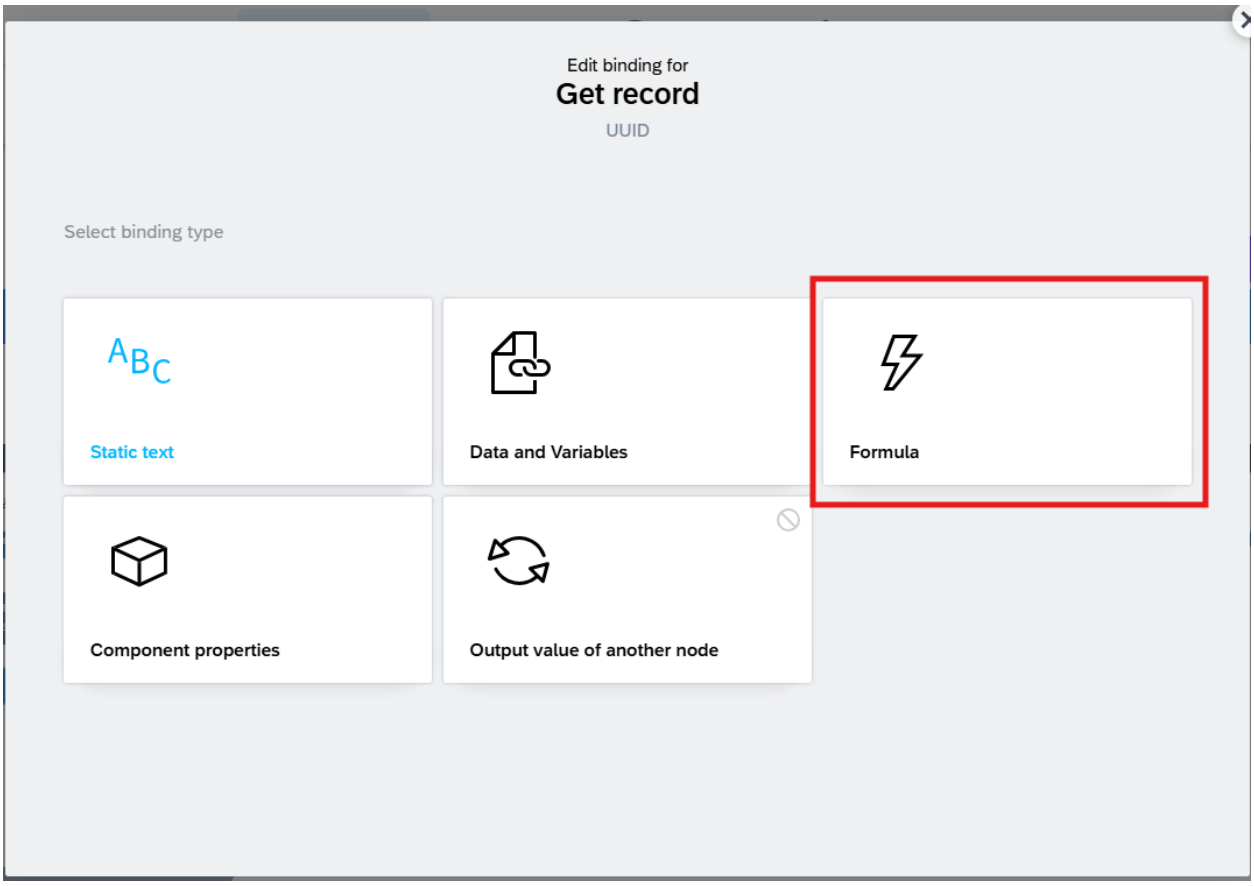


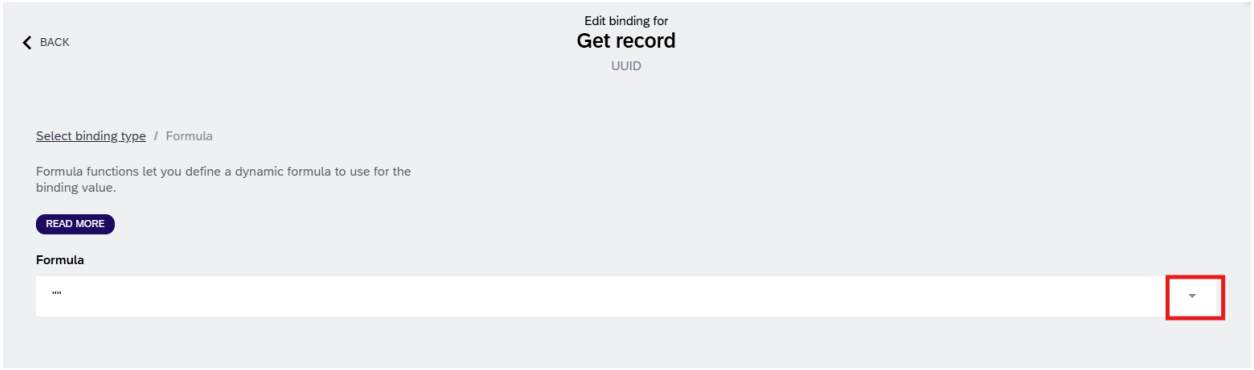
Step 49: Now in logic canvas, select the **Get record**

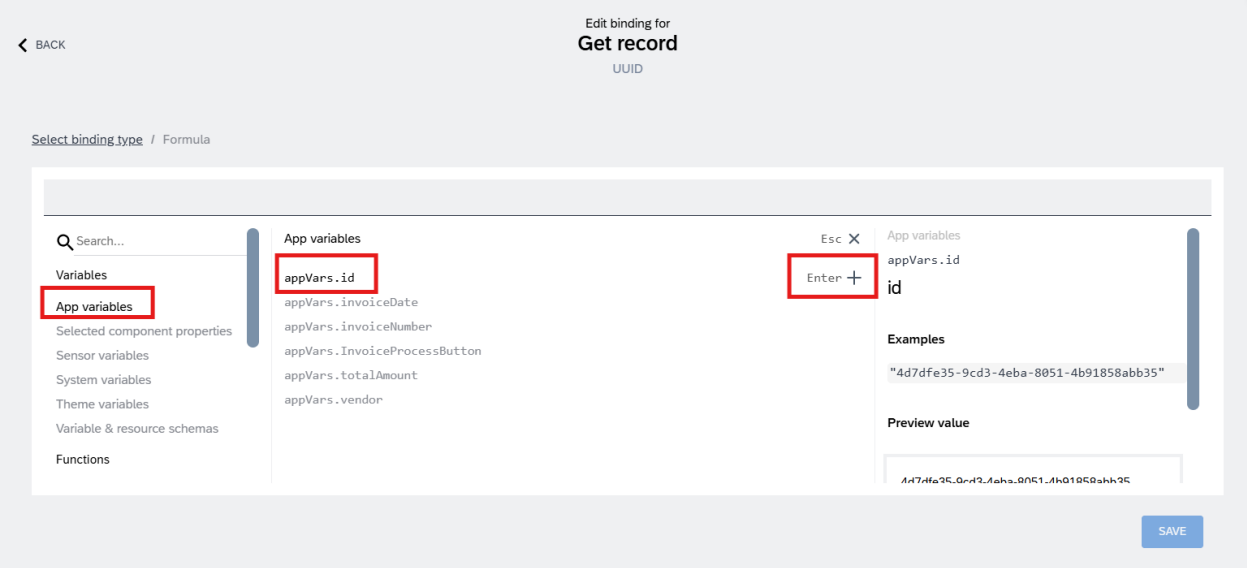


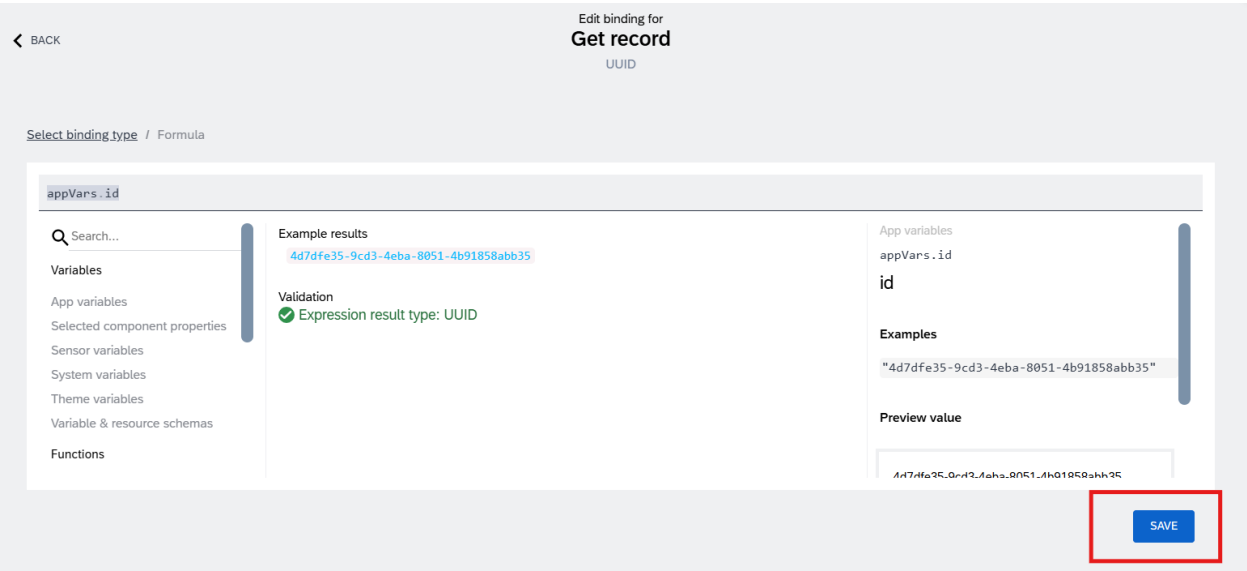
Step 50: Click on “ABC” option and **select the formula and** paste this formula **appVars.id** and save it



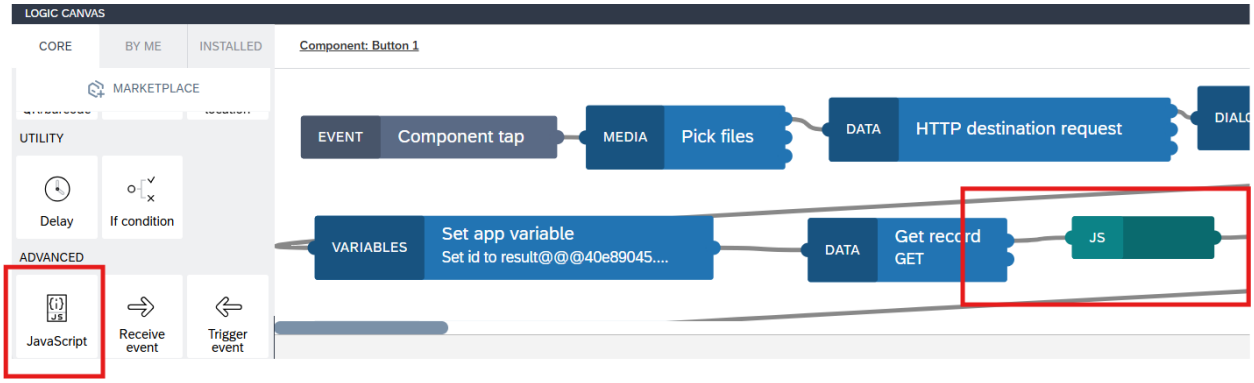

Edit binding for
Get record
UUID
Select binding type
Static text
Data and Variables
Formula
Component properties
Output value of another node


BACK
Edit binding for
Get record
UUID
Select binding type / Formula
Formula functions let you define a dynamic formula to use for the binding value.
READ MORE
Formula



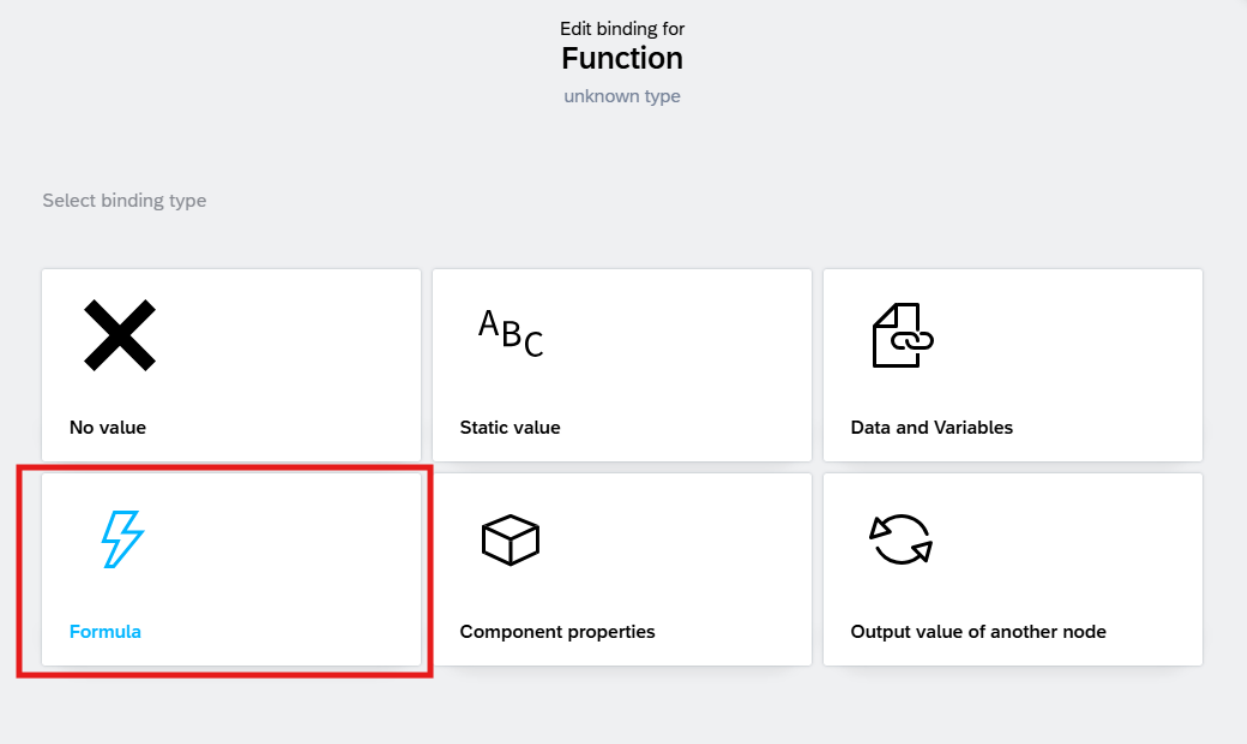


Step51: Drag JavaScript from Logic Canvas and drop it next to “Get Record” and connect the nodes.



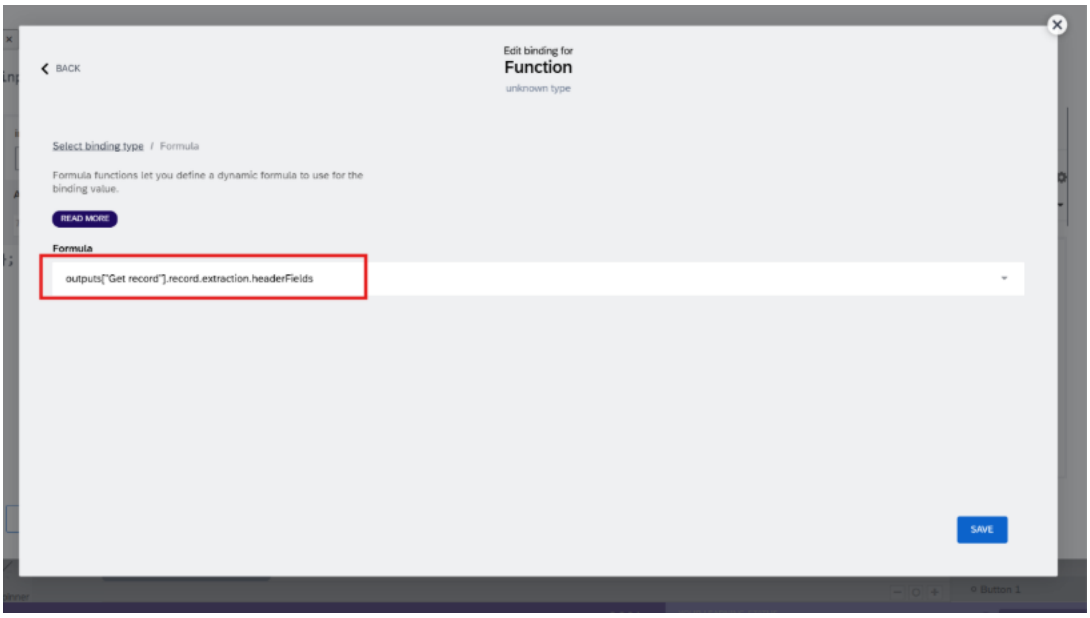
Step 52: Now let's write JavaScript to read the response and store the invoice details in app variables. Under advanced there is a java script: Firstly, select the “ABC” option in input1 and select the Formula

Select Formula



Copy the below snippet and paste it:

outputs["Get record"].record.extraction.headerFields



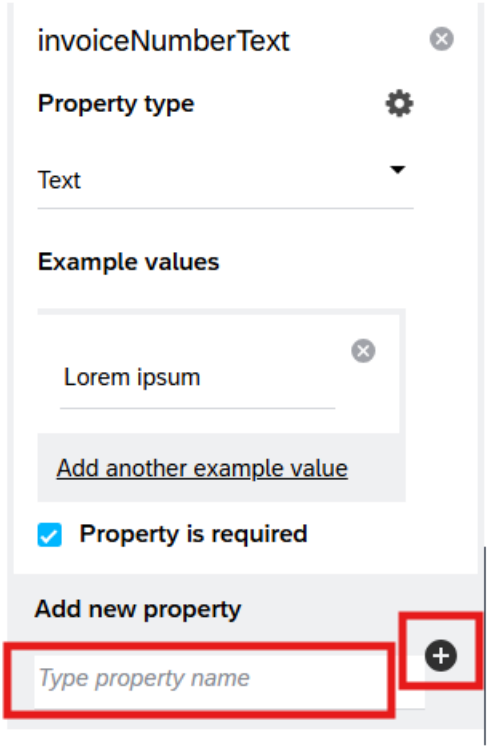
Add the below code snippet to JS

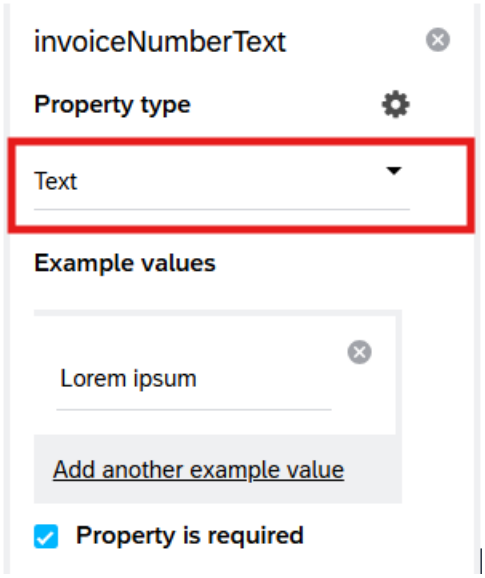
```
const headerFields = inputs.input1; const getValue = (name) => {
const match = headerFields.find(field => field.name === name);
 return match ? match.value : "";
};

let invoiceNumber = getValue("invoiceId");
if(!invoiceNumber) {
invoiceNumber = getValue("documentNumber");
}

const invoiceDate = getValue("documentDate");
let vendorName = getValue("vendorName");
if(!vendorName) {
```

```
vendorName = getValue("senderName");
}

let totalDue = getValue("grossAmount");
if(!totalDue){
  totalDue= getValue("netAmount");
}

return {
  invoiceNumberText: String(invoiceNumber),
  documentDateText: String(invoiceDate),
  vendorNameText: String(vendorName),
  grossAmountText: String(totalDue)
};
```

Step 53: We need to make sure we add Output object new properties and value type “Text” in Output (right side). Every time you add one property select Add new property.

Outputs

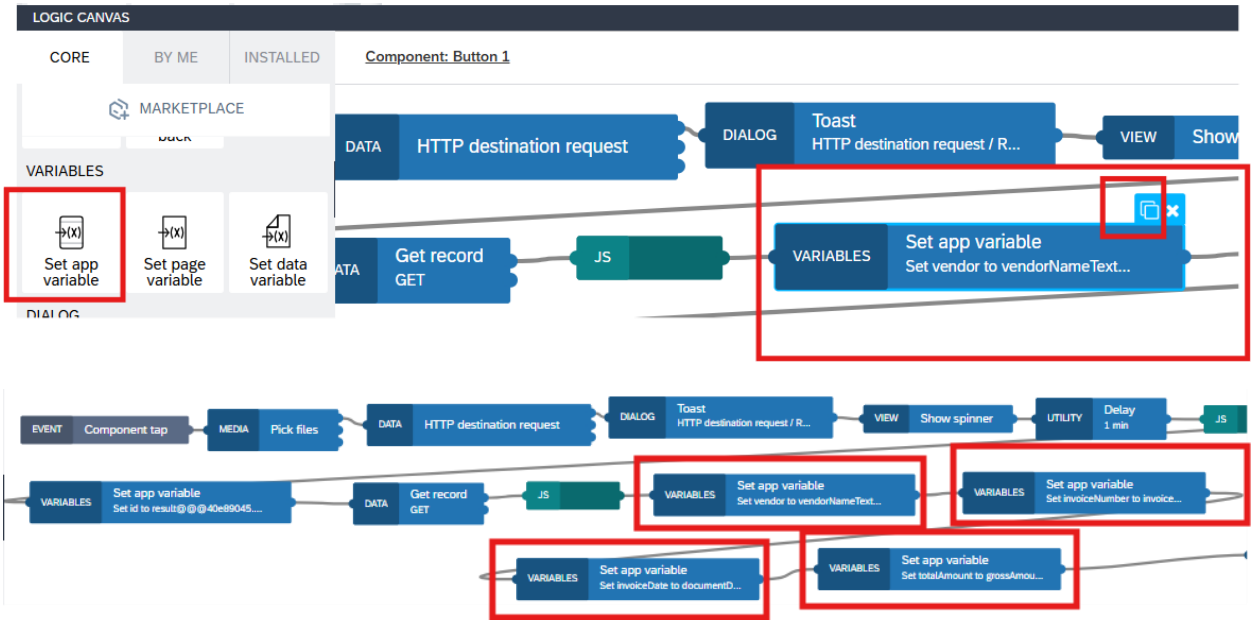




Property
invoiceNumberText

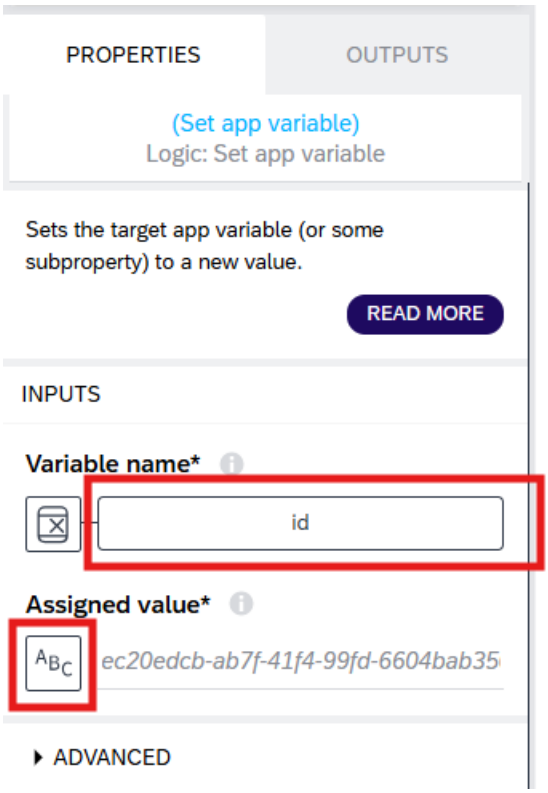
documentDateText
vendorNameText
grossAmountText

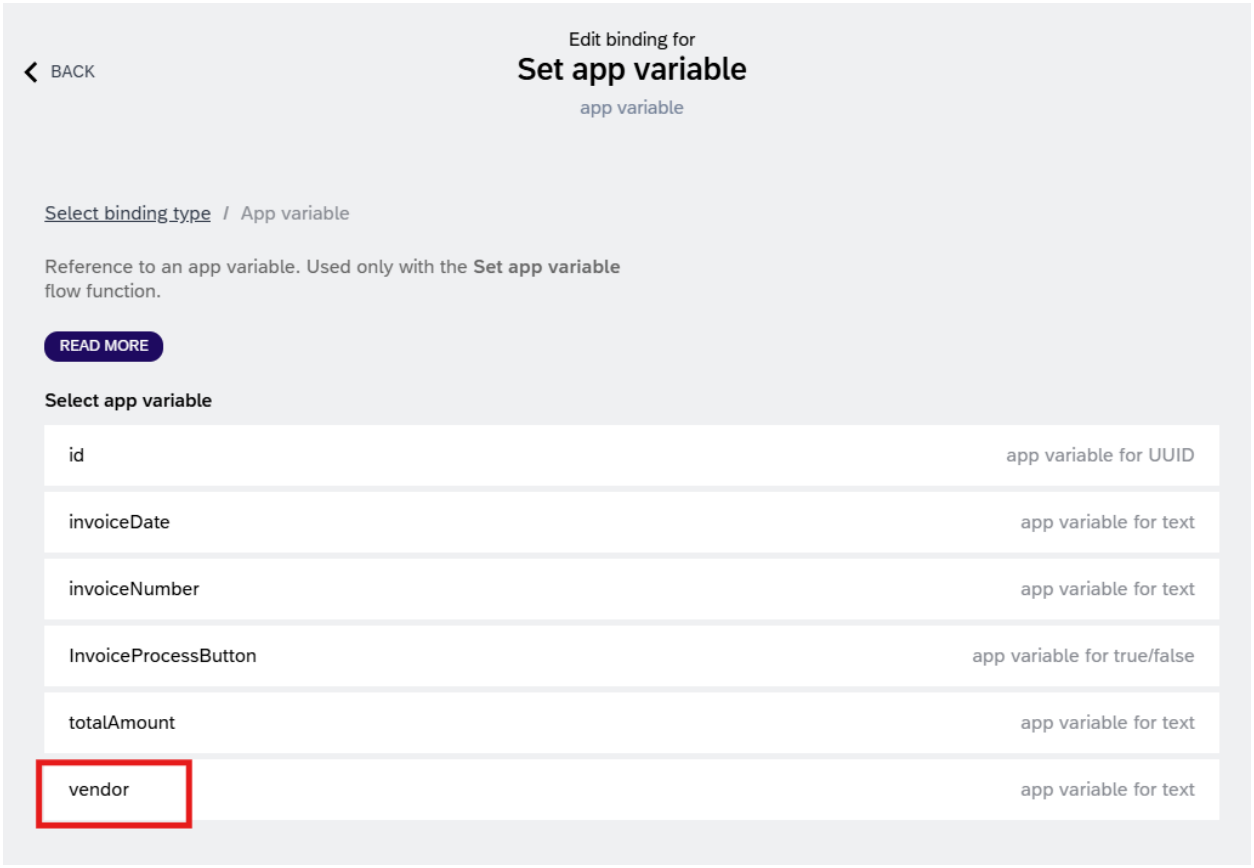
Step 54: Now let's drag and drop set app variables and connect to the previous node. Since we need 4 values, you can use Copy icon and create 3 more components and connect it to the previous node.



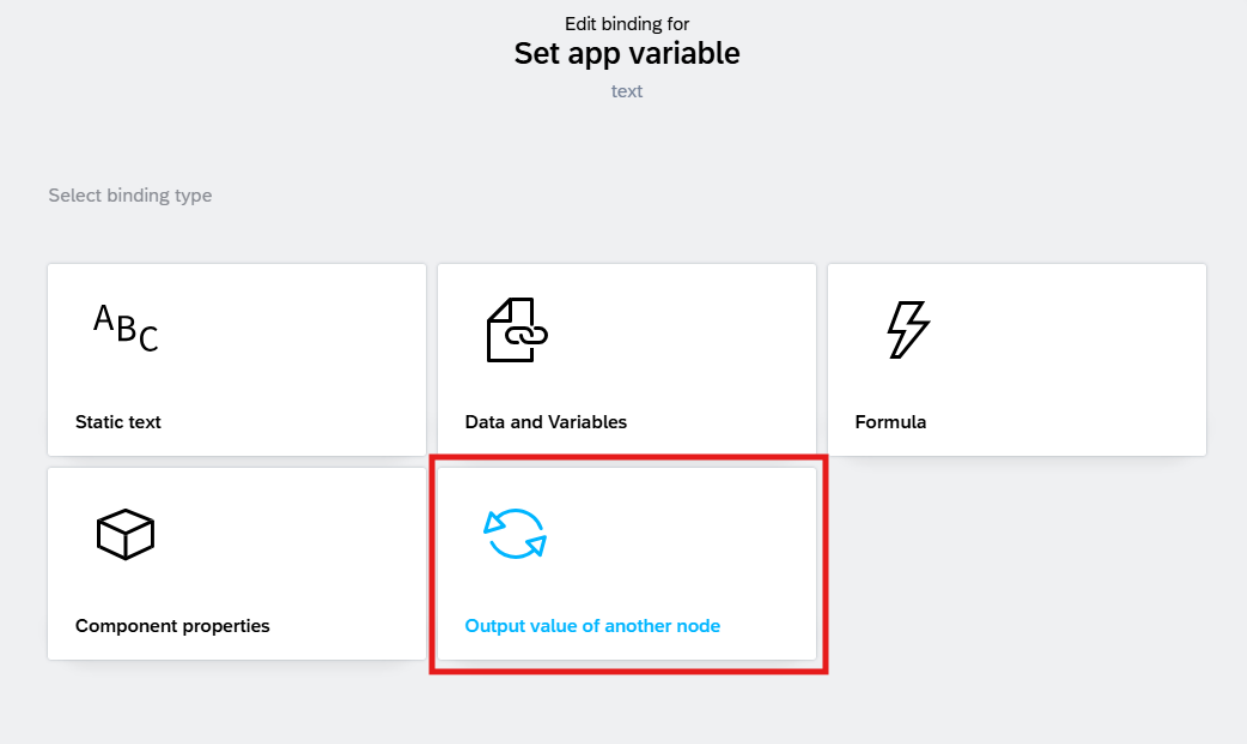
Step 55: Now lets select one by one all 4 App variables component and add property details to it.

Select Variable Name -> by default it will display first one you can select from drop down the required variable

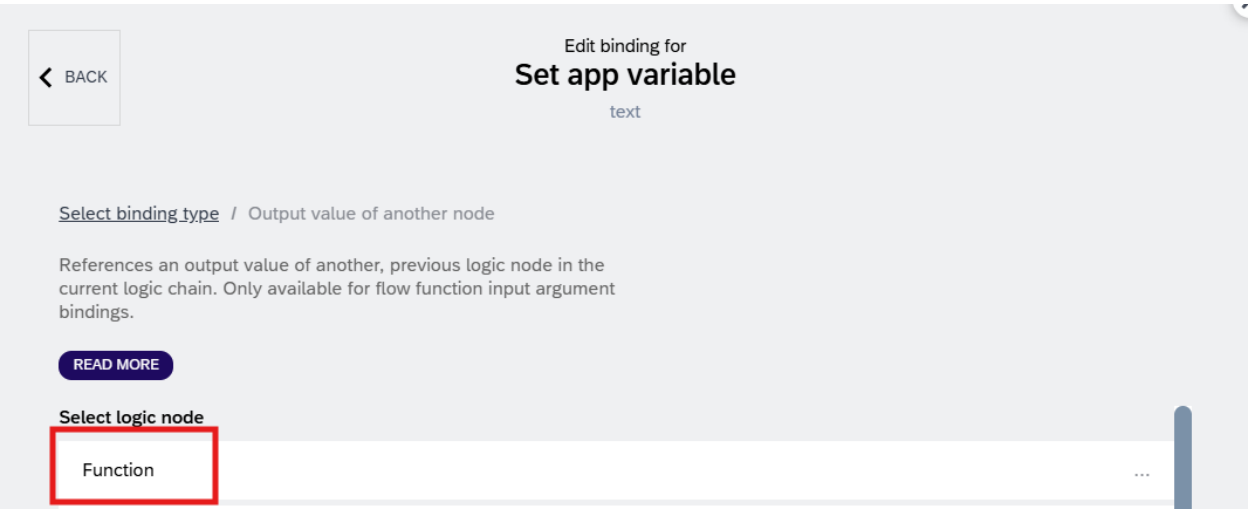

PROPERTIES
OUTPUTS
(Set app variable)
Logic: Set app variable
Sets the target app variable (or some subproperty) to a new value.
READ MORE
INPUTS
Variable name*
id
Assigned value*
ec20edcb-ab7f-41f4-99fd-6604bab35
ADVANCED


BACK
Edit binding for
Set app variable
app variable
Select binding type / App variable
Reference to an app variable. Used only with the Set app variable flow function.
READ MORE
Select app variable
id
app variable for UUID
invoiceDate
app variable for text
invoiceNumber
app variable for text
InvoiceProcessButton
app variable for true/false
totalAmount
app variable for text
vendor
app variable for text

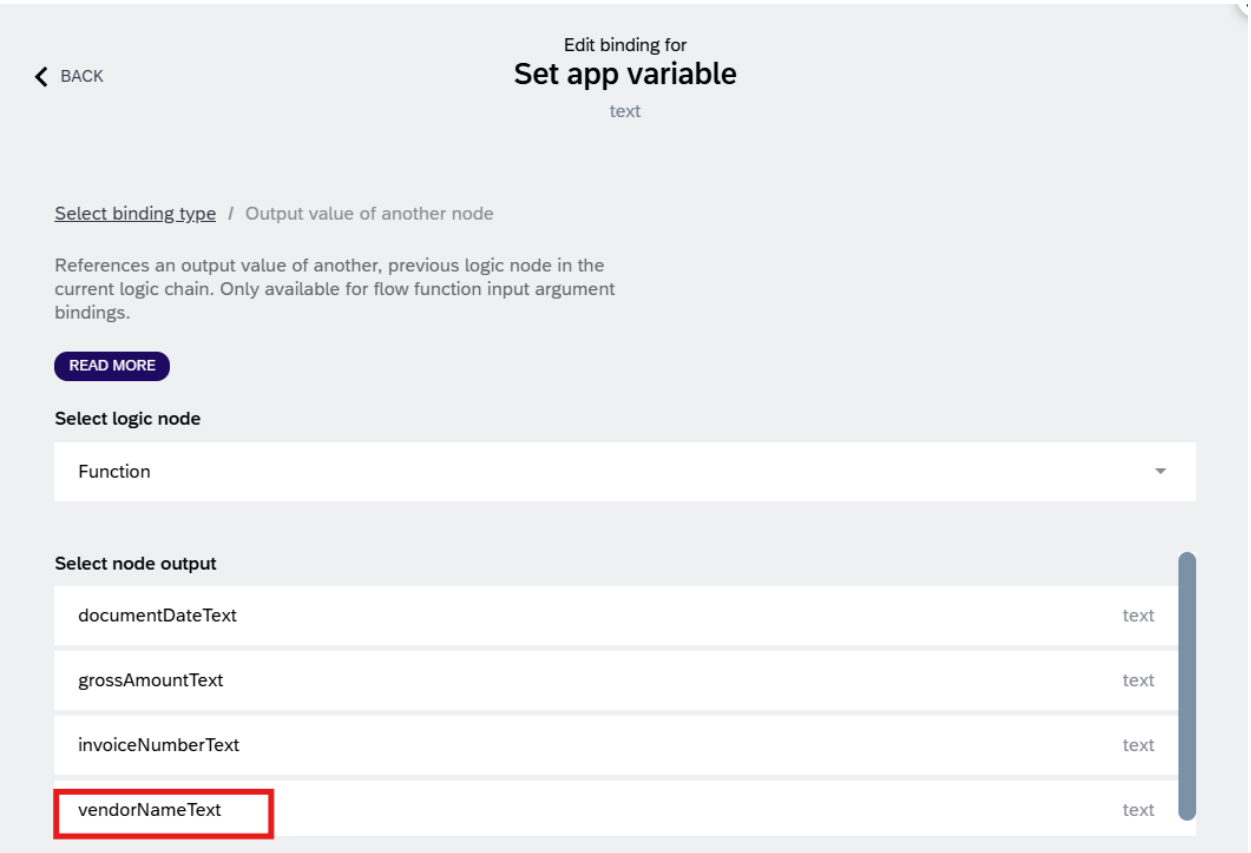
Now Click on “ABC” option from Assign to and select the “Output from another node”



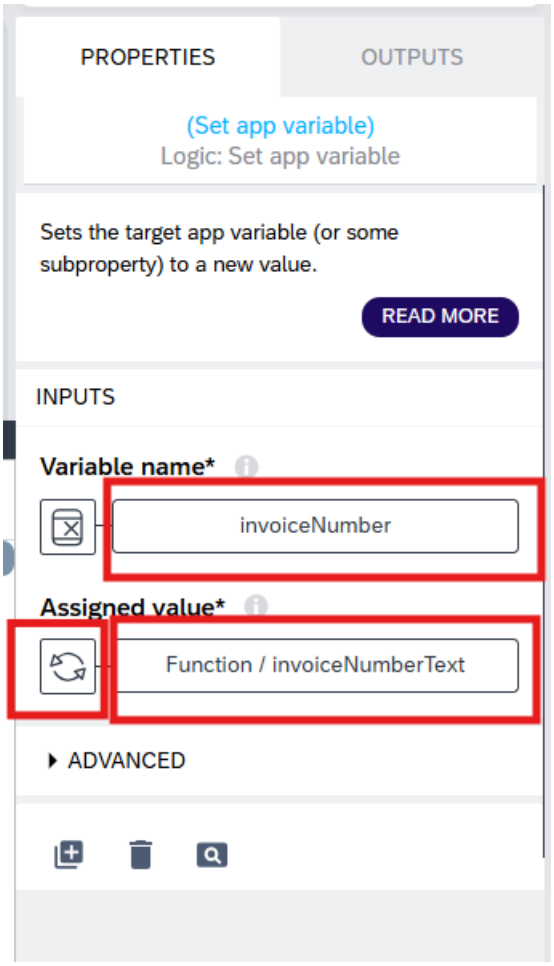
Select Function

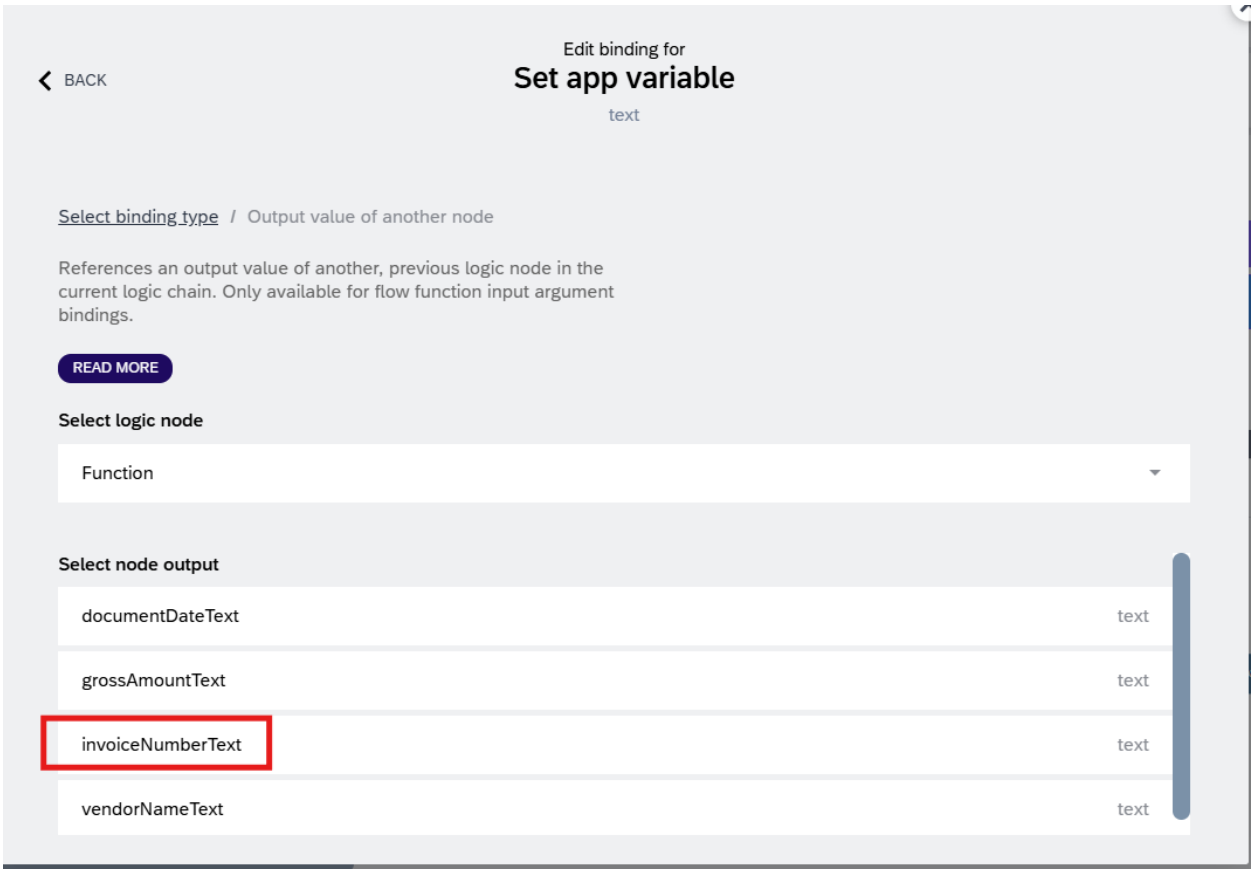


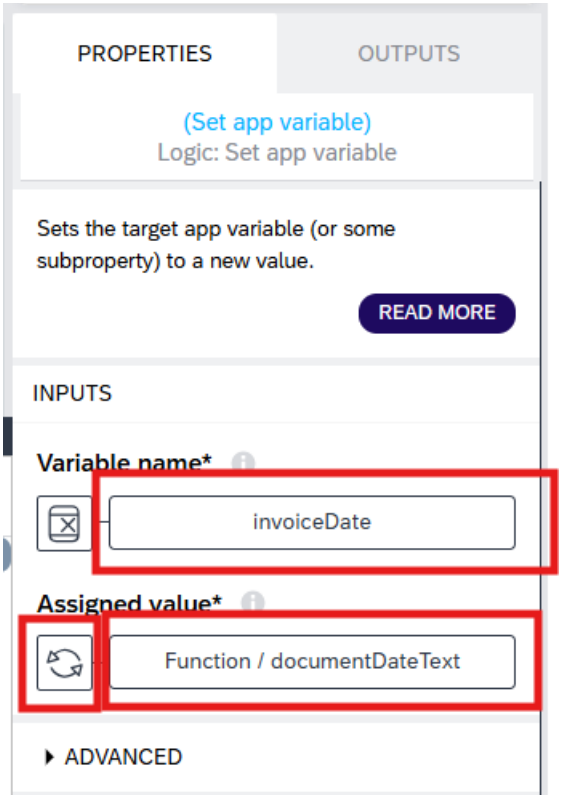
Select VendorNameText

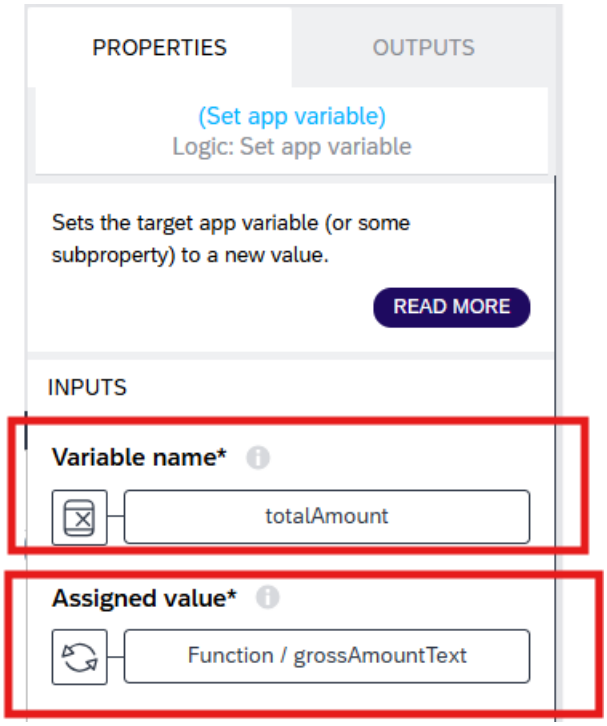


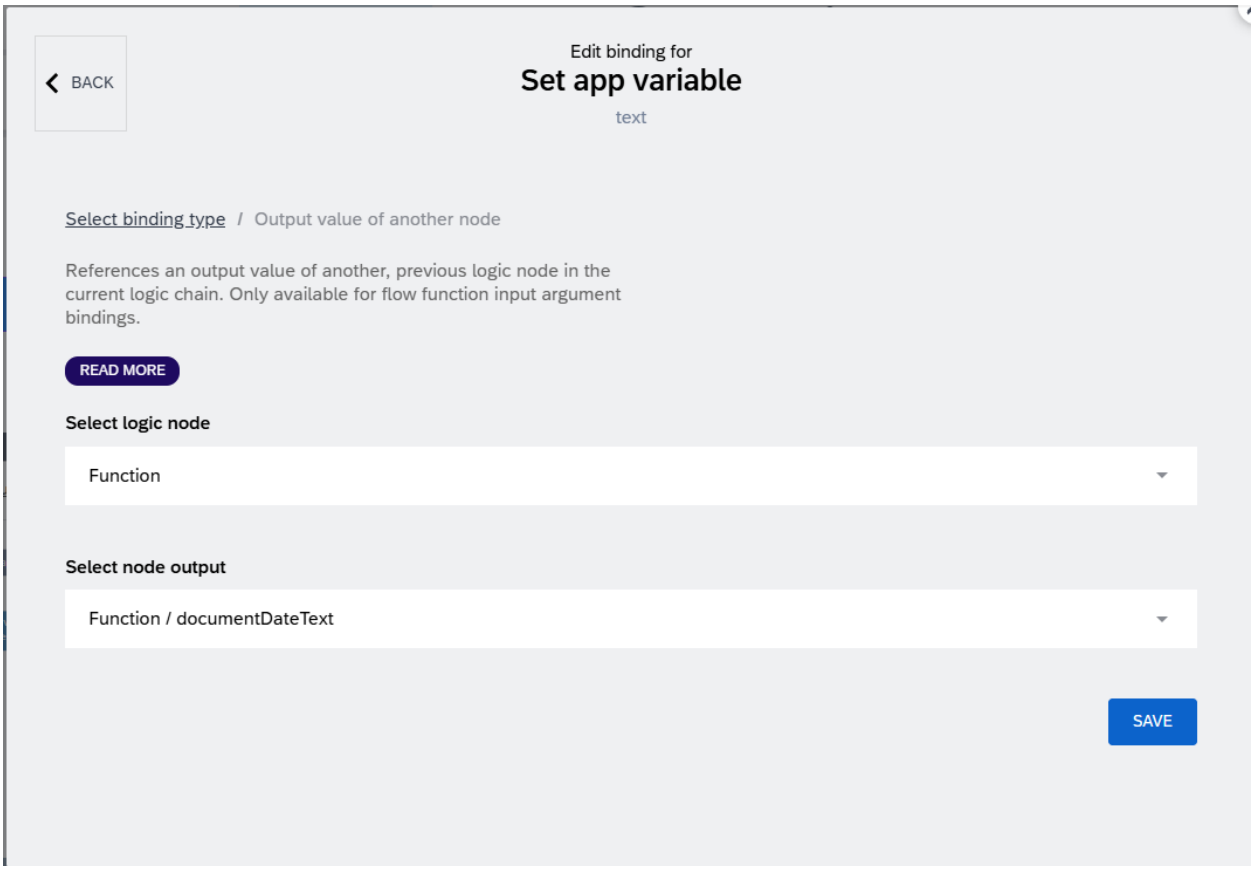
Step 56: Repeat the previous step for next 3 variables and bind the values

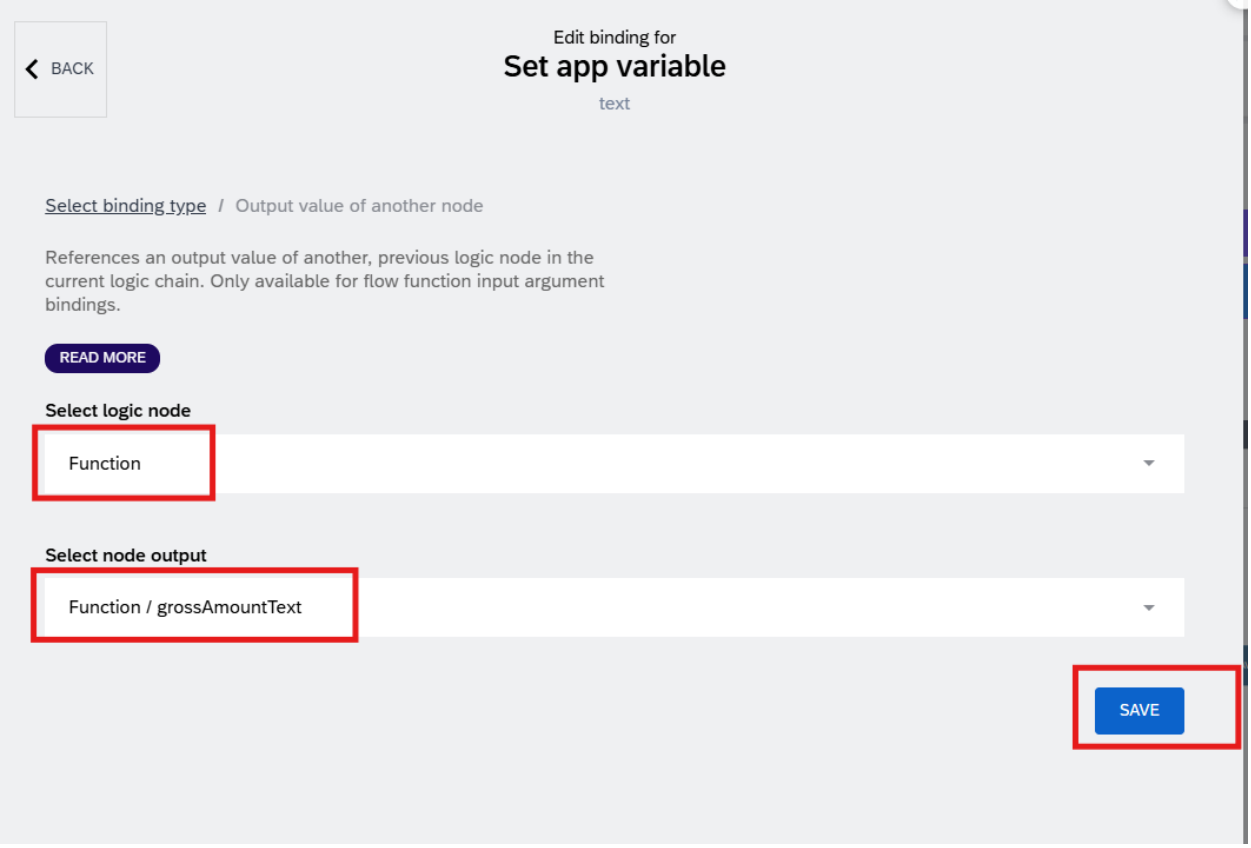

PROPERTIES
OUTPUTS
(Set app variable)
Logic: Set app variable
Sets the target app variable (or some subproperty) to a new value.
READ MORE
INPUTS
Variable name*
invoiceNumber
Assigned value*
Function / invoiceNumberText
ADVANCED


BACK
Edit binding for
Set app variable
text
Select binding type / Output value of another node
References an output value of another, previous logic node in the current logic chain. Only available for flow function input argument bindings.
READ MORE
Select logic node
Function
Select node output
documentDateText
text
grossAmountText
text
invoiceNumberText
text
vendorNameText
text

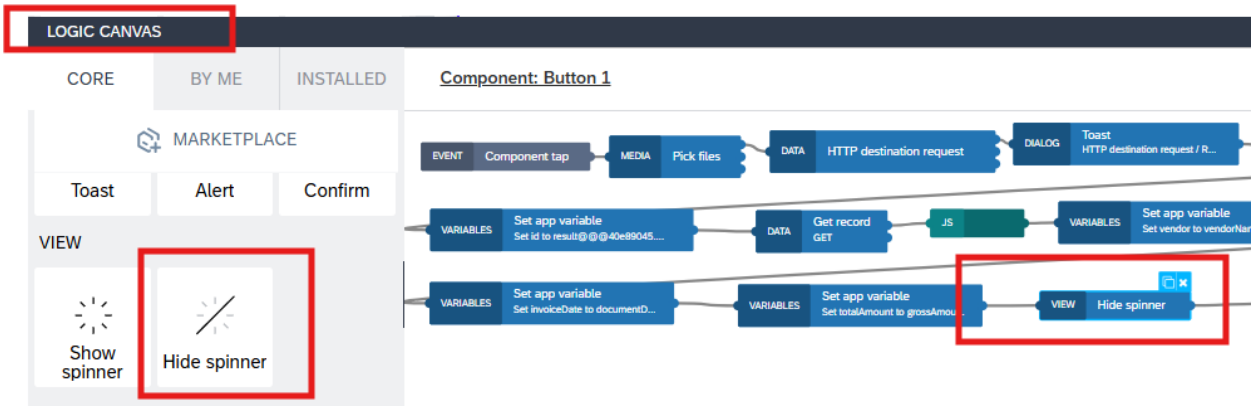

PROPERTIES
OUTPUTS
(Set app variable)
Logic: Set app variable
Sets the target app variable (or some subproperty) to a new value.
READ MORE
INPUTS
Variable name*
invoiceDate
Assigned value*
Function / documentDateText
ADVANCED


PROPERTIES
OUTPUTS
(Set app variable)
Logic: Set app variable
Sets the target app variable (or some subproperty) to a new value.
READ MORE
INPUTS
Variable name*
totalAmount
Assigned value*
Function / grossAmountText

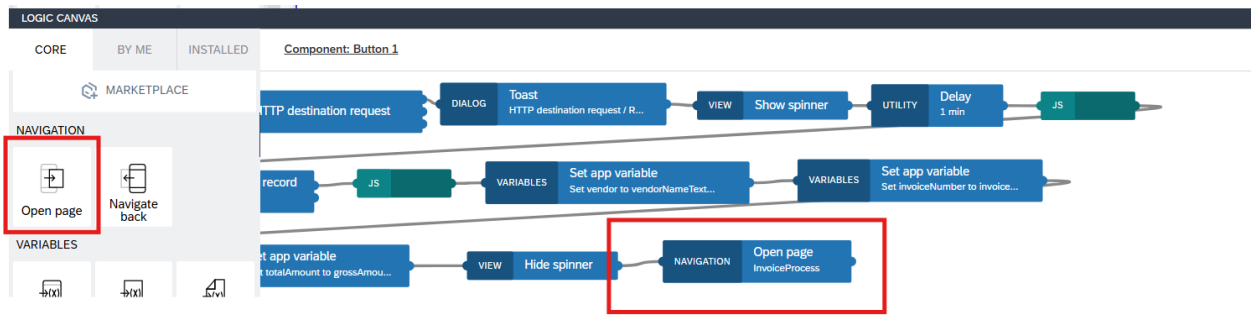

BACK
Edit binding for
Set app variable
text
Select binding type / Output value of another node
References an output value of another, previous logic node in the current logic chain. Only available for flow function input argument bindings.
READ MORE
Select logic node
Function
Select node output
Function / documentDateText
SAVE



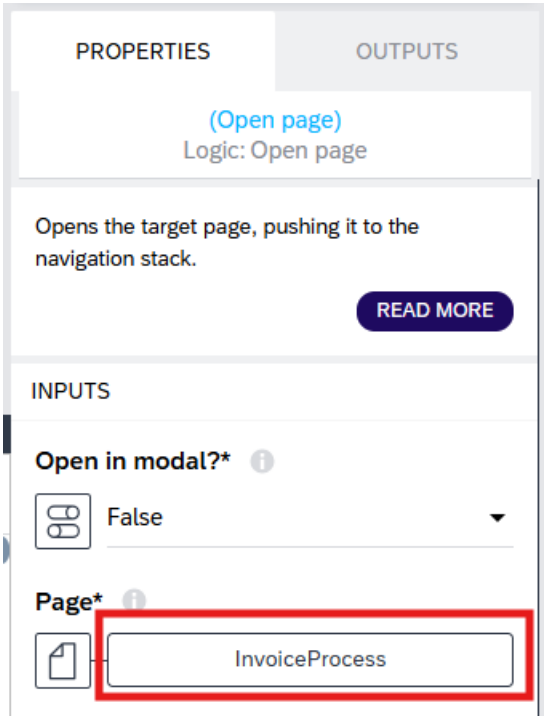
Step 57: Drag Hide Spinner from Logic Canvas and drop it after set app variable component. Connect Hide Spinner to previous node.

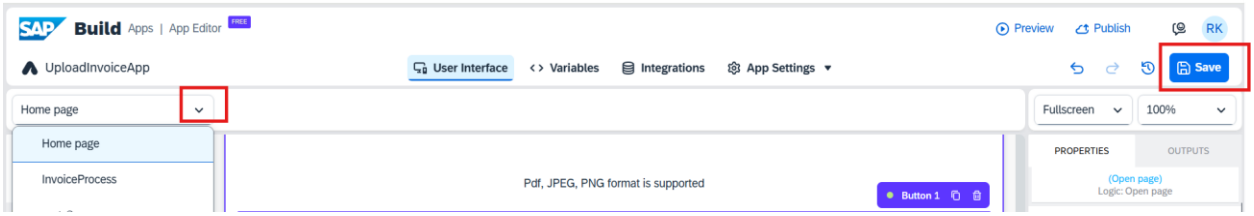


Step58: Drag Open Page from Logic Canvas and drop it after Hide Spinner component. Connect Open page to previous node.

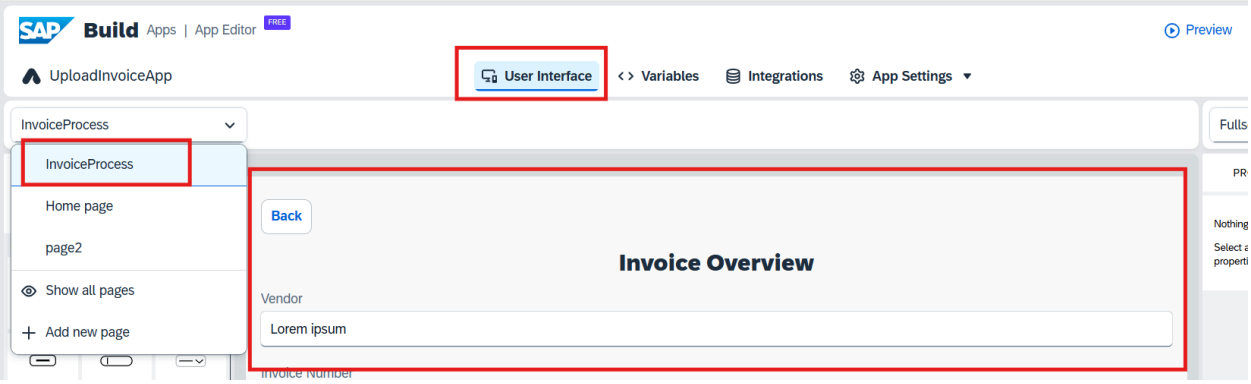


Step59: Add Page detail in Properties. Click on the Open page component and on the right side you can see Properties. If you're not sure about page name, you can find it from drop down (refer second screenshot)

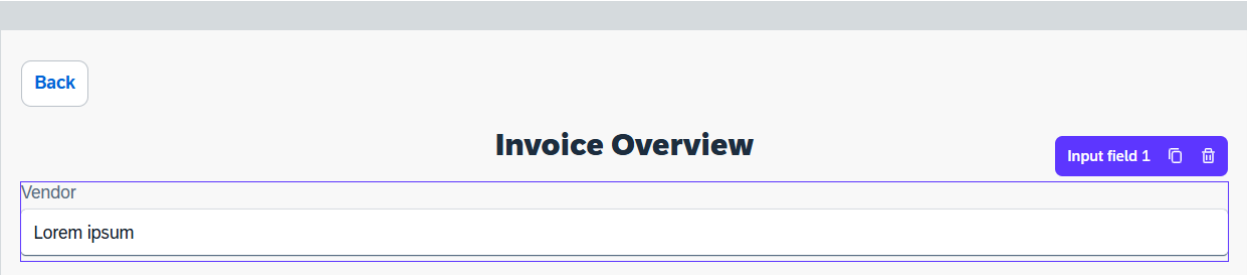


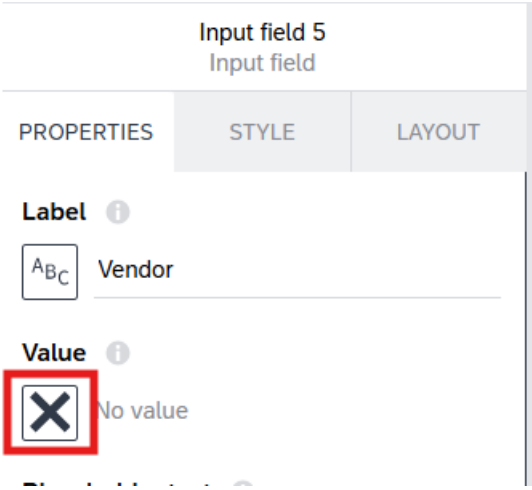


Step60: Select the second Screen from the drop down available at the header of the canvas.

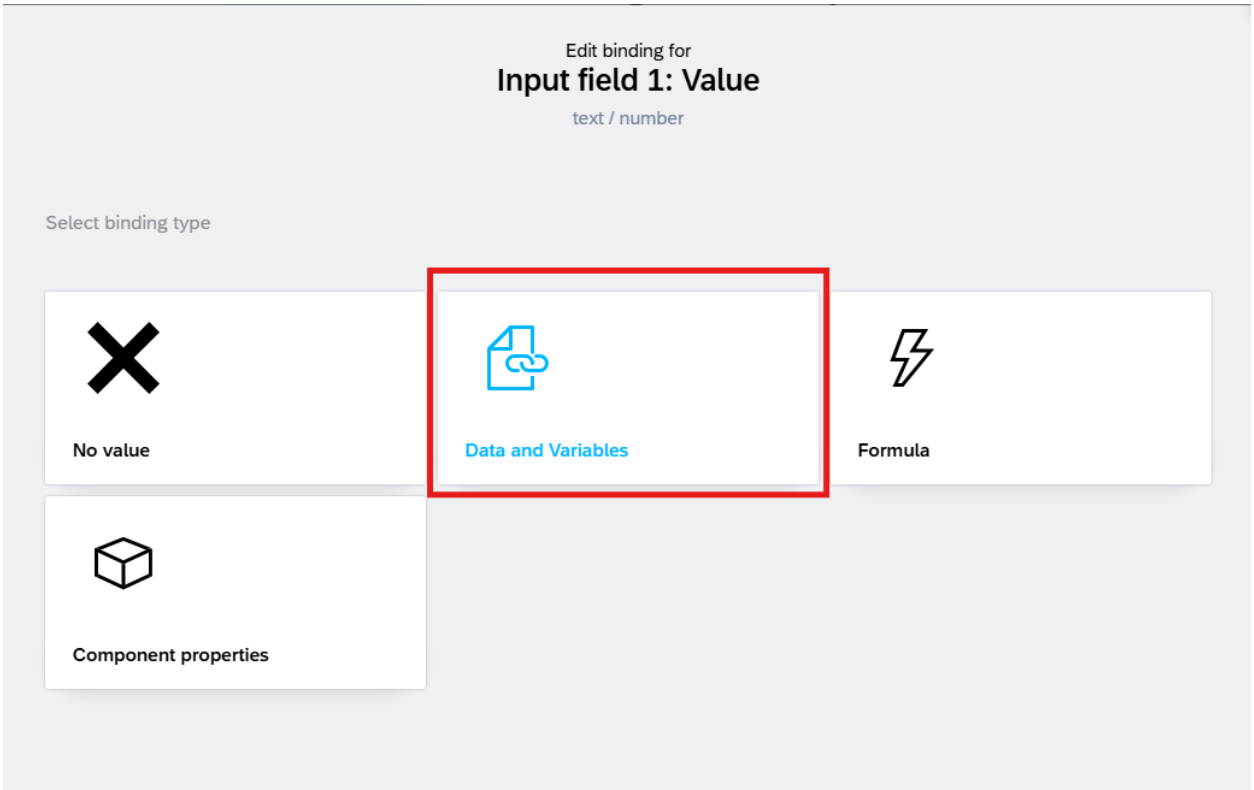


Step 61: Click on First Input Field and in right you see the properties option available for it. (We must repeat the same step for next 3 variables too)

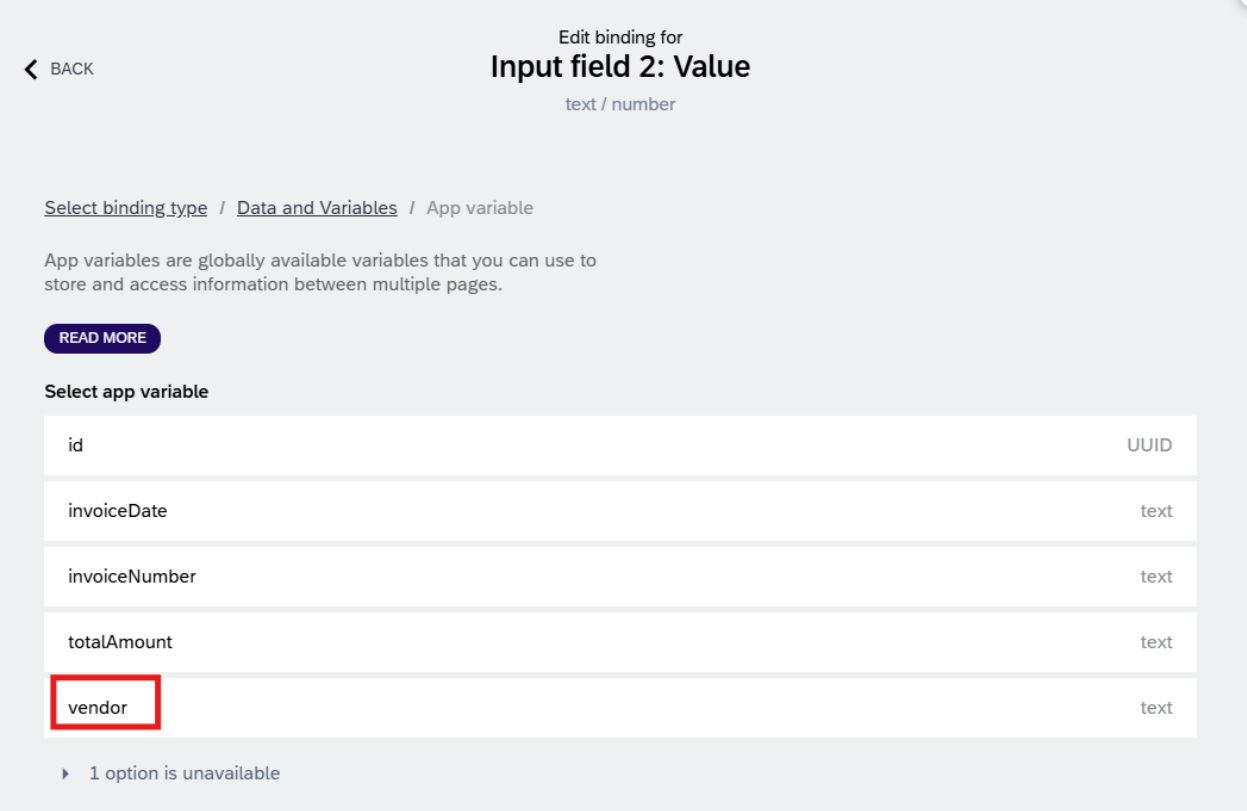




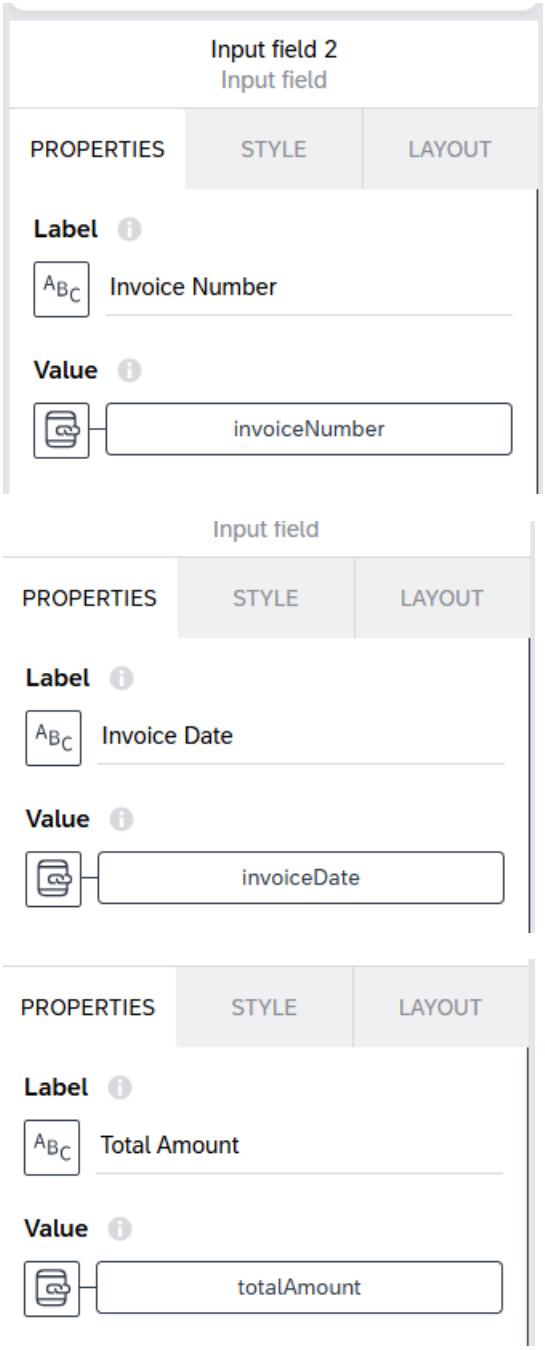
Step 62: Click on X icon and select the Data and variables



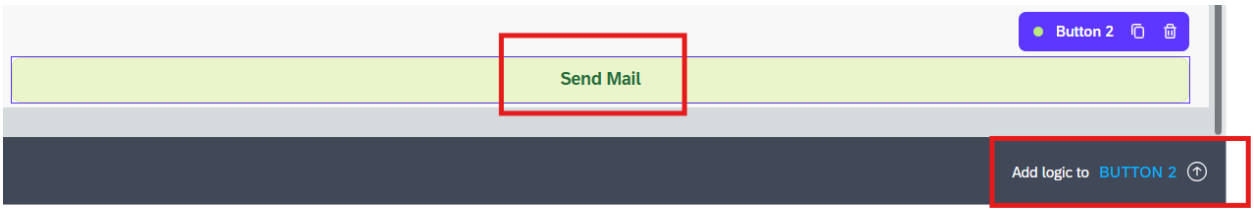
Step 63: Select the variable



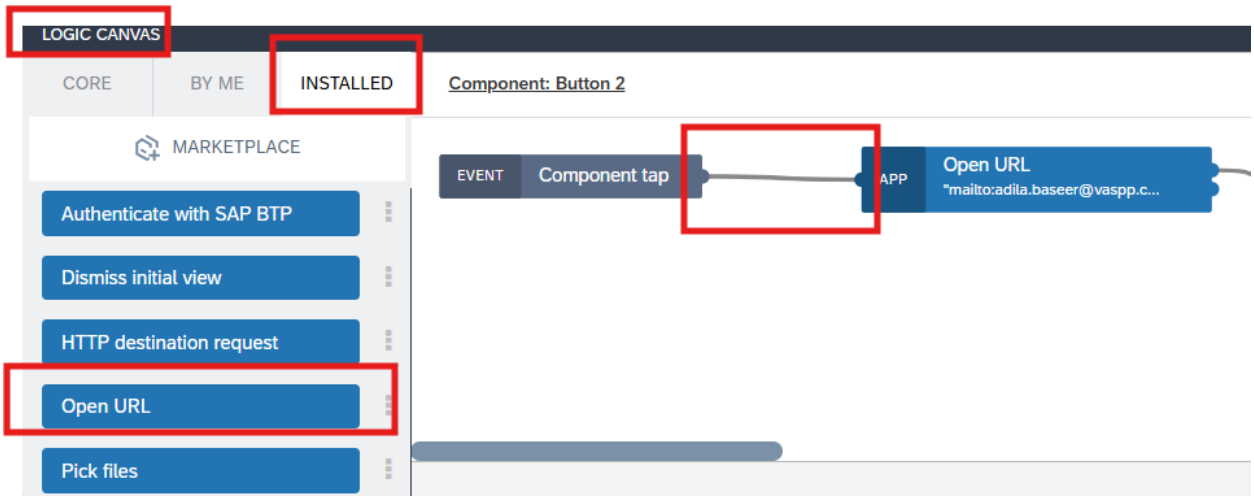
Step 64: Repeat the step for other input fields too.



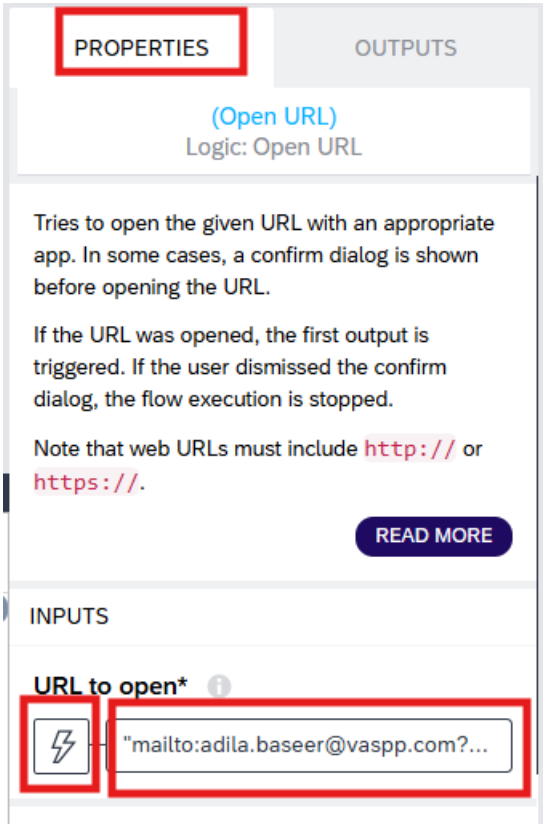
Step65: Click on Send Mail button and click on “Logic editor”

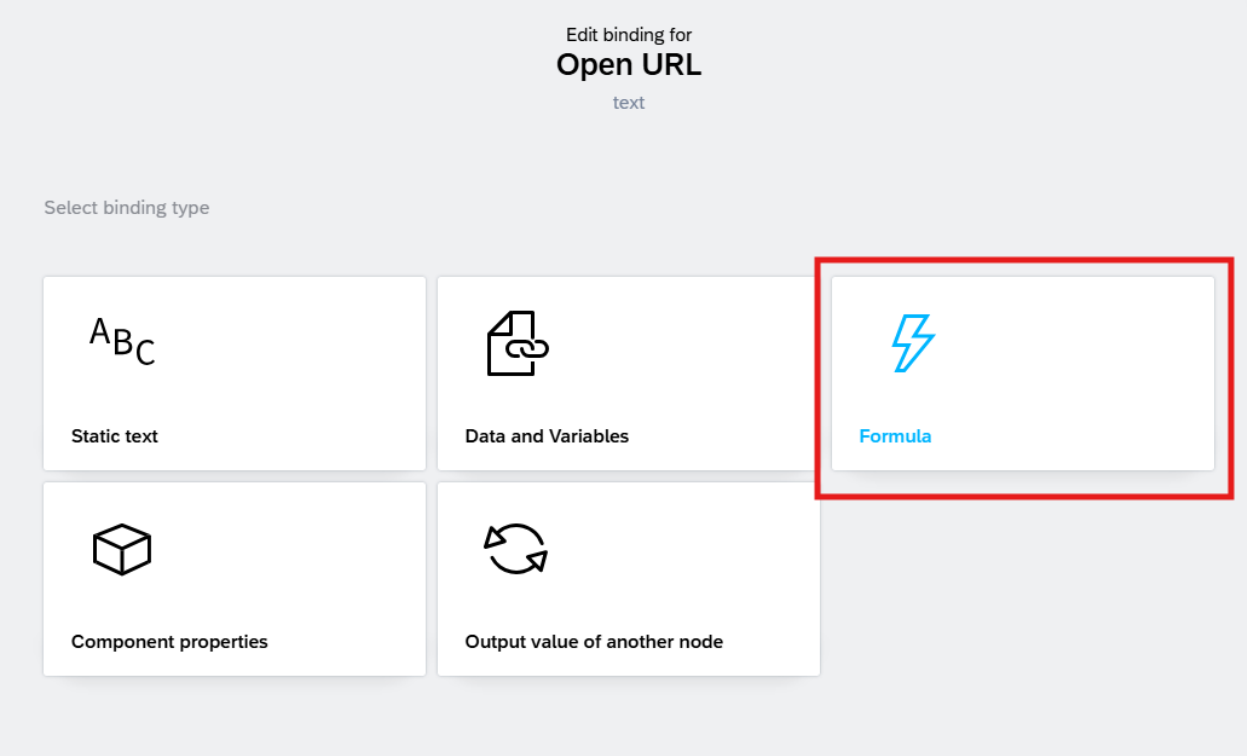


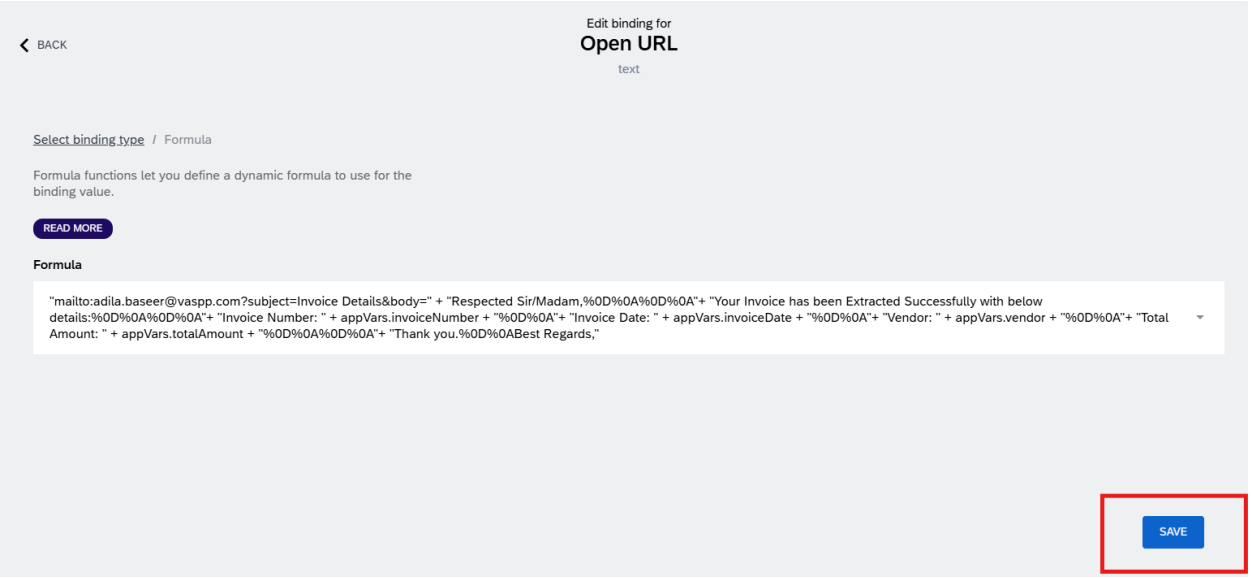
Step66: Go to Logic Canvas -> go to Installed tab --> Drag Open URL and Drop it on logic editor -> Connect it to previous node.



Step67: Select Open URL -> go to right side Properties --> Click on “ABC” option and select the formula option -> Copy and Paste below snippet







Code Snippet: Note: make sure you replace <Enter_Receiver_MailID> with actual mail ID.

"mailto: <Enter_Receiver_MailID>?subject=Invoice Details&body=" + "Respected Sir/Madam,%0D%0A%0D%0A"+ "Your Invoice has been Extracted Successfully with below details:%0D%0A%0D%0A"+ "Invoice Number: " + appVars.invoiceNumber + "%0D%0A"+ "Invoice Date: " + appVars.invoiceDate + "%0D%0A"+ "Vendor: " + appVars.vendor + "%0D%0A"+ "Total Amount: " + appVars.totalAmount + "%0D%0A%0D%0A"+ "Thank you.%0D%0ABest Regards,"

Step64: Go to “Logic Canvas” --> Drag “Navigate back” component and drop it after “Open URL “ Component --> Connect it with previous Node