Creating Dashboards Using Ignition Edge

ABSTRACT:

- Generally in monitoring the data visualization plays a major role.
- The main goal of this document is to get complete information regarding the exposure of the dashboards from the data that will be extracted.
- The problem in this is finding the errors that occurred during the process and finding it is so difficult but that is for a minor reason due to the complexity of handling many servers at a time.
- So, this helps to solve those major errors which are for minor reasons.
- Make easier to do dashboards in Ignition Edge under the proper research explaination.

AGENDA

- Introduction
- Steps from extraction to visualization
- Set Up KepserverEX and Ignition Edge
- Extraction of data into KepserverEX
- Configuration of the data from KepserverEX to Ignition Edge Designer
- Using Perspective Module of Designer
- Working
- Conclusion
- Result

INTRODUCTION

- **Visualization** is the representation of the data in a visual format, such as charts, graphs, diagrams, or images, to make it easier to understand, analyze, and interpret but coming to the real time data representation it is necessary to have a platform that needs to handle large amount of data that is generated from the industries and also to store the history of data to analyze it.
- **Ignition Edge** is an Industrial Automative platform which allows to receive and store the data which is in large amount and make an easy approach to built the dashboard but here the data is not directly generated.
- The data that is generated is industrial-based and is stored according to the protocols and the data is mostly generated and stored in the data logs.
- So, the data which is stored in a log is going to extract from the platform which is the **KEPserverEX** based upon the different protocols and it is a Connectivity platform that just connects to the servers to extract or to transfer the data.
- In this project, data that tranfered to the OPC UA protocol and based upon that, data transfer takes place to the Ignition Edge.
- The visualization is processed in the inbuilt application of Iginition i.e, **Designer Launcher** in which the data is converted into a visual representation.

STEPS FROM EXTRACTION TO VISUALIZATION

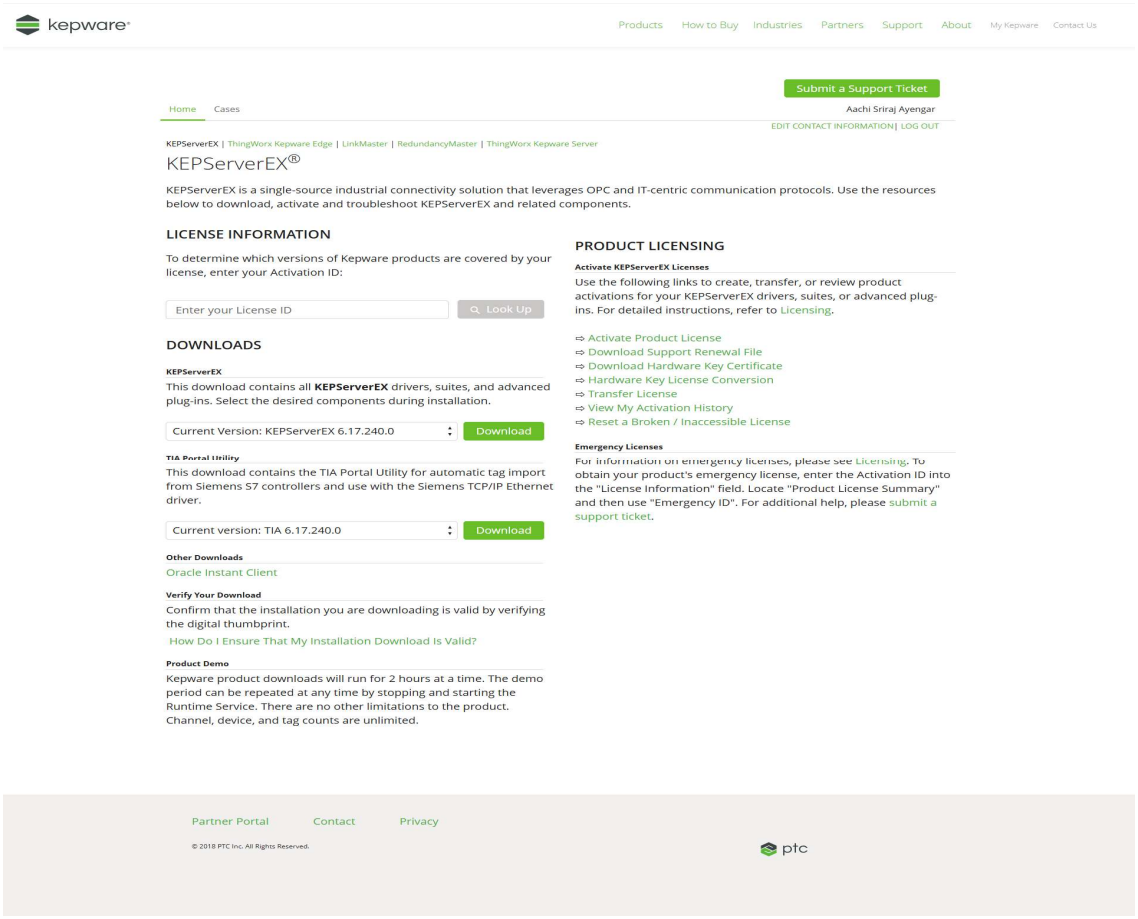
There are various steps in the process of creating dashboards from the extraction of the data but the major things that are included in the process are:

- Set up of the servers
- Extraction of data into KepserverEX
- Configuration from KepserverEX to Ignition Edge Designer
- Using Perspective Module of the Designer

SET UP KEPSERVER-EX AND IGNITION-EDGE

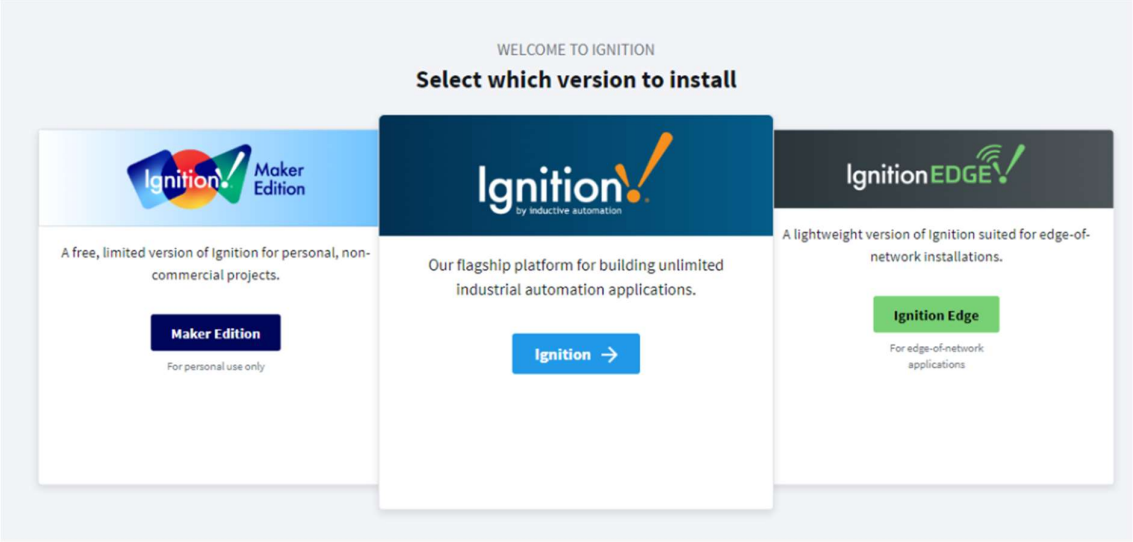
- **Installation Of KepserverEX**:

1. Install KepserverEX through the link "https://my.kepware.com/s/login/SelfRegister".
2. Register to create a Kepware account and after verification of the account, after login to that account through the link "https://my.kepware.com/s/login/", select the latest version and install the KepserverEX.



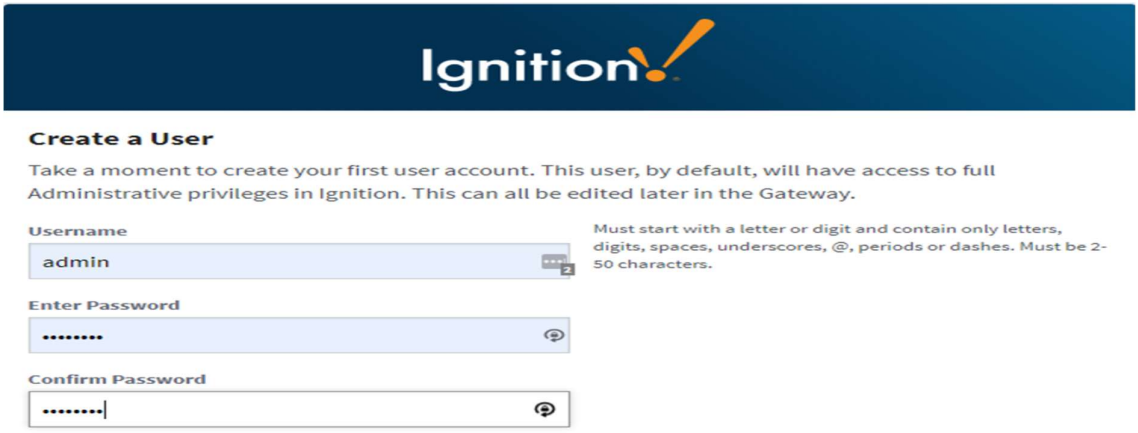
- **Installation Of Ignition Edge**:

1. Install Ignition Edge through the link "https://inductiveautomation.com/downloads".
2. After download, during installation of the server, select "Ignition Edge" in the 'Setup' and click on the 'Next'.



Note: In trail mode of the Ignition Edge, "SQL Database" is not permitted and only "Edge Historian" is only used as data base.

3. Create username and password



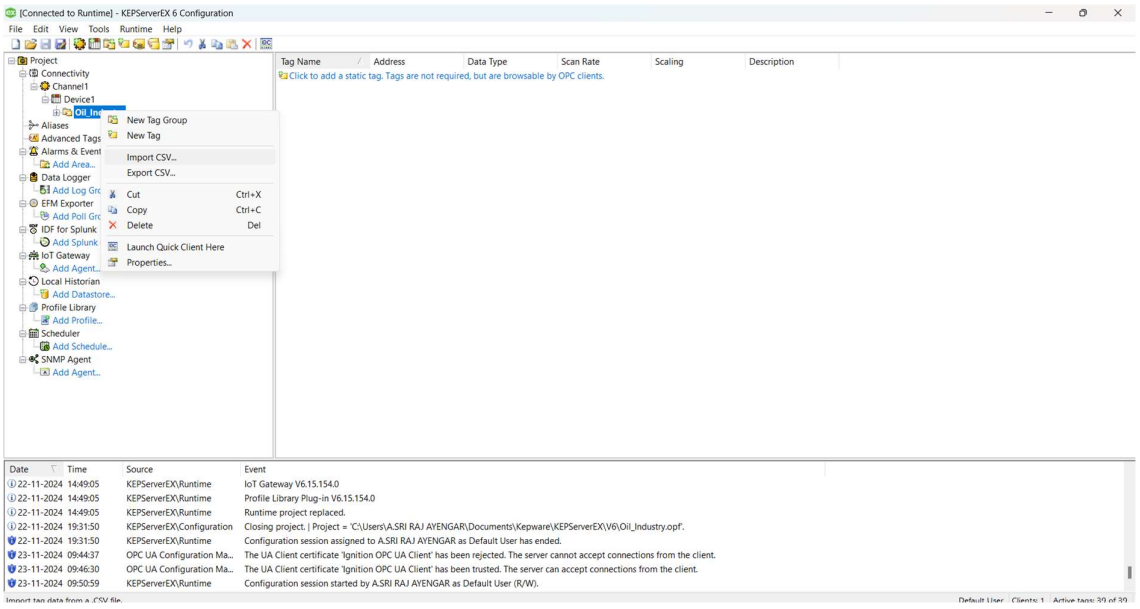
Note: Note the username and password, it is the security option that provided for the Designer

4. After complete installation it navigates automatically to the End point URL i.e., “localhost:8088”.

EXTRACTION OF DATA INTO KEPSERVER-EX

- There are two ways to add the .CSV file into the KepserverEX:

1. Open the KepserverEX Configuration application and create the channel of specific client example Simulator and after creating device under it just right click on it and click option to import '.CSV' file of that specific client.
2. Otherwise after opening the Configuration application go to the Data Logger and select option 'Yes' of 'Enable' under the configuration and select the right corner option of the DSN tab under the Data source then a window is going to open in which we can find the configure their after selecting that we can import the .CSV file.

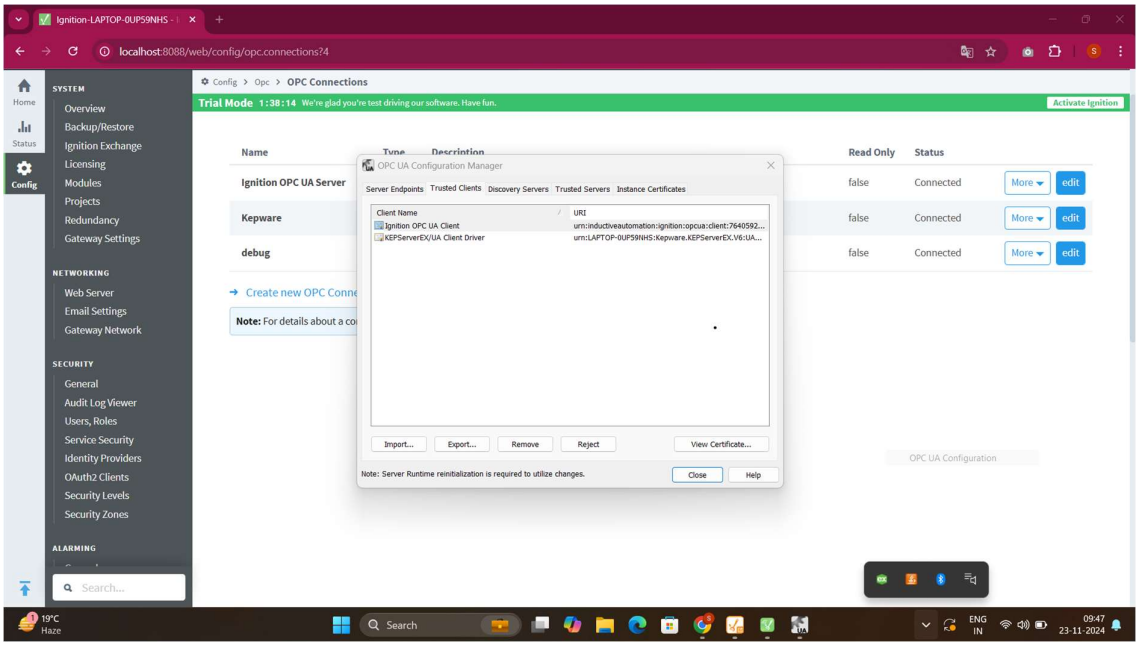


SET UP CONFIGURATION BETWEEN KEPSERVER-EX AND IGNITION-EDGE DESIGNER

- There are few steps to Set up configuration between kepserverEX and Ignition -Edge Designer, they are

- **Configuration of KepserverEX and Ignition:**

1. To configure the Ignition Edge with KepserverEX, it is possible only with OPC UA protocal.
2. Open the services application or window and start Kepserver's OPC.NET service if the service is already started then close the window.
3. Install Ignition Edge through this link and create user and password to open the Designer application of Ignition Edge and Ignition Edge Gateway.
4. After the Ignition Edge gateway is opened which have the port 8088, under configuration click on 'OPC Connections' of 'OPC client' and go to 'Create a new OPC Connection'.
5. Select OPC UA, a window opens with 'Endpoint URL' and go to hidden icons which is at the bottom of the pc and right click on the KepserverEX logo and go to OPC UA configuration their we can find the 'Endpoint URL' which is in the form of "localhost:43920".
6. Press Next after entering the Endpoint and select the Security policy and Security mode as "Basic256Sha256" and "SignAndEncrypt" and complete the process of saving OPC configuration details and at the end page give the name of the Connection and save it.

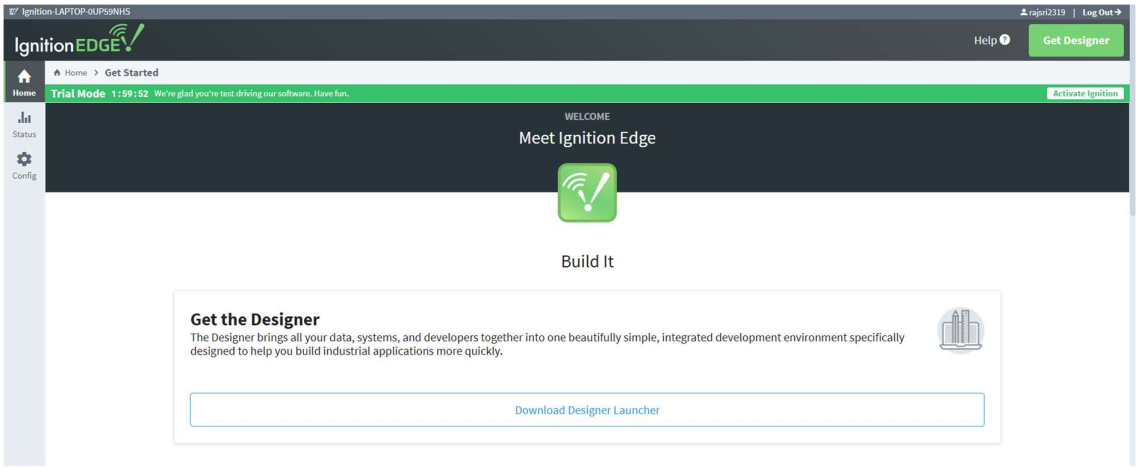


Note: It is necessary to run the Kepserver's OPC services in the background if the kepserver should connect with Ignition Edge.

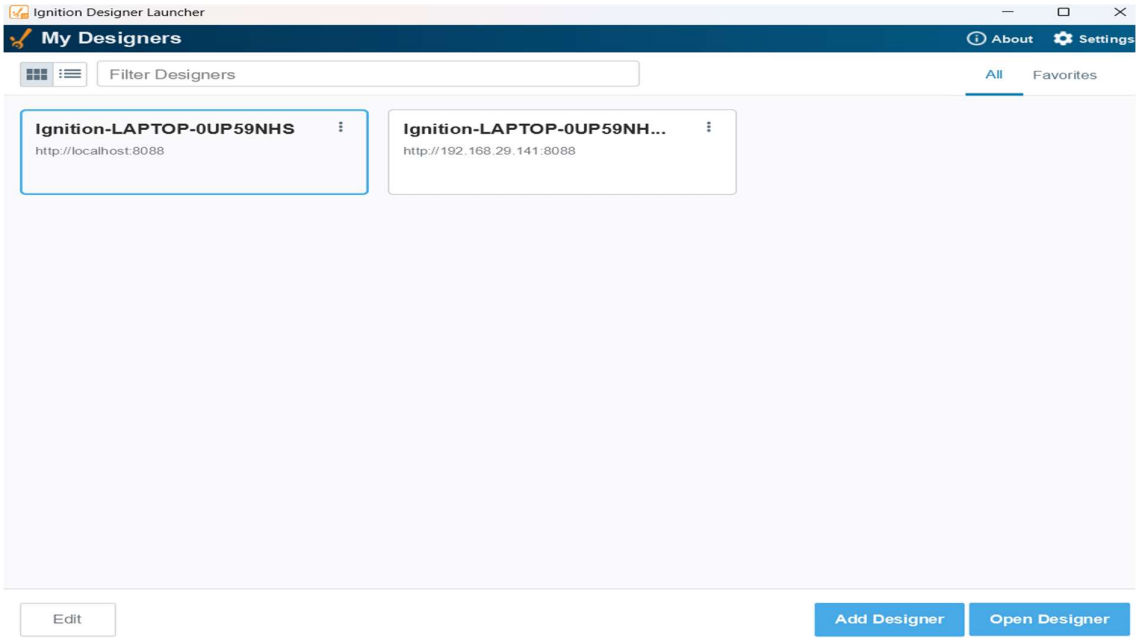
7. Then immediately OPC UA configuration window as mentioned previously, click on 'Trusted Clients' and we can find Ignition Edge OPC UA option right click on that and selected 'Trusted', Ignition Edge is connected to the KepserverEX that we can find in Ignition Edge gateway in OPC connections option of Configuration tab.

- **BROWSING THE TAGS AND ADDING TO THE DATA BASE**

1. To check whether the data is transferred to the Ignition Edge, download the **Designer Launcher** application which is available in official gateway website of Ignition Edge under the ‘Home’ option.

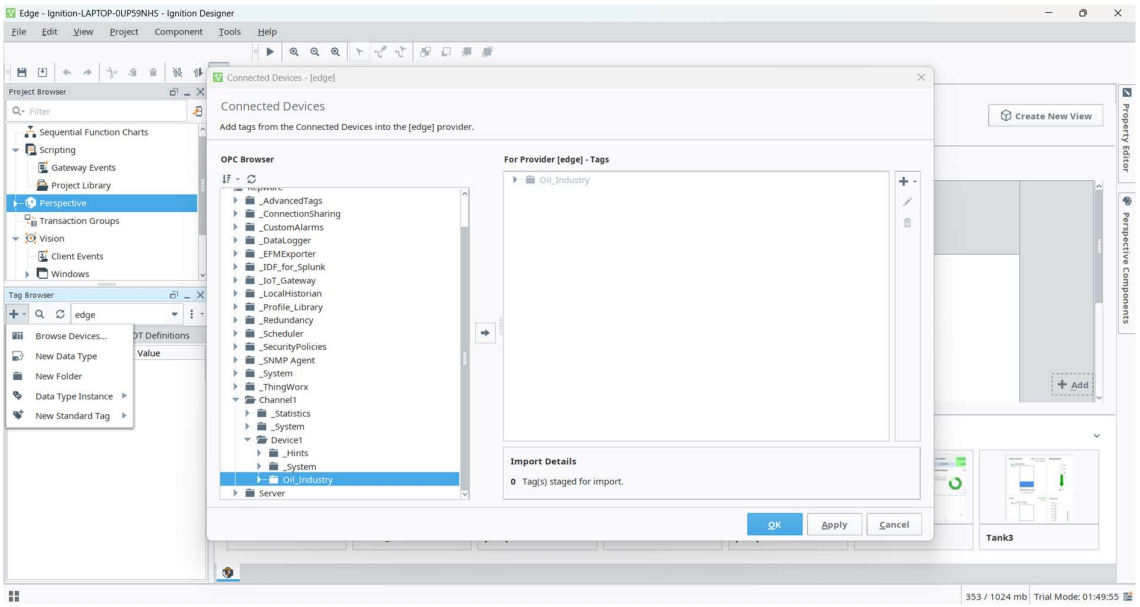


2. After completion of installing Designer select the default designer i.e, open the inbuilt OPC UA configuration user and password which are given while installing the Ignition Edge.



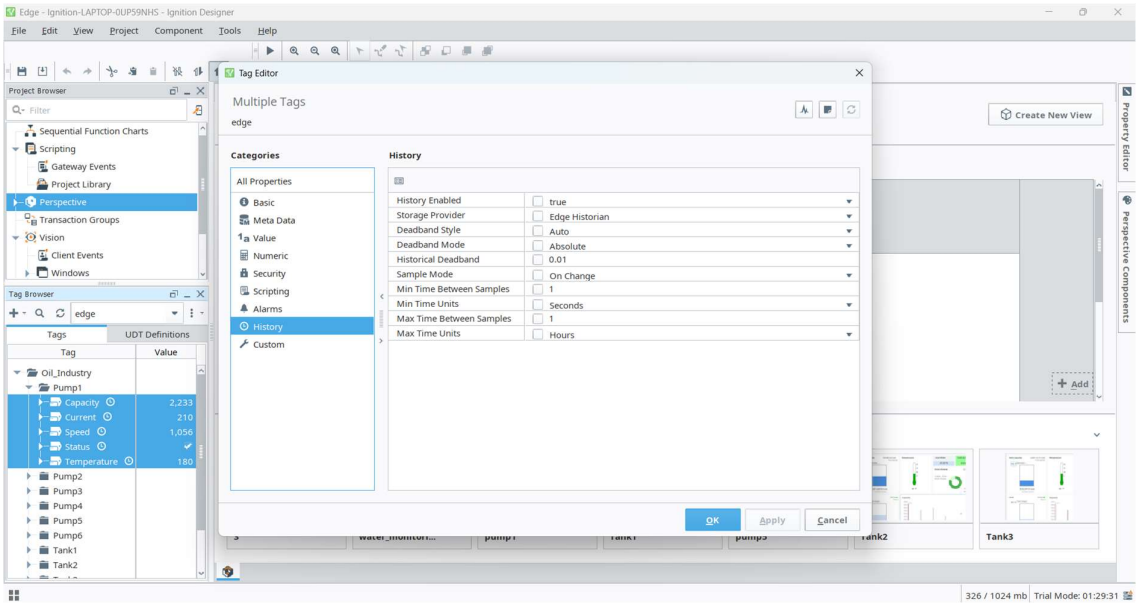
Note: New Designer can be added other than inbuilt localhost by clicking on Add Designer and click on “Manual”, their enter the URL of the host as the second designer URL and add it.

3. Under Tag Browser Click on ‘+’ option and select Browse Devices, their a window is opened in which we can see the folder of KepserverEX that which is saved previously.
4. Add the folder into Tag browser by tapping on right click widget on that window after selecting that folder and press ‘ok’ then we can find whole KepserverEX’s format of tags in the ‘Tag Browser’.



Note: No, need to pull of all the whole data folder, as in this project all the necessary data is stored in the ‘Oil_Industry’ folder.

5. To connect or to configure the Ignition Edge with KepserverEX, it is possible only with OPC UA protocal.
6. Select the tags which are going to be visually represented and right click on it, a window is opened under the “History” make ‘History Enabled’ is ‘true’, “Storage Provider” is ‘Edge Historian’, “Sample Mode” is ‘On Change’ and “Max Time Between Samples” is ‘1’ Hour and click on Okay.
7. This data of tags are stored to Database ‘Edge History’ and this is useful at the visual reprentations like graph and the representations which works based upon the previous and current values.

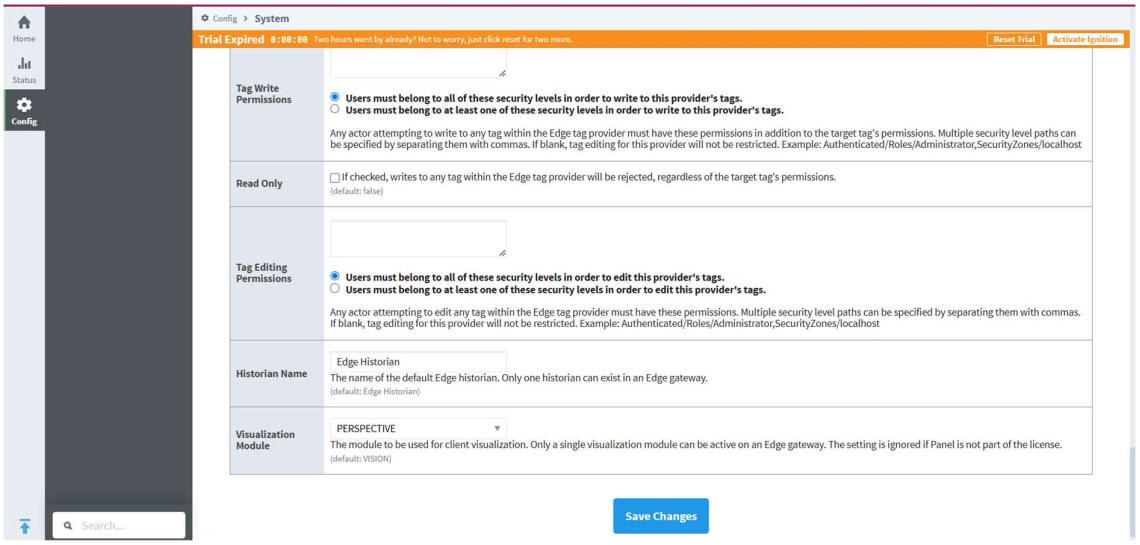


Note: In this project, the tags as used for 6 tanks and 6 pumps data as their properties and their production or monitoring of them.

USING PERSPECTIVE MODULE OF DESIGNER

- **Set Up Perspective Module:**

1. In this project, **Perspective Module** instead of **Vision Client** as in perspective we can find more number of widgets and it is also helpful for the visualization in mobiles.
2. To select perspective module go to the official gateway of Ignition Edge under ‘System’ of ‘Configuration’ select “General Settings” and select the module Visualization mode as “PERSPECTIVE” which is under ‘Edge Settings’ of ‘System’.



Note: We can operate only one module for one gateway ie., either Perspective or a Vision. It is necessary to check to activate the trial mode as it is given for every 2 hours. For every 2 hours activate the trial mode.

Perspective:

1. The Perspective Module is a web-based visualization system in Ignition that leverages HTML5, CSS, and JavaScript to create fully responsive, platform-independent interfaces.
2. It allows for the design and deployment of industrial applications that adapt to desktops, tablets, and smartphones without the need for additional configuration.
3. It provides mobile-friendly, responsive HMIs and dashboards.
4. It provides real-time data visualization and interaction.
5. It enable access to industrial data from any location using a web browser.

Perspective Elements:

a. Perspective Views:

1. A View is a single screen or interface in a Perspective project.
2. It acts as a container for components and is the building block of a Perspective application.
3. **Primary Views:** Main screens designed to display data or controls.
4. **Popup Views:** Smaller, floating windows that can display additional information or controls.
5. **Docked Views:** Fixed panels that appear alongside the main view, often used for navigation or toolbars.
6. **Embedded Views:** Views embedded within other views for reusability.
7. Each view has its own layout and can be designed to be responsive.
8. Views can be linked to URL paths, allowing for navigation.

b. Containers:

1. Containers are layout components used to organize and arrange other components within a view.
2. They define how child components are positioned and resized.
3. **Coordinate Container:** Allows absolute positioning of components based on x/y coordinates.
4. **Flex Container:** Arranges components in a row or column, flexing them to fill available space.
5. **Column Container:** Organizes components into responsive columns that adjust based on screen size.
6. **Tab Container:** Displays components within tabbed sections.
7. Breakdown Container: Automatically rearranges components based on screen size for responsive design.

c. Perspective Components:

1. Components are the individual elements that make up a user interface.
2. They include basic elements like buttons and text fields, as well as complex charts and tables.
3. **Basic Components:** Labels, buttons, text inputs, images.
4. **Display Components:** LED displays, progress bars, gauges.
5. **Input Components:** Dropdowns, checkboxes, numeric inputs, sliders.
6. **Chart Components:** Time-series charts, bar charts, pie charts, power charts.
7. **Navigation Components:** Menus, links, tab strips.
8. **Miscellaneous Components:** Maps, web browsers, video players.

d. Property Editor:

1. The Property Editor is the interface in Ignition's Designer where you configure the properties of components, containers, and views.
2. **Property Binding:** Allows properties to be linked to tags, expressions, or scripts for dynamic updates.
3. **Property Types:** Can include basic types (text, numbers) or complex types (objects, arrays).
4. **Styles and Classes:** Configure appearance by assigning styles and CSS classes.
5. **Event Handlers:** Define actions based on user interactions (clicks, hover, etc.).

e. Binding:

1. Bindings link a component's properties to data sources, allowing real-time updates.
2. **Tag Binding:** Links a property to an Ignition tag.
3. **Expression Binding:** Uses expressions to calculate a property's value dynamically.
4. **Property Binding:** Links one component property to another.
5. **Script Binding:** Uses Python scripts for complex logic.

f. Event Handlers:

1. Event handlers define how components respond to user interactions, such as clicks or mouse movements.
2. On Click: Trigger actions when a user clicks a component.
3. On Mouse Enter/Leave: Detect when a mouse enters or leaves a component.
4. On Key Press: Respond to keyboard input.

g. Sessions and Security:

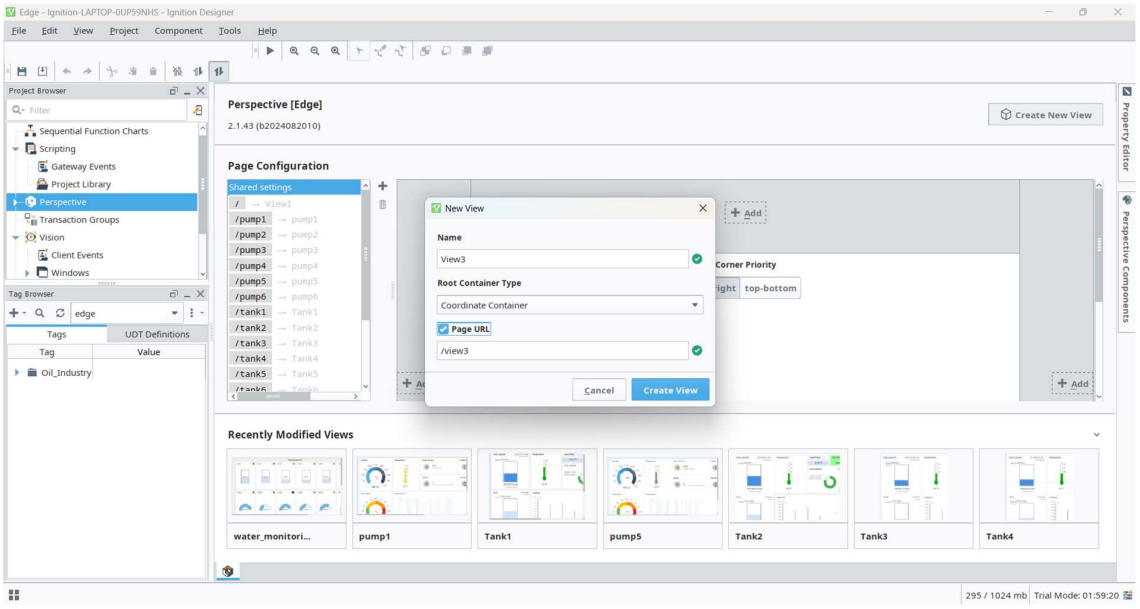
1. A session represents a user's interaction with a Perspective project.
2. Sessions track user state and data across views.
3. Perspective supports user authentication and role-based access control, allowing you to restrict access to certain views or components based on user permissions.

Development Process in Perspecvtive:

1. **Create a View:** Start by creating a new view in the Designer.
2. **Add Containers:** Use containers to structure your layout.
3. **Add Components:** Drag and drop components into the containers.
4. **Configure Properties:** Use the Property Editor to customize each component.
5. **Bind Data:** Set up bindings to connect components to tags or other data sources.
6. **Add Event Handlers:** Define interactions and user actions.
7. **Test and Deploy:** Preview your design in different screen sizes and publish your project.

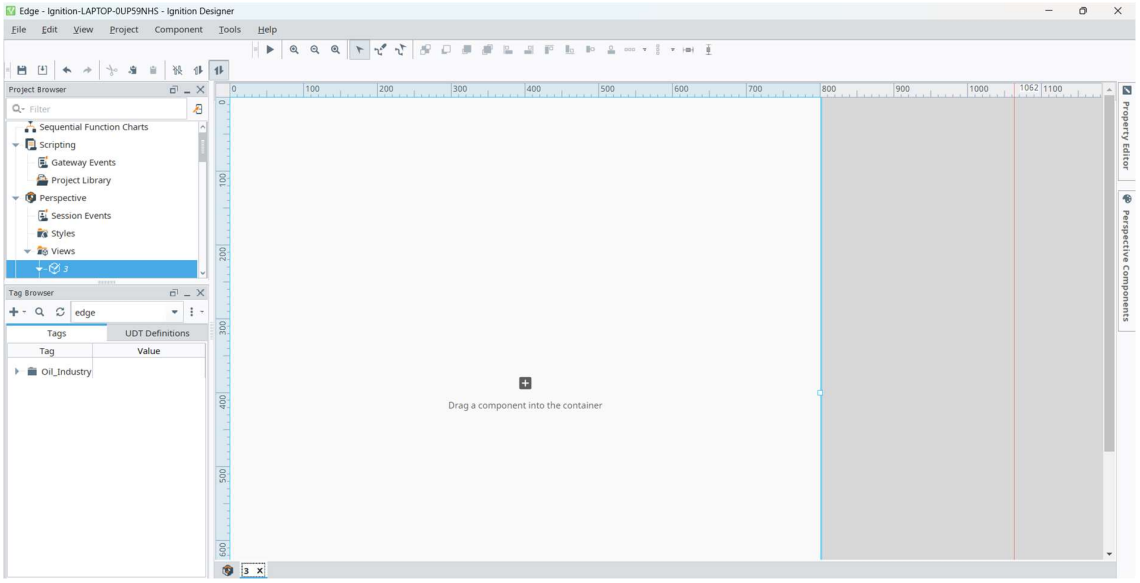
- **Development of Perspective Designer Project:**

1. On the left corner of the Designer under 'Project Browser' click on the 'Perspective', click on the 'Create New View' which is present at the right.
2. Now give a name to the view and enable the tick mark to get the visualization on the webpage and select the container type as 'Cordinate Container' as the container allows to get the visualization that fits to the pc parameter.



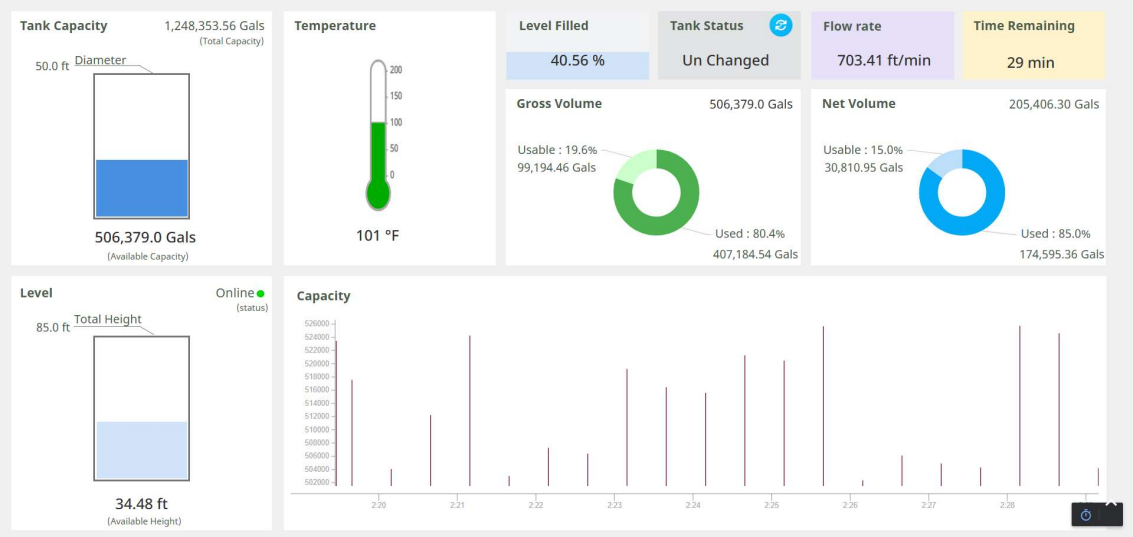
Note: Under the page Configuration as we have given Page URL as '/view3' and name as 'view 3', it is saved as '/view3 - > view3'.

3. Now after clicking on 'Create View', it directs to that view.
4. On the right corner we can see the 'Perspective components' where we can find the widgets and also 'Property Editor' where we can bind the tags.

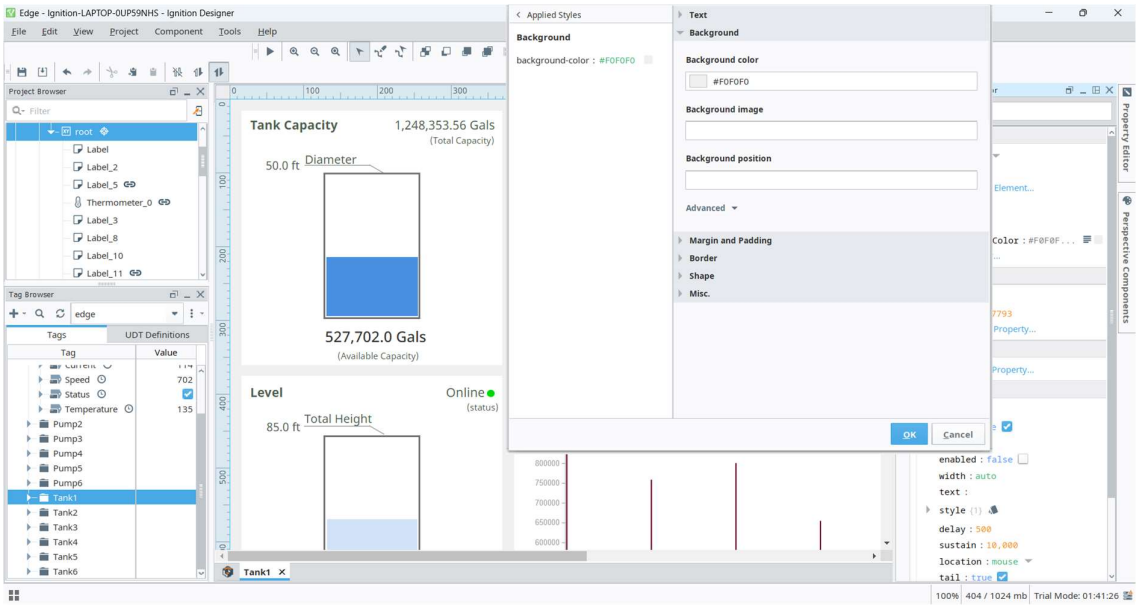


5. In this project, tags are created for the 6 tanks and 6 pumps from 1 to 6 as their properties, production or monitoring of them.
6. Accordingly 6 dashboards of tanks and pumps respectively with one monitoring dashboards are created.
7. From Monitoring Dashboard we can get contact with the 12 pumps and tanks dashboards.
8. As all the tanks and pumps dashboards having different values with the same widget.

Tank dashboard:



- Click on the white panel and move the mouse to the ‘Property Editor’ and under the ‘Props’ select the black box which is the side of the style and change the background colour to the ash colour of colour code ‘#F0F0F0’ type that color code in the back ground colour tab which is present in the background of the black box as mentioned.



- Tank Capacity widget:

1. After creating the view as previously mentioned go to the perspective component and select the ‘label’ widget.
2. Move the mouse to the Property Editor and under the position edit width and height as 354 and 345 respectively and drag the position as given below and under the props go to the black box side to the style and change the background colour to white previously it is transparent.
3. Now add another label from ‘Perspective Components’ with 131 and 198 of width and height respectively which is under the position tab of the ‘Property Editor’ and make sure to keep the position in middle of the background label widget and add in the styles black box under the props tab, in the border option give the ‘border style’ as solid and ‘border width’ as ‘3’ with the border color code ‘#757575’ and click on okay, this is for the outer layer of the tank.
4. For the inner water level take another label widget from the Components of height and width 190 and 130 as the maximum height of the water level widget is 190, note the maximum height of the widget and y position at that height
5. For the tag binding of the water level widget as we need to adjust the height of widget, open the Property Editor and under the postion move the mouse or the cursor to the height and right click on that and click on the configure binding their we can find a window for the tag binding.
6. Select the expression and in the tag browser window and give the available capacity of the tank divided by total capacity i.e.,

```
abs(({[edge]Oil_Industry/Tank1/Capacity}/((({[edge]Oil_Industry/Tank1/Diameter
}^2)*7.48*{[edge]Oil_Industry/Tank1/Height}*3.1415)/4)))
```

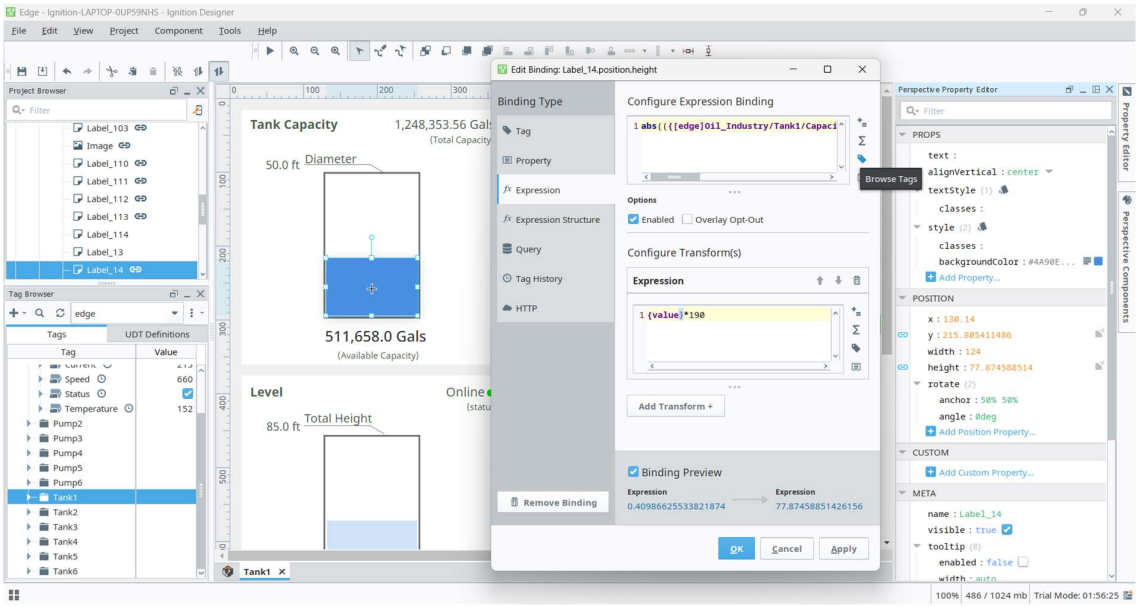
where the tags are indicated with ‘{}’ in the script so there are three tags and we can browse the tags by clicking on the tag symbol present on the right side of the script of the tag binding window. Here we have taken example of

‘Tank1’ and add transform and select expression to give the product of result and maximum height as shown below.

7. Also bind the y position like height and write the expression as previous one, in this project the expression is,

```
103.68 + (190-{this.position.height})
```

where ‘190’ is the maximum height and ‘103.68’ is the y position at that height and give the back ground color of color code “#4A90E2” in the style black box.



Note: For the labels which are taken for levels and any other background works, remove the text content in the Props of the property editor and also we can bind the tag with property based upon the need of it.

8. Now add another four labels to get total capacity of the tank and to get the available capacity another two for mentioning them.
9. For the Total capacity binding, open the configure binding as mentioned previously of the text tab of the props of the property editor, in this project the expression for binding is,

```
((({[edge]Oil_Industry/Tank1/Diameter}^2)*7.48*{[edge]Oil_Industry/Tank1/Height}*3.1415)/4)
```

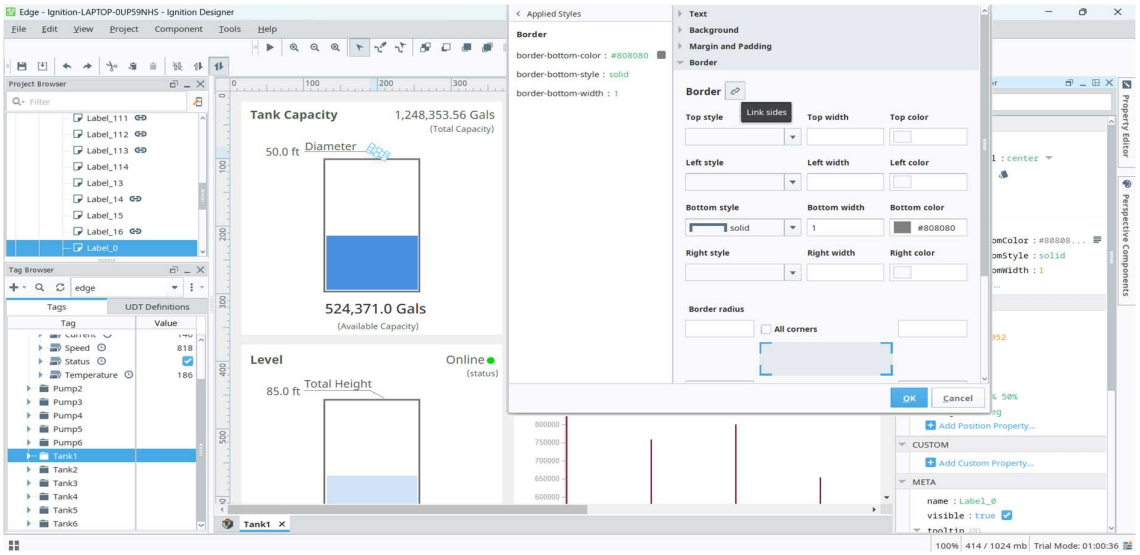
as it is the formula to get the tank volume and converting into gallons i.e, “pi*r^2*h*7.48” and give ‘add transform’ and give expression for the decimal format “numberFormat({value},"###,###.00")+" "+"Gals"” as the capacity units are taken as ‘Gals’.

10. The available capacity is directly generated in the tag as,

```
{[edge]Oil_Industry/Tank1/Capacity}
```

in expression binding of the text property of that label.

11. For the Diameter and the under lined widget and the value take three labels, for the Diameter and the rotated label go to the style of the props and under the border click on the symbol that right side to the border and under the bottom take the border style as solid and width as 1 and border color code aa ‘#808080’ and for diameter give name in the text of that label and for rotated label give the angle as ‘30 deg’ under the position of property editor.



12. For value the bind the 3rd label under the tag binding of the text of props by clicking on the tag symbol, as in this project the tag is directly generated which is the constant as it is the diameter of the tank and add transform as expression to display the units of the property as,

```
numberFormat({value}, "###,###.0")+" "+"ft"
```

- Level widget:

1. As previously how the tank widget is built in the same way the level widget is built.
2. Coming to the level of the water inside the tank, the binding of the tank is same as the taking percentage but in this case taking water level height to the total height of the tank binding maximum height is same to the tank capacity but different with the caliculation and the calculation that used in expression binding of the height is,

```
(({[edge]Oil_Industry/Tank1/Capacity}/(3.1415*(({[edge]Oil_Industry/Tank1/Diam
eter}/2)^2)*7.48)))/{[edge]Oil_Industry/Tank1/Height}
```

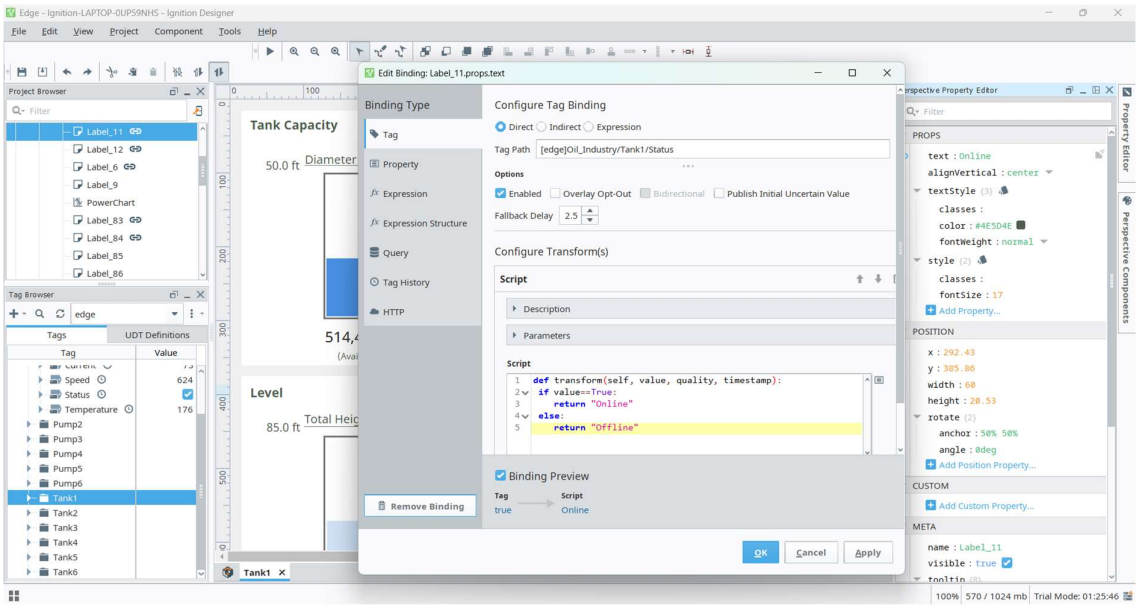
   where the formula is ratio of water_level_height and the tank height and other expression is same as the tank's liquid capacity as multiplying to the maximum height ie., 190 and in binding of y, y position is 459 ie., "459 + (190-{this.position.height})" and give the background color as the color code '#d1e3f8'.
3. Coming to the available height the binding expression caliculation is

```
({[edge]Oil_Industry/Tank1/Capacity}/(3.1415*(({[edge]Oil_Industry/Tank1/Diame
ter}/2)^2)*7.48))
```

it is the caliculation of height from the cylinder volume formula i.e, ‘V/((r^2)*pi*7.48)’.

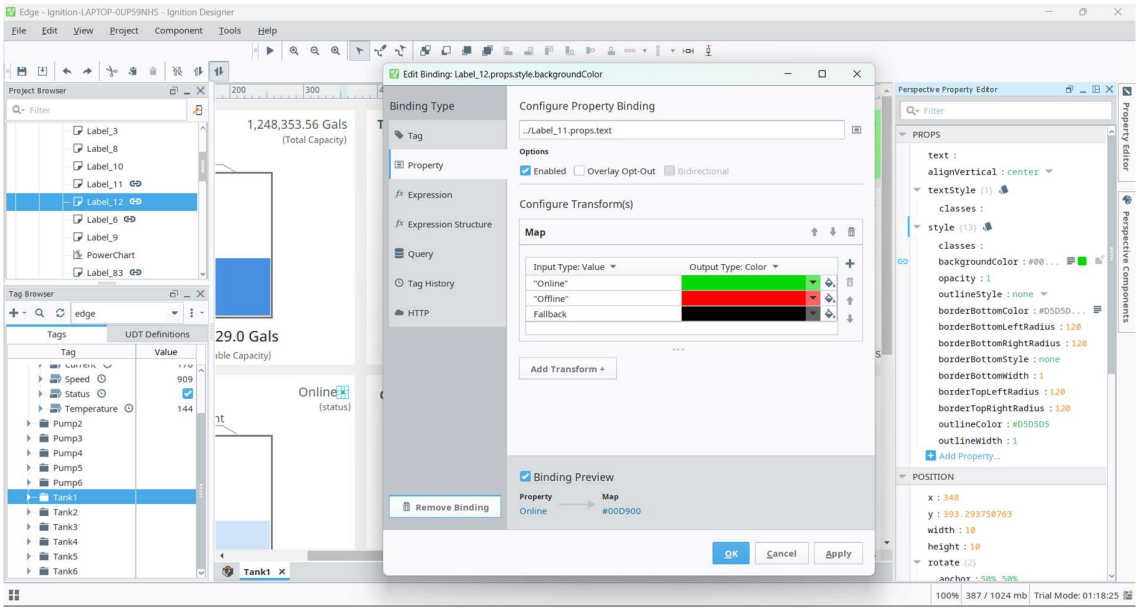
4. For the status script take a new label and bind the text as tag binding as navigate to that specific tank status tag example Tank1 status and click on ‘Add Tansform’ and select for script and type,

```
if value==True:
    return "Online"
else:
    return "Offline"
```



5. For the status symbol take a label and in the style property of the property editor, in the border section give the radius for the four sides as 120 and height and wight are 10 respectively.
6. Under the background color, first give a color then click on ‘Ok’, then under style some properties are saved.
7. In that, right click on the background and give property binding as the property in which the status script is selected, as it can be visible under the perspective

of the project browser which is present at left now select that label in the property binding and click on ‘Add transform’binding and click on the mapping and map, the colour codes as for ‘Online’ and ‘Offline’ are ‘#00D900’ and ‘#FF0000’ respectively as shown below.



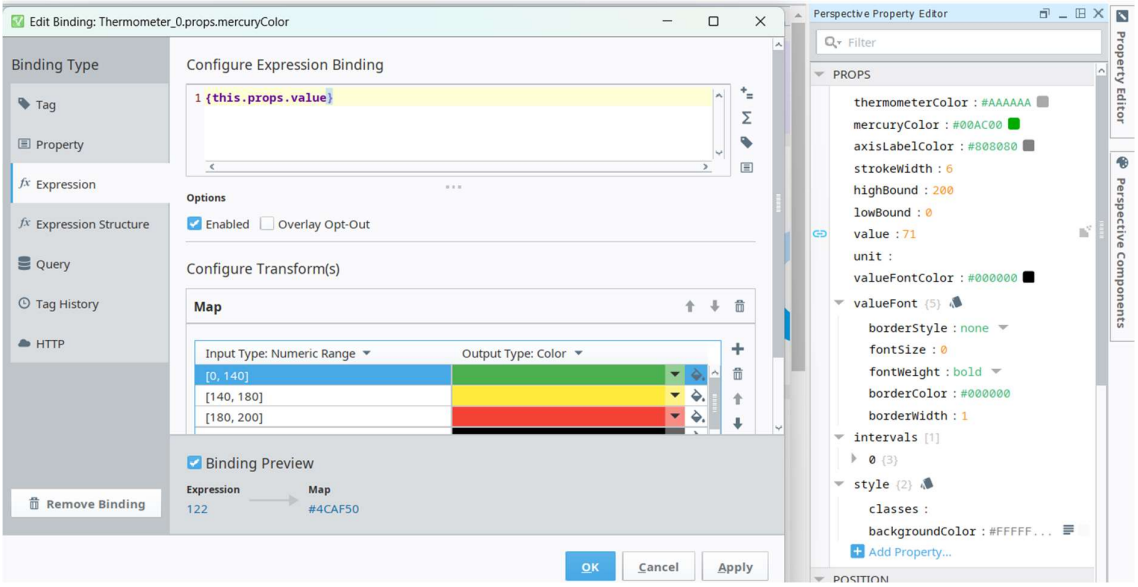
Note: To get the any property name say for status script widget it is ‘label_11’ click on that widget, on left under project bowser, that that label is marked so that it is used whenever the property binding takes place.

8. For the status and available height, built in the same as the available capacity and total capacity of the tank capacity widget.
9. Add the Total_Height and the slightly bent widget with the value as previously mentioned Diameter widget in the tank capacity widget but here height of the tank tag is binded and in the expression for the units, the script is pasted,

```
numberFormat({value}, "###,###.0")+" "+"ft"
```

- Temperature widget:

1. Except the back ground widget the another component i.e., thermometer is taken directly from the perspective component.
2. Give value of thermometer at the value side heading of the property editor by tag binding i.e., bind that specific tag from the designer's tag browser.
3. Give the color code in the property editor at mercury color as for 0 to 140 "#4CAF50" and 141 to 180 "#FFEB3B" and for 180 to 200 "#F44336".
4. For the value of the temperature take another label and at the value give property binding of the value of the thermometer and give 'Add transform' as mapping with output type as 'color'.



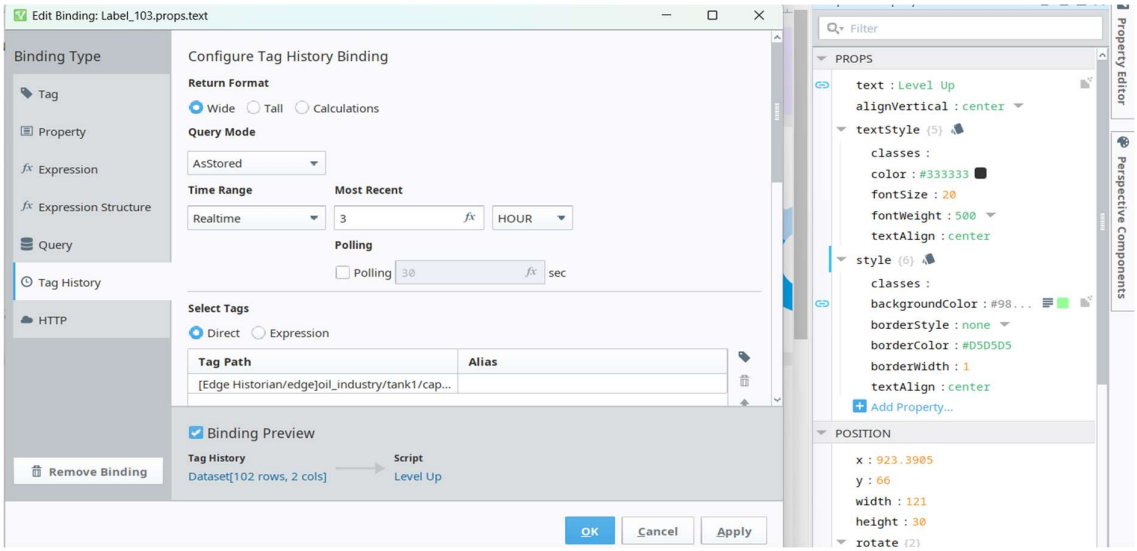
- Four small widgets(Level Up, Tank Status, Flow rate and Time Remaining):

1. Take four back ground label and for the first label i.e, for the four widgets give height and width as 93 and 194.09 respectively.
2. Give background colors "#F2F4F6", "#98FF98", "#e6e0f8", "#fff3cd" for the Level Up, Tank Status, Flow rate and the Time Remaining respectively.
3. Take 4 labels for the values and place them in that widgets.
4. For the Level Up widget for bind the expression as

```
(({[edge]Oil_Industry/Tank1/Capacity})/((({[edge]Oil_Industry/Tank1/Diameter}^
2)*7.48*{[edge]Oil_Industry/Tank1/Height}*3.1415)/4))*100
```

5. As the above formula is available capacity by Total capacity and for the level of water widget update the bindings same as the Tank Capacity or Level but it is important to check whether that new label maximum height and y value of that height is checked or not and expression formula is given in Tank capacity.
6. For the Tank Status, for the label in which value is stored, add ‘Tag history’ binding and give the tag options as given below and add the tag under the direct and it is important to add the below script that which verifies and gives the status from the previous values.

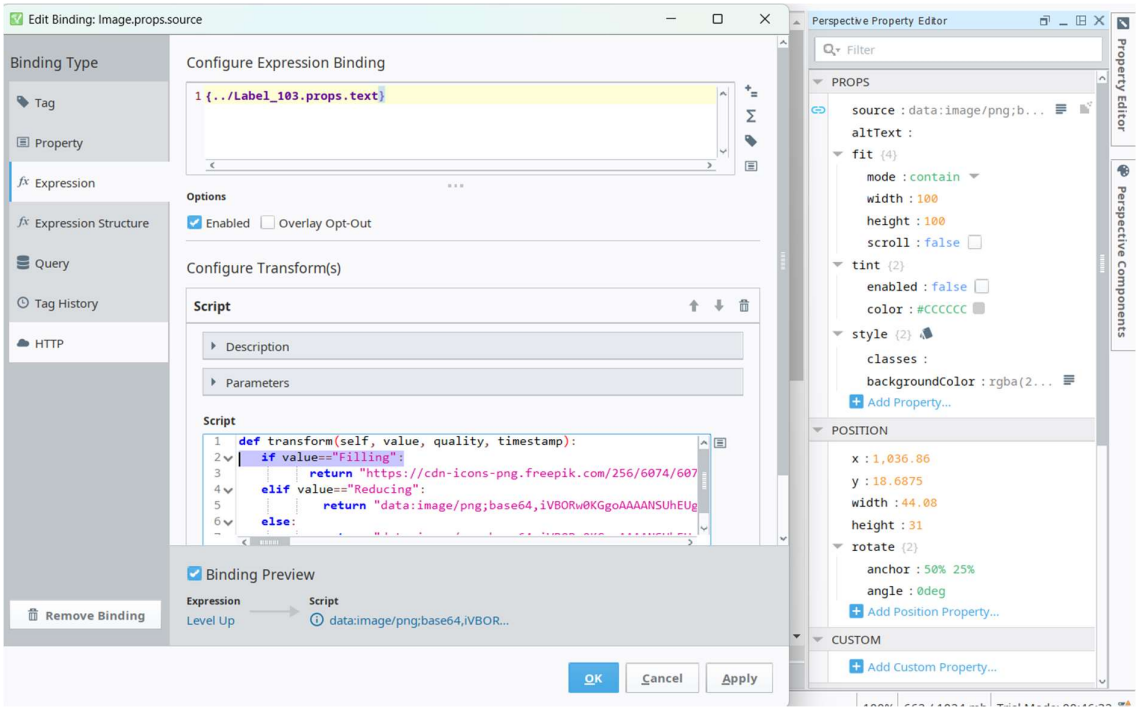
```
# Check if the dataset has at least 2 rows
 a,i,c=0,0,value.getValueAt(0,1)
 if self.getSibling("Label_11").props.text=="Online" and
value.getRowCount()>=1:
  while i<value.getRowCount():
   if i+1==value.getRowCount():
        if c>value.getValueAt(i,1):
            return "Reducing"
        else:
            return "Level Up"
   i=i+1
 else:
     return "Un Changed"

    # Compare the values
```



7. For the back ground colour use the property binding with the add transform as mapping same as the temperature widget and give color codes for Level Up “#98FF98”, for Reducing “#F8D7DA”, for Un changed “#E2E3E5” just replace in the inputs as the strings of the status and for the main panel back ground color add the property binding of that label and with the property of the background color of the value label as check on which label the value is display and in binding navigate to that label.
8. For displaying symbols, add a component with the name ‘Image’ from the perspective components and at the source do property binding as navigate to the value property of label in ehich the status is displaying and add transform as scripts as

```
if value=="Filling":
  return “url_of_image_of_increasing”
Elif value== “Reducing”:
  return “url_of_image_of_decreasing”
Else:
  return “url_of_image_of_reloading”
```

9. Give the values of the image component as given in the below image



Note: If the image is copied directly from the internet copy the direct address of that image, if image is not from the internet if it is already present in the pc then download the url of that image by converting it from the internet and copy that url and paste in the script.

10. To calculate the Flow rate first add the Time remaining Tag binding in the time remaning label and remain the same background color.
11. For the Flow rate give the Tag history binding of tag capcity in which available capacity is stored and add two expression add transforms, in the first add the below as it will caliculate the available volume or capacity difference of the previous and current values.

```
# Check if the dataset has at least 2 rows
 i,a=value.getRowCount()-1,0
 while i<value.getRowCount():
  if value.getValueAt(i,1)!=value.getValueAt(value.getRowCount()-2,1):
    a= abs(value.getValueAt(value.getRowCount()-2,1)-value.getValueAt(i,1))
    break
```

```
  i=i-1
 return a
    # Compare the values
```

12. The second expression add the script in which it will caliculate the ratio of that difference of that volume with the time and the time is which previously added in the Time remaining widget i.e, adding value property of that label in the divisor place.

```
numberFormat(({value}/{[edge]Oil_Industry/Tank1/Time}),"###,###.00")+"
"+"ft/min"
```

- Gross Volume widget:

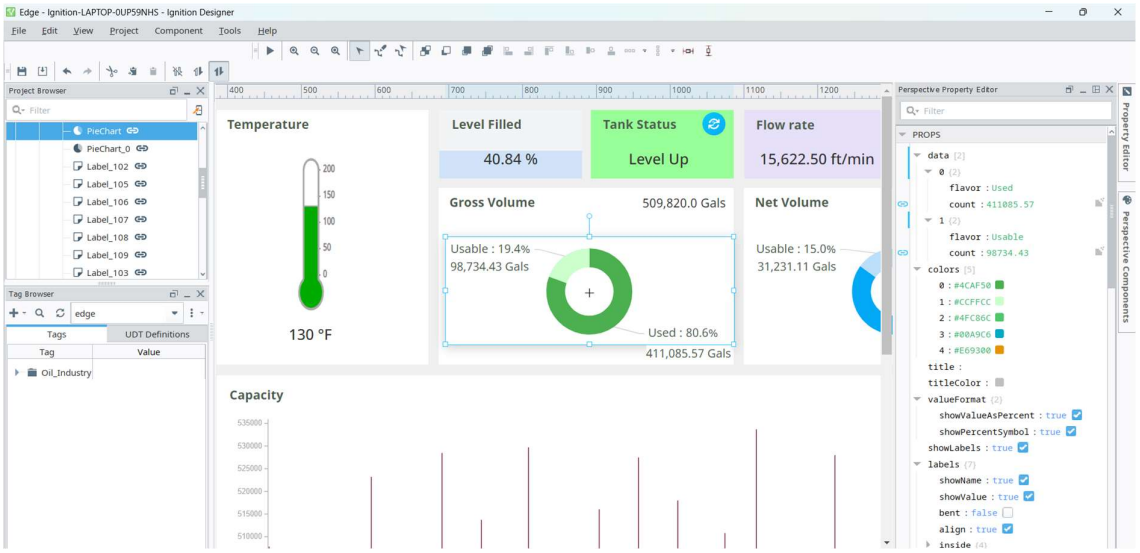
1. For the gross volume add a pie chart component from perspective component and for the total gross volume add a label with value of tag binding as available capacity.
2. For the pie chart for the used under the 0 as given below in the image add binding for the count and in the favour replace with used, give the expression binding as,

```
({[edge]Oil_Industry/Tank1/Capacity})-
(3.1415*(({[edge]Oil_Industry/Tank1/Diameter}^2)/4)*(85-
({[edge]Oil_Industry/Tank1/Capacity}/((3.1415*(({[edge]Oil_Industry/Tank1/Diam
eter}/2)^2)*7.48)))))
```

3. The expression gives the capcity difference with the available capacity and the remaning capacity.
4. For unused or usable the expression binding is

```
({[edge]Oil_Industry/Tank1/Capacity})-(({[edge]Oil_Industry/Tank1/Capacity})-
(3.1415*(({[edge]Oil_Industry/Tank1/Diameter}^2)/4)*(85-
({[edge]Oil_Industry/Tank1/Capacity}/((3.1415*(({[edge]Oil_Industry/Tank1/Diam
eter}/2)^2)*7.48))))))
```

5. That is total gross capacity and used gross capacity difference.



Note: Place only 0 and 1 and delete remaining in the data of the props for used and usable.

6. Add color codes as shown in the above image for 0 and 1 and click on the show percent and show percent symbol.
7. Add two labels for the values of the used and usable gross volumes and copy and tag properties in the values of them.
8. Scroll down in the props of property editor and give ‘cutoutRadius’ as 45 which gives donut shape.

- Net Volume widget:

1. Same as the above add a label for the Net Volume value and give expression binding for this as,

```
({[edge]Oil_Industry/Tank1/Capacity}*(({[edge]Oil_Industry/Tank1/Capacity})/(
(({[edge]Oil_Industry/Tank1/Diameter}^2)*7.48*{[edge]Oil_Industry/Tank1/Height
}*3.1415)/4))))
```

2. The net volume formula in the expression is product of the Gross volume and the ratio of the total volume to available volume.
3. The used and usable formulas are,

```
{[edge]Oil_Industry/Tank1/Capacity}*(({[edge]Oil_Industry/Tank1/Capacity}/(3.1
415*(({[edge]Oil_Industry/Tank1/Diameter}/2)^2)*7.48)))/100
```

```
({[edge]Oil_Industry/Tank1/Capacity}*(({[edge]Oil_Industry/Tank1/Capacity})/(
(({[edge]Oil_Industry/Tank1/Diameter}^2)*7.48*{[edge]Oil_Industry/Tank1/Height
}*3.1415)/4))))-
({[edge]Oil_Industry/Tank1/Capacity}*(({[edge]Oil_Industry/Tank1/Capacity}/(3.
1415*(({[edge]Oil_Industry/Tank1/Diameter}/2)^2)*7.48)))/100)
```

4. For the used net volume formula is used product of the Gross volume and the same ratio and usable is difference between used and total gross volumes.
5. Reaming properties of the pie chart is same as the Gross pie chart.

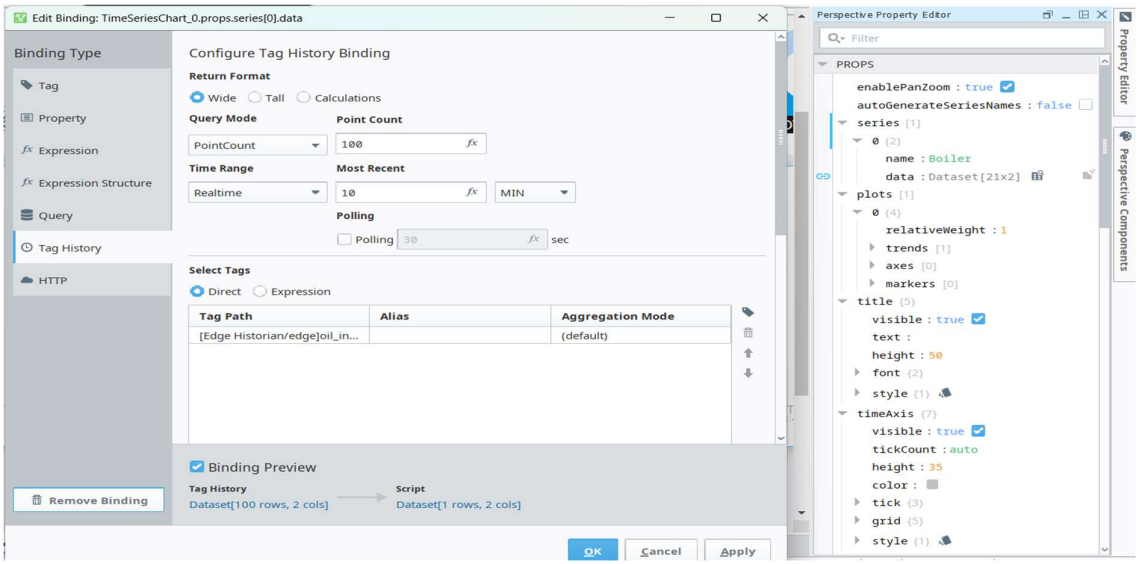
- Bar Chart:

1. Take a Time series Bar chart from the Perspective components under the 0 of series of props of the property editor, under the data delete all the previous data from 0 to 9.
2. Bind the Tag Hitory under by right clicking on the data as given below.
3. Add the tag of available capacity and click on add transform and select script and follow the script as,

```
    import system.dataset

    # Check if 'value' is a dataset by confirming it has the required methods
    if not hasattr(value, "getColumnNames") or not hasattr(value,
"getValueAt"):
        return value  # Return the original value if it's not a dataset

    # Extract headers (columns) and dataset dimensions
    headers = list(value.getColumnNames())  # The first column is assumed to
be timestamps
    row_count = value.getRowCount()
    col_count = value.getColumnCount()

    # Use a set to track unique values across all columns (excluding
timestamps)
    unique_values = set()
    new_rows = []

    for row_index in range(row_count):
        # Get the timestamp (assumed to be the first column)
        timestamp = value.getValueAt(row_index, 0)

        for col_index in range(1, col_count):  # Skip the first column
(timestamps)
            cell_value = value.getValueAt(row_index, col_index)

            # If the value is unique, add it to the set and the new dataset
            if cell_value not in unique_values:
```

```
                unique_values.add(cell_value)
                new_rows.append([timestamp, cell_value])

    # Create new headers for the resulting dataset
    new_headers = ["Timestamp", "Value"]

    # Return the new dataset
    return system.dataset.toDataSet(new_headers, new_rows)
```

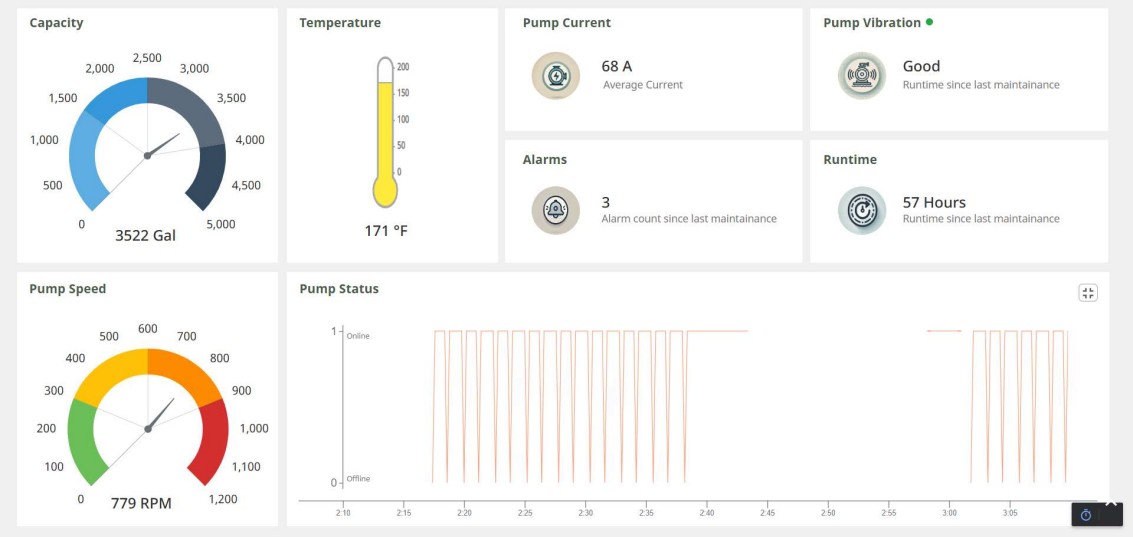
4. The above script plays a key role in add the distinct values into a new dataset.



Note:

1. Add the titles with the label of font size as 17 and color of the text is “#4E5D4E” with font weight as “bold” which are present in the text of the style option in the props of the property editor of that label.
2. Font size and colour of the subtitles like Total capcity in the Tank capacity widget or the diameter with its value are 17 and “#4E5D4E” without bold.

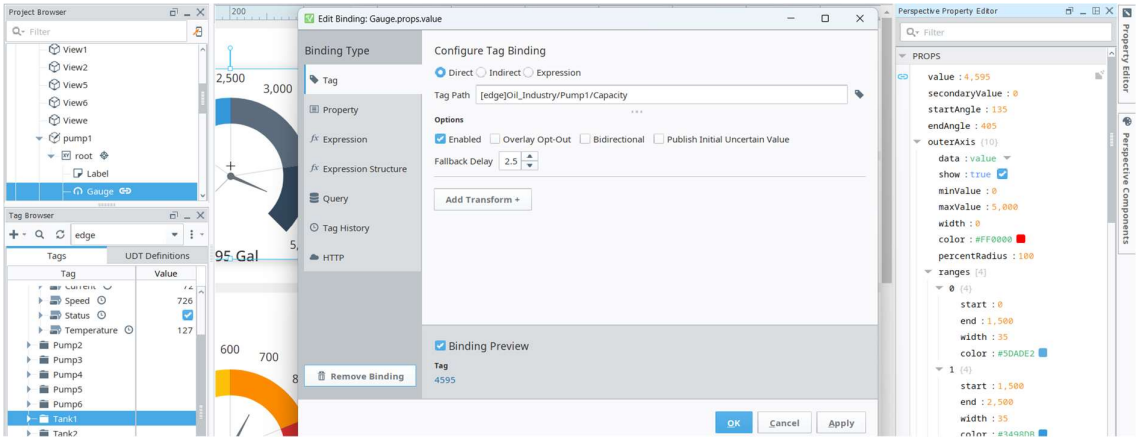
Pump Dashboard:



1. Create a new view as the tank dashboard, give name of the dashboard pump1, select the containor as coordinate and enable the URL.
2. After creating it will automatically navigates to that Design.

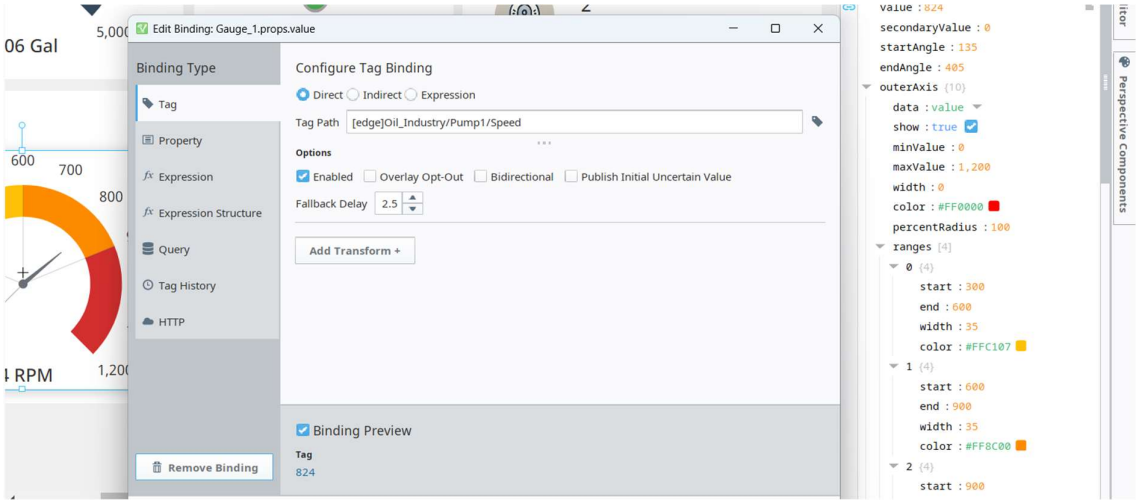
- Guage widgets (Capacity and pump speed):

3. Take a Guage component from the perspective component select full axis Guage under the Guage
4. As the background panel is common for all the dashboards which is mentioned previously in the tank dashboard.
5. Duplicate the Guage by right click on that widget and selecting duplicate.
6. Bind the tag for the first Guage as tag binding of capacity and give the colours as 0 to 1500 “#5DADE2”, 1500 to 2500 “#3498DB”,2500 to 4000 “#5D6D7E”, 4000 to 5000 “#34495E”.



Note: It is possible to delete the ranges or adding the ranges as per required by right click on that series.

7. For the speed widget guage add the speed of pump tag by the pump binding and add the colors.
8. For the values take property binding of the value of Guage for speed and capacity and add transform expression to display units.



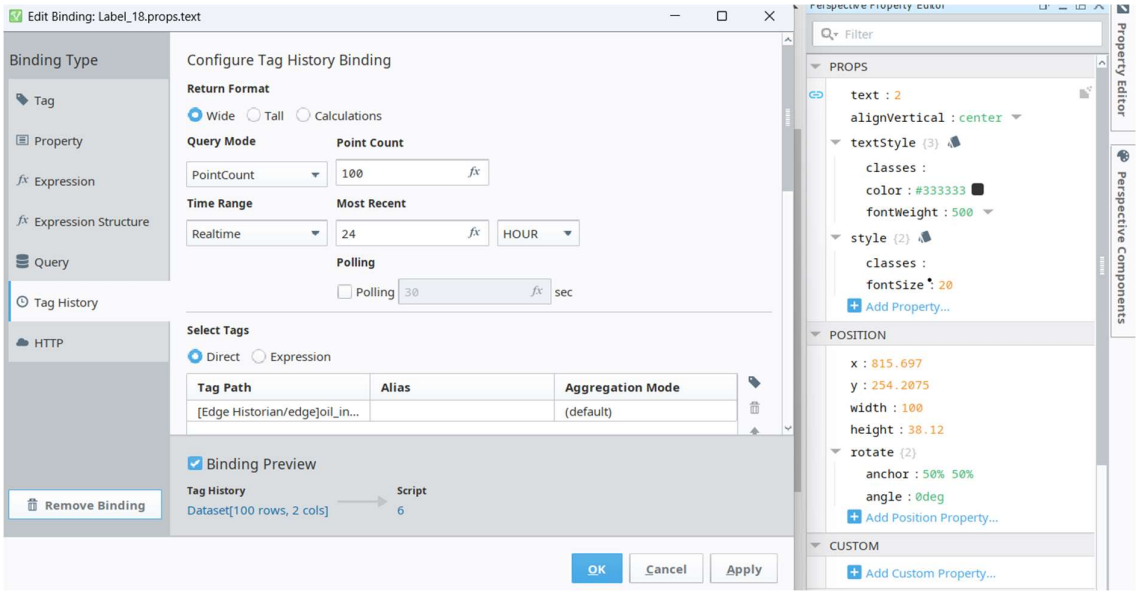
Note:

1. Add the temperature widget and give tag binding of the pump temperature.
2. All the properties of temperature widget is same as the tank widget.

- Four rectangular widgets (Pump Current, Pump Vibration, Alarms, Run time):

1. From perspective component select four image components and give the radius which is present in the border of style of props of the property editor as 150 for four corners and give width and height as same in both the position and in props, and in the props select the mode as fill.
2. Paste the urls in the source of the property editor of the four images respectively.
3. Select four label components for the values and for the Pump Current bind the tag binding of the current tag of the pump and add a transform of expression for the units of the current.
4. For the Alarm widget give the Tag History binding to allow the script as to count the alarms.

```
# Check if the dataset has at least 2 rows
 a,i=0,0
 while i<value.getRowCount():
  if (value.getValueAt(i,1)>=50 and value.getValueAt(i,1)<=60) or
(value.getValueAt(i,1)>=230 and value.getValueAt(i,1)<=250):
        a=a+1
  i=i+1
 return a
    # Compare the values
```



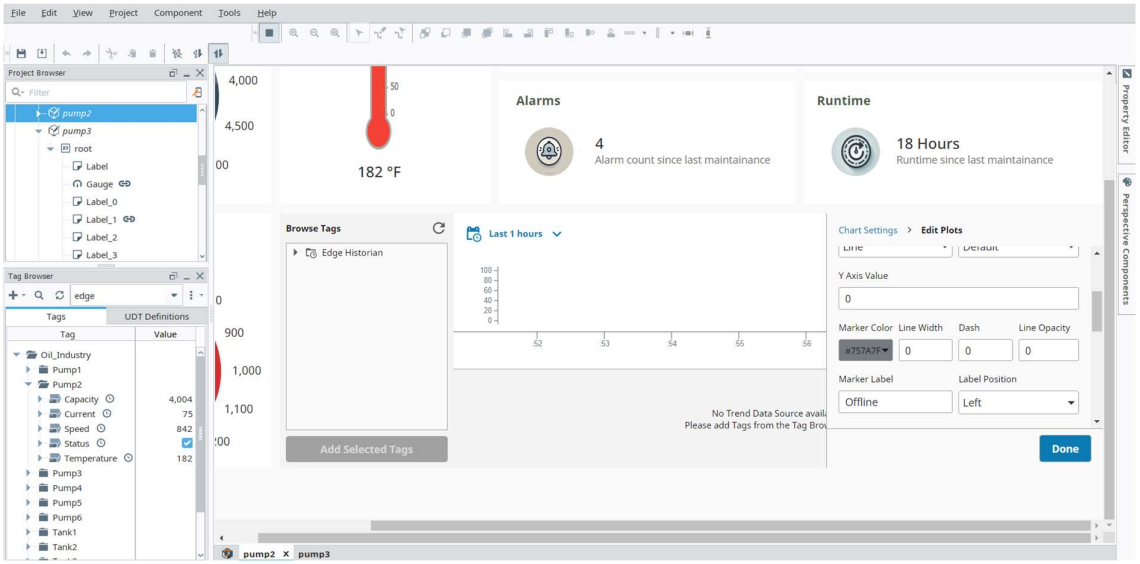
5. For Run time widget value select tag binding for the extraction run time tag and write an expression by adding transform for the units.
6. In Pump Vibration, in the property editor of the label select the tag binding of the value of Current tag or select the tag in the expression and add transform of script to get the status as,

```
if value>=0 and value<=(200*0.5):
    return "Good"
elif value>(200*0.5) and value<=(200*0.8):
    return "Moderate"
else:
    return "Severe"
```

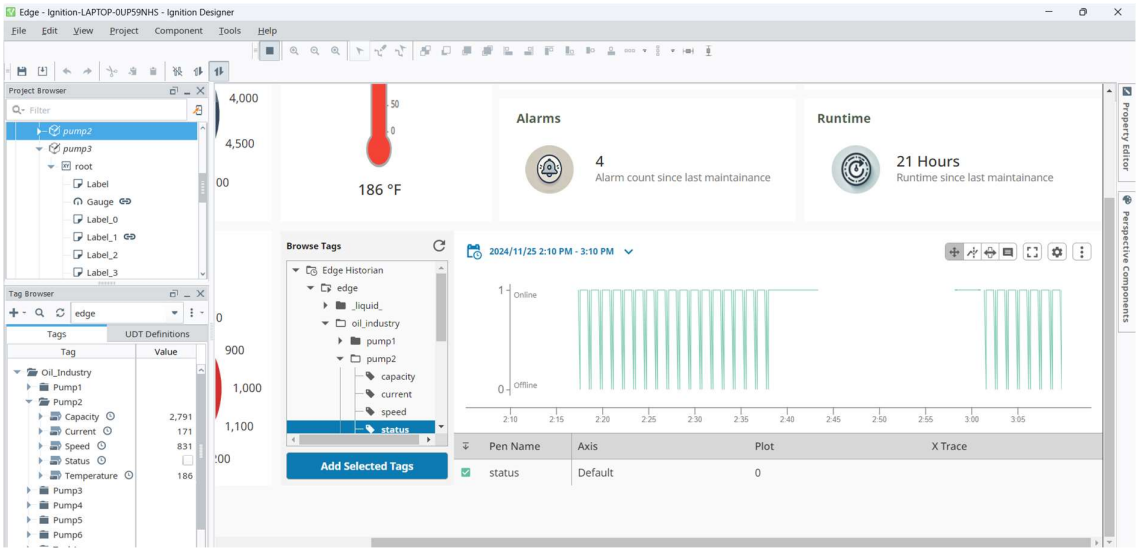
7. For the status symbol take a label component and give the border radius as 150 for all sides to get circular shape.
8. Select the property binding for the back ground color of that label and add transform of mapping the output type as color and input type as value.
9. Add the colours for “Good”, “Moderate”, “Severe”.

- Chart widget:

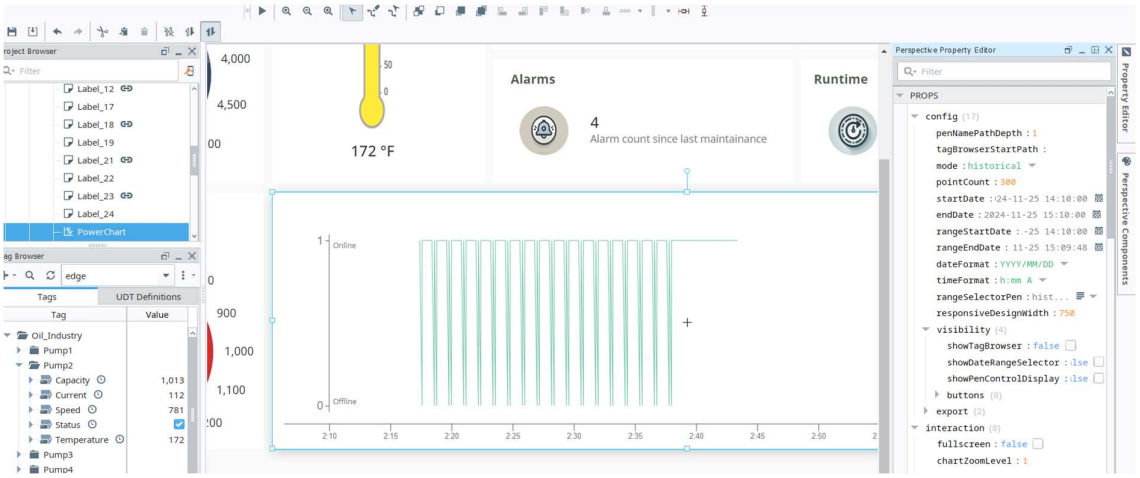
1. Select Power chart from the Components and add run the project by click play symbol on the top .
2. Click on the settings symbol on the power chart and add in the plot add the two markers as given below,



3. For the second marker Y Axis Value is 1 and Marker Label is “Offline”, remaining properties are same as the marker0.
4. Click on the tag browser and and add the status tag and select the time as historical and get select the from and to dates and time.



5. In the property under the configuration of props property editor under the visibility disable the ‘showtagbrowser’, ‘showDateRangeSelector’ and ‘showPenControlDisplay’.

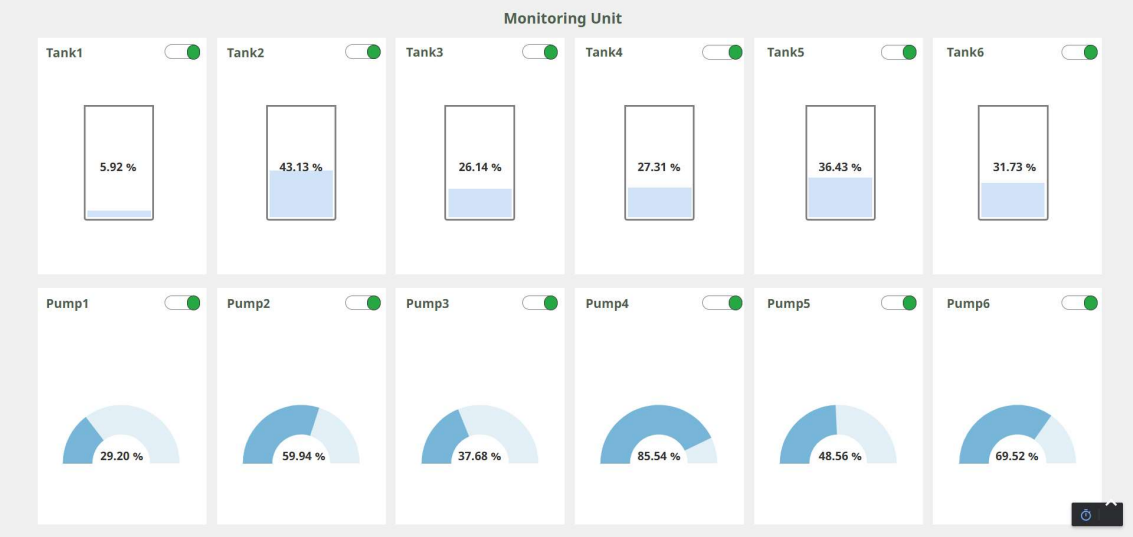


6. In property editor under the title enable visibility and in the gap of the title keep the label for the title.

Note:

1. The title fonts are same as for the tank visualization.
2. Click on the perspective present in the project browser.
3. Duplicate the pump and tank dashboards by creating views and copy the components in that particular design which are present in the project browser.

Water Monitoring Dashboard:



Create new view with ‘Water_Monitoring’ and select Coordinate Container and enable the URL.

- Online Status widget:

1. Select a label component and give the border in the style of the props of property editor as border width is 1 and take the width more as the rectangle.
2. Give the broder radius as 12 for four sides so that it looks as slide bar symbol.
3. Take another label component and make it circular as take same height and width in the position of property editor and give 120 for all the sides so that it look like a circle.
4. For the x position of the property editor of that circular label give the tag binding and add the script in the add transform.
5. Insert the circle in the slide bar and update the script as,

```
if value == True :
 return 253.347221375
```

```
    else:
        return 223.444442749
```

6. Where the under the True the right end side position of that circle on the slide bar and another is left side position.
7. In the slide bar, bind the background colour property which is present in the style of the property editor to the status of that particular tank or pump and give the mapping of the value input and output type is colour as for ‘True’ it is white and ‘False’ offline its is black.
8. In the circular widget also bind the background color property but after binding tag select the white if it is ‘False’ and green if it is ‘True’.

- Simple guage widget :

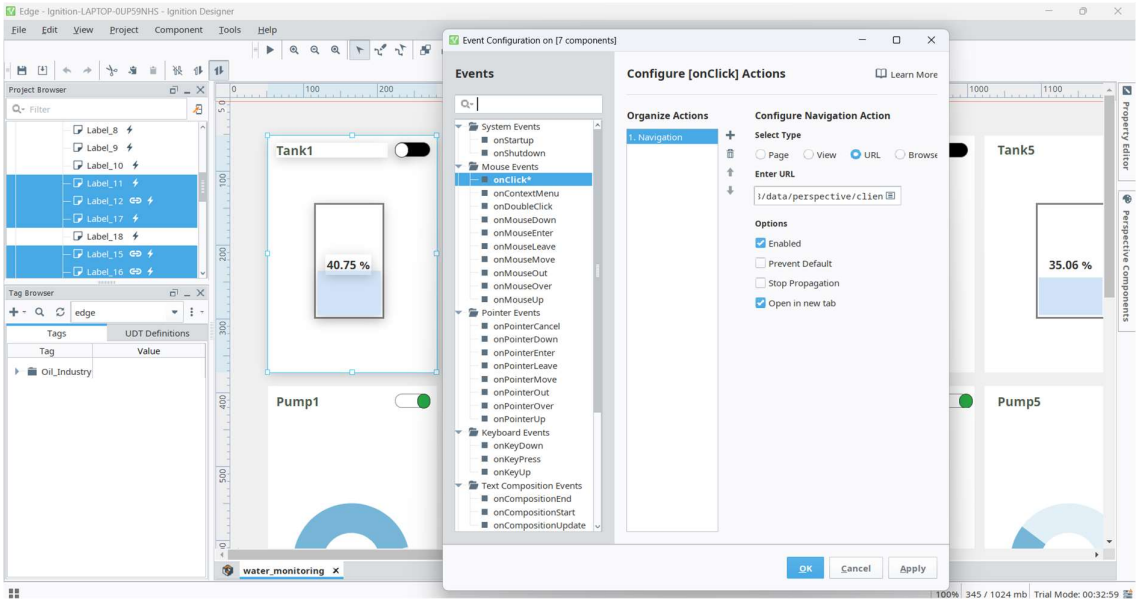
1. Select a simple guage component and give the max value in the props of the guage as 5000 and bind the value as tag binding of the capacity of specific pump.
2. Give the width of 40 under the arc under the props of the property editor.
3. Add a label and give the property binding of the value of the guage value and add the transform as expression for the percent, take total capacity as 5000.

Note:

1. For the tank widget, building it is same as the ‘Level Filled’ widget of the tank dashboard and the height is same for as the tank capacity widget of the tank dashboard.
2. Duplicate into more 5 for creating 12 panels of 6 tanks and 6 pumps including status and value widgets and replace the tags in the tag binding if it is required.

- Adding Events for the navigation of the particular dashboard :

1. In the project browser select all the components of a panel like ‘Tank1’ and in the designer right click on the selected widgets and select ‘Configure Events’.
2. Under the ‘Mouse Events’ click on the ‘On click’ and click on plus symbol and select ‘Navigation’.
3. Select URL enter the URL in of that specific tank or pump(can get in the browser by typing the host name and at the end with the name of the view as ‘http://localhost:8088/data/perspective/client/Edge/tank1’) and select enable and ‘open in new tab’.

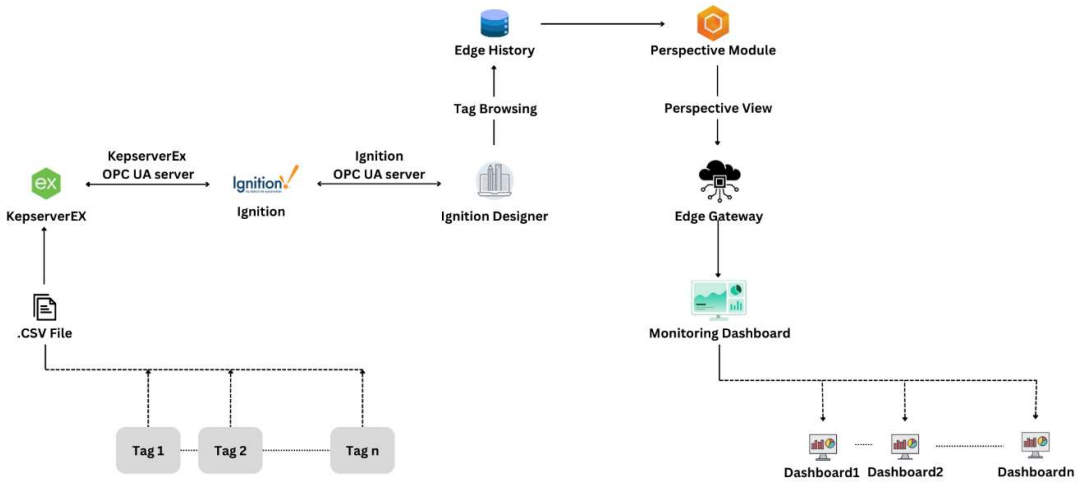


Note:

1. For all other panels give event configuration and select URL for that specific tank or pump.
2. Save all the dashboards and enter that specific URL in browser for visualizing the dashboard through the gateway.

WORKING:

- The whole set up works on the in-built OPC UA servers of KepserverEX and Ignition Edge.
- Data which is extracted in the KepserverEX is configured to Ignition Edge gateway.
- So, the data is transferred to the in-built Designer application which connected through the Host URL as the bowsing or extracting the tags takes place.
- These tags are enabled to send the data to the Database Edge Historian which are stored with the time.
- Hence in the visualization the data which is going to be representated are extracted from the change of the data from the database time-line.
- All the views that present in the project, which contains widgets that are mathematically binded tags saved as a single project and in that for every view, their edge URL is saved as to visualize the dashboard through that URL in the browser.



- The entire view is operated as a single dashboard by perspective Module in which tags of Historain database are worked according to the bind given to the widgets and the data which are saved to the Edge Historian changes due to the change in KepserverEX because of configuration or live connection from the KepserverEX OPC UA server.

CONCLUSION:

Creating dashboards using Ignition Edge is a robust and flexible process, enabling the visualization of real-time and historical data for various industrial applications. The Ignition Edge platform offers tools such as Perspective and Vision, allowing seamless integration of data from sources like Edge Historian into interactive and intuitive dashboards. By leveraging its modular design and scripting capabilities, users can develop custom dashboards tailored to specific operational requirements.

The process emphasizes the importance of understanding user needs, selecting relevant data sources, and using dynamic components (e.g., Power Charts, graphs, and widgets) for effective decision-making. Adopting best practices in data design ensures that dashboards are both aesthetically appealing and functional.

RESULT:

- The dashboards created in Ignition Edge improved operational oversight by presenting real-time metrics like tank capacity, pump efficiency, and process trends in an accessible format.
- Utilization of Ignition Edge's drag-and-drop interface and scripting features allowed for tailoring dashboards to meet specific user preferences (e.g., qualitative vibration statuses, temperature in Fahrenheit, capacity in gallons).
- By integrating tools like the Power Chart, the dashboards enabled operators to analyze historical trends and make informed decisions about process optimization.
- The final dashboards were designed to cater to various users, from operators to managers, providing concise yet comprehensive insights.
- The dashboards demonstrated scalability, enabling future expansion to include additional data points, equipment, or processes without significant rework.