|  |  |
| --- | --- |
| IC Reports framework  2025 | Abstract  This framework leverages React.js and Highcharts to create high-performance, interactive IC reports. It provides a modular, component-based approach for efficient report development and maintenance, ensuring seamless integration with existing systems and easy scalability for future requirements.  Vicky Thakur |

Contents

[1. Framework Introduction 2](#_Toc194580887)

[2. What It Covers 2](#_Toc194580888)

[3. Component-Based Approach 2](#_Toc194580889)

[4. Nothing Changes in IC Configuration and Reports Setup 10](#_Toc194580890)

[5. What It Takes to Onboard a New Report 10](#_Toc194580891)

[6. Ongoing Maintenance 13](#_Toc194580892)

[8. Skillset Required for Associates 14](#_Toc194580893)

[9. Pre-Requisites 14](#_Toc194580894)

**SOP for React.js and Highcharts Based IC Reports**

# 1. Framework Introduction

The new reporting framework is built on the powerful React.js framework and leverages Highcharts for data visualization. This approach ensures high performance, interactive reports, and seamless user experience. React.js is a JavaScript library for building user interfaces, while Highcharts is a charting library that provides a wide variety of interactive charts and graphs.

# 2. What It Covers

This SOP covers the implementation of the new reporting framework, including the configuration of new reports, how to use pre-existing components, make changes as applicable, integration with existing systems, and ongoing maintenance. It aims to provide clear guidelines for creating and managing IC reports using React.js and Highcharts. The document will guide you through the entire process, from initial setup to long-term maintenance.

# 3. Component-Based Approach

The framework follows component-based architecture, which allows for quick and efficient report creation. Each report is broken down into reusable React Components, making it easier to manage and update. This modular approach ensures that changes in one part of the report do not affect other parts, thus enhancing maintainability and scalability. The same components are applicable for SalesIQ web and UFE.

**Steps to Follow:**

1. **Identify Components**: Break down the report into smaller, reusable components.
2. **Develop Components**: Write the code for each component using React.js and Highcharts.
3. **Assemble Components**: Combine the components to form the complete report.
4. **Test Components**: Ensure each component works as expected individually and as part of the whole report.

**Components:**

1. **RHSPanel component –** 
   * **Purpose –** The RHSPanel component is used to present specific content related to the report's details. It accepts a content prop, which holds an array of objects, each containing a data key that represents the textual information to be displayed. This component can be reused across different parts of the report where similar content needs to be displayed. By utilizing the content prop, the RHSPanel component remains flexible and can adapt to various types of data without changing its structure.
   * **Configuration –** The RHSPanel component accepts the following parameter:
     + content (Array of Objects) **-** data (String): A textual representation of the content to be displayed on the RHSPanel. This is passed as an array of objects, each containing the data key.
   * **Sample Data -** Below is a sample of the RHSData array used in the RHSPanel component:

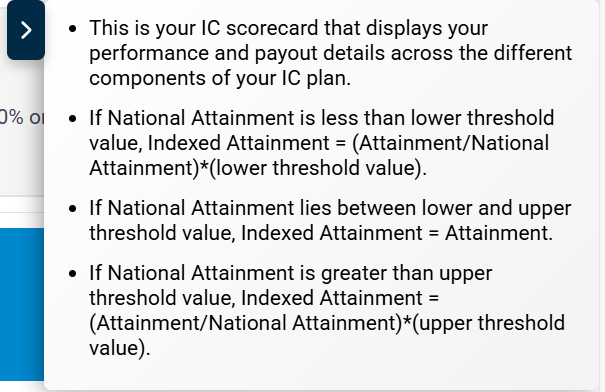
const RHSData = [

{ data: "This is your IC scorecard that displays your performance and payout details across the different components of your IC plan." },

{ data: "If National Attainment is less than lower threshold value, Indexed Attainment = (Attainment/National Attainment)\*(lower threshold value)." }

];

* + **Screenshot**

****

* + **How to use –  
    Import the Component**: import the RHSPanel component into your desired file.  
    import RHSPanel from './path/to/RHSPanel';  
    **Render the RHSPanel Component:** render the RHSPanel component and pass the RHSData as the content prop.  
    <RHSPanel content={RHSData} />

* **DataTable Component –** 
  + **Purpose -** This component is responsible for rendering tables with various interactive features, allowing users to view and manage data efficiently. The DataTable component is highly customizable and supports multiple features such as search, sorting, filtering, pagination, and data download, making it versatile for displaying a variety of data in a structured format.
  + **Configuration -** The DataTable component accepts the following parameters:
    - **columns -** This prop holds the structure of the table, typically an array of objects representing the column headers, their field names.
    - **data -** This is the dataset that will be displayed in the table. It’s typically an array of objects where each object represents a row in the table.
    - **isSearch -** When set to true, a search input field will be rendered above the table, allowing users to filter the data dynamically.
    - **isDownload -** If true, a button will be shown to allow users to download the table data.
    - **isFilter -** If true, popup will open where you can apply multiple filter conditions to filter the data.
    - **isSort -** Enabling sorting allows users to click on column headers to sort the data in an ascending or descending order.
    - **isPagination -** When set to true, pagination controls will be shown to allow users to navigate through large datasets, ensuring that only a subset of the data is displayed per page.
    - **showRecordDropdown -** If true, a dropdown menu will be shown for users to select how many records to display per page.
    - **isFixedHeader -** When enabled, the table header will remain fixed at the top of the page as users scroll down through the data.
    - **defaultPageSize -** Specifies the number of rows to display by default on each page of the table.
    - **isColumnShowHide -** The pop up will open which allows users to show or hide specific columns in the table.
    - **csvDownload -** This controls whether the table data can be downloaded in CSV format.
    - **fileName -** The file name used when downloading the data.
  + **Sample Data -** Here’s an example of how the props for the DataTable component might look in practice:  
      
    teamSummaryTableColumns = [

        { name: 'Team', type: 'text', fieldName: 'TEAM' },

        { name: 'Level', type: 'text', fieldName: 'geoName'},

        { name: 'Role', type: 'text', fieldName: 'ROLE'},  
];  
  
filteredData = [

        { TEAM: 'IPF', geoName: 'Geo1', ROLE: 'SC' },

        { TEAM: 'IPF', geoName: 'Geo2', ROLE: 'AD' },

        { TEAM: 'IPF', geoName: 'Geo3', ROLE: 'NSD' },  
];

<DataTable

columns={teamSummaryTableColumns}

data={filteredData}

isSearch=true/false

isDownload=true/false

isFilter=true/false

isSort=true/false

isPagination=true/false

showRecordDropdown=true/false

isFixedHeader=true/false

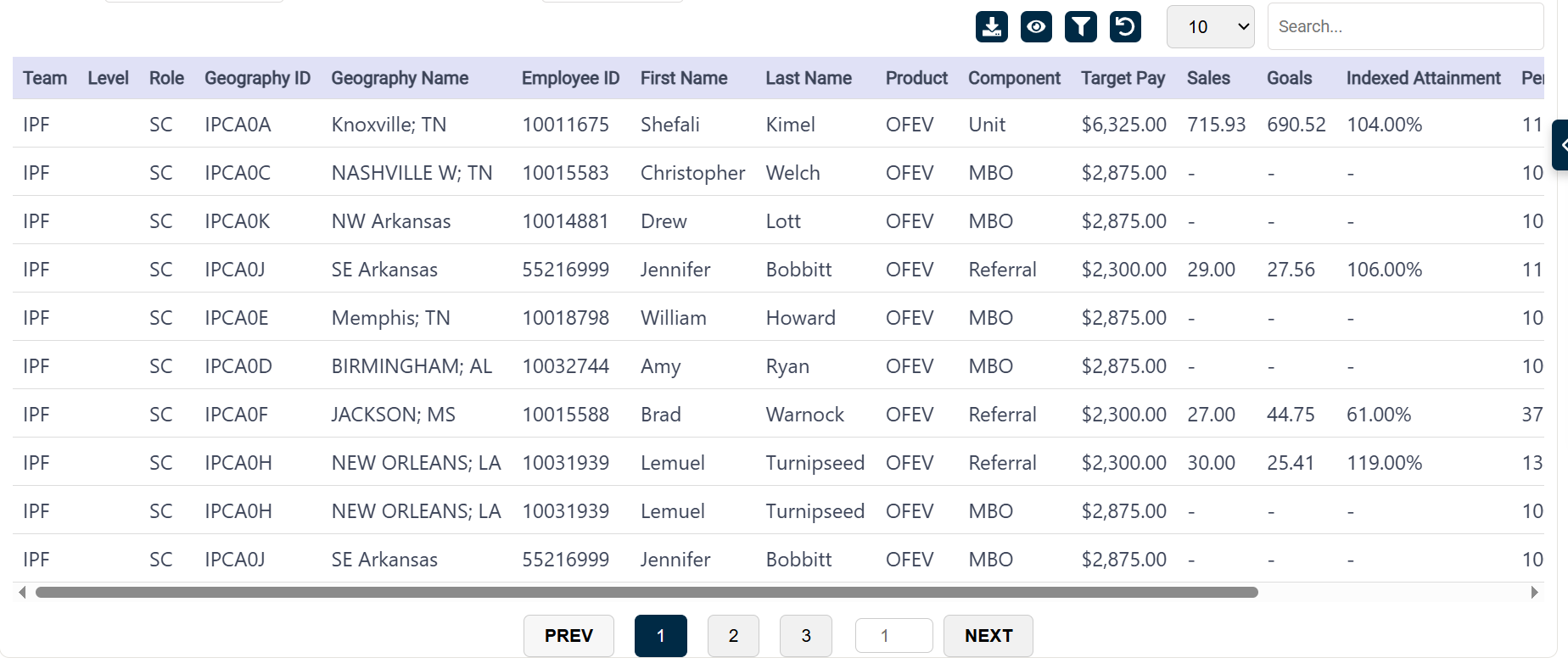
defaultPageSize=10

isColumnShowHide=true/false

csvDownload=true/false

fileName=DownloadFileName

/>

* + **Screenshot –   
    **
  + **How to use –   
    Import the Component:** Import the DataTable component into your file where you want to use it.  
    import DataTable from './path/to/DataTable';

**Render the DataTable Component:** render the DataTable component and pass the values to the parameters.  
<DataTable

columns={teamSummaryTableColumns}

              data={filteredData}

              isSearch={true}

              isDownload={true}

              isFilter={true}

              isSort={true}

              isPagination={true}

              maxHeight={true}

              showRecordDropdown={false}

              isFixedHeader={false}

               defaultPageSize={10}

               isColumnShowHide={true}

               csvDownload={false}

               fileName={`Performance\_Summary(${geoId})`}

/>

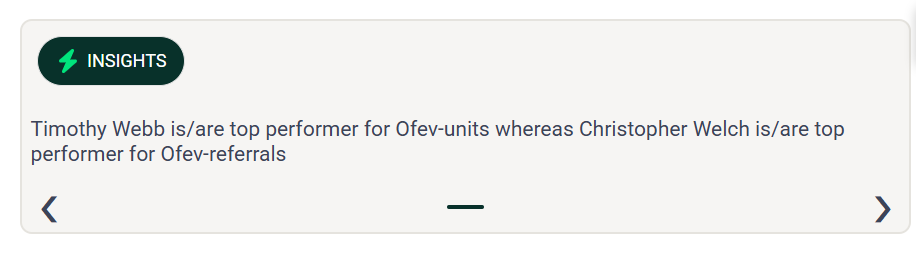
* **Insights Component –** 
  + **Purpose -** Insights component provides a dynamic and interactive way to display key insights related to the report. It supports a carousel layout, allowing users to cycle through different insights with navigation controls.
  + **Configuration -** An array of objects, each containing a text property that holds the insight content.
  + **Sample Data -**   
    const insightData = [

{ text: "This is a valuable insight into the performance of your team over the last quarter." },

{ text: "Consider improving your strategies based on the declining trends in the current quarter." },

{ text: "Here is a prediction of the future trends based on historical data and performance metrics." }

];

* + **Screenshot -   
    **
  + **How to use –**

**Import and Use the Insights Component:** Import the Insights component into your parent component and pass the insights data as a prop.  
import Insights from './path/to/Insights';  
<Insights insights={insightData} />

* **Chart Component –** 
  + **Purpose -** The Chart component is designed to display various charts using Highcharts, which is a powerful charting library. The component allows users to visualize data in an interactive format. It takes in custom configuration options as a prop to generate the chart, providing flexibility in visual representation for various datasets.
  + **Configuration -** The Chart component accepts a single prop:
    - **Options -** An object containing the configuration for the Highcharts instance. This configuration includes various properties like chart type, title, x and y axes settings, series data, and plot options.
    - Key Configuration Fields in options:
      * **chart** - Defines the chart type and properties like height.  
        Example: { type: 'column', height: 195 }
      * **title**: Specifies the chart title and its style.  
        Example: { text: '', style: { fontSize: '14px', fontWeight: 'bold' } }
      * **xAxis**: Configuration for the x-axis, such as categories and title.  
        Example: { categories: ['Territory 1', 'Territory 2'], title: { text: 'Territory count' } }
      * **yAxis**: Configuration for the y-axis, including minimum value, title, and label settings.

Example: { min: 0, title: { text: 'Earnings Count' } }

* + - * **series**: An array of series data to be plotted on the chart.  
        Example: [ { name: 'Earnings', data: [10, 20, 30], color: '#928BDE' } ]
      * **plotOptions**: Additional settings for customizing the appearance of the chart's columns, such as data labels, point width, etc.  
        Example: { column: { dataLabels: { enabled: true, style: { fontWeight: 'normal' } }, pointWidth: 35 } }
      * **legend**: Configuration for the chart legend.  
        Example: { symbolRadius: 0, layout: 'vertical', align: 'right', verticalAlign: 'top' }
* **Sample Data -**const teamSummaryChartOptions = {

chart: { type: 'column', height: 195 },

title: { text: '', style: { fontSize: '14px', fontWeight: 'bold' } },

xAxis: {

categories: teamSummaryBarChartValue.xAxis,

title: { text: 'Territory count' },

lineColor: '#EDEFF2',

lineWidth: 1,

},

yAxis: {

min: 0,

title: { text: 'Earnings Count' },

labels: { enabled: true },

gridLineWidth: 0,

},

series: [

{ name: 'Earnings', data: teamSummaryBarChartValue.bar1, color: '#928BDE', type: 'column' },

],

plotOptions: {

column: {

dataLabels: {

enabled: true,

formatter: function () { return this.y + ''; },

style: { fontWeight: 'normal', color: '#000000' },

},

pointWidth: 35,

},

},

legend: {

symbolRadius: 0,

layout: 'vertical',

align: 'right',

verticalAlign: 'top',

floating: true,

y: -10,

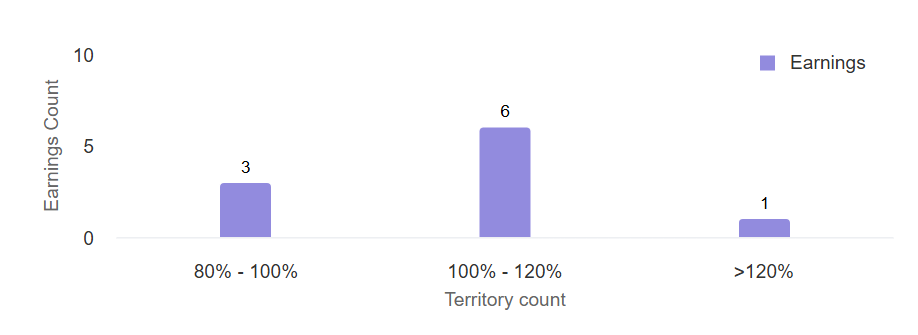
x: -10,

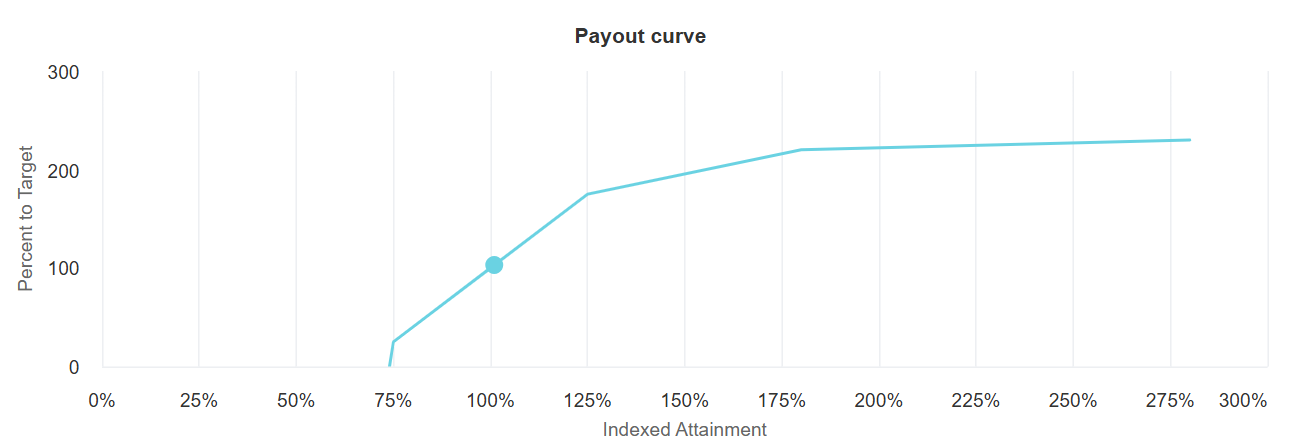
symbolWidth: 20,

symbolHeight: 10,

},

};

* **Screenshot -   
  **

****

* **How to use -   
  Prepare Chart Data:** Create the configuration object for the chart, including data for the x-axis, y-axis, series, etc.  
  **Import and Use the Chart Component**: Import the Chart component into your parent component and pass the configuration object.  
  import Chart from './Chart';  
  <Chart options={teamSummaryChartOptions} />

# 4. Nothing Changes in IC Configuration and Reports Setup

The new framework integrates seamlessly with the existing IC configuration and report setup. There is no need for additional configuration or changes to the current setup of ‘Report Setup’ in SalesIQ.

This ensures that the transition to the new framework is smooth and does not disrupt existing operations.

# 5. What It Takes to Onboard a New Report

To begin a new report, follow these steps:

1. **Identify Data Sources**: Determine the data sources and requirements for the new report
2. **Develop Report Components**: Reuse the existing components and make the necessary tweaks.
3. **Test the Report**: Ensure the report meets performance and accuracy standards.
4. **Deploy the Report**: Test on the Sandbox and deploy to production after QA signoff.

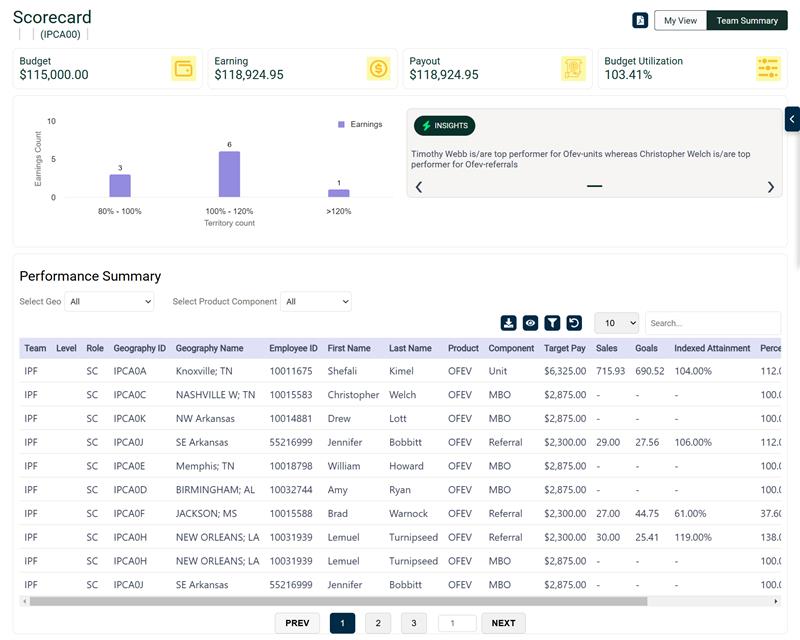
**Reports implemented in Current Implementations**

**BI-US Pilot**: A report for the Boehringer Ingelheim engagement team in the US

1. **Scorecard**

A screenshot of a computer

AI-generated content may be incorrect.



1. **Weekly Performance Report**

A screenshot of a data report

AI-generated content may be incorrect.

* **Ferring SIQ Implementation**: A report for the Ferring SIQ project.

# 6. Ongoing Maintenance

If there are changes to the plan, such as new requirements or updates to the existing reports, follow these steps:

1. **Change in IC Report Setup**: These changes will be usual, as in the current process
2. **Assess Impact**: Evaluate the impact of the changes on the ‘Reporting layer’
3. **Update Components**: Modify the relevant components and update the data wiring
4. **Test Updates**: Ensure the updated reports meet the new requirements

# 8. Skillset Required for Associates

Associates working on these reports must have the following skills:

* **Knowledge of IC Reports**: Understanding of the current IC reports framework and configuration.
* **React.js Skills**: Proficiency in using React.js for developing user interfaces.
* **Highcharts Skills**: Experience with Highcharts for creating interactive charts and graphs.
* **Problem-Solving**: Ability to troubleshoot and resolve issues that may arise during report development and maintenance.
* **Communication**: Strong communication skills to effectively collaborate with team members and stakeholders.

# 9. Pre-Requisites

Before starting the development of new reports, ensure the following prerequisites are in place:

1. **iPad** (for testing)
2. **Salesforce Org Details**
3. **Server Details**
4. **IIS Server Enabled**
5. **Report Data Published**
6. **Domain to Host Python API & React App**
7. **Node.js Installed on Server**
8. **SMFT Credentials**
9. **Snowflake Credentials**