IC REPORTS FRAMEWORK

Version 1.0
Last Updated:

1 CONTENTS

2 DOCUMENT DETAILS

Document Title	IC Reports Framework
Date of Release	
Version No.	
Document Owner	
Document Author	
Document Approver	

3 VERSION CONTROL

Version No	Release Date	Revision Notes	Date Approved	Approved By
1.0		First draft for stakeholders		
2.0				

4 DOCUMENT APPROVERS

From organization:

Approver	Approver Name	Date Approved	Signature
1	Sahil Mahajan	24-04-2024	
2			

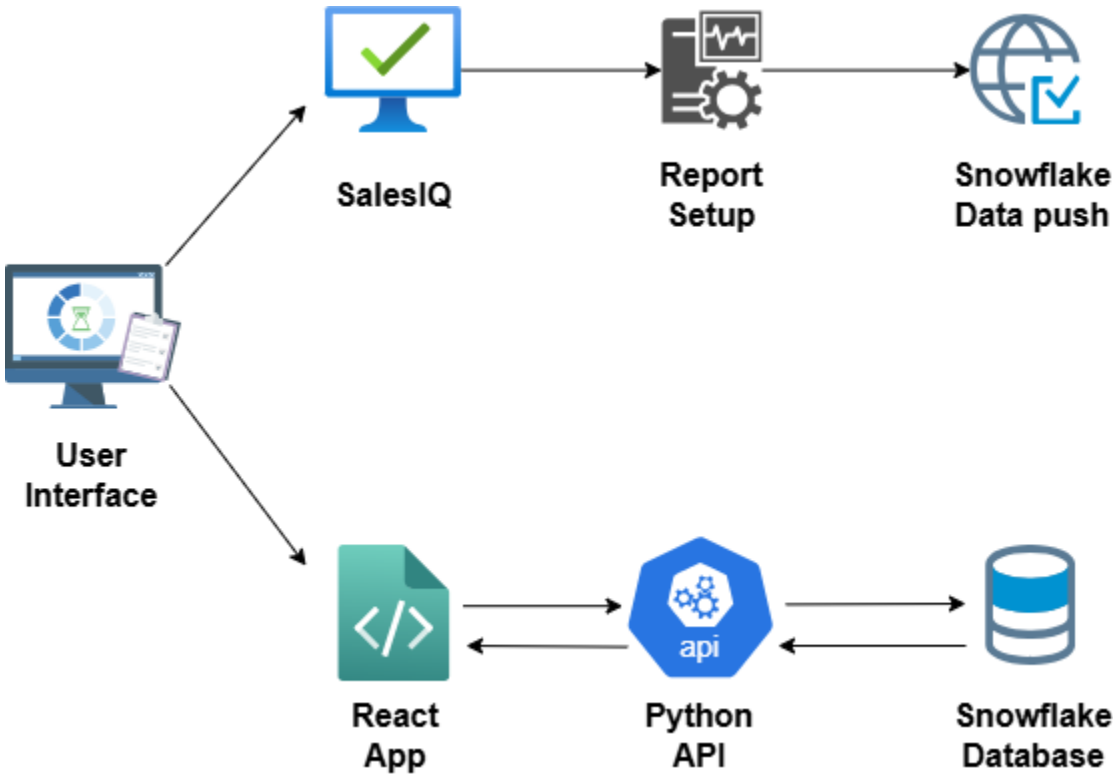
5 FRAMEWORK INTRODUCTION

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.

6 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.

7 ARCHITECTURE



8 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.

8.1 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.

8.2 Goalcard Report – Example use case

The goalcard report provides an overview of an individual's or team's performance against predefined goals, showcasing key metrics, historical sales data and insights. Below, we break down the structure of the Goalcard Report and how it can be implemented using reusable React components.

1. **Goalcard Report Header** - The **HeaderSection1** component is used to show report name, report period and additional information about the user along the client logo.
2. **Goalcard Report Details -** The **RHSPanel** component is used to show additional information about the Goalcard.
3. **Historical Sales Chart -** The **Chart** component is used to display historical sales data over time. A line chart or column chart can be used to represent the sales trends
4. **Insights and Recommendations -** The **Insights** component is used to provide additional commentary or recommendations based on the report's data. This section helps guide the user with actionable insights derived from sales performance.

8.3 Components Library

Components Library

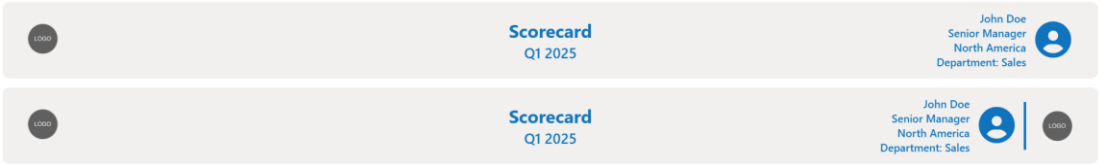
- **HeaderSection1 Component –**
  - **Purpose –** The HeaderSection1 component is designed to display the header section of a report or dashboard. It showcases essential report metadata such as the report title, reporting period, user information, and logos (company or brand). The component is highly configurable, enabling or disabling elements like the right-side logo based on props.
  - **Configuration –** The HeaderSection1 component accepts the following parameter:
    - **title (String)**: The name or title of the report.
    - **reportPeriod (String)**: A string representing the reporting period.
    - **leftLogo (Image Path)**: Path or import reference to the image/logo displayed on the left side of the header.
    - **rightLogo (Image Path)**: Path or import reference to the image/logo displayed on the right side of the header.
    - **userDetails (Array of Strings)**: An array of strings displaying user-specific information such as name, role, region, or department.
    - **showRightLogo (Boolean)**: A flag to control the visibility of the right-side logo. If set to `true`, the right logo is displayed; if `false`, it's hidden.

  - **Sample Data -** Below is a sample of the array used in the HeaderSection1 component:

```
import leftLogo from '../Images/logo-placeholder-image.png';
import rightLogo from '../Images/ logo-placeholder-image.png';

const reportName = 'Scorecard';
const reportPeriod = 'Q1 2025';
const userDetails = [
   "John Doe",
   "Senior Manager",
   "North America",
   "Department: Sales"
];
```

  - **Screenshot**



- **How to use –**
  Import the Component – First, import the HeaderSection1 component and any required assets (like logos) into your desired file:

  ```
  import HeaderSection1 from './path/to/HeaderSection1';
  import leftLogo from '../Images/ logo-placeholder-image.png';
  import rightLogo from '../Images/ logo-placeholder-image.png';
  ```

  Render the HeaderSection1 Component – Render the component.

  ```
  <HeaderSection1
      title={reportName}
      reportPeriod={reportPeriod}
      leftLogo={leftLogo}
      rightLogo={rightLogo}
      userDetails={userDetails}
      showRightLogo={false}
    />
  ```

- **HeaderSection2 Component –**
  - **Purpose –** The HeaderSection2 component is designed to display the header section of a report. On the left side it showcases essential report metadata such as the report title, reporting period, user information. It allows the user to switch between different views such as "My View", "Team Summary", and "Executive Overview". The component is highly configurable with options for customizing button labels, handling user interactions, and displaying user information.
  - **Configuration –** The HeaderSection2 component accepts the following parameter:
    - **title (String)**: The name or title of the report.
    - **userDetails (Array of Strings)**: An array of strings displaying user-specific information such as name, role, region, or department.
    - **buttonLabels (Array of Strings**): An array of button labels that define the views available to the user (e.g., ["My View", "Team Summary", "Executive Overview"]).
    - **onButtonClick (Function)**: A callback function triggered when a button is clicked. This function should update the currentView state to reflect the selected view.
    - **defaultView (String)**: The default view selected when the component first renders (e.g., "My View").

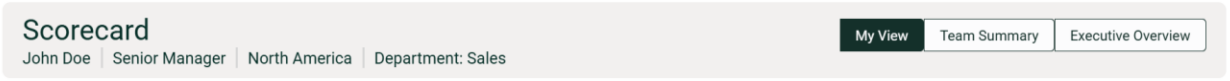
  - **Sample Data -** Below is a sample of the array used in the HeaderSection2 component:

    ```
    const userDetails = [
      "John Doe",
      "Senior Manager",
      "North America",
      "Department: Sales"
    ];
    ```

```
const buttonLabels = ["My View", "Team Summary", "Executive Overview"];
const defaultView = buttonLabels[0]; // or choose from props, config, etc.

const [currentView, setCurrentView] = useState(defaultView);
```

- **Screenshot**



- **How to use –**
  Import the Component – First, import the HeaderSection2 component and any required assets into your desired file:

```
import HeaderSection2 from './path/to/HeaderSection2';
```

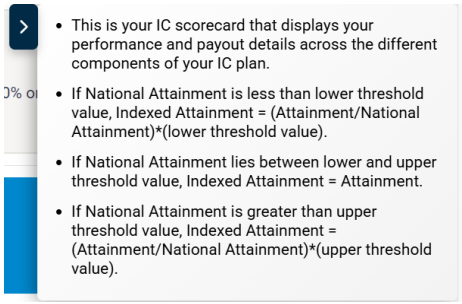
  Render the HeaderSection2 Component – Render the component.

```
<HeaderSection2
    title="Scorecard"
    userDetails={userDetails}
    buttonLabels={buttonLabels}
    onButtonClick={setCurrentView}
    defaultView={defaultView}
  />
```

- **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**

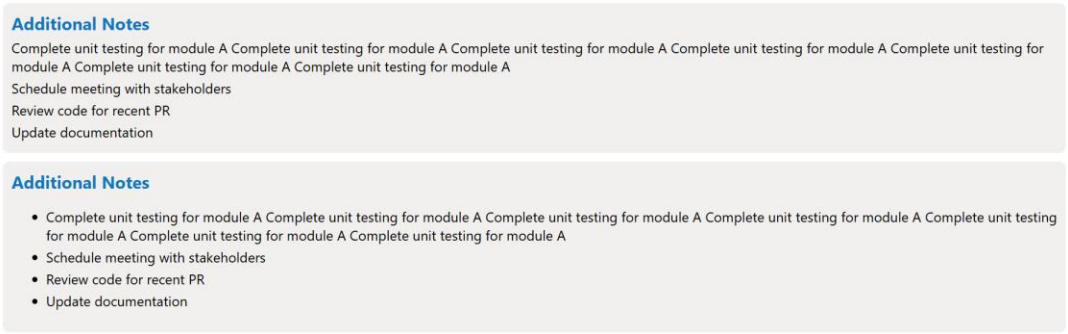


- **How to use –**
  Import the Component: import the RHSPanel component into your desired file.
  import RHSPanel from './path/to/RHSPanel';
  Render the RHSPanel Component: render the RHSPanel component and pass the RHSData as the content prop.
  <RHSPanel content={RHSData} />

- **AdditionalNotes Component –**
  - **Purpose –** The AdditionalNotes component is used to display a list of additional notes. It is highly configurable and can handle different sets of notes while offering options for displaying them as bullet points or as plain text. The component takes in several props, such as the notes, a title for the section, an empty message to display when no notes are available, and a flag to toggle whether to show the notes as bullet points.
  - **Configuration –** The AdditionalNotes component accepts the following parameters:
    - **notes (Array of Strings)**: An array of strings, each representing a separate note. These are the main contents displayed by the component.
    - **notesHeader (String)**: A string that serves as the header/title of the notes section.
    - **emptyMessage (String)**: A string to display when no notes are available or when the notes array is empty or null.
    - **useBullets (Boolean)**: A flag that determines whether the notes are displayed as bullet points. By default, this is set to true. If set to false, the notes will be displayed as plain text.
  - **Sample Data -** Below is a sample of how to configure the `AdditionalNotes` component with data:

```
const additionalNotesArray = [
  "Complete unit testing for module A Complete unit testing for module A Complete ",
  "Schedule meeting with stakeholders",
  "Review code for recent PR",
  "Update documentation"
];
const notesHeaderTitle = "Additional Notes"
const emptyNotesMessgae = "No Notes Available"
```

- **Screenshot**



- **How to use –**
  Import the Component: import the AdditionalNotes component into your desired file.
  import AdditionalNotes from './path/to/AdditionalNotes';
  Render the RHSPanel Component: render the AdditionalNotes component and pass the necessary data.

```
<AdditionalNotes
        notes={additionalNotesArray}
        notesHeader={notesHeaderTitle}
        emptyMessage={emptyNotesMessgae}
        useBullets={false}
/>
```

- **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:

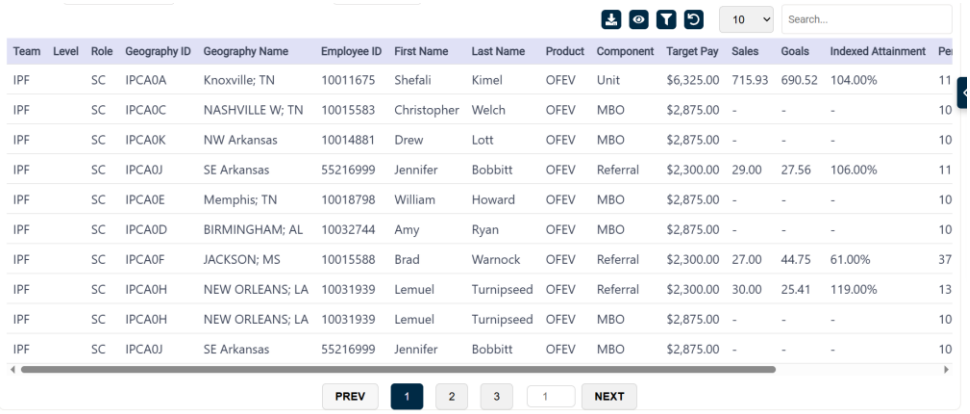
```
teamSummaryTableColumns = [
    { name: 'Team', type: 'text', fieldName: 'TEAM' },
    { name: 'Level', type: 'text', fieldName: 'geoName'},
    { name: 'Role', type: 'text', fieldName: 'ROLE'},
];

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 –**



Team	Level	Role	Geography ID	Geography Name	Employee ID	First Name	Last Name	Product	Component	Target Pay	Sales	Goals	Indexed Attainment	Pe
IPF		SC	IPCA0A	Knoxville; TN	10011675	Shefali	Kimel	OFEV	Unit	$6,325.00	715.93	690.52	104.00%	11
IPF		SC	IPCA0C	NASHVILLE W; TN	10015583	Christopher	Welch	OFEV	MBO	$2,875.00	-	-	-	10
IPF		SC	IPCA0K	NW Arkansas	10014881	Drew	Lott	OFEV	MBO	$2,875.00	-	-	-	10
IPF		SC	IPCA0J	SE Arkansas	55216999	Jennifer	Bobbitt	OFEV	Referral	$2,300.00	29.00	27.56	106.00%	11
IPF		SC	IPCA0E	Memphis; TN	10018798	William	Howard	OFEV	MBO	$2,875.00	-	-	-	10
IPF		SC	IPCA0D	BIRMINGHAM; AL	10032744	Amy	Ryan	OFEV	MBO	$2,875.00	-	-	-	10
IPF		SC	IPCA0F	JACKSON; MS	10015588	Brad	Warnock	OFEV	Referral	$2,300.00	27.00	44.75	61.00%	37
IPF		SC	IPCA0H	NEW ORLEANS; LA	10031939	Lemuel	Turnipseed	OFEV	Referral	$2,300.00	30.00	25.41	119.00%	13
IPF		SC	IPCA0H	NEW ORLEANS; LA	10031939	Lemuel	Turnipseed	OFEV	MBO	$2,875.00	-	-	-	10
IPF		SC	IPCA0J	SE Arkansas	55216999	Jennifer	Bobbitt	OFEV	MBO	$2,875.00	-	-	-	10

- **How to use –**

  Import the Component: Import the DataTable component into your file where you want to use it.

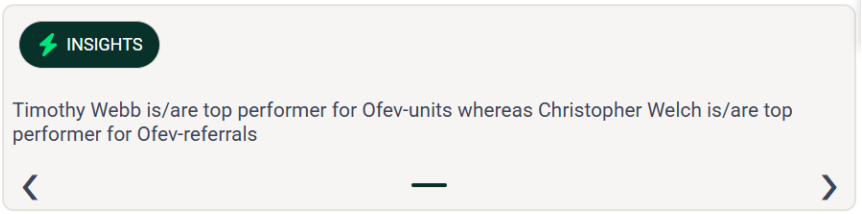
  import DataTable from './path/to/DataTable';

  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. Different types of charts are supported like Bar chart, Line chart, Pie chart, Column chart.
    - 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' } }

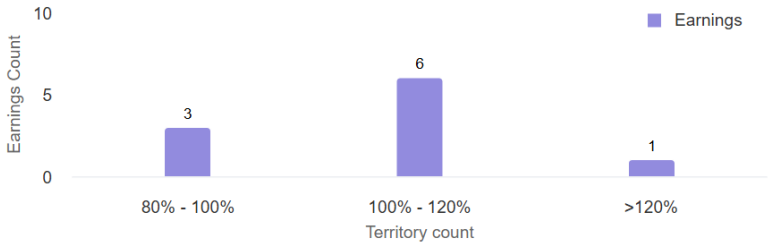
- **yAxis**: Configuration for the y-axis, including minimum value, title, and label settings.
  Example: { min: 0, title: { text: 'Earnings Count' } }
- **series**: An array of series data to be plotted on the chart.
  Example: [{ name: 'Earnings', data: [10, 20, 30], color: '#928BDE' }]
- **plotOptions**: 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' }

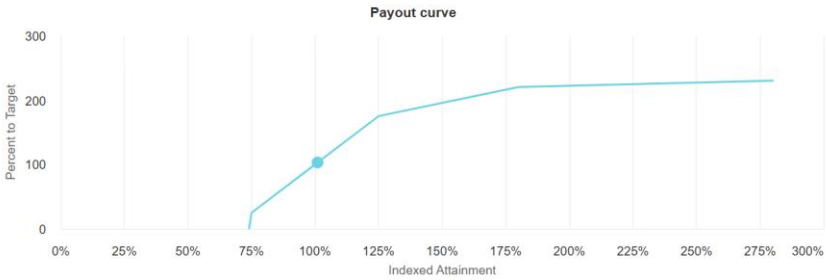
- **Sample Data -**

```
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} />

9 CHANGES IN EXISTING IC PROCESSING

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.

10 ROLLOUT STRATEGY

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**: Create the report components using React.js and Highcharts.
3. **Integrate with IC Configuration**: Integrate the new report with the existing IC configuration.
4. **Test the Report**: Ensure the report meets performance and accuracy standards.
5. **Deploy the Report**: Make the report available to users.

Examples of Current Implementations:

- **BI-US Pilot**: A pilot report for the Business Intelligence team in the US. The following React reports have been developed:
    - Weekly Performance Report
    - Scorecard
    - Goalcard
    - Rank Report
    - HCO HCP Report
    - What If Calculator

- **Ferring SIQ Implementation**: A report for the Ferring SIQ project. The following React reports have been developed:
    - Scorecard
    - Goalcard

11 WHAT IF MY PLAN CHANGES

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

1. **Assess Impact**: Evaluate the impact of the changes on the current framework.
2. **Update Components**: Modify the relevant components and configurations.
3. **Test Updates**: Ensure the updated reports meet the new requirements.
4. **Communication**: Inform all stakeholders about the changes and their implications.

12 ONGOING MAINTAINANCE

Ongoing maintenance involves regular updates to the framework and reports to ensure they remain efficient and accurate. This includes:

1. **Monitor Performance**: Regularly check the performance of the reports.
2. **Address Issues**: Fix any issues or bugs that arise.
3. **Update Framework**: Keep the framework and components up to date with the latest versions.
4. **Provide Training**: Offer training and support to users to help them understand and use the reports effectively.

13 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.
- **Snowflake Knowledge**: Familiarity with Snowflake for data querying and management.
- **Python for APIs**: Experience using Python to develop and integrate APIs.
- **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.

14 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**

15 BENEFITS OF USING REACT REPORTS

- **High Performance** - React Reports deliver fast and responsive performance, even with large datasets. Reports are getting rendered within 5 seconds.
- **Cost-Effective** - React Reports can be up to 10 times less expensive compared to Power BI, especially for large-scale implementations. There are no per-user licensing fees, making it a scalable solution for growing teams.
- **Look and Feel** - You can design React reports to look exactly how you want with clean and modern appearance.
- **Easy to Customize and Reuse** - React lets you build reusable pieces (called components), so it's easy to make changes and use the same parts in different places. This saves time and effort.

16 EFFORTS ESTIMATION AND COST

- **Implementation Effort (when all Reports are in Legacy Setup):**

  These React report efforts are estimated considering that the initial setup and report data have already been pushed before the start of the task.

Task	Effort
Initial Setup (IT Tasks)	5

Type (Per Report Effort)	Effort
React Report with DTR report setup (as per the components present in Library)	5
React Report with Non DTR report setup (as per the components present in Library)	6
React Report with small changes	7
React Report with medium changes	10
React Report with complex changes	On Demand

Below are examples of changes along with their corresponding categories.

Changes	Category
Font and color changes	Small
New chart configuration (may require time for POC)	Medium
Calculation logic change (IC Calculator)	Complex

- **Implementation Effort (when Reports are migrated from Golden Image with no Changes): -**

Task	Effort
Initial Setup (IT Tasks)	5

Type (Pack of 5 Report)	Effort
React Report with DTR report setup (as per the components present in Library)	6
React Report with Non DTR report setup (as per the components present in Library)	6

Assumptions: -

1. Database should be Snowflake and there is no change in table structure.

- **Maintenance Effort:**
  Any changes or enhancements required in the reports, including bug fixes, support requests, or quarterly updates, will be managed and implemented by the React development team. This includes front-end adjustments, data-binding issues, and UI/UX improvements as needed.

- **Cost Implications (Licenses, Infrastructure, Resourcing):**
  The project requires two dedicated resources – one for front-end development (React) and one for back-end/API development. Additionally, infrastructure access is essential, including:
  - Server access
  - Snowflake database access for data integration.
  - IIS Admin access for hosting both the APIs and the React application

17 KNOWN PROBLEMS AND CHALLENGES

- **Technical Limitations:**
  - Map-based visualizations are not currently supported in reports. POC is pending.
  - Reports support only the English language currently.
  - Print view functionality for reports is yet to be explored through a POC.
  - Download functionality for reports on iPad devices is currently unsupported

- **UI/UX Limitations:**

  No UI/UX limitations have been identified so far.

- **Performance or Data latency concerns:**
  Reports typically load within 5 seconds. In case any latency is observed, it is recommended to consider upgrading the Snowflake to ensure optimal performance.

18 COMPARE AND CONTRAST (VETTED BY PBI TEAM)

Feature	React (With Components)	Power BI Embedded
Export to PDF/Excel	✅ The **Datatable** component includes options to download data in both Excel and CSV formats.	✅ Native
Print Views	❌ Currently not supported. POC needs to be done	✅ Available out-of-box
Custom Format Downloads	✅ We've added advanced filter functionality to the **Datatable** component, allowing users to download the filtered data.	⚠️ Limited to standard formats
Chart Variety	✅ We have a **Chart** component that supports rendering various chart types using the Highcharts library.	✅ Wide but predefined
Custom Design	✅ Full freedom (branding, interactions)	⚠️ Customization limited to themes
Dynamic/Interactive UIs	✅ 100% possible	✅ Built-in for standard interactions
Hover Details	✅ Fully customizable	✅ Native tooltips
Drill-down Navigation	✅ You control logic (e.g., via router/state)	✅ Built-in
Cross-chart Filtering	✅ If implemented across components	✅ Ready-made
Row-Level Security	✅ Possible via backend logic	✅ Native RLS support
User-Specific Filters	✅ We're receiving user-specific filtered data in React from Snowflake.	✅ Supported
Whitelisting/IP Controls	✅ We can handle it in salesforce	✅ Controlled via Azure
Custom UX & Layout	✅ Pixel-perfect freedom	⚠️ Limited to Power BI visual framework
Dev Time	🚧 Relatively high	🧩 Low, mostly configuration
Reusability	✅ We're using a component-based structure, so each component is reusable. That makes it easy to make changes and use components wherever we need	⚠️ Limited reusability/custom visuals
Embedded Analytics Flow	✅ Seamless with React routing/state	✅ Via iframe or API
Filter Panels	✅ Custom design, can save state	✅ Slicers available
Bookmarks/Views	✅ Can build via user config saving	✅ Native support
Cross-Filtering	✅ Possible with state sync logic	✅ Native
Per-user config (e.g., column select)	✅ Easy via JSON configs/state mgmt.	⚠️ Complex via bookmarks/APIs
White-labeling	✅ Fully skinnable	⚠️ Limited

Multi-tenant Custom Logic	✅ Possible with smart component design	⚠️ Needs complex Power BI setups

19 TRANSITION PLAN FROM PBI TO REACTJS

- **Assessment of Existing Power BI Report:**
  - Review the existing Power BI report to understand the data sources, visualizations, calculations, and overall logic.
  - Identify key performance indicators (KPIs), charts, tables, filters, and other components used in the report.
- **Identify Data Sources:**
  - Identify the data sources used in Power BI.
  - Ensure data structure compatibility between Power BI and the ReactJS environment.
- **Report Configuration:**
  - If any changes are needed in the current Power BI report setup, the IC Team will take care of it.
  - If a non-DTR report needs to be converted to a DTR report, the IC Team will handle the conversion.
  - Any changes to the table structure or data types will be handled by the IC Team.
- **React Development Phase:**
  - Implement the data-fetching logic using Python APIs and procedures to retrieve data from Snowflake.
  - Recreate a similar layout and design in ReactJS by using various components from the component library to replicate the look and feel of the Power BI report.
- **Testing and Validation**
  - Test data rendering and UI performance with large datasets to ensure smooth operation.
  - Ensure that the report is responsive and user-friendly, considering it will be access on Web and ipad.

20 REPORT DEPLOYMENT STRATEGY

It follows the existing process.

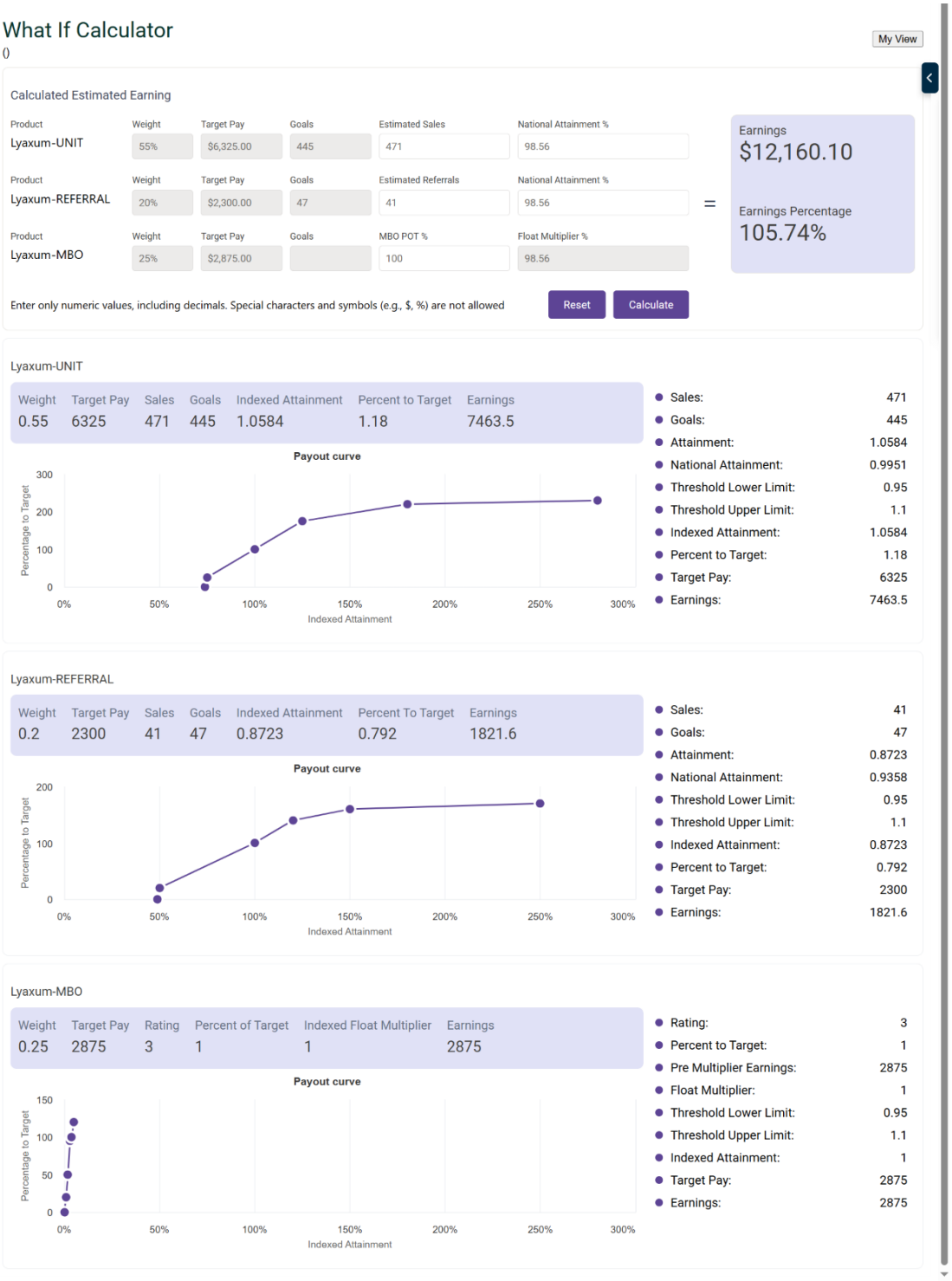
21 MULTI-TENANCY CONSIDERATIONS

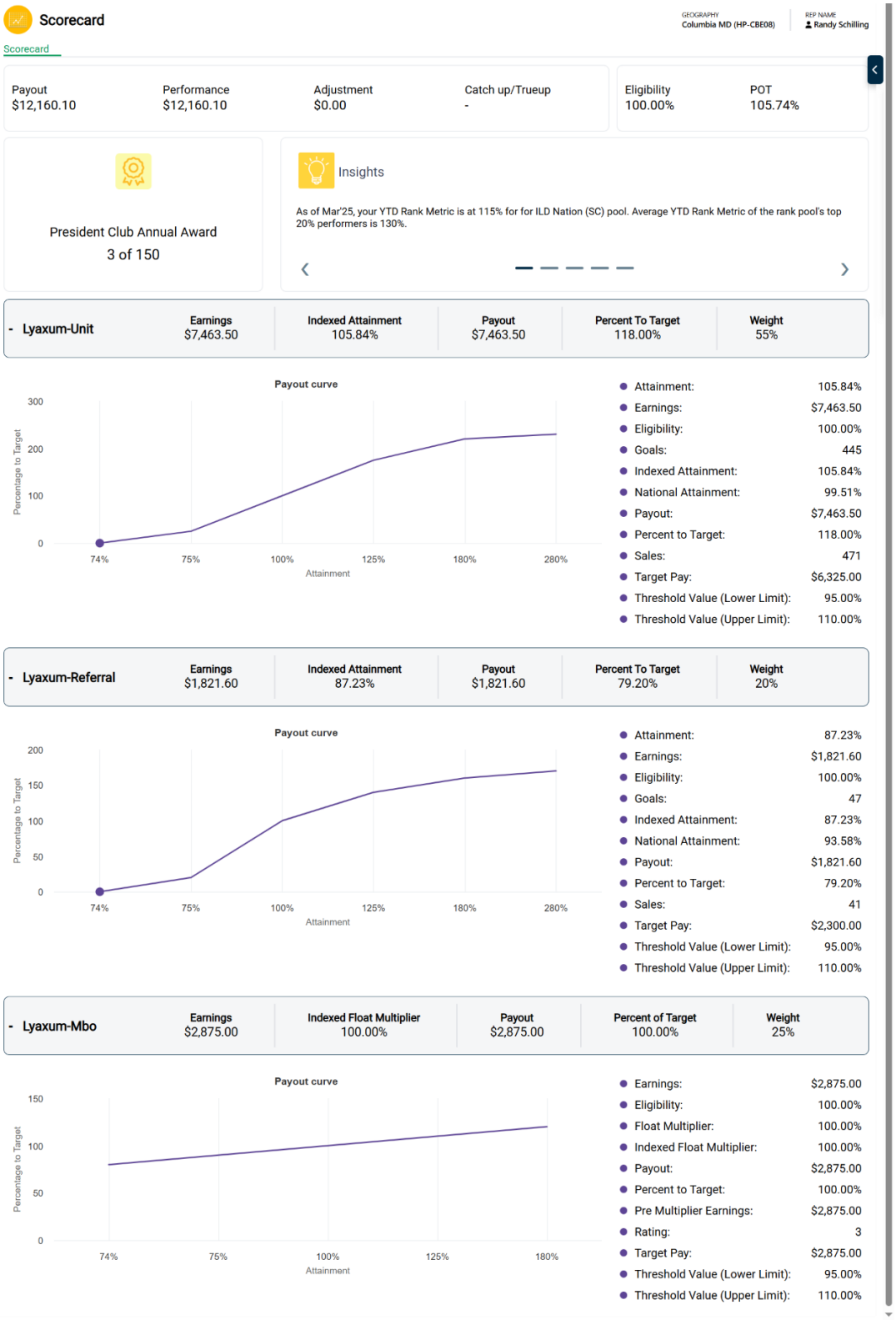
It follows the existing process.

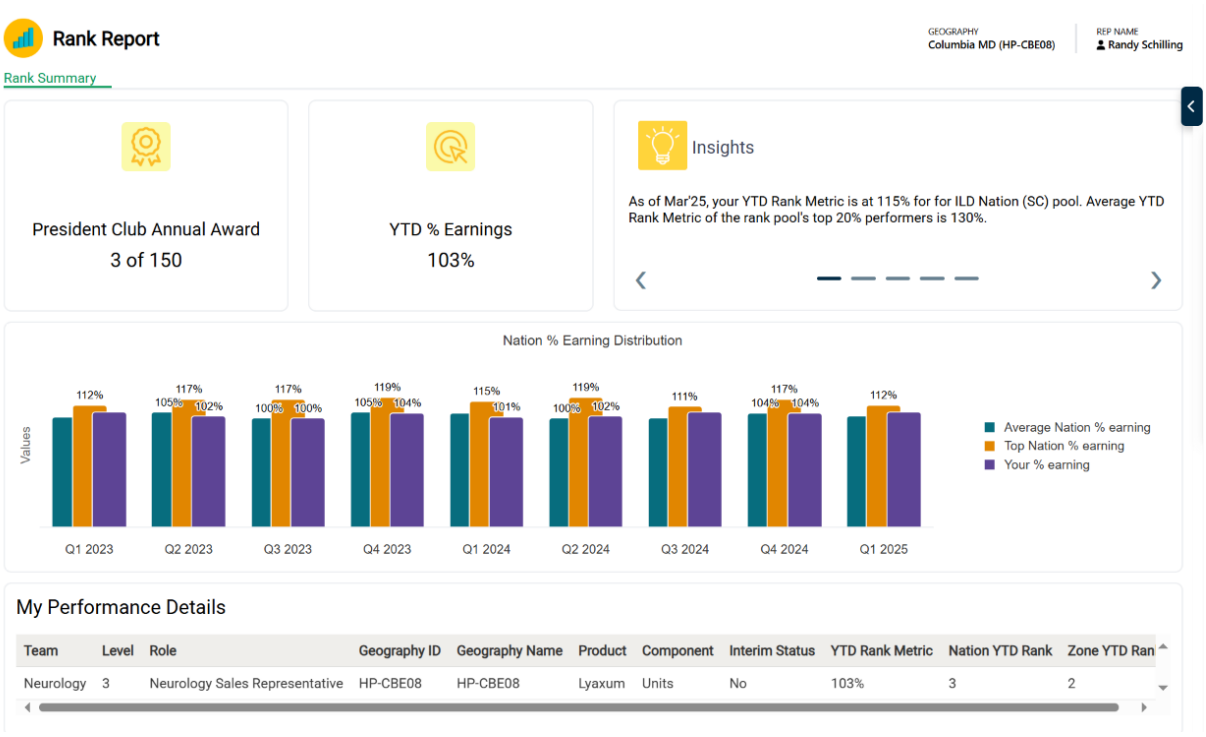
22 CUSTOMER CUSTOMIZATION CAPABILITY

This includes business team-driven data configurations, such as customizing KPI labels, table column headers, and the sequencing of columns. These types of changes can be easily managed based on business requirements.

23 SCREENSHOTS







My Performance Details

Team	Level	Role	Geography ID	Geography Name	Product	Component	Interim Status	YTD Rank Metric	Nation YTD Rank	Zone YTD Ran
Neurology	3	Neurology Sales Representative	HP-CBE08	HP-CBE08	Lyaxum	Units	No	103%	3	2