### ****ThingsBoard Dashboard System Overview****

The **ThingsBoard Dashboard System** is a key feature of the ThingsBoard platform, designed to provide real-time monitoring and visualization for Internet of Things (IoT) devices. The system allows users to create customizable, interactive dashboards to display device data, telemetry, and control various aspects of IoT devices in a visual manner.

**Dashboards**

* **Definition:** Dashboards are collections of widgets (charts, gauges, maps, tables) that visualize IoT data in a user-friendly interface.
* **Purpose:** Provides insights into device status, performance, or other metrics in real-time.
* **Features:**
  + **Multiple Dashboards**: You can create multiple dashboards for different purposes or different groups of users.
  + **Customizable Layouts**: Drag-and-drop layout design, resizing, and widget organization.
  + **Interactive Widgets**: Allows for interaction with data (e.g., clicking on a device to show detailed information).
  + **Multi-tenant Support**: Dashboards can be customized and segmented for different tenants or user groups.

**2️⃣ Widgets**

* **Definition:** A widget is a component that displays a specific piece of information. This could be a chart, gauge, table, map, or any other UI element that renders data dynamically.
* **Widget Types**:
  + **Charts**: Bar charts, line charts, pie charts, etc., to visualize time-series data.
  + **Gauges**: Show values like temperature, humidity, or pressure in a circular dial.
  + **Maps**: Display device locations on a geographical map (with geospatial data).
  + **Tables**: Display a tabular list of data, useful for device status or logs.
  + **Indicators**: Used to show on/off statuses or other boolean data.

**3️⃣ Widget Bundles**

* **Definition:** Widget Bundles are groups of related widgets packaged together. They define how widgets are rendered, configured, and styled.
* **Purpose:** Widget Bundles allow you to package multiple widgets with preset configurations for easy reuse and sharing across dashboards.

**4️⃣ Templates**

* **Definition:** Templates are predefined widget configurations that allow for quick dashboard creation and consistency in design.
* **Use Case:** If you have similar dashboards across devices or tenants, templates can be used to apply the same configuration (e.g., widget layout, theme, widget type) across multiple dashboards.

**5️⃣ Themes and Layouts**

* **Themes:** Themes define the overall look and feel of the dashboard, including colors, fonts, and styling.
* **Layouts:** Layouts allow users to position and resize widgets on the dashboard. It supports **drag-and-drop** functionality to easily customize the dashboard.

**6️⃣ Real-Time Data & Telemetry**

* **Telemetry Data:** Dashboards can visualize telemetry data, which includes real-time or historical data sent by IoT devices. ThingsBoard provides a streaming mechanism for sending this data.
* **WebSocket Streaming:** Real-time data is pushed from the server to the dashboard using WebSockets, ensuring that data is updated instantaneously without the need for refreshing.

**7️⃣ Device Control**

* **Remote Device Control:** Dashboards can allow users to send commands to devices directly through the dashboard interface. For example, you can turn on/off a device, set parameters, or trigger specific actions based on the data shown in the dashboard.
* **Device and Asset Management:** Besides viewing the data, dashboards can also interact with devices (e.g., changing their configurations or settings).

**8️⃣ User & Role-Based Access**

* **User Roles:** ThingsBoard supports role-based access control (RBAC), allowing you to assign different permissions to users based on their role (e.g., Admin, User, Viewer).
* **Multi-Tenant Support:** Different dashboards can be made available to different tenants or user groups.

**9️⃣ API Integration**

* **REST APIs**: ThingsBoard provides REST APIs that allow users to programmatically interact with dashboards, widgets, and device data. This means you can automate dashboard creation, modify widget configurations, or fetch telemetry data using the ThingsBoard backend.
* **Widget API Integration**: Custom widgets can be integrated into dashboards using APIs.

**Dashboard System Workflow:**

1. **Create a Dashboard**: Users can create a new dashboard from the ThingsBoard UI and begin adding widgets.
2. **Customize Widgets**: Widgets are added to the dashboard and configured to show specific device data or telemetry.
3. **Apply Layouts**: Widgets can be arranged in grids, resized, or set with templates to create the dashboard layout.
4. **Monitor Devices**: Once the dashboard is live, real-time data is streamed into the widgets, providing users with live device status and telemetry.
5. **User Interaction**: Users can interact with the dashboard by clicking on devices, opening detailed views, or controlling devices remotely.

**Dashboard System Features:**

* **Drag-and-Drop Layouts**: Rearrange widgets easily.
* **Customizable Widget Settings**: Change chart type, set thresholds, and apply custom styles.
* **Real-Time Data Streaming**: WebSocket-based streaming of telemetry data.
* **Role-Based Access Control (RBAC)**: Different users have different access to dashboards.
* **Multiple Dashboards**: Support for multiple dashboards per tenant/user.
* **Reusable Templates & Widgets**: Easy reusability of widget bundles and templates.
* **Advanced Visualizations**: Support for advanced chart types like heatmaps, time-series graphs, and histograms.

**Use Cases for ThingsBoard Dashboard System:**

1. **Smart City Monitoring**: Visualize and control IoT devices such as smart streetlights, cameras, and sensors across a city.
2. **Industrial IoT (IIoT)**: Monitor industrial equipment like pumps, machines, and motors in factories, along with real-time telemetry.
3. **Energy Management**: Manage energy consumption, track performance of renewable energy sources (e.g., solar, wind), and visualize energy metrics.
4. **Fleet Management**: Monitor vehicle telemetry, such as location, speed, and fuel consumption in real-time.

**How to Integrate Custom Dashboards**

If you need custom dashboards or widgets, you can:

* **Extend ThingsBoard** using the **Widgets Library**.
* **Create Custom Widgets** with **React** or **Angular**.
* **Programmatically Create Dashboards** using ThingsBoard **REST APIs**.

If you're looking to **customize the dashboard system further** or integrate it with your own platform, let me know, and I can help with specific use cases or code examples!

**Project Structure**

You will create **two separate React projects**:

1. **Dashboard Web App** - The main application that hosts the dashboard UI.
2. **Widget Library** - A separate React component library containing reusable widgets, which can be plugged into the dashboard.

**Create the Widget Library as a Separate React Component Library**

This will be a standalone React library that exports various widgets.

**Step 1: Create the Widget Library Project**

Run the following command to create a new React component library using **Vite**:

sh

CopyEdit

npx create-vite widget-library --template react-ts

Go inside the project:

sh

CopyEdit

cd widget-library

Initialize it as a library:

sh

CopyEdit

npm init -y

Update package.json to make it a library:

json

CopyEdit

{

"name": "widget-library",

"version": "1.0.0",

"main": "dist/index.js",

"module": "dist/index.esm.js",

"types": "dist/index.d.ts",

"files": ["dist"],

"scripts": {

"build": "vite build",

"prepublishOnly": "npm run build"

},

"peerDependencies": {

"react": "^18.0.0",

"react-dom": "^18.0.0"

}

}

**Step 2: Create a Widget Component**

Inside src/components, create a sample widget component:

📂 widget-library/src/components/ChartWidget.tsx

tsx

CopyEdit

import React from "react";

import { LineChart, Line, XAxis, YAxis, Tooltip, ResponsiveContainer } from "recharts";

const ChartWidget = ({ data }: { data: any[] }) => {

return (

<ResponsiveContainer width="100%" height={300}>

<LineChart data={data}>

<XAxis dataKey="time" />

<YAxis />

<Tooltip />

<Line type="monotone" dataKey="value" stroke="#8884d8" />

</LineChart>

</ResponsiveContainer>

);

};

export default ChartWidget;

**Step 3: Export Components from Library**

Create src/index.ts to export all components:

📂 widget-library/src/index.ts

tsx

CopyEdit

export { default as ChartWidget } from "./components/ChartWidget";

**Step 4: Build and Publish the Library**

Run:

sh

CopyEdit

npm run build

If you want to use it locally, link the package:

sh

CopyEdit

npm link

Alternatively, you can publish it to **npm**:

sh

CopyEdit

npm publish --access public

**2️⃣ Create the Dashboard Web App**

This is the main React project where the widget library will be used.

**Step 1: Create the Dashboard Web App**

Run:

sh

CopyEdit

npx create-vite dashboard-app --template react-ts

cd dashboard-app

**Step 2: Install the Widget Library**

If using a local package:

sh

CopyEdit

npm link widget-library

If published on npm:

sh

CopyEdit

npm install widget-library

**Step 3: Use Widgets in the Dashboard**

Modify src/components/Dashboard.tsx:

📂 dashboard-app/src/components/Dashboard.tsx

tsx

CopyEdit

import React from "react";

import { ChartWidget } from "widget-library";

const Dashboard = () => {

const data = [

{ time: "10:00", value: 50 },

{ time: "10:05", value: 55 },

{ time: "10:10", value: 60 },

];

return (

<div>

<h1>Dashboard</h1>

<ChartWidget data={data} />

</div>

);

};

export default Dashboard;

**3️⃣ Running the Projects**

Start the widget library (if using local linking):

sh

CopyEdit

cd widget-library

npm run build

npm link

Start the dashboard app:

sh

CopyEdit

cd ../dashboard-app

npm run dev

Now, the dashboard app will dynamically import widgets from the **widget library**!

**✅ Benefits of This Approach**

* **Separation of Concerns**: The widget library is independent and can be reused in other projects.
* **Versioning & Updates**: The widget library can be versioned separately and updated in the dashboard.
* **Plug-and-Play Architecture**: New widgets can be added to the widget library without modifying the dashboard app.

Would you like **drag-and-drop functionality** or **dynamic widget loading** next?