***Angular Container and react repo module federation***

***Steps to be Performed in React repo***

**Step 1:** **Create react repo in typescript using following command:**

npx create-react-app my-app --template typescript

**Step 2: Add following packages and scripts to package.json.**

"dependencies": {

    "copy-webpack-plugin": "^8.1.1",

  },

  "devDependencies": {

    "@svgr/webpack": "^8.1.0",

    "css-loader": "^7.1.2",

    "sass": "^1.77.8",

    "sass-loader": "^12.6.0",

    "style-loader": "^4.0.0",

    "ts-loader": "^9.5.1",

    "webpack-cli": "^5.1.4",

    "webpack-dev-server": "^5.1.0"

  }

  "scripts": {

    "start": "webpack serve --mode=development --devtool=source-map",

    "build": "webpack --mode=production"

  },

**Step 3: Create a webpack.config.ts file and add this code and run npm i.**

const ModuleFederationPlugin = require("webpack/lib/container/ModuleFederationPlugin");

const CopyWebpackPlugin = require("copy-webpack-plugin");

const webpack = require("webpack"); // Make sure to import webpack

module.exports = (options) => {

  return {

    entry: './src/index.tsx', // Entry file is TSX

    output: {

      filename: 'bundle.js',

      publicPath: "auto",

      uniqueName: "mfe1"

    },

    module: {

      rules: [

        {

          test: /\.(ts|js)x?$/, // Match both .ts, .tsx, .js, and .jsx files

          exclude: /node\_modules/,

          use: {

            loader: "babel-loader", // Use babel-loader instead of ts-loader

            options: {

              presets: [

                "@babel/preset-env", // Transpile modern JS features

                "@babel/preset-react", // Transpile JSX/React code

                "@babel/preset-typescript", // Transpile TypeScript

              ],

            },

          },

        },

        {

          test: /\.css$/, // Rule for CSS files

          use: ['style-loader', 'css-loader'] // Use style-loader and css-loader

        },

        {

          test: /\.scss$/, // Rule for SCSS files

          use: [

            'style-loader', // Inject styles into DOM

            'css-loader', // Translates CSS into CommonJS modules

            'sass-loader' // Compiles Sass to CSS

          ],

        },

        {

          test: /\.svg$/, // Rule for handling SVG files

          oneOf: [

            {

              issuer: /\.[jt]sx?$/, // If the import is from a JS/TSX file

              use: ['@svgr/webpack'], // Use svgr for SVG as React Component

            },

            {

              type: 'asset', // Default file handling for SVGs (URL path)

              resourceQuery: /url/, // If the import has ?url, treat as a URL

            },

          ],

        }

      ],

    },

    resolve: {

      extensions: ['.tsx', '.ts', '.js', '.jsx'], // Add .tsx, .ts, .js, and .jsx as resolvable extensions

    },

    plugins: [

      new ModuleFederationPlugin({

        name: "react",

        library: {

          type: "var",

          name: "react"

        },

        filename: "remoteEntry.js", // Expose entry point for remote access

        exposes: {

          './web-components': './src/App.tsx', // Expose your App component

        },

        shared: {

          react: {

            singleton: true,

            eager: true,  // Ensure React is loaded only once and early

            requiredVersion: false

          },

          'react-dom': {

            singleton: true,

            eager: true, // Ensure ReactDOM is loaded only once and early

            requiredVersion: false

          }

        }

      }),

      new CopyWebpackPlugin({

        patterns: [

          { from: './public/index.html', to: 'index.html' } // Adjust this path as per your public directory

        ]

      }),

      new webpack.ProvidePlugin({

        React: "react", // Automatically imports React wherever needed

      })

    ],

    devServer: {

      port: 4209, // You can give any port number

      static: {

        directory: './public' // Serve files from the public directory

      },

    }

  };

};

**Step 4: Now, add this code to app.tsx file.**

// ReactMfe component rendering the App component

const ReactMfe = () => {

  return <App />;

};

// Register the custom web component to be used in HTML

class WebComponent extends HTMLElement {

  connectedCallback() {

    const root = createRoot(this); // Use createRoot to create the React root

    root.render(<ReactMfe />);

  }

}

// Define the custom element

customElements.define("react-element", WebComponent);

const [dashboardValue, setDashboardValue] = useState("");

  const [countManageValue, setCountManageValue] = useState("");

  useEffect(() => {

    // Listener for the profileData event from Angular

    console.log("Received event:", new Date());

    const eventListener = (res: any) => {

      setDashboardValue(res.detail.profileData?.domainId); // Profile Data

      setCountManageValue(res.detail.businessUnit); // Business Unit

      console.log(

        "Received Event Data in React repo:",

        res.detail,

        dashboardValue,

        countManageValue

      );

    };

    // Immediately add event listener for profileData

    window.addEventListener("profileData", eventListener);

    // Clean up the event listener when the component is unmounted

    return () => {

      window.removeEventListener("profileData", eventListener);

    };

  }, []);

Complete App.tsx file would look like this

import "./App.css";

import React, { useEffect, useState } from "react";

import { BrowserRouter, Routes, Route } from "react-router-dom";

import IdCardRequest from "./components/id-card-request/IdCardRequest";

import IDcard from "./components/id\_card/id\_card";

import BonafideLetterLanding from "./components/BonafideLetterLanding";

import { createRoot } from "react-dom/client";

const App: React.FC = () => {

  const [dashboardValue, setDashboardValue] = useState("");

  const [countManageValue, setCountManageValue] = useState("");

  useEffect(() => {

    // Listener for the profileData event from Angular

    console.log("Received event:", new Date());

    const eventListener = (res: any) => {

      setDashboardValue(res.detail.profileData?.domainId); // Profile Data

      setCountManageValue(res.detail.businessUnit); // Business Unit

      console.log(

        "Received Event Data in React repo:",

        res.detail,

        dashboardValue,

        countManageValue

      );

    };

    // Immediately add event listener for profileData

    window.addEventListener("profileData", eventListener);

    // Clean up the event listener when the component is unmounted

    return () => {

      window.removeEventListener("profileData", eventListener);

    };

  }, []);

  return (

    <BrowserRouter>

      <div style={{ padding: "32px" }}>

        <Routes>

          <Route path="/" element={<BonafideLetterLanding />} />

          <Route path="/idcard-request" element={<IdCardRequest />} />

          <Route path="/id-card" element={<IDcard />} />

        </Routes>

      </div>

    </BrowserRouter>

  );

};

// ReactMfe component rendering the App component

const ReactMfe = () => {

  return <App />;

};

// Register the custom web component to be used in HTML

class WebComponent extends HTMLElement {

  connectedCallback() {

    const root = createRoot(this); // Use createRoot to create the React root

    root.render(<ReactMfe />);

  }

}

// Define the custom element

customElements.define("react-element", WebComponent);

export default App;

**Step 5: comment this line in index.html file**

    <div id="root"></div>

And add these lines above the commented line

    <react-element></react-element>

    <script src="./bundle.js"></script>

**Step 6: Last step in react repo is to run this command “npm run build”.**

**Steps to be Performed in angular container:**

**Step 1: Add this package in package.json file of angular container and run “npm i” command.**

  "dependencies": {

    "@angular-architects/module-federation-tools": "^18.0.6",

  },

**Step 2: Add this code to app-routing.module.ts file in angular container.**

import {

  WebComponentWrapper, WebComponentWrapperOptions

} from '@angular-architects/module-federation-tools';

  {

    path: 'vmsv2',

    component: WebComponentWrapper,

    data: {

      type: 'script',

      remoteEntry: 'http://localhost:4209/remoteEntry.js',

      // remoteEntry:

      // environment.ROUTING\_URL + "/vmsv2/remoteEntry.js",

      remoteName: 'react',

      // remoteName: 'pmc\_rc\_securityservice\_v2',

      exposedModule: './web-components',

      elementName: 'react-element',

    } as WebComponentWrapperOptions,

    // canActivate: [LoginService],

  },

**Now, make sure that the port number in react repo matches the above remoteEntry path and remoteName should match the react repo these names, you can give it unique name.**

    plugins: [

      new ModuleFederationPlugin({

        name: "react",

        library: {

          type: "var",

          name: "react"

        },