Task 2: You are working on a dashboard using ReactJS and Tailwind CSS. The dashboard displays data charts which are fetched from a REST API. However, the charts are not updating in real-time as new data arrives from the API.

Question: How would you troubleshoot and resolve this issue ensuring the charts update in real-time as the data changes? Explain your approach, potential challenges and the overall thought process.

Answer: By following a systematic method, I can troubleshoot and resolve the issue of charts not updating in real-time as new data arrives from the API in a React JS and Tailwind CSS dashboard.

Firstly I check the API response status, how it delivers, and if it has any error. Verify the update of state using the 'useState' hook when it gets new data. I will use the 'useEffect' hook to check real-time updates of data. For handling error I'll check the unsuccessful status and place an error to the console. Lastly, I'll check if there are any unwanted burdens or huge data on the server.

Some of the challenges and considerations of the given problem I have to check Cross-origin Resource Sharing (CORS) and make sure that React JS application can send queries to the API, particularly if it is hosted on a different domain. Then make sure that the backend API – whether via WebSocket or other methods – is built to accommodate updates in real-time. Besides handling Large Data Sets where I have to consider how to efficiently handle and update large datasets without causing performance issues on the client side. Finally, I have to concentrate on security and implement secure communication practices, especially if dealing with sensitive data or real-time updates.

Understanding how the API offers real-time changes, choosing a suitable real-time communication method (WebSocket, polling), and making sure the React state is updated appropriately are crucial to fixing the problem. Crucial components of the troubleshooting process include testing, debugging, and knowing what the API is capable of.