Approaches :

1. Check Data Flow :
    - Examine the data flow from the REST API to the charts. Ensure that the data fetched from the API is reaching the chart components.
    - Use (console.log) or debugging tools to log the received data and confirm that it reflects the real-time changes.
2. Verify API Connection :
    - Confirm that the API is indeed sending real-time updates. Check the API documentation and test the API using tools like Postman or Curl to verify that new data is being pushed or available at the API endpoint.
3. Update State Correctly :
    - If using React state or a state management library (e.g., Redux), ensure that the state is being updated correctly when new data arrives. Check the component state or the Redux store to see if it reflects the latest data.
4. React Component Lifecycle :
    - Understand the React component lifecycle, especially the (ComponentDidUpdate) lifecycle method or the (UseEffect) hook if using functional components. Ensure that the chart components are re-rendering when the state changes.

Potential Challenges :

1. API Data Limiting :
    - The API might have rate-limiting policies that could impact real-time updates. Check API documentation for any restrictions.
2. Compatibility Issues :
    - Ensure compatibility between the version of the charting library, ReactJS, and other dependencies. Incompatibilities could lead to unexpected behavior.

Overall Thought Process :

1. Isolate The Issues :
    - Break down the problem by checking each component of the system separately (API, state management, chart components).
2. Log And Monitor :
    - Utilize logging statements and monitoring tools to track the flow of data and identify any irregularities.
3. Colaboration :
    - If the problem persists, I can take help from colleagues, online communities, or forums where React and Tailwind CSS experts may provide insights.