Troubleshooting and resolving real-time updating issues in a ReactJS and Tailwind CSS dashboard fetching data from a REST API involves a systematic approach. Here are the steps can be taken:

1. Understand the Data Flow:

- Ensure that your React components are correctly fetching and updating data from the REST API.
- Verify that the API response contains the latest data.

2. Check State Management:

- Confirm that the data fetched from the API is being stored correctly in the React component's state.
- Ensure that the state changes trigger re-renders.

3. Use React Hooks Effect:

- Utilize the `useEffect` hook to handle side effects such as fetching data.
- Make sure that the `useEffect` hook is set up to run when the component mounts and whenever there is a change in relevant dependencies (e.g., the data fetched from the API).

4. Update Charts on Data Changes:

- Check if the chart library that are using has proper mechanisms to update the charts dynamically.
- Ensure that the chart component receives the updated data as props whenever there is a change in the data.

5. Debugging:

- Use browser developer tools to inspect the network requests and responses to confirm data is being fetched correctly.
- Utilize console logs or debugging tools to track the flow of data through your components and functions.

6. Real-Time Updates:

- If real-time updates are required, consider implementing a WebSocket connection for live data streaming.
- Alternatively, we can implement a periodic data polling mechanism to fetch the latest data at regular intervals.

Potential Challenges:

1. **State Immutability:** Ensure that you are updating state immutably to trigger re-renders.
2. **Async Operations:** Be mindful of asynchronous operations when fetching data. Make sure to handle promises correctly.
3. **Chart Library Compatibility:** Check if the chart library you're using supports dynamic updates and ensure you are using it correctly.
4. **API Rate Limits:** Confirm that you are not hitting API rate limits, especially if you are polling for data frequently.

Overall Thought Process:

1. **Data Flow and State:** Verify that data is flowing correctly from the API to the React state.
2. **UseEffect for Side Effects:** Implement the `useEffect` hook to manage the side effect of fetching data.
3. **Update Chart Components:** Ensure that the chart components receive updated data when the state changes.
4. **Debugging:** Use debugging tools to trace the flow of data and identify any issues.
5. **Real-Time Considerations:** Implement real-time updates if necessary, considering the requirements of your application.

By following this systematic approach, we should be able to troubleshoot and resolve the issue of charts not updating in real-time as new data arrives from the API.