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.

Troubleshoot Techniques:

- `Console tab`: Understand whether updated data is coming or not, debugging.
- `Network tab`: Examine network activity to ensure API calls are being made and responses are as expected.
- `React dev-tool`: Inspect React component state, props, and lifecycle events.

Scenario-I: UI isn't updated in real-time

In this code, the chart only gets data from a specific URL once, when it first appears. After that, it doesn't fetch updated data in real-time. If you check the network activity, you'll notice that there's only one request made when the chart is shown for the first time.

Scenario-I

```jsx
import { useState, useEffect } from "react";

function Chart() {
  const [chartData, setChartData] = useState(null);

  // Fetch data only once after the first mount
  useEffect(() => {
    fetch('URL')
      .then((res) => res.json())
      .then((data) => {
        console.log(data);
        setChartData(data);
      })
      .catch((err) => console.log(err));
  }, []);

  return <p>{chartData}</p>;
}

export default function App() {
  const [show, setShow] = useState(false);

  return (
    <>
      <button onClick={() => setShow(!show)}>
        {show? "Unmount": "Mount"} the component
      </button>
      {show && <hr />}
      {show && <Chart />}
    </>
  );
}
```

The chart component fetches data only once when it's first displayed. If you want the data to update in real-time, you need to modify the code to fetch new data periodically. This can be done using techniques like setting intervals. Adjusting the code to handle continuous updates will ensure the chart reflects the most recent information.

Scenario-II: Unwanted network calls despite unmounting

In the updated code, real-time updates are achieved by fetching data every second using setInterval. However, a crucial issue arises when the Chart component is unmounted: the fetch API continues to make unnecessary network calls, leading to inefficient resource utilization.

Scenario-II

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

function Chart() {
  const [chartData, setChartData] = useState(null);

  //! Fetch data every 1s, but continues even after being unmounted
  useEffect(() => {
    setInterval(() => {
      fetch('URL')
        .then((res) => res.json())
        .then((data) => {
          console.log(data);
          setChartData(data);
        })
        .catch((err) => console.log(err));
    }, 1000);

  }, []);

  return <p>{chartData}</p>;
}

export default function App() {
  const [show, setShow] = useState(false);

  return (
    <>
      <button onClick={() => setShow(!show)}>
        {show? "Hide": "Show"} the chart
      </button>
      {show && <hr />}
      {show && <Chart />}
    </>
  );
}
```

Scenario-III: Real-Time UI data updates and no unwanted network calls

In the following code modification, the issue of unnecessary network calls is addressed, and the UI now receives real-time updates as intended. The setInterval fetches data every second, ensuring the display reflects the latest information. Importantly, when the Chart component is unmounted, the cleanup function effectively clears the interval, preventing unwanted network calls.