NAME: BATTULA BHAVANA
ROLL NO: CH.EN.U4AIE21010

Case Study 1: Real-Time Stock Market Dashboard

1. How would you implement a WebSocket connection in a React component for real-time data fetching?

To implement a WebSocket connection in a React component, you can use the WebSocket API within the use Effect hook. In the use Effect, create a new instance of WebSocket and set up event handlers for onopen, onmessage, onerror, and onclose. When a message is received (onmessage), you can parse the data and update the component's state to reflect the latest stock prices. The WebSocket connection should be closed when the component unmounts to prevent memory leaks.

2. Describe how you would create a responsive table to display stock prices.

A responsive table can be created using CSS Flexbox or Grid. The table should be structured with thead and tbody elements for the headers and data rows. Using CSS, you can style the table to ensure it adapts to different screen sizes. For example, applying max-width and width properties to the table and using media queries can help ensure the table is easily readable on mobile devices.

3. How can you implement a search bar to filter stocks based on the user's input?

To implement a search bar, create an input field that tracks the user's input using the useState hook. Attach an onChange event handler to the input that filters the stock list based on the query. This handler should update the displayed stocks by filtering the original stock data array, allowing users to see only the stocks that match their input.

4. Explain the steps for handling connection errors and displaying appropriate messages to the user.

To handle connection errors, you can set up an onerror event handler in your WebSocket connection. When an error occurs, update a state variable to reflect the error status, which can be used to conditionally render an error message in the UI. Additionally, you can provide a retry mechanism or connection status checks to inform the user of the connection state.

5. How would you ensure the efficient updating of stock prices in the UI without performance degradation?

To ensure efficient updating of stock prices, utilize React's useMemo and useCallback hooks. These hooks help optimize performance by memoizing values and functions, thus preventing unnecessary re-renders. Additionally, you can limit the frequency of updates by batching updates or using techniques like throttling or debouncing when processing incoming stock price data.