Day 3CC2: **Component Styling & Lifecycle Coding Challenge**

Problem Statement:

You are developing a dashboard for a real-time stock market tracking application. The dashboard should display live stock prices, allow users to customize styles, and showcase the component lifecycle methods in action.

User Stories:

1. Styling the Dashboard

- As a user, I want to have a visually appealing dashboard with custom styles, so that I can clearly view stock prices.
- As a developer, I should be able to apply CSS and Bootstrap to improve UI consistency.

2. Managing Component Lifecycle

- As a developer, I want to implement lifecycle methods to fetch stock data when the component mounts, so that users see live stock prices immediately.
- As a developer, I want to update stock prices dynamically using lifecycle methods, so that users see real-time updates.

3. Handling Controlled & Uncontrolled Components

- As a user, I want to input a stock symbol using a controlled component, so that I can fetch stock data dynamically.
- As a user, I should see previously entered stock symbols stored in an uncontrolled component, so that I can track my searches.

4. Real-Time Data with WebSockets

- As a user, I want to see stock price changes in real-time, so that I can make quick investment decisions.
- As a developer, I should use ExpressJS and Socket.io to stream live stock data updates.

Demo Implementation Steps:

1. Styling & Bootstrap Implementation

- Create a React component for the stock market dashboard.
- Apply CSS styles and Bootstrap classes for layout and design.

2. *Implementing Lifecycle Methods*

- Use `componentDidMount` to fetch initial stock data.
- Use `componentDidUpdate` to refresh stock prices dynamically.

3. *Controlled & Uncontrolled Components*

- Implement an input field for stock symbol using controlled components.
- Store previous searches using uncontrolled components.

4. *Setting Up ExpressJS & Socket.io*

- Set up an Express server to fetch stock prices from an external API.
- Use Socket.io to send real-time stock updates to the React frontend.

Final Objective:

By the end of this challenge, users will have a fully functional stock market dashboard with styled components, controlled and uncontrolled inputs, and real-time stock updates powered by WebSockets.

Bonus Challenge:

- Add a feature to allow users to customize the dashboard theme dynamically.
- Display stock trends using charts (e.g., with Chart.js).