- Displaying the chat message and deploying your chat app

Chat Message Component:

The Chat Message component is created to display chat messages with different positions (left or right) and mood emojis (happy, sad, or neutral).

Sentiment Emoji Constants:

Three constants are defined for sentiment emojis: SAD\_EMOJI, HAPPY\_EMOJI, and NEUTRAL\_EMOJI. These constants store Unicode code points for the corresponding emojis.

Chat Component Modifications:

The Chat component is modified to render chat messages based on the state's chats array.

Chat Message Rendering:

For each chat message in the chats array, the component determines the position (left or right) based on whether the sender is the active user.

The sentiment score is used to determine the mood emoji (happy, sad, or neutral) for the message.

The component also checks whether the message is the first in the list, follows a message from another user, or has a delay of over 1 minute from the previous message of the same user. Based on these conditions, it displays the user's name and mood emoji before the message.

The ChatMessage component is used to render the chat message itself, with the position prop determining its position on the screen.

Emoji Rendering:

The String.fromCodePoint(...mood) method is used to render the mood emoji based on the Unicode code points defined earlier.

**React Native**

React Native is a popular framework for mobile app development, and its usefulness is attributed to several key advantages.

Aspects of React Native:

Cross-Platform Compatibility: React Native allows you to write code once and deploy it on both iOS and Android platforms. This saves time and effort in developing and maintaining separate codebases for each platform.

Code Reusability: With React Native, a significant portion of your codebase, including components and business logic, can be reused between web and mobile applications if you're using React on the web.

Fast Development: React Native streamlines the development process with its hot-reloading feature, enabling developers to see real-time changes without rebuilding the entire app. This speeds up development and debugging.

Large and Active Community: React Native has a vast and active community of developers, which means you can find plenty of resources, libraries, and third-party components to help you with your projects. This community support can be invaluable for problem-solving and learning.

Performance: React Native offers near-native performance thanks to its use of native modules and components. It provides a smooth and responsive user experience.

Cost-Effective: Building a cross-platform app with React Native typically requires fewer resources (developers and time) than developing separate native apps for iOS and Android. This can result in cost savings for businesses.

Live Updates: React Native supports over-the-air (OTA) updates, allowing you to make updates to your app without having to go through the app store approval process. This is especially useful for bug fixes and minor feature updates.

Third-Party Plugins: React Native has an extensive library of third-party plugins and components available through the npm ecosystem. This means you can easily add functionality to your app by leveraging existing packages.

Strong Developer Experience: React Native has excellent developer tools and a comprehensive ecosystem, including libraries for navigation, state management (e.g., Redux), and testing, which enhances the development experience.

Support from Facebook: React Native is backed by Facebook, which continues to invest in its development and improvement. This provides confidence in its long-term sustainability.

Smooth User Interface: React Native allows for the creation of a highly responsive and smooth user interface, making it suitable for various types of applications, including those with complex animations and interactions.