Internship Task Documentation

Torq ReactJS Development

CHANDRA SAI TEJA ADHIKARLA

28 May 2023

Logic and approaches used to achieve Real-time messaging , Theme Adaption and Code Highlighting

Real-Time Messaging :

- Establishing WebSocket Connection :

  The client-side uses the ‘socket.io-client’ library to establish a WebSocket connection with the server . The connection is set up in the ‘useEffect’ hook using the ‘io()’ function provided by ‘socket.io-client’. The ‘socket’ object represents the WebSocket connection and is stored in the ‘socket.current’ ref for access throughout the component.
- Sending Messages :

  When the user clicks the send button , the ‘handleMessageSend’ function is triggered. The function emits a ‘sendMessage’ event to the server using the ‘socket.emit()’ method . The message content , sender information , and a timestamp are sent as data in the event .
- Receiving Messages :

  The server broadcasts the received message to all connected clients using the ‘message’ event. On the client-side , the “socket.on(‘message’)” event listener is used to receive and handle the incoming messages. The ‘setMessages’ function updates the ‘messages’ state by appending the new message to the existing message array.

Theme Adaptation :

- React State Management :

  The current theme state is managed using the ‘useState’ hook . The initial theme state is set to ‘light’. The ‘theme’ variable holds the current theme , and the ‘setTheme’ function is used to update the theme.
- Theme Toggle Handling :

  The ‘handleThemeToggle’ function is triggered when the user clicks the theme toggle switch . It toggles the current theme between ‘light’ and ‘dark’ using a conditional statement . The ‘setTheme’ function is called with the updated theme to apply the new theme.

- Applying Theme Styles :

  CSS variables are used to define theme-specific values , such as colours and background . The ‘theme’ class is dynamically added to the chat sidebar element based on the current theme state. CSS rules with theme-specific variables are defined using the ‘.theme-light’ and ‘.theme-dark’ selectors .

Code Highlighting :

- Highlighting Code Snippets :

  The ‘handleCodeHighlight’ function is used to replace code snippets within triple backticks (```) with highlighted HTML code. Prism.js (JavaScript Syntax Highlighting Library) is used for code syntax highlighting .
- Replacing Code Snippets :

  The ‘replace’ method with a regular expression is used to identify code snippets in the message content . Each code snippet is replaced with an HTML string containing a ‘<pre>’ element with the appropriate class for Prism.js highlighting .

Theme Adaptation Documentation

React State Management :

- Establishing WebSocket Connection :

  The current theme state is managed using the ‘useState’ hook from React. The initial value for the theme state is set to ‘light’ using the ‘useState’ hook: ‘const [theme,setTheme] = useState(‘light);’ . The ‘theme’ variable holds the current theme and the ‘setTheme’ function is used to update the theme .
- CSS Variables :

  CSS variables are utilised to define theme-specific values , such as colours and background . The CSS variables are defined in the ‘ChatSidebar.css’ file . For example the light theme background color is defined as ‘background-color: #fff;’ , while the dark theme background is defined as ‘background-color: #262626;’ . The theme-specific CSS rules are defined using the ‘.theme-light’ and ‘.theme-direct’ selectors in the ‘ChatSidebar.css’

Code Highlighting Documentation

Approach :

- ❖ Regular Expression :

  A regular expression (‘codeRegex’) is used to identify code snippets in the message content . It matches text enclosed within triple backticks (```) and captures the code content .
- ❖ Handling Code Highlighting :

  The ‘handleCodeHighlight’ function in ‘ChatSidebar.js’ processes the message content to identify code snippets.
- ❖ Prism.js Integration :

  Prism.js library and CSS are imported for syntax highlighting capabilities .
- ❖ Applying Syntax Highlighting :

  The ‘replace’ function replaces code snippets in the message content with highlighted HTML elements . Each code snippets is passed to Prism.js to apply syntax highlighting . The Highlighting code is wrapped in ‘<pre>’ and ‘<code>’ HTML elements.

Methodology :

- ❖ Extraction of Code Snippets :

  The ‘codeRegex’ regular expression identifies code snippets in the message content .
- ❖ Applying Syntax Highlighting :

  A callback function applies syntax highlighting to each code snippet using Prism.js. The highlighted code is returned as a replacement for the code snippet .
- ❖ Rendering Highlighted Content :

  The highlighted message content is rendering using ‘dangerouslySetInnerHTML’. The HTML content is inserted into the element while bypassing React’s XSS protection.