Sure, here are the test cases for the provided React component Message with 100% coverage:

TypeScript

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import Message from './Message';

jest.mock('axios');

describe('Message Component', () => {
  it('should render the message with the correct content for bot
messages', () => {
    const botMessageProps = {
      role: 'bot',
      content: [
        {
          context: 'This is the first bot response.',
          book: 'Book Title 1',
          section_title: 'Section Title 1',
          hyperlink: 'https://example.com/book1',
          generated_resp: 'Bot response 1',
        },
        {
          context: 'This is the second bot response.',
          book: 'Book Title 2',
          section_title: 'Section Title 2',
          hyperlink: 'https://example.com/book2',
          generated_resp: 'Bot response 2',
        },
      ],
      theme: 'dark',
    };

    render(<Message {...botMessageProps} />);

    const messageContent = screen.getByRole('listitem');
    expect(messageContent).toContainText('This is the first bot
response.');
    expect(messageContent).toContainText('Book Title 1');
    expect(messageContent).toContainText('Section Title 1');

expect(messageContent).toContainText('https://example.com/book1');
    expect(messageContent).toContainText('Bot response 1');
```

```
    const tabs = screen.getAllByRole('tab');
    expect(tabs.length).toBe(2);

    const tabPanels = screen.getAllByRole('tabpanel');
    expect(tabPanels.length).toBe(2);
  });

  it('should render the message with the correct content for user
messages', () => {
    const userMessageProps = {
      role: 'user',
      content: 'This is a user message.',
      theme: 'light',
    };

    render(<Message {...userMessageProps} />);

    const messageContent = screen.getByRole('listitem');
    expect(messageContent).toContainText('This is a user message.');
  });

  it('should render the loading spinner for bot messages when
botIsTyping is true', () => {
    const loadingMessageProps = {
      role: 'bot',
      botIsTyping: true,
      theme: 'dark',
    };

    render(<Message {...loadingMessageProps} />);

    const loadingSpinners = screen.getAllByRole('img');
    expect(loadingSpinners.length).toBe(3);
  });

  it('should handle the copy icon click', () => {
    const mockHandleCopy = jest.fn();
    const messageProps = {
      role: 'bot',
      content: [
        {
          context: 'This is the first bot response.',
          book: 'Book Title 1',
          section_title: 'Section Title 1',
          hyperlink: 'https://example.com/book1',
          generated_resp: 'Bot response 1',
        },
      ],
```

```
      theme: 'dark',
      handleCopy: mockHandleCopy,
    };

    render(<Message {...messageProps} />);

    const copyIcon = screen.getByRole('button', { name: 'Copy Icon'
});
    fireEvent.click(copyIcon);

    expect(mockHandleCopy).toHaveBeenCalledTimes(1);
  });

  it('should handle the feedback icons click', () => {
    const mockHandleFeedback = jest.fn();
    const messageProps = {
      role: 'bot',
      content: [
        {
          context: 'This is the first bot response.',
          book: 'Book Title 1',
          section_title: 'Section Title 1',
          hyperlink: 'https://example.com/book1',
          generated_resp: 'Bot response 1',
        },
      ],
      theme: 'dark',
      handleFeedback: mockHandleFeedback,
    };

    render(<Message {...messageProps} />);

    const thumbUpIcon = screen.getByRole('button', { name: 'Hand
Thumb Up Icon' });
    fireEvent.click(thumbUpIcon);

    expect(mockHandleFeedback).toHaveBeenCalledWith('up');

    const thumbDownIcon = screen.getByRole('button', { name: 'Hand
Thumb Down Icon' });
    fireEvent.click
```