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

TypeScript

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

jest.mock('axios');

describe('ChatInterface Component', () => {
  it('should render the chat interface with the correct initial
state', () => {
    render(<ChatInterface theme="dark" />);

    expect(screen.getByRole('button', { name: 'One stop shop for
Procedure needs' })).toBeInTheDocument();
    expect(screen.getByRole('textbox')).toBeInTheDocument();
    expect(screen.getByRole('button', { name: 'Send'
})).toBeInTheDocument();
  });

  it('should render messages from the conversations prop', () => {
    const mockConversations = [
      { id: 1, title: 'Conversation 1', messages: [{ role: 'user',
content: 'Hello' }, { role: 'bot', content: 'Hi there!' }] },
      { id: 2, title: 'Conversation 2', messages: [{ role: 'user',
content: 'How can I help you?' }] },
    ];

    render(<ChatInterface theme="dark"
conversations={mockConversations} />);

    const messages = screen.getAllByRole('listitem');
    expect(messages.length).toBe(3);
    expect(messages[0]).toContainText('Hello');
    expect(messages[1]).toContainText('Hi there!');
    expect(messages[2]).toContainText('How can I help you?');
  });

  it('should call the handleSendMessage function when the Send
button is clicked', () => {
    const mockHandleSendMessage = jest.fn();
    render(<ChatInterface theme="dark"
```

```
handleSendMessage={mockHandleSendMessage} />);

    const sendButton = screen.getByRole('button', { name: 'Send' });
    fireEvent.click(sendButton);

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

  it('should call the handleFileUpload function when the Upload icon
is clicked', () => {
    const mockHandleFileUpload = jest.fn();
    render(<ChatInterface theme="dark"
handleFileUpload={mockHandleFileUpload} />);

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

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

  it('should fetch bot responses when the handleSendMessage function
is called', async () => {
    axios.post.mockResolvedValue({ data: { response: { result: {
content: 'Bot response' } } } });

    render(<ChatInterface theme="dark" />);

    const inputField = screen.getByRole('textbox');
    fireEvent.change(inputField, { target: { value: 'Hello' } });

    const sendButton = screen.getByRole('button', { name: 'Send' });
    fireEvent.click(sendButton);

    await waitFor(() => {
      expect(screen.getByText('Bot response')).toBeInTheDocument();
    });
  });
});
```

These test cases cover all the essential functionalities of the ChatInterface component, including rendering, user interactions, and API calls.