TESTING QUESTIONS

1. How do you test React components?

Answer:

Unit Testing with Jest and React Testing Library:

jsx

```jsx
// Button.test.js
import React from 'react';
import { render, fireEvent, screen } from
'@testing-library/react';
import '@testing-library/jest-dom';
import Button from './Button';

describe('Button Component', () => {
  test('renders button with correct text', () => {
    render(<Button>Click me</Button>);
    expect(screen.getByRole('button')).toHaveTextContent('Click
me');
  });

  test('calls onClick handler when clicked', () => {
    const handleClick = jest.fn();
    render(<Button onClick={handleClick}>Click me</Button>);

    fireEvent.click(screen.getByRole('button'));
    expect(handleClick).toHaveBeenCalledTimes(1);
  });

  test('is disabled when disabled prop is true', () => {
    render(<Button disabled>Click me</Button>);
    expect(screen.getByRole('button')).toBeDisabled();
  });

  test('shows loading state', () => {
    render(<Button loading>Click me</Button>);
    expect(screen.getByRole('button')).toBeDisabled();
    expect(screen.getByTestId('spinner')).toBeInTheDocument();
  });
});

// TodoApp.test.js
import React from 'react';
import { render, fireEvent, screen, waitFor } from
'@testing-library/react';
```

```
import userEvent from '@testing-library/user-event';
import TodoApp from './TodoApp';

describe('TodoApp', () => {
  test('adds new todo', async () => {
    const user = userEvent.setup();
    render(<TodoApp />);

    const input = screen.getByPlaceholderText('Add a new
todo...');
    const addButton = screen.getByText('Add');

    await user.type(input, 'New todo item');
    await user.click(addButton);

    expect(screen.getByText('New todo item')).toBeInTheDocument();
    expect(input).toHaveValue('');
  });

  test('toggles todo completion', async () => {
    const user = userEvent.setup();
    render(<TodoApp />);

    // Add a todo first
    const input = screen.getByPlaceholderText('Add a new
todo...');
    await user.type(input, 'Test todo');
    await user.click(screen.getByText('Add'));

    // Toggle completion
    const checkbox = screen.getByRole('checkbox');
    await user.click(checkbox);

    expect(checkbox).toBeChecked();
  });

  test('filters todos correctly', async () => {
    const user = userEvent.setup();
    render(<TodoApp />);

    // Add todos
    const input = screen.getByPlaceholderText('Add a new
todo...');
    await user.type(input, 'Active todo');
    await user.click(screen.getByText('Add'));

    await user.type(input, 'Completed todo');
    await user.click(screen.getByText('Add'));
```

```
    // Complete second todo
    const checkboxes = screen.getAllByRole('checkbox');
    await user.click(checkboxes[1]);

    // Filter active
    await user.click(screen.getByText('Active'));
    expect(screen.getByText('Active todo')).toBeInTheDocument();
    expect(screen.queryByText('Completed
todo')).not.toBeInTheDocument();

    // Filter completed
    await user.click(screen.getByText('Completed'));
    expect(screen.queryByText('Active
todo')).not.toBeInTheDocument();
    expect(screen.getByText('Completed
todo')).toBeInTheDocument();
  });
});
```

Integration Testing:

jsx

```
// ApiIntegration.test.js
import React from 'react';
import { render, screen, waitFor } from '@testing-library/react';
import { rest } from 'msw';
import { setupServer } from 'msw/node';
import UserList from './UserList';

// Mock API server
const server = setupServer(
  rest.get('/api/users', (req, res, ctx) => {
    return res(
      ctx.json([
        { id: 1, name: 'John Doe', email: 'john@example.com' },
        { id: 2, name: 'Jane Smith', email: 'jane@example.com' }
      ])
    );
  })
);

beforeAll(() => server.listen());
afterEach(() => server.resetHandlers());
afterAll(() => server.close());

describe('UserList Integration', () => {
  test('loads and displays users', async () => {
```

```
    render(<UserList />);

    expect(screen.getByText('Loading...')).toBeInTheDocument();

    await waitFor(() => {
      expect(screen.getByText('John Doe')).toBeInTheDocument();
      expect(screen.getByText('Jane Smith')).toBeInTheDocument();
    });
  });

  test('handles API error', async () => {
    server.use(
      rest.get('/api/users', (req, res, ctx) => {
        return res(ctx.status(500));
      })
    );

    render(<UserList />);

    await waitFor(() => {
      expect(screen.getByText(/error/i)).toBeInTheDocument();
    });
  });
});
```