Answer the following

1. **Describe snapshot testing. How does it work, and what benefits does it offer in testing React components?**

Ans –

Snapshot testing is a testing technique used to ensure that the output of a component has not changed unexpecte**dly**. It works by capturing a serialized representation (snapshot) of a component's rendered output and comparing it to previous snapshots.

If the output changes, the test fails, indicating a potential unintended change in the UI.

How Does Snapshot Testing Work?

1. The test framework (like Jest) renders the component and takes a snapshot of its output.
2. The snapshot is stored in a file inside a __snapshots__ directory.
3. On subsequent test runs, Jest compares the new output with the stored snapshot.
4. If differences are detected, the test fails, prompting the developer to review and update the snapshot if needed.

Example of Snapshot Testing in React

```
import React from "react";

import renderer from "react-test-renderer";

import Button from "./Button";


test("renders Button correctly", () => {

  const tree = renderer.create(<Button label="Click Me" />).toJSON();

  expect(tree).toMatchSnapshot();

});
```

Benefits of Snapshot Testing

- **Detects Unintended UI Changes** – Ensures UI elements remain consistent.
- **Easy to Implement** – Requires minimal test code.
- **Works Well for Components** – Ideal for stateless components or components with predictable outputs.
- **Fast and Automated** – Quickly verifies UI structure across different states.

When to Avoid Snapshot Testing?

- For highly dynamic component**s** that frequently change (e.g., components with random values or time-dependent rendering).
- When testing complex interactions (e.g., button clicks, API calls), where functional testing is more suitable.

2. **What is integration testing, and why is it valuable in the development of React applications?**

Ans –

Integration testing ensures that multiple components or modules work together correctly. Unlike unit tests (which focus on isolated functions/components), integration tests validate the interaction between components, APIs, and third-party services.

Why is Integration Testing Valuable in React Development?

1. **Ensures Components Work Together** – Validates how UI components interact with each other.
2. **Tests API and Data Flow** – Verifies that API calls return expected data and state updates properly.
3. **Identifies Edge Cases** – Detects issues that unit tests might miss.
4. **Improves Confidence Before Deployment** – Reduces the risk of breaking functionality after updates.

Example of Integration Testing in React

Using React Testing Library to test a component that fetches data from an API:

```
import { render, screen, waitFor } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import ProductList from "./ProductList";
import { server } from "./mocks/server"; // Mock API server

test("fetches and displays products", async () => {
  render(<ProductList />);

  // Wait for products to load
  await waitFor(() => expect(screen.getByText("Product 1")).toBeInTheDocument());

  // Simulate user interaction
  userEvent.click(screen.getByText("Add to Cart"));

  expect(screen.getByText("1 item in cart")).toBeInTheDocument();
});
```