**Snapshot testing** is a **feature of Jest**

**✅ Snapshot Testing: Origin and Purpose**

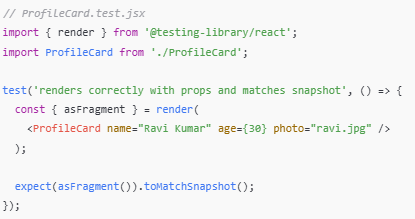
* **Library:** Jest provides the snapshot testing mechanism.
* **Purpose:** To capture and store the **rendered output** of a component in a text format (usually .snap files).
* **Usage:** If the component output changes later, Jest compares it to the saved snapshot and alerts you to differences.

🔍 How **React Testing Library Fits** In

**React Testing Library** is used to **render React components** in a test-friendly way. When combined with Jest, you can:

* Use React Testing Library’s render() to create the component DOM.
* Use Jest’s toMatchSnapshot() to save or compare that DOM output.





### When to Update Snapshots ?

If you **intentionally change** your component (e.g., UI design), you’ll need to update the stored snapshot using:

npm test -- -u

Brish Assignment : Component Rendering and Snapshot Testing  
**Objective:** Learn how to test component rendering and use snapshot testing.  
**Tasks:**

* Create a simple Profile Card component with props
* Write tests to verify the component renders correctly with different props
* Implement snapshot testing to capture the expected output
* Update the component and understand how to update snapshots

**Concepts Covered:**

* Component props testing
* Snapshot testing
* Test expectations and matchers
* Component updates and test maintenance

Working:

<https://jestjs.io/docs/snapshot-testing>

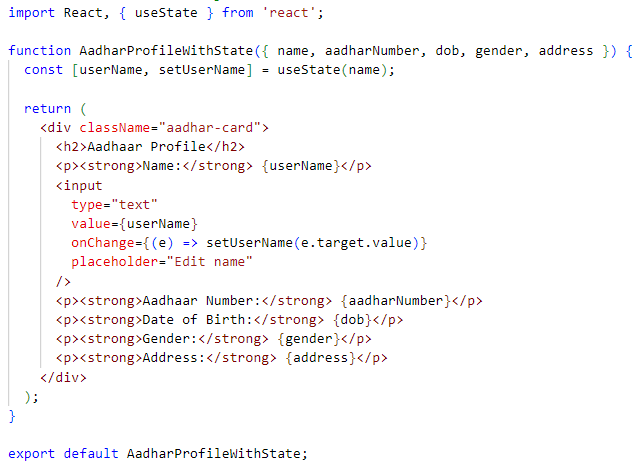


Fig: Build the component

  
Fig: test file for **AadharProfileWithState** component with a snapshot

NOTE: In above test.js file Dynamic interactions(like clicking buttons or input textbox) not reflected in snapshot unless you simulate it manually using renderer.root.find() or use React Testing Library

When you run  
 npm test AadharProfileWithState.test.js

Jest will generate a snapshot file i.e., **AadharProfileWithState.test.js.snap** automatically under the --snapshots-- folder(next to your **AadharProfileWithState.test.js** file)

This .snap file is stored to ensure **future renders of the component do not unintentionally change the structure or content**. If any visual/DOM-level output changes (e.g., adding/removing a field), Jest will alert you with a **snapshot mismatch** during testing.

Jest creates a **AadharProfileWithState.test.js.snap** file like this:

// Jest Snapshot v1

exports[`AadharProfileWithState Snapshot renders correctly with useState 1`] = `

<div

className="aadhar-card"

>

<h2>

Aadhaar Profile

</h2>

<p>

<strong>

Name:

</strong>

Indraja

</p>

<input

placeholder="Edit name"

type="text"

value="Indraja"

/>

<p>

<strong>

Aadhaar Number:

</strong>

1234-5678-9012

</p>

<p>

<strong>

Date of Birth:

</strong>

01-01-1990

</p>

<p>

<strong>

Gender:

</strong>

Female

</p>

<p>

<strong>

Address:

</strong>

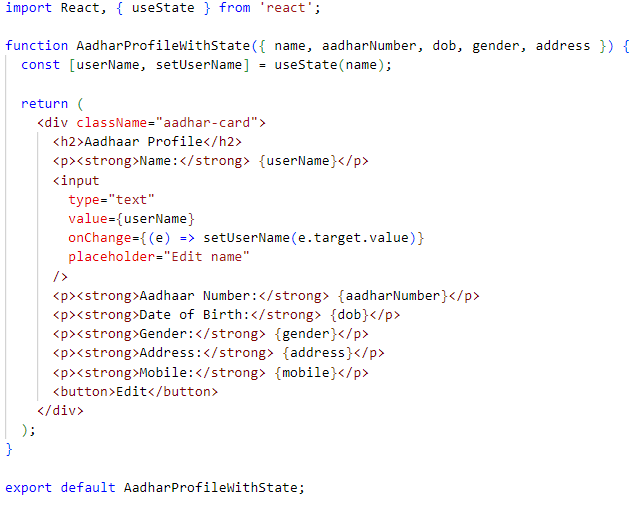
Kadapa, Andhra Pradesh

</p>

</div>

`;

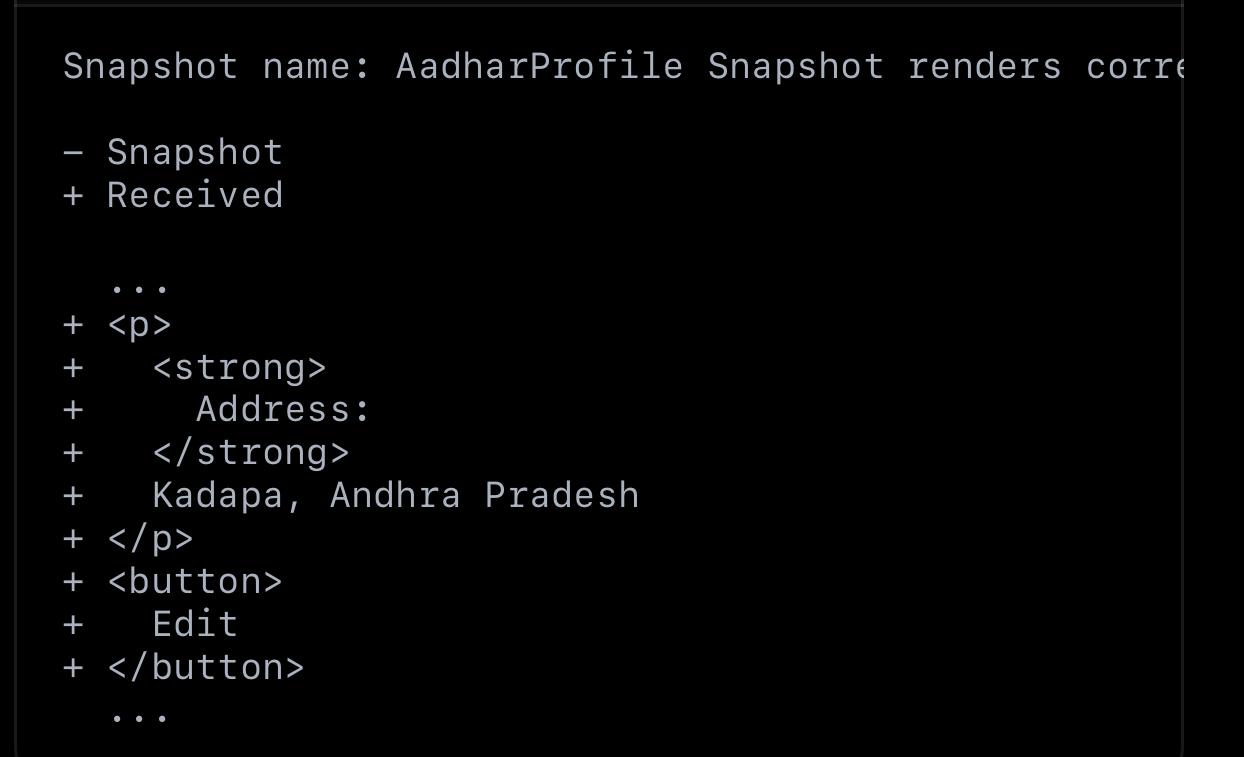
Let’s say you update **AadharProfileWithState.js** to add a new **Mobile** filed and an **Edit** button:





When you run  
 npm test AadharProfileWithState.test.js

Jest will fail the snapshot test and show:



To accept the new snapshot, If you’re intentionally updating the UI, run:

npm test -- -u or jest -u

**NOTE:** use -u flag to accept the new snapshot.

Finally,

Snapshot testing only checks the component’s rendered structure at a point in time.

Snapshot only captures initial render.

Hence developers prefer React Testing Library for realistic test