React JS

React Testing

Agenda

01 Types of Testing

02 Unit testing using Jest

03 Mocking using Jest

04 Snapshot Testing

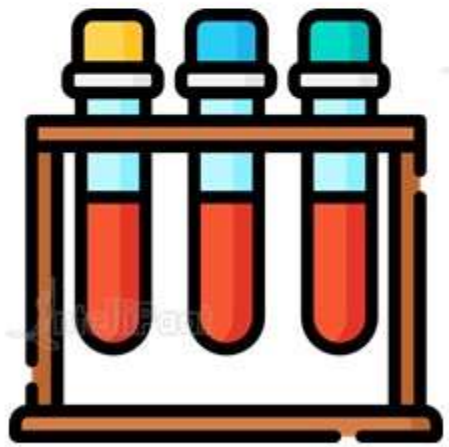
05 Testing Redux

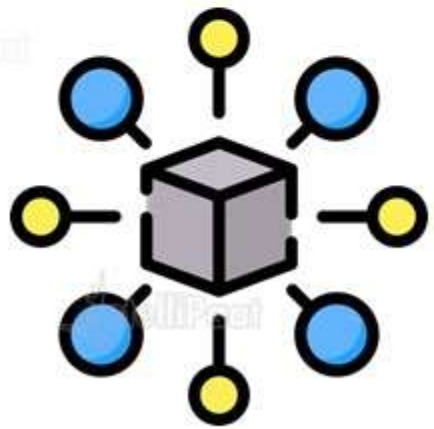
06 Testing Components

Types of Testing

Types of Testing

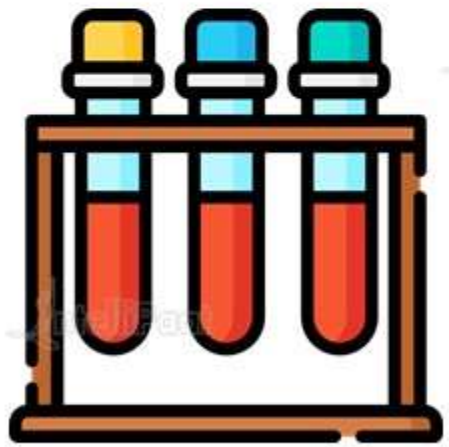
Testing is done while an application is being developed so that the newly created features can be verified to be working correctly

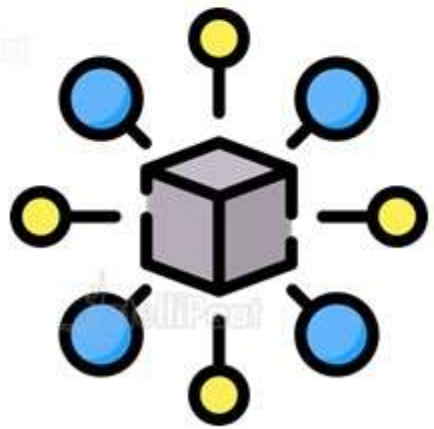




Types of Testing

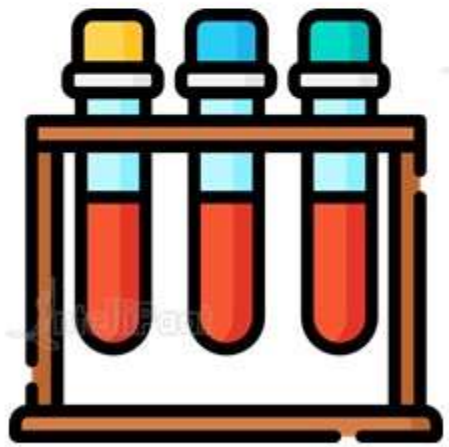
Since there are different levels at which we can test the code, there are also different kinds of tests we can write

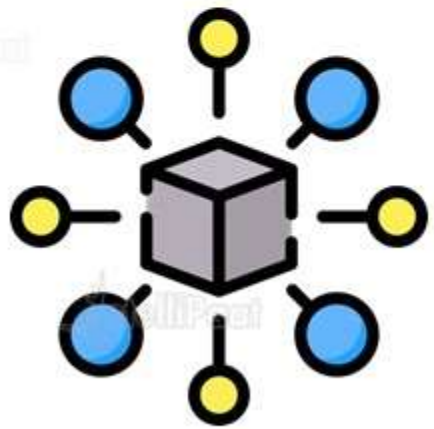




Types of Testing

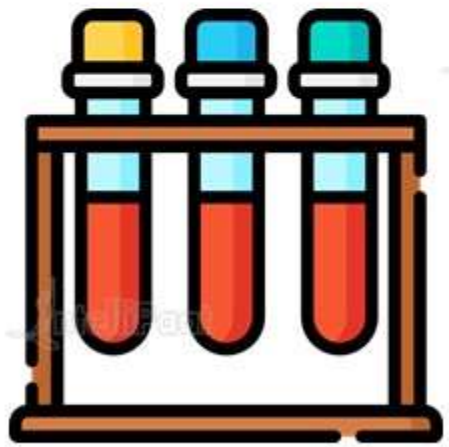
These different kinds of test provide us the ability to test individual units in isolation , integration multiple units or the user flow in the application

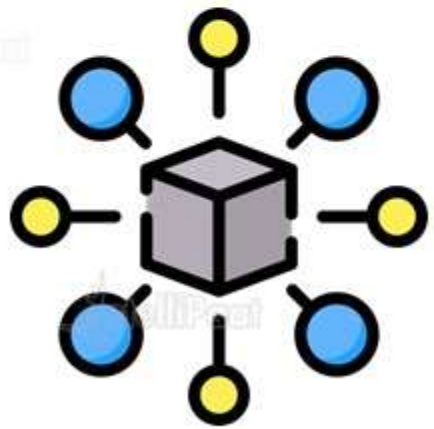




Types of Testing

These tests are called unit tests, integration tests and acceptance tests respectively

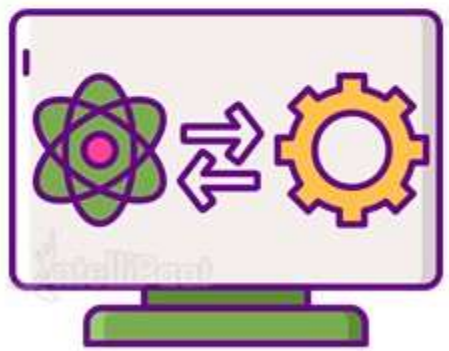




Unit testing using Jest

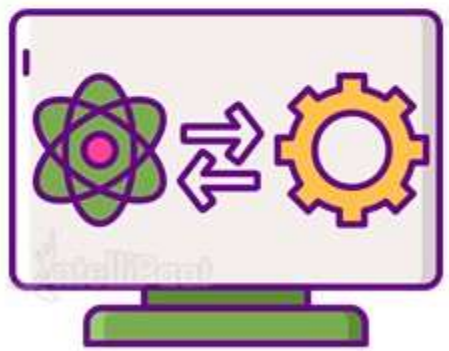
Unit testing using Jest

Unit test is the most common and also the fastest kind of test that we can write



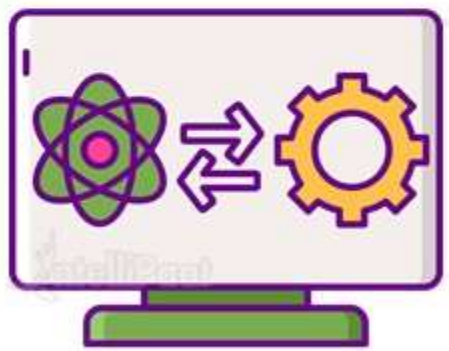
Unit testing using Jest

Unit tests allow us to check if an individual unit of code is being working as we expect it to, in isolation



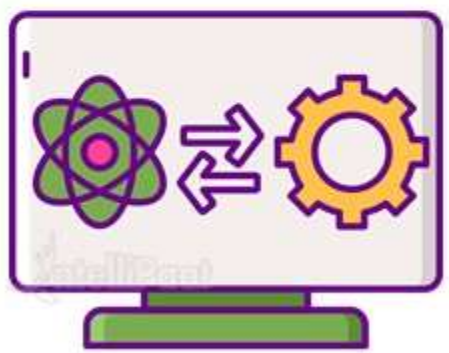
Unit testing using Jest

The reason we do unit test in isolation is so that, they are fast and faults in other units do not impact our tests



Unit testing using Jest

Some times it can be very difficult to isolate modules in cases like this we use a technique called mocking



Hands On: Unit Testing using Jest

Mocking using Jest

Mocking using Jest

Mocking is a technique which is used to provide a fake implementation for a unit so it can be used in another unit for testing purposes

Mocking using Jest

For example if we have to unit test a module that uses makes a request to an external service, then we need to mock it

Mocking using Jest

In mocking we simply would bypass making a request and return a hardcoded response

Mocking using Jest

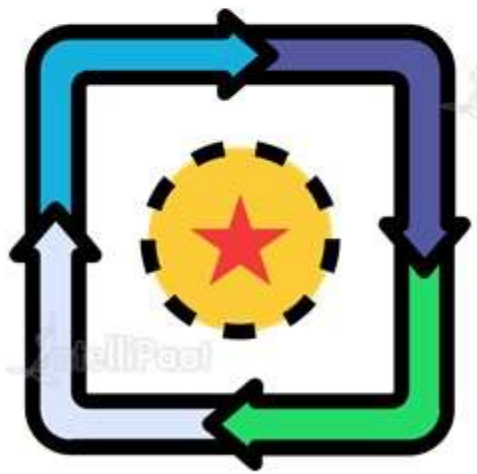
This makes our test fast, reliable and most importantly it makes our code deterministic

Hands On: Mocking

Snapshot Testing

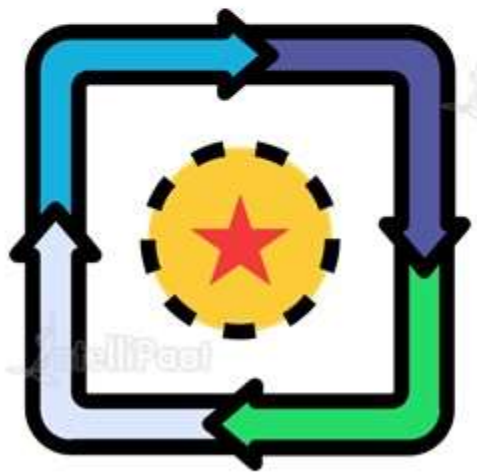
Snapshot Testing

Snapshot tests are a very useful tool whenever you want to make sure your UI does not change unexpectedly



Snapshot Testing

In Snapshot Testing we save a snapshot of current UI so that it can be used later

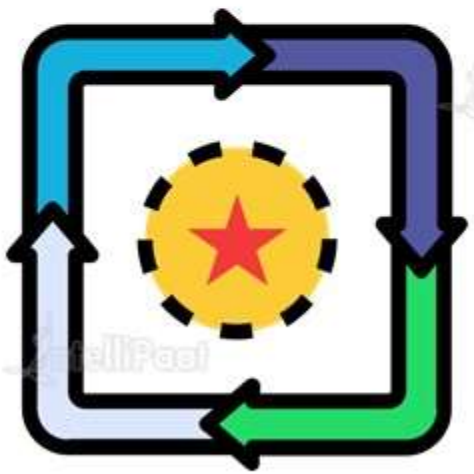


Snapshot Testing

When changes are made to the code but we do not wish to make changes to the UI we can use Snapshot Testing

Snapshot Testing

Snapshots can be updated easily when we do make changes and those snapshots are used for further testing

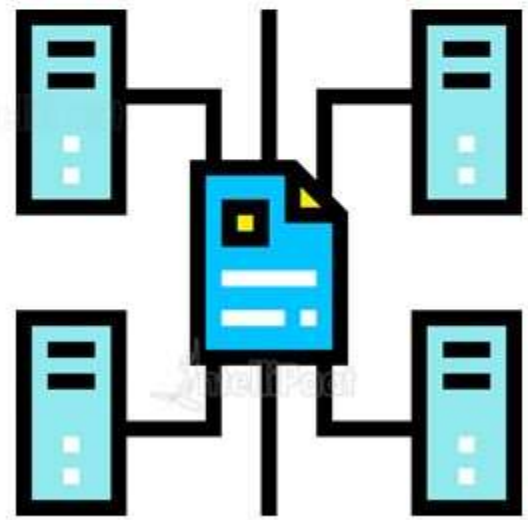


Hands On: Snapshot Testing

Testing Redux

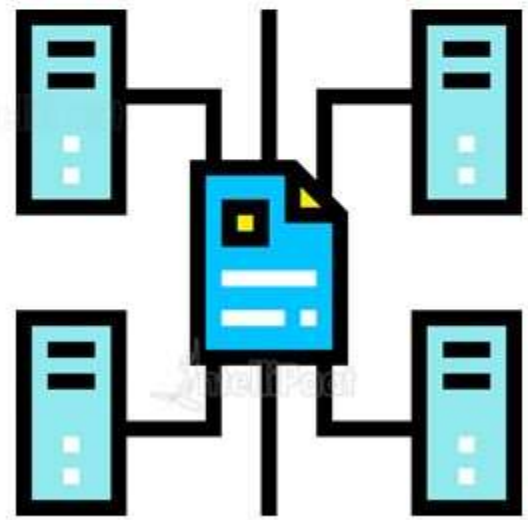
Testing Redux

Since Redux is just a state container they are very easy to write and run unit test



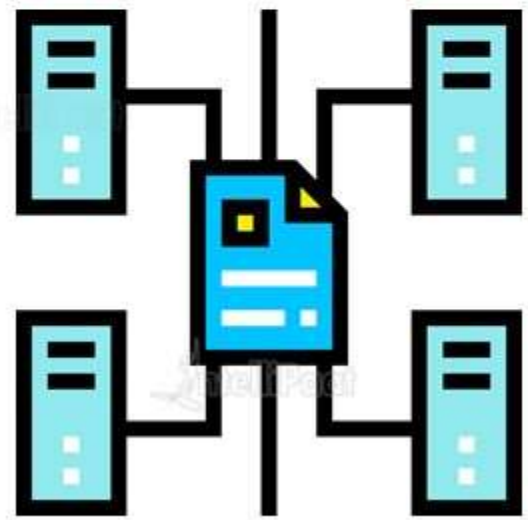
Testing Redux

For testing redux we can firstly test reducer by passing in state and asserting new state



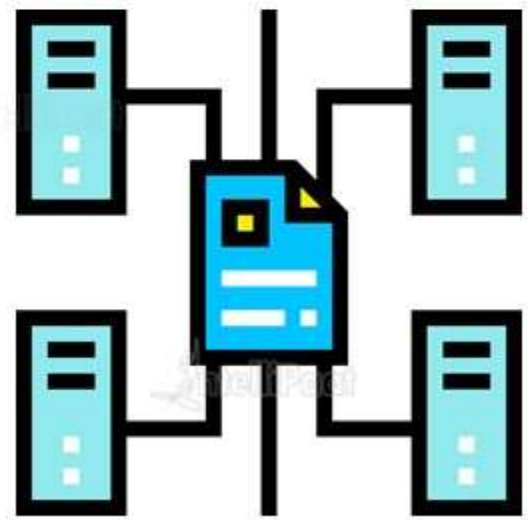
Testing Redux

After reducers we can also test our action creators by passing in values and asserting an object of specific shape



Testing Redux

Finally, we can test our stores by simply dispatching actions and checking state

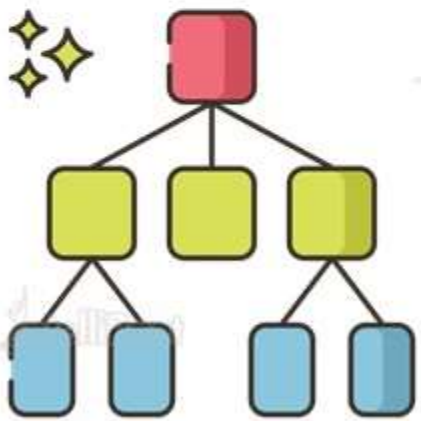


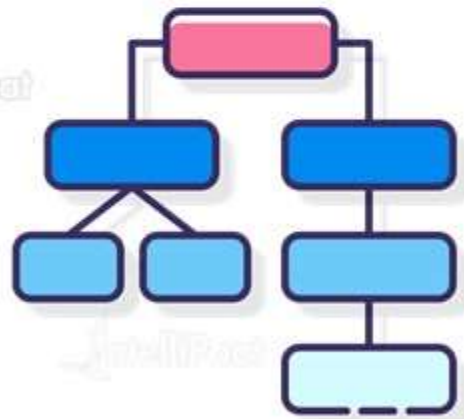
Hands On: Redux Testing

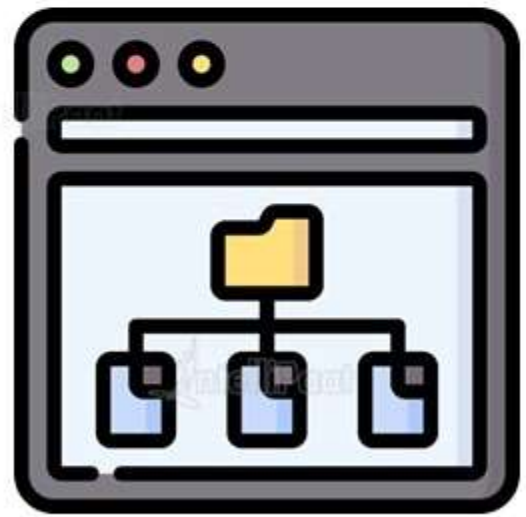
Testing Components

Testing Components

Since React Components work in the browser the cannot be tested completely in isolation

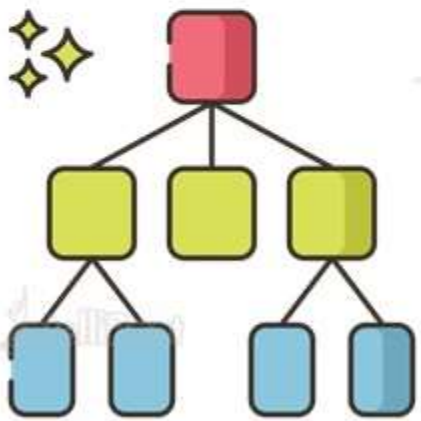


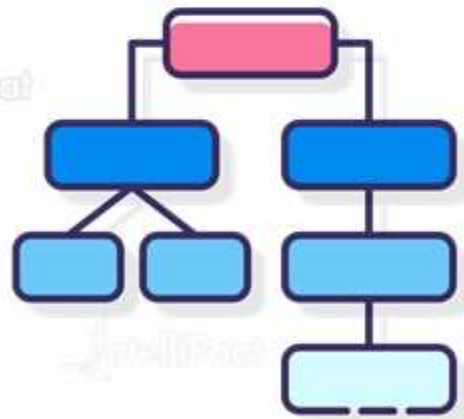


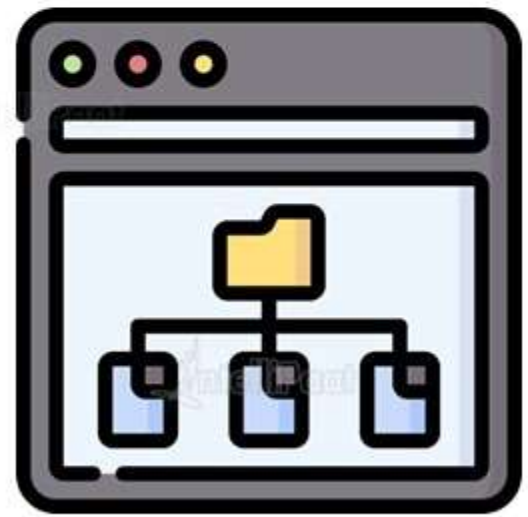


Testing Components

We have to use testing libraries and test the UI and user intreactions with the UI

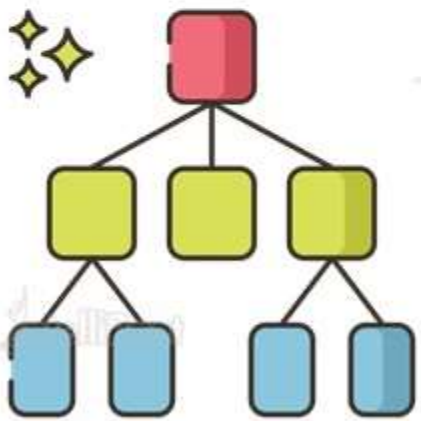


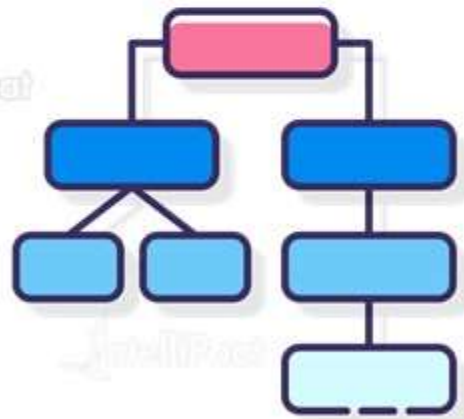


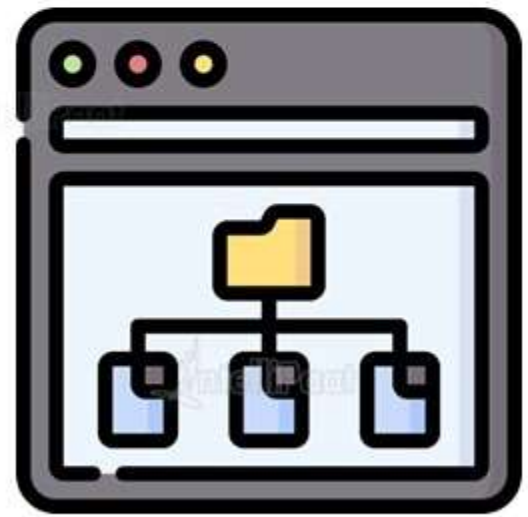


Testing Components

For this we can use a testing library like testing-library/react or enzyme

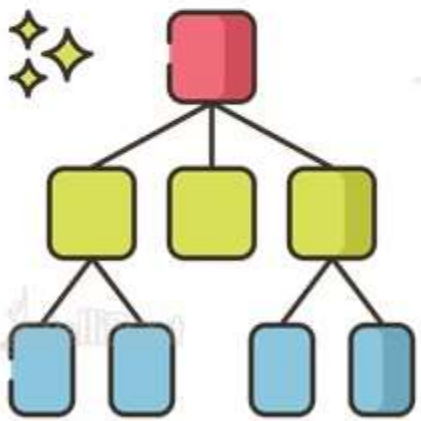


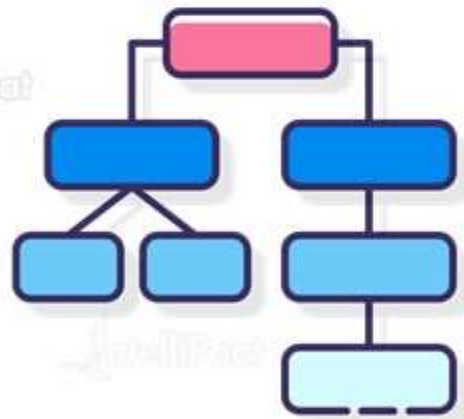


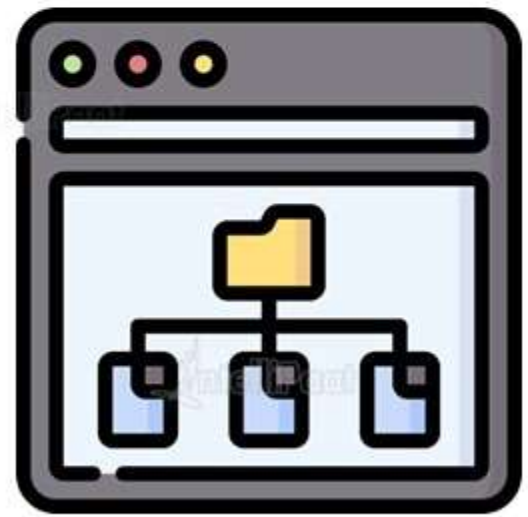


Testing Components

These libraries convert components to dom elements that we can interact with and make assertions







Hands On: Testing React Components

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor