# Unit Testing in React: Isolation, Mocking, and Jest

## 1. Understanding Need for Isolation in Testing

Isolation in testing means ensuring that a test focuses solely on the unit of code being tested—without interference from external dependencies such as APIs, databases, or unrelated modules. This is important because:  
  
• It ensures consistency: Isolated tests are not affected by changes in unrelated code.  
• It improves reliability: Tests won’t fail due to external issues like API downtime.  
• It simplifies debugging: Failures point directly to the unit in question.  
• It speeds up testing: Mocked or stubbed dependencies run faster than real implementations.

## 2. Understanding the Concept of Mocking

Mocking is the practice of replacing real functions or modules with dummy implementations that simulate behavior. Mocks are used to:  
  
• Simulate API calls or responses.  
• Verify how functions are called (arguments, call count, etc.).  
• Return controlled data to test edge cases or specific logic flows.  
  
Mocking is essential in unit testing to keep tests isolated and to test modules independently.

## 3. Using Jest for Unit Testing and Mocking

Jest is a powerful testing framework developed by Meta (Facebook) that supports unit testing and mocking out-of-the-box.   
  
Key Features:  
• Zero configuration setup.  
• Built-in mocking and assertion APIs.  
• Snapshot testing and code coverage tools.  
  
Examples:  
  
Mocking a module:  
```  
jest.mock('axios');  
axios.get.mockResolvedValue({ data: [...] });  
```  
  
Testing a function with Jest:  
```  
test('adds 2 numbers', () => {  
 expect(sum(2, 3)).toBe(5);  
});  
```  
  
Jest works well with React Testing Library and Enzyme to test components effectively.