



---

# TESTING TOOL DOCUMENTATION

---



NOVEMBER 11, 2025

BANGLADESH UNIVERSITY OF PROFESSIONALS

## Tools Available:

For unit and integration testing in a **Next.js + React** project, the top competitors are:

Tool	Type	Pros	Cons
<b>Jest</b>	Unit & Integration	Zero-config, snapshot testing, built-in mocks, coverage reporting, works for frontend & backend, fast	Not full E2E, needs Cypress/Playwright for browser tests
<b>Mocha + Chai + Sinon</b>	Unit & Integration	Flexible, widely used, works with many libraries	Requires configuration, separate mocking library, slower setup, no snapshot testing
<b>Vitest</b>	Unit & Integration	Modern, fast, Vite/Next.js optimized, Jest-compatible API	Smaller community, fewer plugins, less mature

## What is Jest?

- Jest is a **JavaScript testing framework** maintained by Meta (Facebook).
- It is primarily used for **unit testing**, **integration testing**, and **snapshot testing** of JS/TS code.
- Works **seamlessly with Node.js**, React, Next.js, and other JS frameworks.
- Provides a **zero-configuration setup** for most projects.

ShebaBondhu uses **Next.js as backend**, and React for frontend:

Feature	Jest Advantage for ShebaBondhu
<b>Integration with Next.js</b>	Jest works smoothly with Next.js API routes, serverless functions, and React components.
<b>Full-stack JS testing</b>	You can test backend (API endpoints) and frontend (React components) in the same framework.
<b>Snapshot testing</b>	Useful for UI regression testing — you can detect accidental changes in components like dashboards, forms, charts.
<b>Mocking</b>	Jest has built-in mocks for functions, modules, and timers. Helps test parts of the system independently.
<b>Speed</b>	Runs tests in parallel, fast for large JS codebases.
<b>Community &amp; support</b>	Huge community: tons of tutorials, plugins, and integration guides for Next.js and React.

## Why Jest is the best choice for ShebaBondhu

- **Stack is Next.js:** Jest has official support and works out-of-the-box.
- **Team size:** Fewer dependencies; zero-config reduces setup time.
- **You want unit + integration tests for both frontend/backend :** Jest covers both.
- **Snapshot testing for UI components :** Helps catch visual regressions in dashboards, forms, interactive elements.
- **Community support :** Lots of plugins for mocking databases, API calls, timers, etc.

Jest is **fast, easy, full-featured**, and a perfect fit for a **Next.js-based ShebaBondhu project**. For unit & integration testing, Jest alone is sufficient.

## HOW TO IMPLEMENT JEST?

```
PS D:\Academic\year4sem1\STM lab\jest\jest> mkdir shebabondhu-demo
>> cd shebabondhu-demo
>>

Directory: D:\Academic\year4sem1\STM lab\jest\jest

Mode                LastWriteTime         Length Name
----                -
d-----         11/12/2025   6:04 AM                shebabondhu-demo

>> npm init -y
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo>
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo> npm init -y
>>
Wrote to D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo\package.json:

{
  "name": "shebabondhu-demo",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  }
}
```

Installed jest mocha chai altogether

```
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo> npm install express
>> npm install --save-dev jest mocha chai
>>

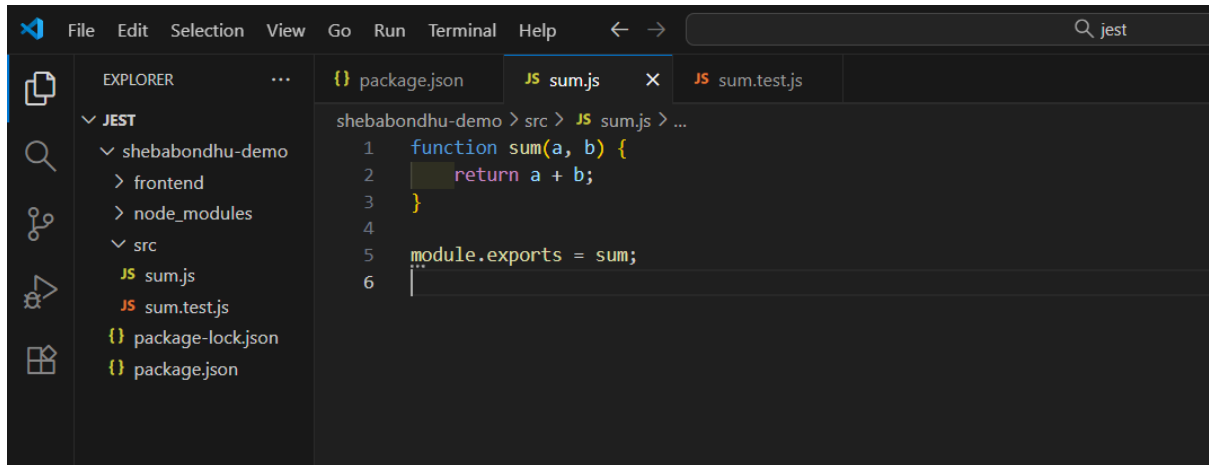
added 68 packages, and audited 69 packages in 6s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and
tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

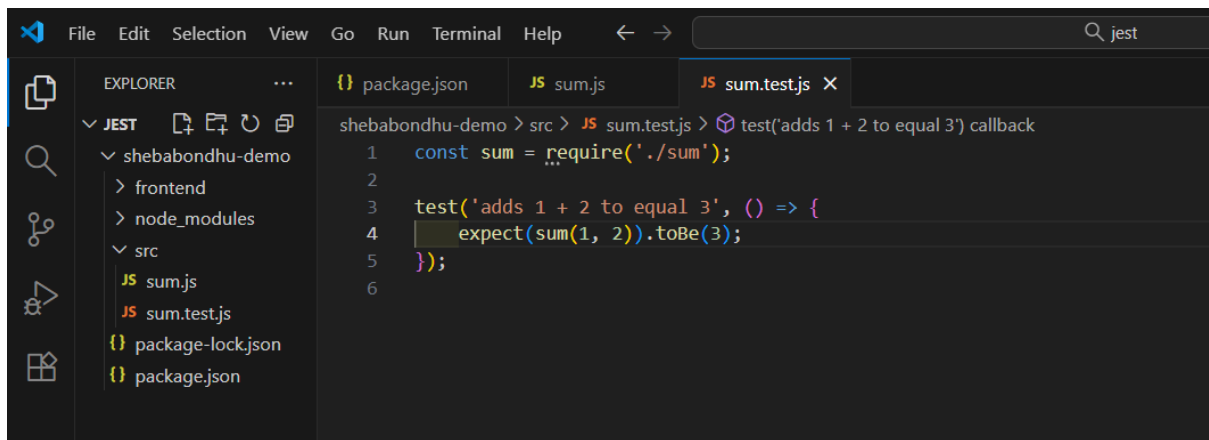
added 318 packages, and audited 387 packages in 1m

69 packages are looking for funding
  run `npm fund` for details
```



The image shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure for 'shebabondhu-demo', including 'frontend', 'node\_modules', and 'src'. The 'src' folder is expanded, showing 'sum.js' and 'sum.test.js'. The Editor view on the right shows the content of 'sum.js'.

```
shebabondhu-demo > src > JS sum.js > ...
1  function sum(a, b) {
2    return a + b;
3  }
4
5  module.exports = sum;
6
```



The image shows the Visual Studio Code interface with the 'sum.test.js' file open in the Editor. The Explorer sidebar on the left shows the same project structure as the previous image. The Editor view on the right shows the content of 'sum.test.js'.

```
shebabondhu-demo > src > JS sum.test.js > test('adds 1 + 2 to equal 3') callback
1  const sum = require('./sum');
2
3  test('adds 1 + 2 to equal 3', () => {
4    expect(sum(1, 2)).toBe(3);
5  });
6
```

```
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo> npm test
>>

> shebabondhu-demo@1.0.0 test
> jest

PASS src/sum.test.js (5.806 s)
  ✓ adds 1 + 2 to equal 3 (3 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        7.343 s
Ran all test suites.
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo>
```

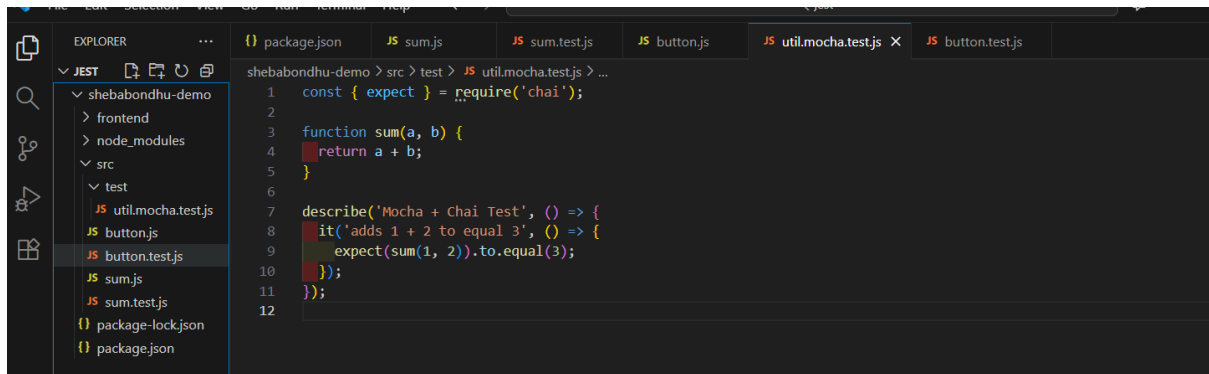
## HOW TO IMPLEMENT MOCHA+CHAI?

```
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo> npm install mocha chai --save-dev
>>

up to date, audited 387 packages in 4s

69 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



The screenshot shows the VS Code interface. The Explorer sidebar on the left shows the project structure: shebabondhu-demo > frontend > node\_modules > src > test. The test directory contains several files: util.mocha.test.js, button.js, button.test.js, sum.js, and sum.test.js. The main editor window displays the content of util.mocha.test.js, which includes a require for 'chai', a sum function, and a Mocha test suite with an 'it' block for 'adds 1 + 2 to equal 3'.

```
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo> cd src
PS D:\Academic\year4sem1\STM lab\jest\jest\shebabondhu-demo\src> npx mocha
>>

Mocha + Chai Test
  ✓ adds 1 + 2 to equal 3

1 passing (6ms)
```

Feature	Jest	Vitest	Mocha
Speed	Fast, but slower in large projects	Super fast (uses Vite's dev server + hot reload)	Slow (runs sequentially)
Ease of setup	Zero config	Zero config with Vite projects	Manual config + Chai required
Mocking	Built-in (jest.mock)	Built-in (vi.mock)	Needs extra library (Sinon)
TypeScript support	Good (with Babel or ts-jest)	Native	Requires setup
Integration testing	Possible but verbose	Simpler with Vite ecosystem	Manual setup
Ecosystem support	Huge (older, stable)	Growing fast (modern, active devs)	Legacy (longstanding)
Best for	Node + React/Next apps	Vite/React/TS modern stacks	Legacy JS or minimal Node scripts