41. Testing & Deployment in React

AGENDA

- Testing
  - Types of tests
  - Libraries used
  - Implement unit tests
  - TDD (Test Driven Development)
- Deployment
  - Netlify / Surge / Vercel

TESTING :

Process of evaluating & validating application's functionality to ensure it meets the desired perf & quality standards.

Way to test

- manual testing ← QA / SDET
- automated testing

Areas to test

- Functional / Integration
- Unit
- Snapshot → it kinda clicks image
- e2e

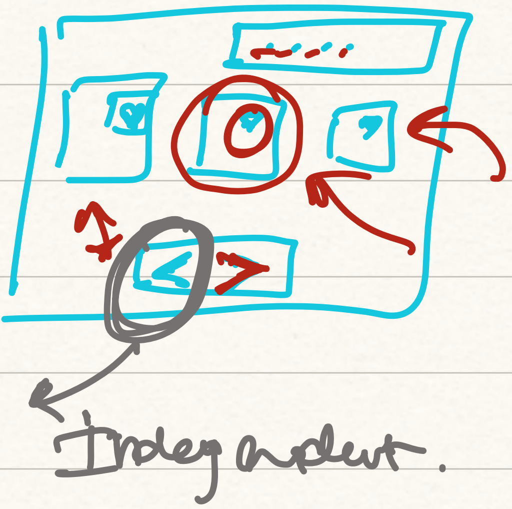
copy of the component.
After you change anything,
it matches that snapshot.

- Security
- Performance
- Stress / Chaos
- Regression

Unit → Verifying smallest unit (component) of the app in isolation

Functional/Integration

How components interact with eachother to achieve specific functionality.



End-to-end (e2e) evaluate entire application.

Stress testing / Chaos

[NFR tests] /

10M active users

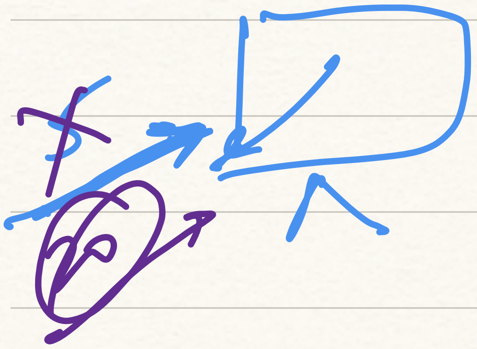
Scale

Load test

1K ✓ 4000 users

↓

Break



Mimic traffic

Spawn users

- Identify breaking point.

Performance test → How app behaves in different devices, network Conditions.
∟ Judge on UX.

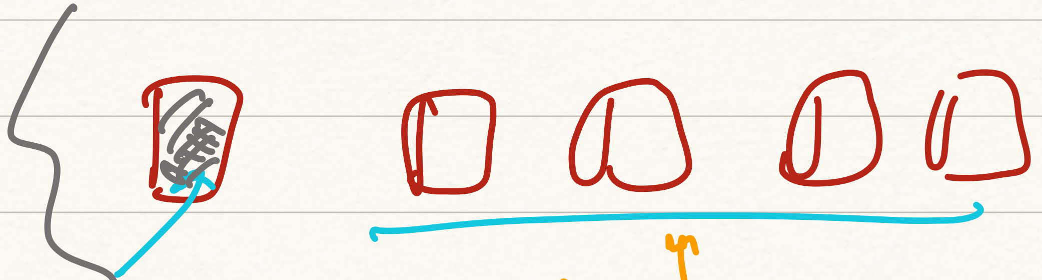
Security test → Identifying vulnerabilities/ weakness in app.

Regression test

↓

Run entire application even after

Code at one place



↑

may be affected

Small code change :(

→

React testing library (rtl)

↳ gives us access to VDOM, events

→ JEST

↳ To actualy run the test

test, describe, expect (common function)

→ {Render, Screen} from '@testing-library/react'

Used to render the React Component into VDOM for testing.

'Screen' object is used to access output of the rendered Component.

```
Const LinkElement = Screen.getByText(/learn react/i);
Expect (linkElement).toBeInDocument()
```

↓

expect fn is from Jest to test whether LinkElement is present in the document or not.

Test Fail → If linkElement not found

- npm run test - - -u

↑
To update snapshot

TDD → Test Driven Development

tests first, feature later.

methodology →

① First write all test cases

↓

then write component

② Refactor all your test cases

↳ optimize your component.