AUTOMATED TESTING FOR WEB APP

By Nongnooch Roongpiboonsopit

August 14th, 2021

https://www.linkedin.com/in/nongnoochr/

IEEE Spectrum / How the Boeing 737 Max Disaster Looks to a Software Develop...

Type to search

FEATURE | AEROSPACE

HOW THE BOEING 737 MAX DISASTER LOOKS TO A SOFTWARE DEVELOPER

Design shortcuts meant to make a new plane seem like an old, familiar one are to blame

Source: https://spectrum.ieee.org/how-the-boeing-737-max-disaster-looks-to-a-software-developer

AGENDA

- What is software testing and why it is important?
- Types of software testing
- How to apply Test Pyramid for a Web App
- Questions

WHAT IS SOFTWARE TESTING AND WHY?

- **Software testing** is an investigation conducted to provide stakeholders with information about **the quality of the software product or service under test**.
  - Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

- Test techniques include the process of executing a program or application with **the intent of finding failures** and **verifying that the software product is fit for use**.

Resource: https://en.wikipedia.org/wiki/Software_testing

There are multiple types of software testing but …

We will only cover

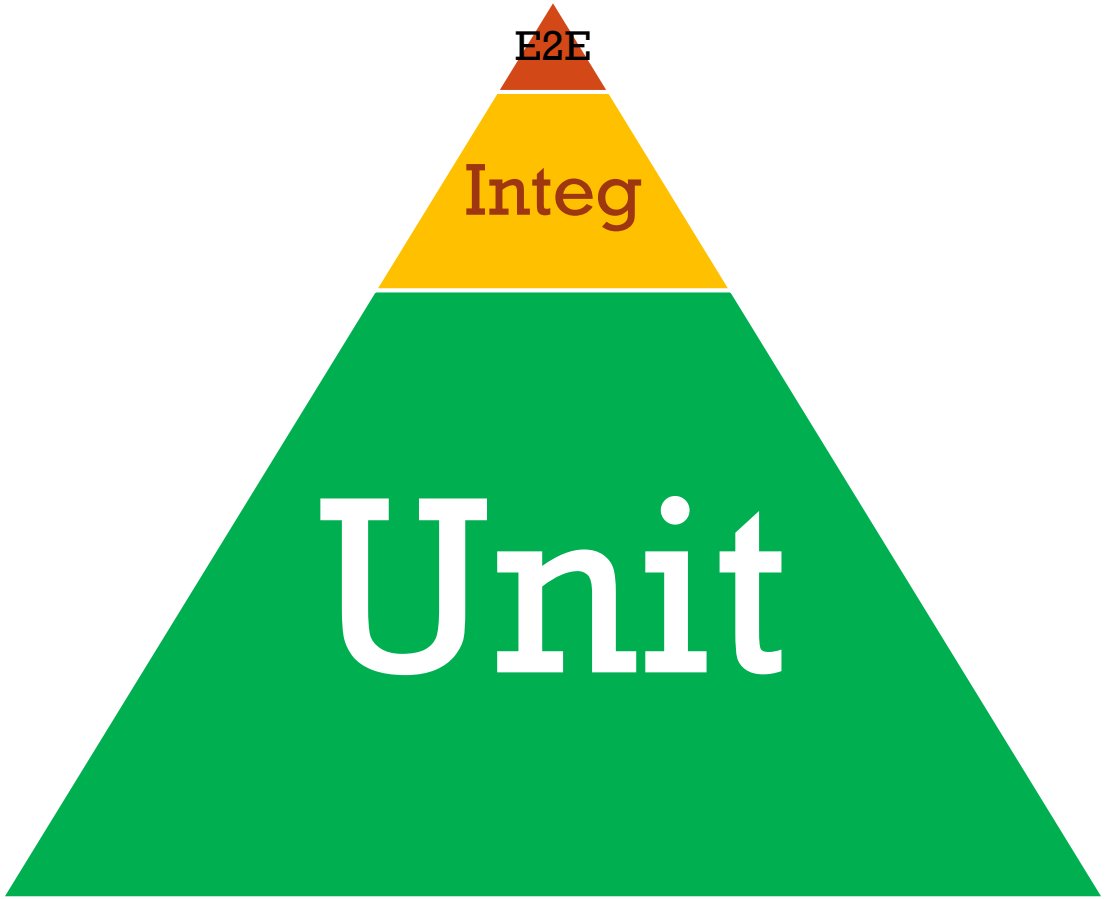
AUTOMATED TESTING

focusing on testing

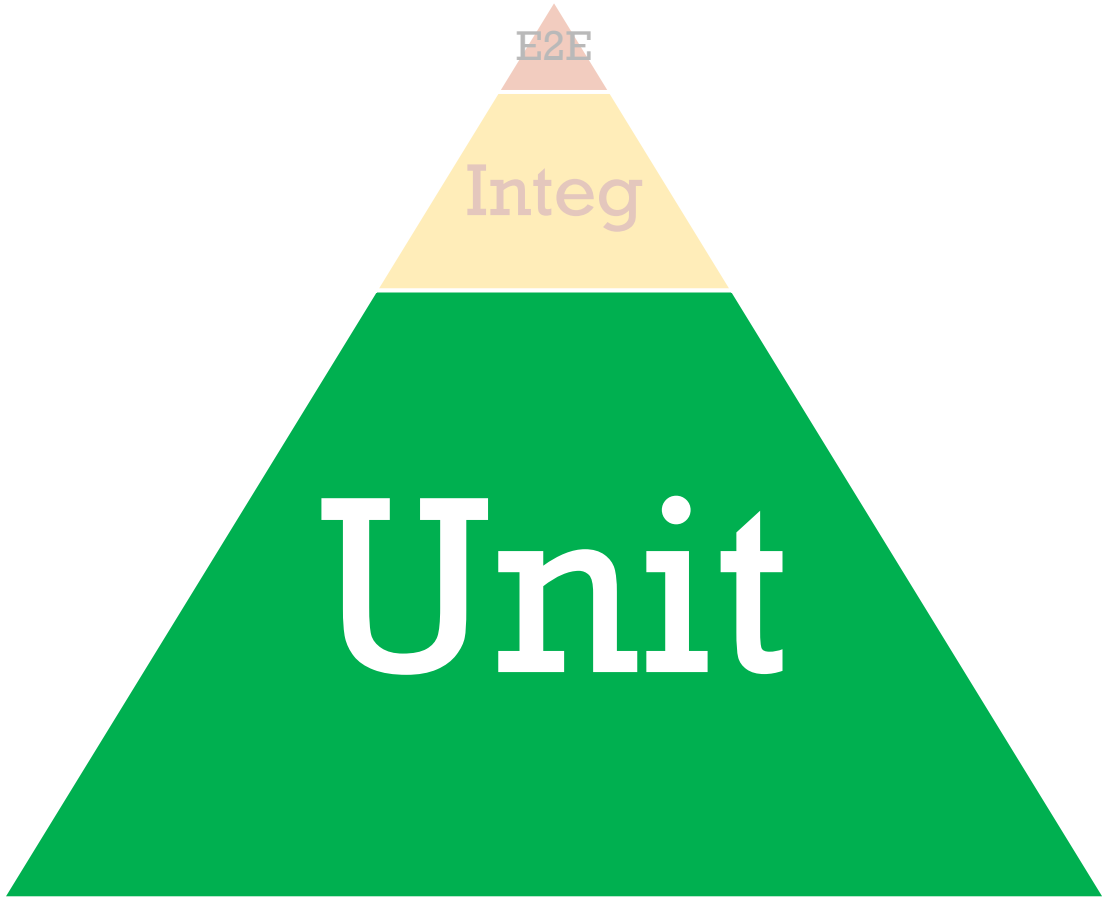
WEB Applications

in this talk

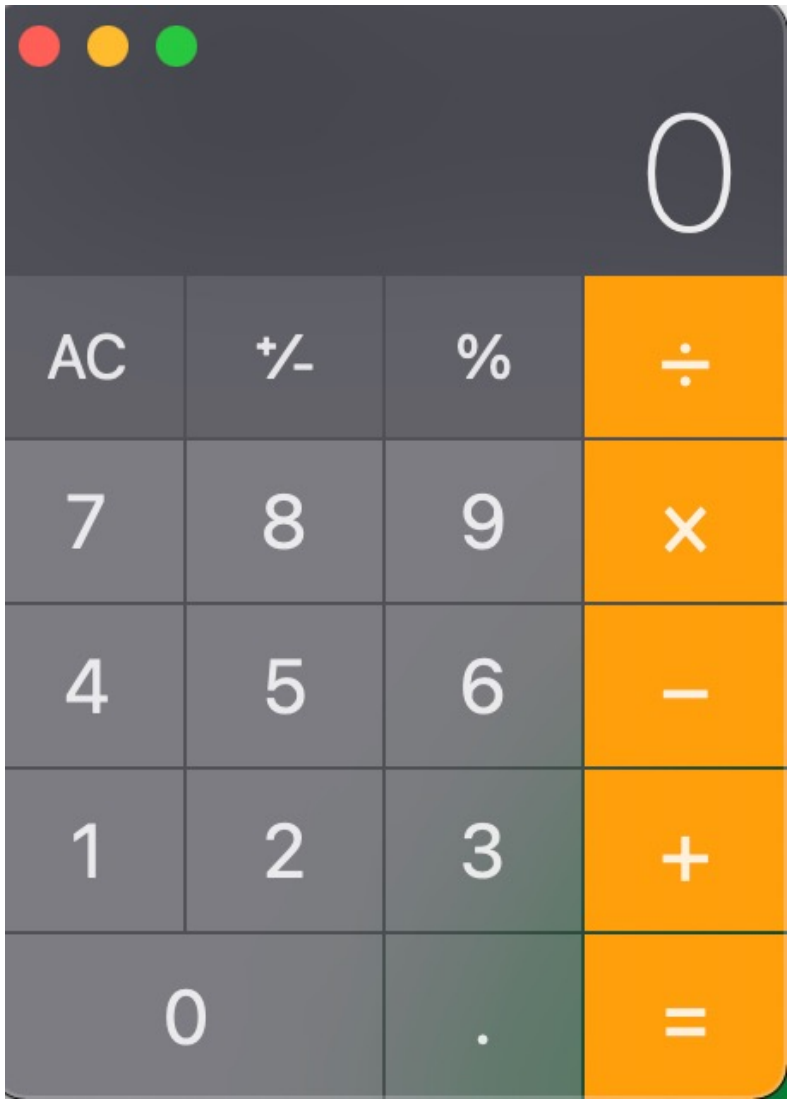
TYPES OF AUTOMATED TESTING



UNIT TESTING



- Cover individual units of code
- Can be tested in isolation
- Typically performed by the developer during development
- **Fast and easy to debug**
  - Tests at this level should be maximized (~70%)



EXAMPLE: UNIT TEST CASES FOR CALCULATOR APP

Possible unit test cases

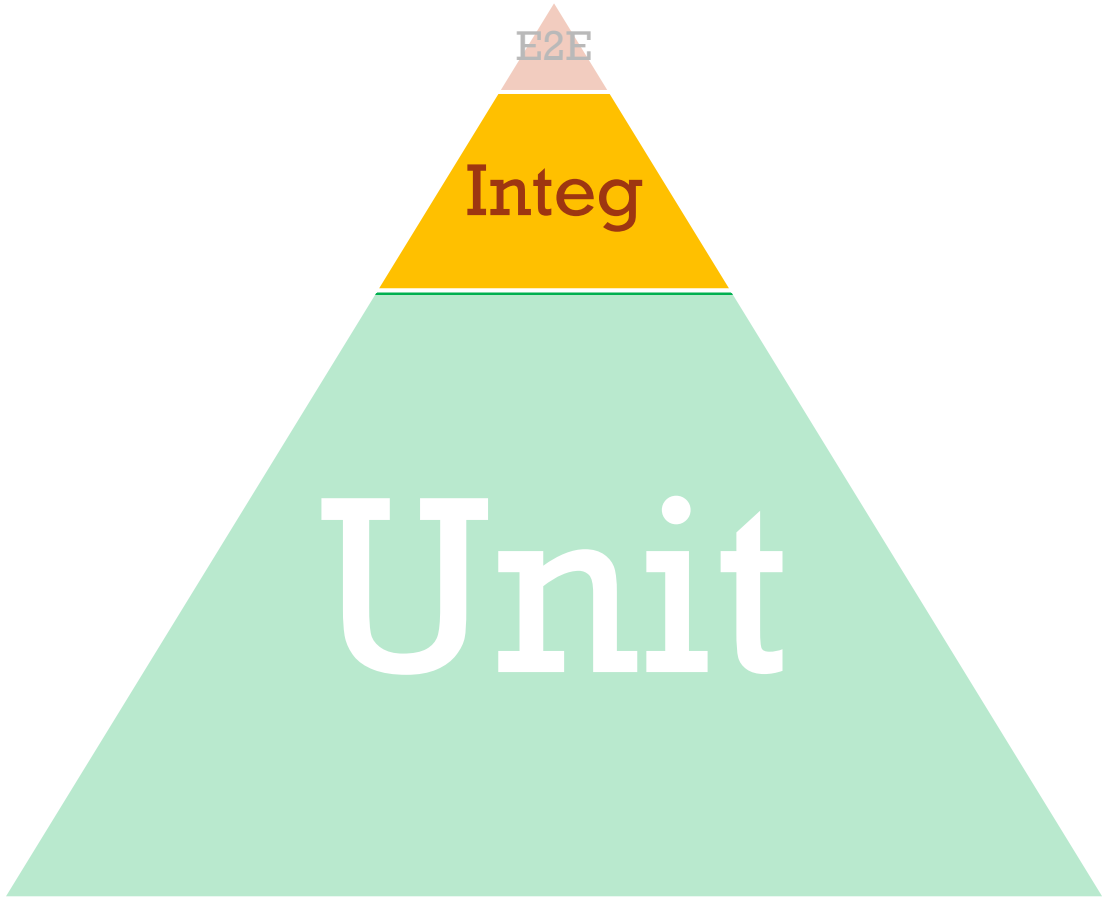
no UI

- Test the math operations (+, -, x, /)
- Test the AC function
- Test the +/- function
- Test the % function

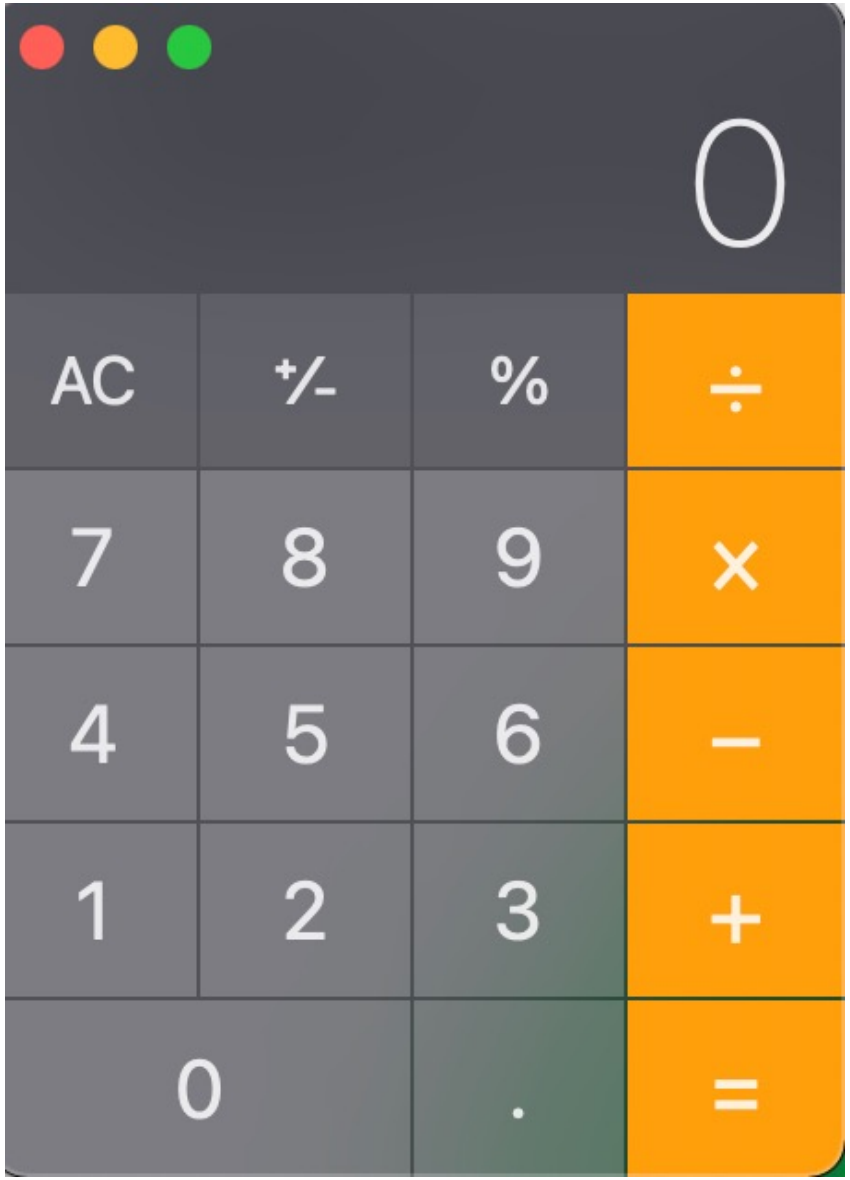
UI Components

- Button component should render an appropriate symbol

INTEGRATION TESTING



- Process of testing the interface between two software units or modules
- Focus on determining the correctness of the interface
  - The purpose of the integration testing is to expose faults in the interaction between integrated units
- **More costly than unit testing** but less expensive than E2E testing
  - There should be around 20% of tests added to this level

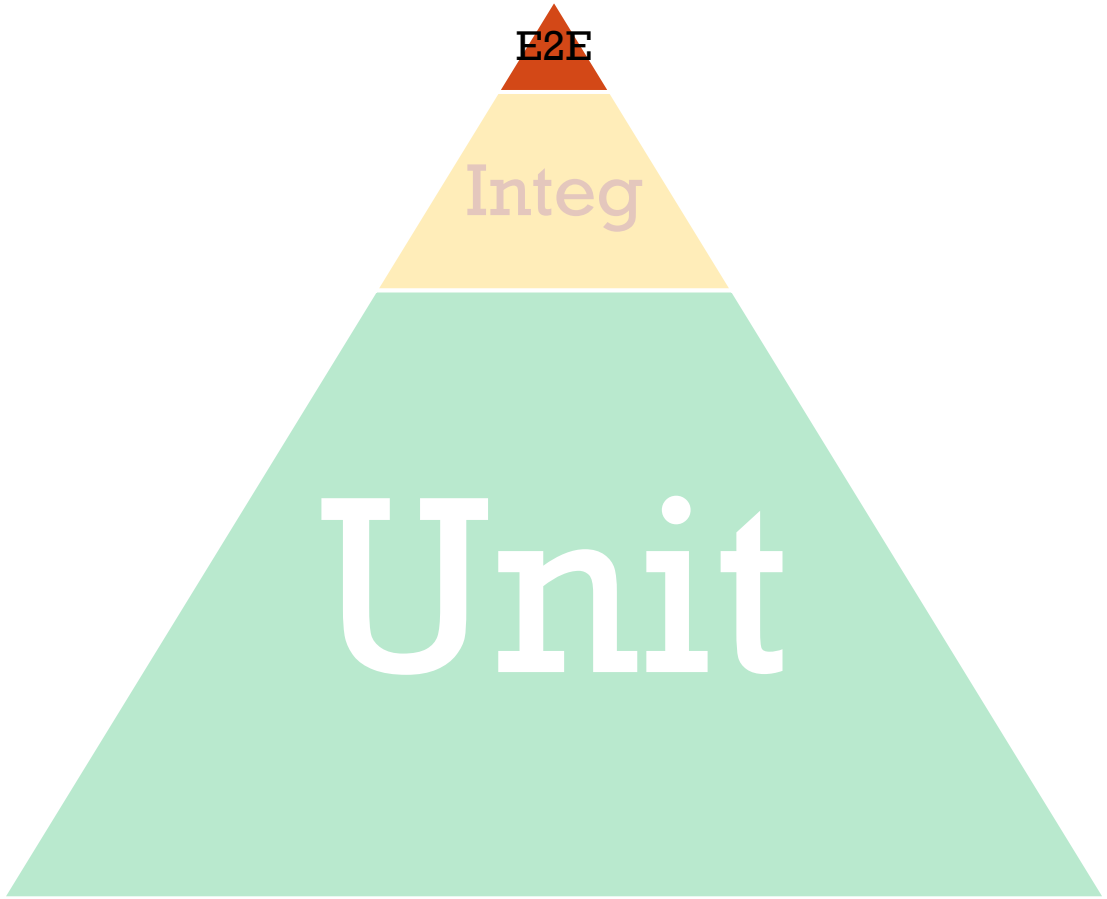


EXAMPLE: INTEG TEST CASES

Possible integ test cases

- Clicking ‘7’, ’+’, ‘8’, ‘=‘ and verify that the display shows ‘15’
- Clicking the ‘AC’ button should reset the display

END-TO-END (E2E) TESTING



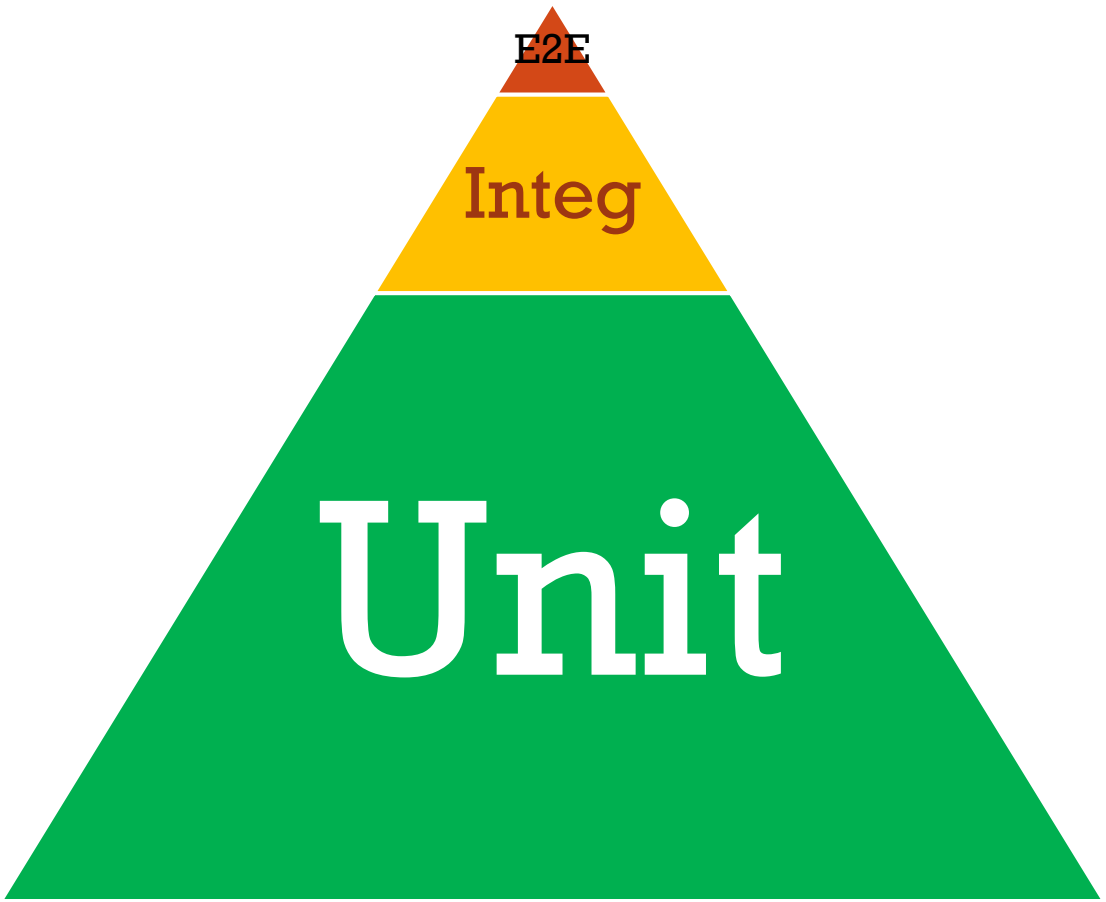
- Simulate a user level experience across the full stack of a software product
  - These tests are highly valuable to implement as they offer assurance that real users are having a smooth bug free experience, even when new commits are pushed
- E2E tests can be expensive up front to capture and record the user flow sequence
  - Tests at this level should saved to capture *design cases and requirements*
  - They can be used to replace manual testing

EXAMPLE: E2E TEST CASES

Possible E2E test cases

- Select a restaurant should show up menu of the selected restaurant
- Click the 'Order NOW' button should place an order in the cart to the server and return an appropriate response to the user

ADHERE TO TEST PYRAMID FOR QUALITY AND TESTABILITY



Slow, Flaky

Fast, Easy to debug

THERE ARE SO MANY JS TESTING LIBRARIES.

WHICH ONE SHOULD I USE?

Frameworks

- Jest, Mocha, Protractor, Jasmine, QUnit, etc.

Browser Controller Libraries

(for E2E Testing)

- Playwright, Puppeteer, FuncUnit etc.

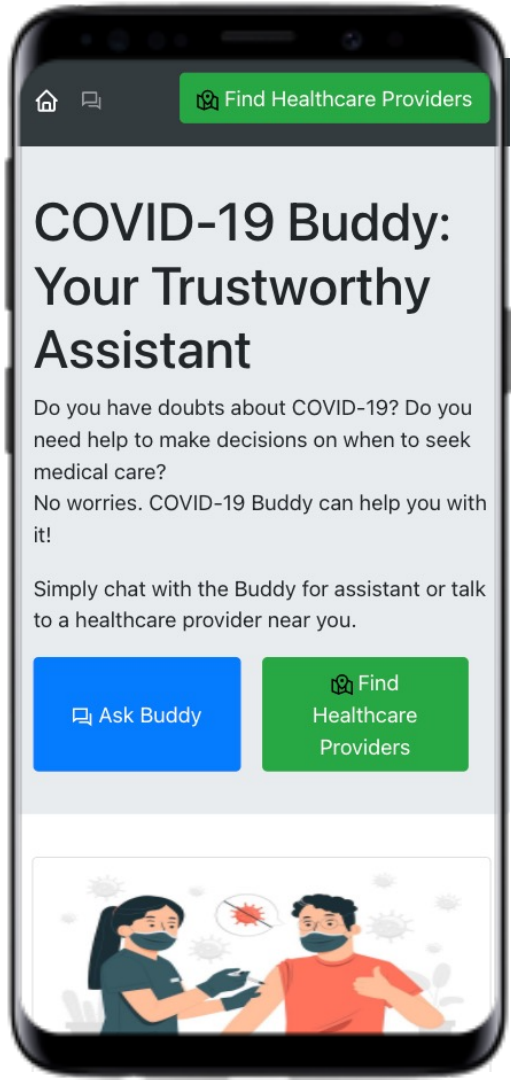
Above are available JS testing libraries (nodejs) as of August 12th, 2021


LET'S APPLY WHAT WE LEARNED TOGETHER!

COVID-19 Healthcare App Challenge

WINNER Honorable Mentions

Project in devpost



+

IQVIA™ HealthCare Locator

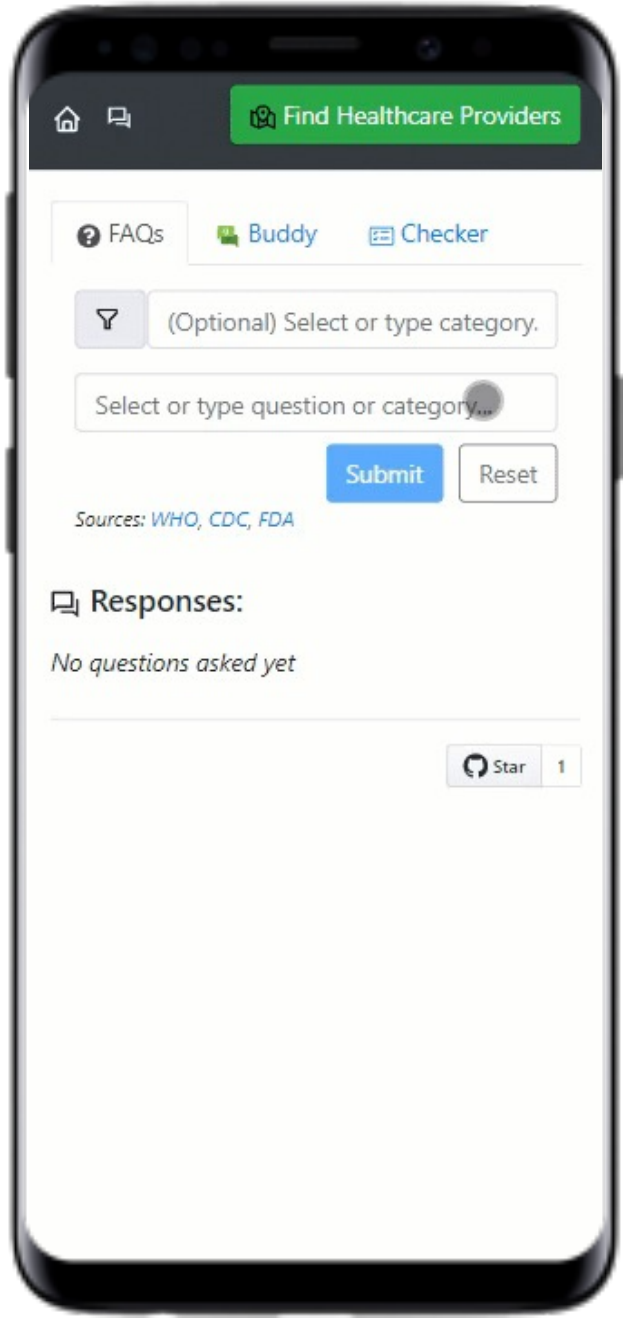
Website: https://covid-buddy.herokuapp.com/

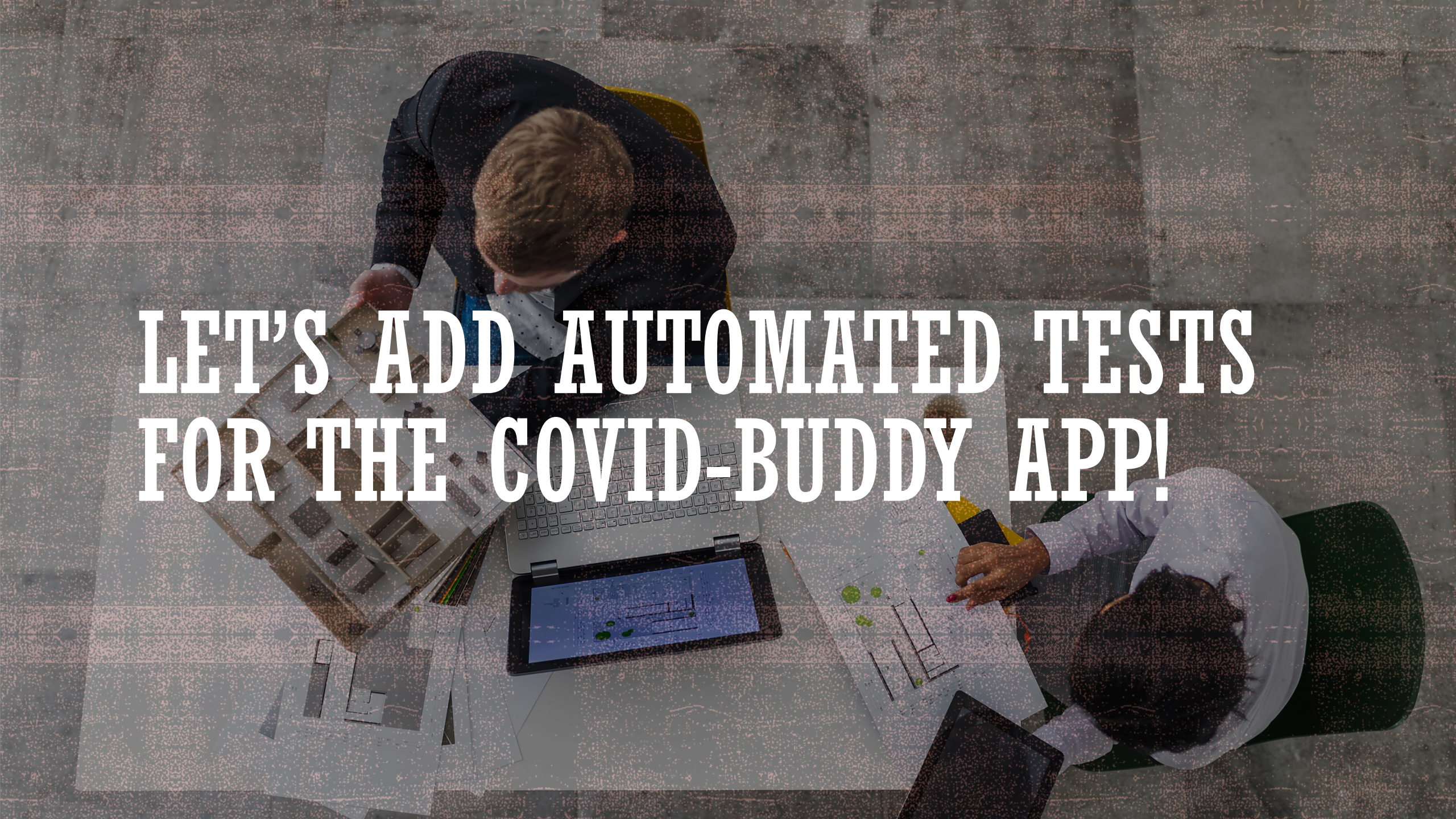
Github: https://github.com/nongnoochr/covid-buddy

FAQs

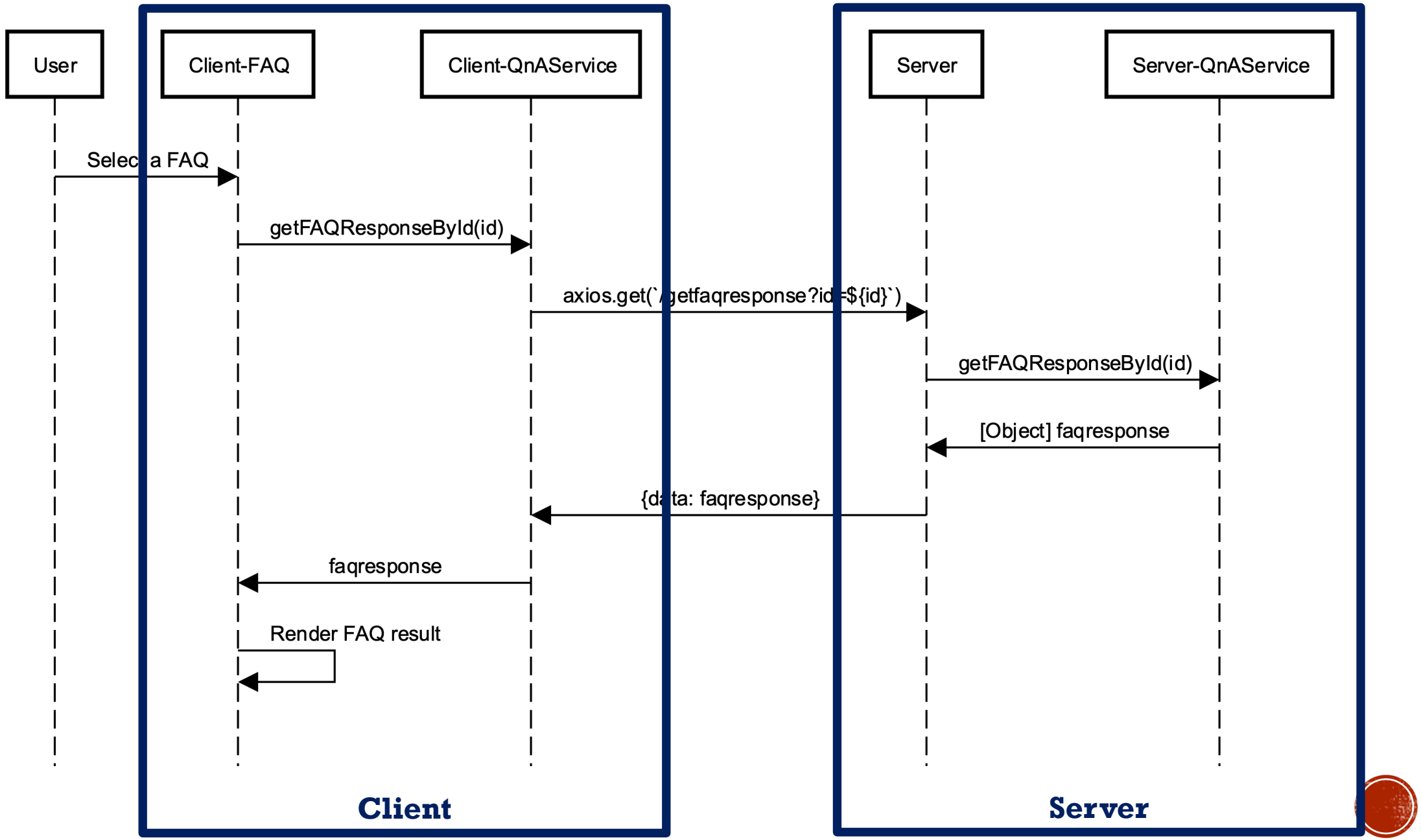
- **794 FAQs** collected from **WHO, CDC, FDA** websites

- **AI + IQVIA HealthCare Locator SDK** are integrated to suggest specialists nearby

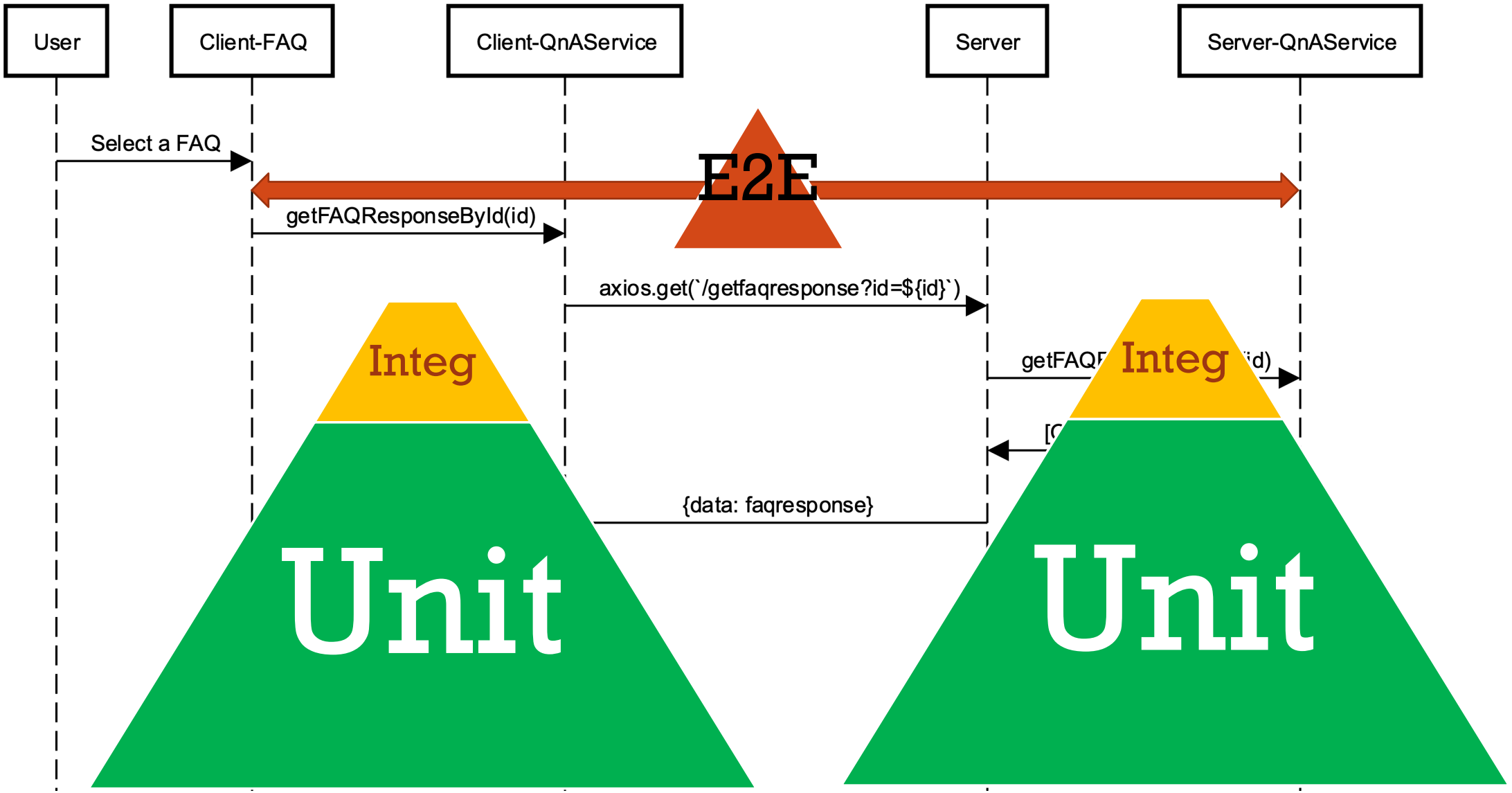



LET'S ADD AUTOMATED TESTS
FOR THE COVID-BUDDY APP!

FAQ Workflow



FAQ Workflow



BEFORE WE START...

Get this repo if you'd like to follow along:

% git clone https://github.com/nongnoochr/covid-buddy.git

% cd covid-buddy

% yarn install (OR % npm install)

% cd client

% yarn install

% yarn build

** Note that the 'Find Healthcare Provider' feature won't work out-of-the-box without specifying a valid apiKey for the IQVIA SDK. For more details, see instructions in README*

If you'd like to run the app in dev mode

Start server

% cd <path/to/covid-buddy>

% node index.js

Start client app in dev mode – do this in a new tab

Then, navigate to http://localhost:3000 in your browser

% cd client

% yarn start

WHAT WE WILL BE USING TO DEVELOP AUTOMATED TESTS?

JS Testing Framework:

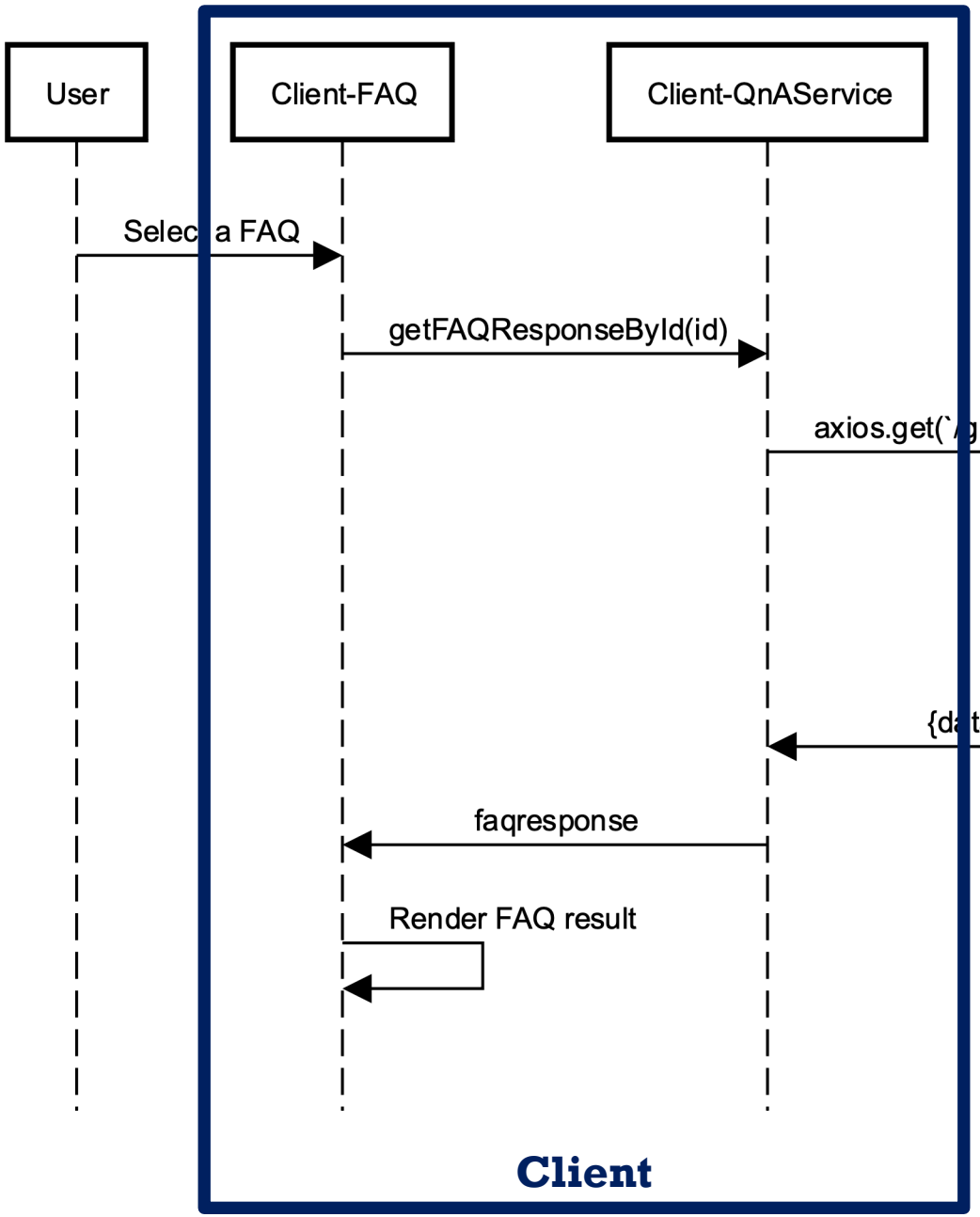
Jest

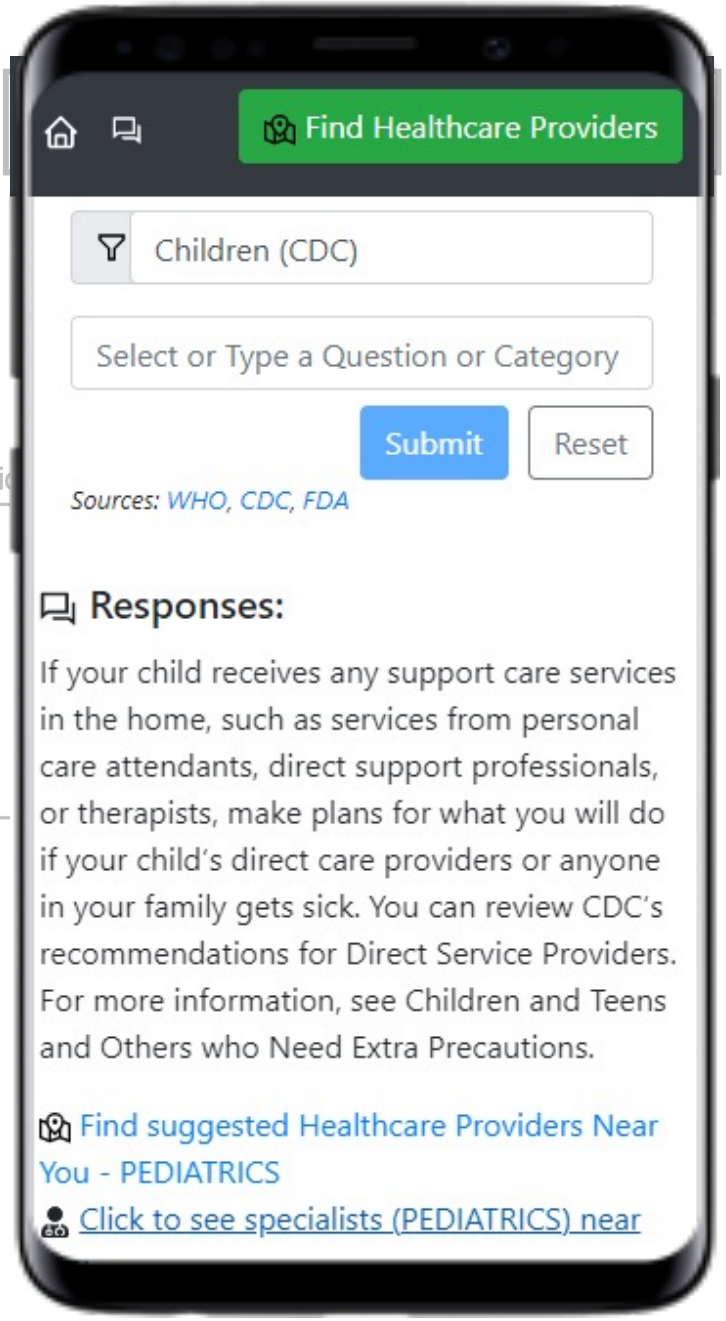
Browser Controller Library (for E2E testing):

Playwright

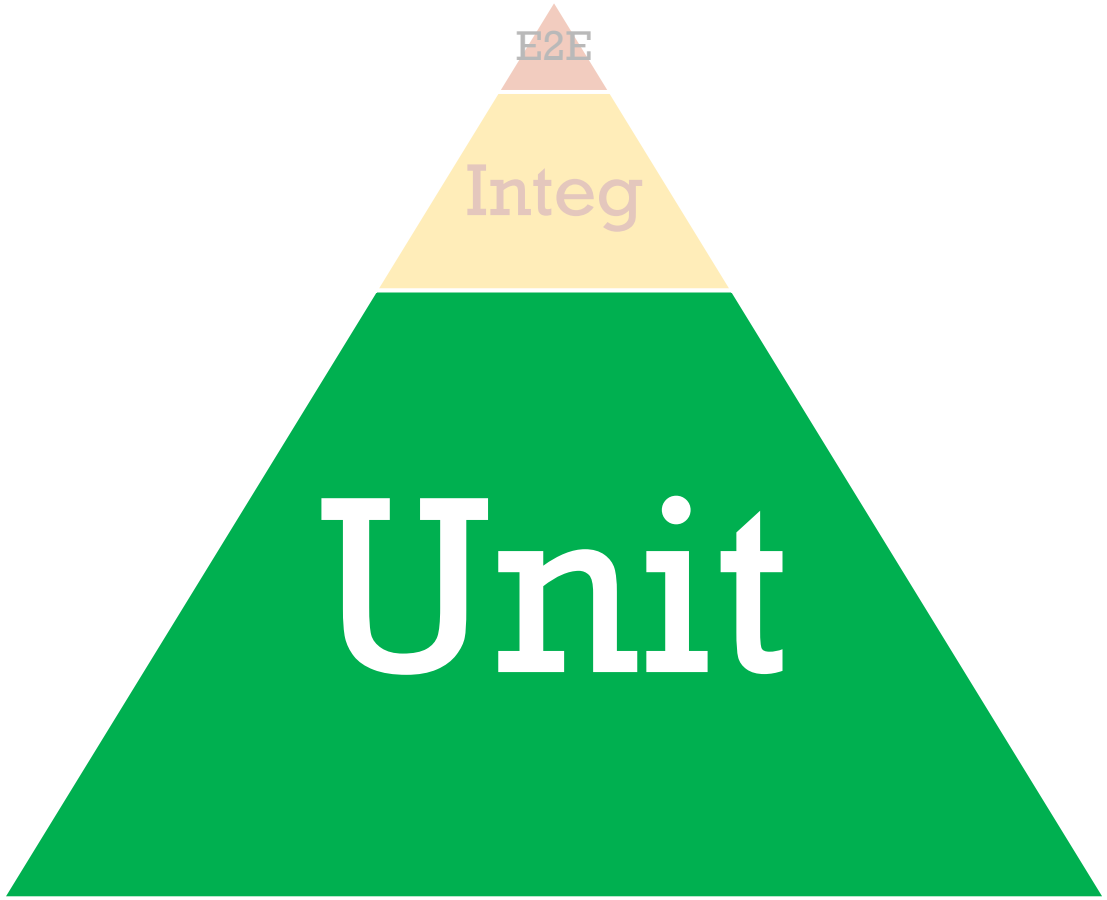
CLIENT-SIDE TESTING

FAQ Workflow





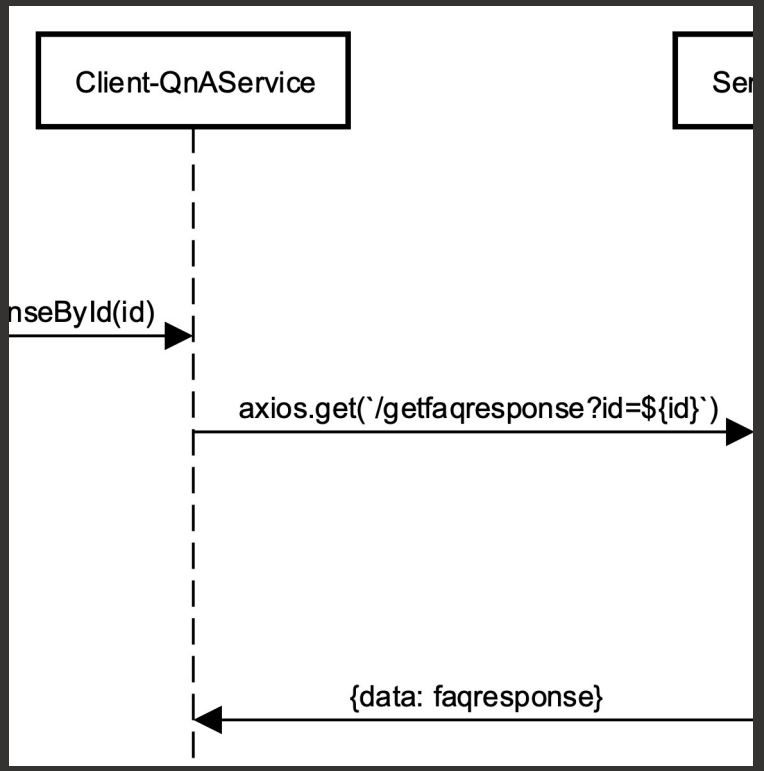
RECAP: UNIT TESTING



- Cover individual units of code
- Can be tested in isolation
- Typically performed by the developer during development
- **Fast and easy to debug**
  - Tests at this level should be maximized (~70%)

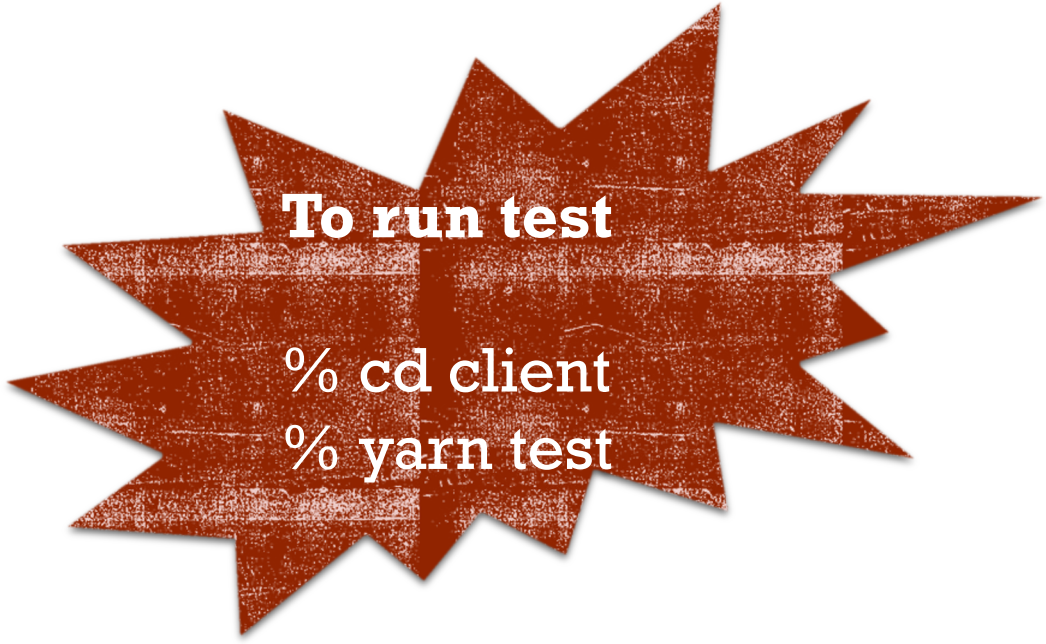
UNIT TEST CASES – NO UI

Ensure that the **getFAQResponseById** function of the **QnAService** module make an **appropriate request**



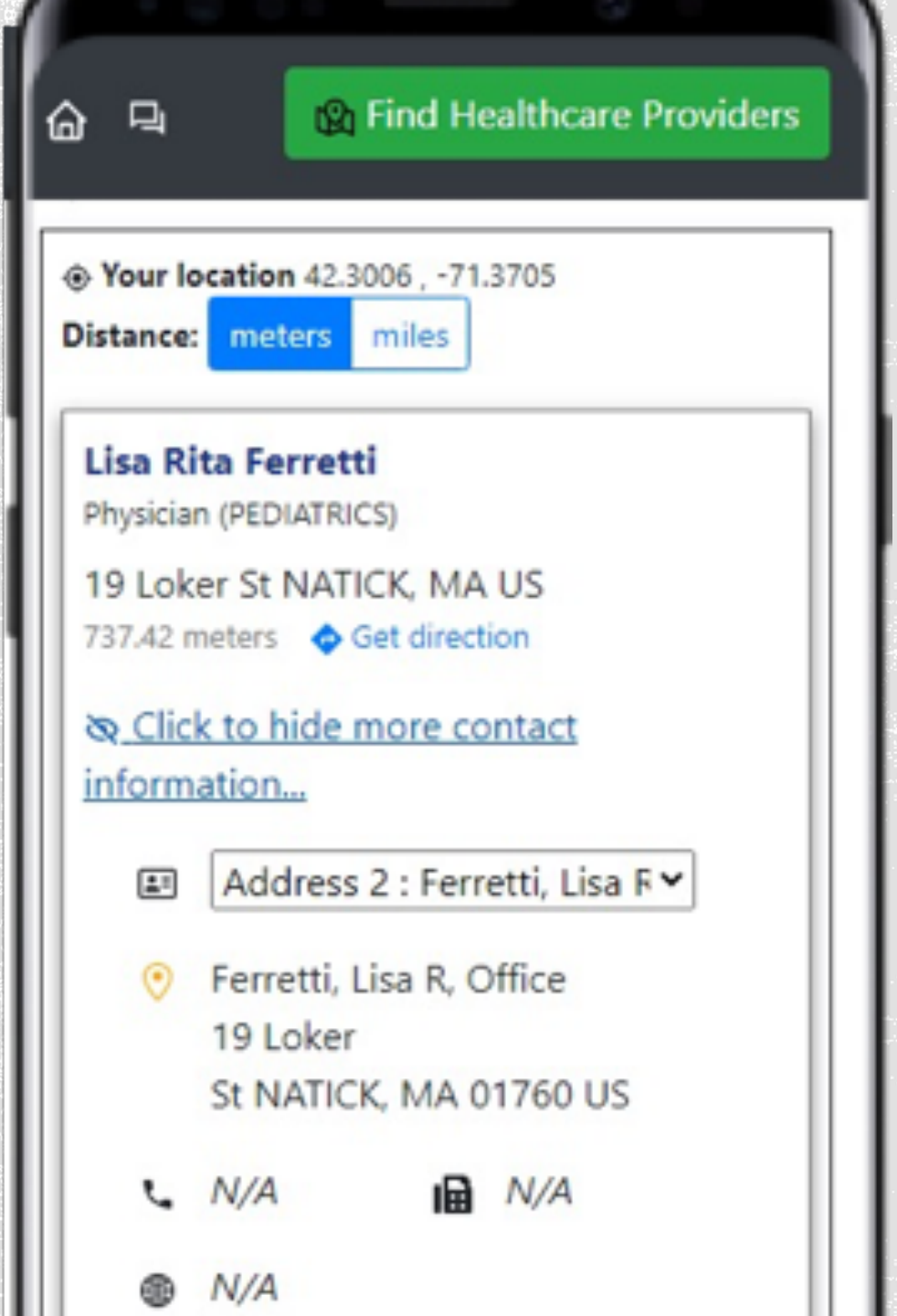
Source: services/QnAService.js

```
/**
 * Request FAQ Response data for a particular FAQ
 * @async
 * @param {string} id Identifier of a requested FAQ
 * @returns {[object]} FAQ Response data
 */
const getFAQResponseById = async (id) => {
    const res = await axios.get(`/getfaqresponse?id=${id}`);
    return res.data;
}
```



Test: tests /services/QnAService.test.js

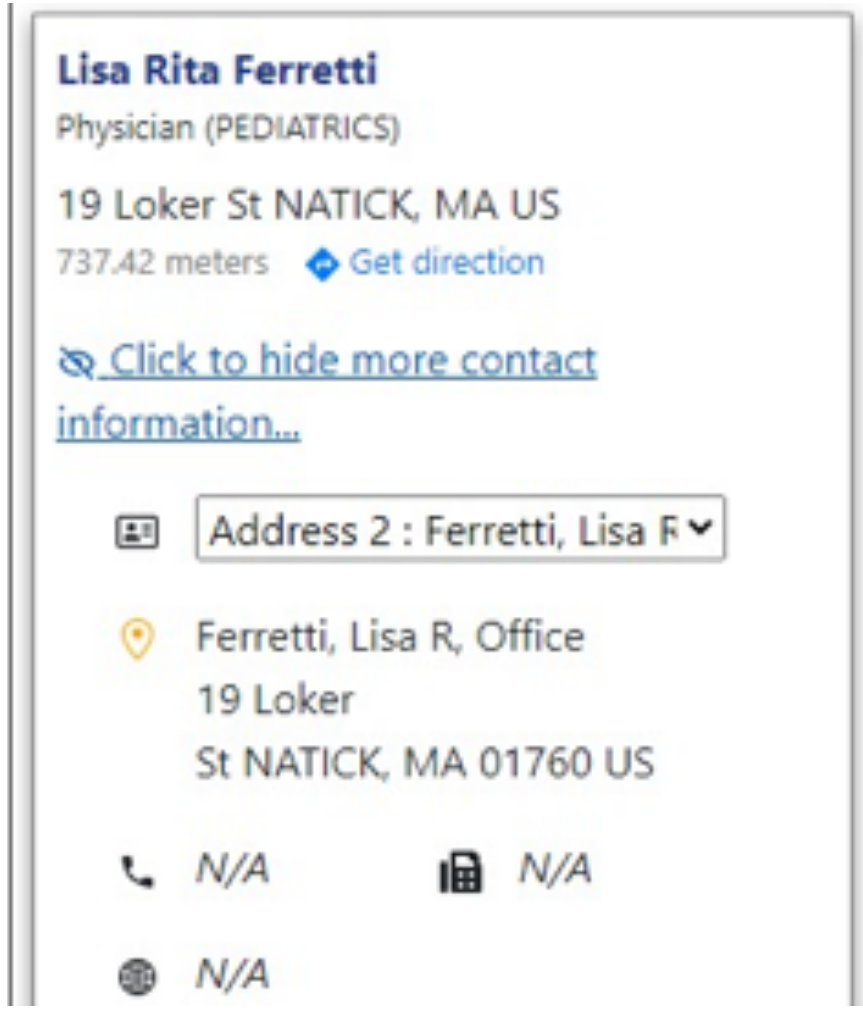
```
it('getFAQResponseById should fetch an appropriate data', async () => {

    const expRes = { data: 'mockedData' };
    axios.get.mockResolvedValue(expRes);

    const myId = 'myId';
    const actRes = await getFAQResponseById(myId);

    expect(actRes).toEqual(expRes.data);
    expect(axios.get).toHaveBeenCalledWith(`/getfaqresponse?id=${myId}`);
});
```



UNIT TEST CASES –UI COMPONENT

Ensure that the **HCPInfo** component **renders appropriate information** based on the **input data**

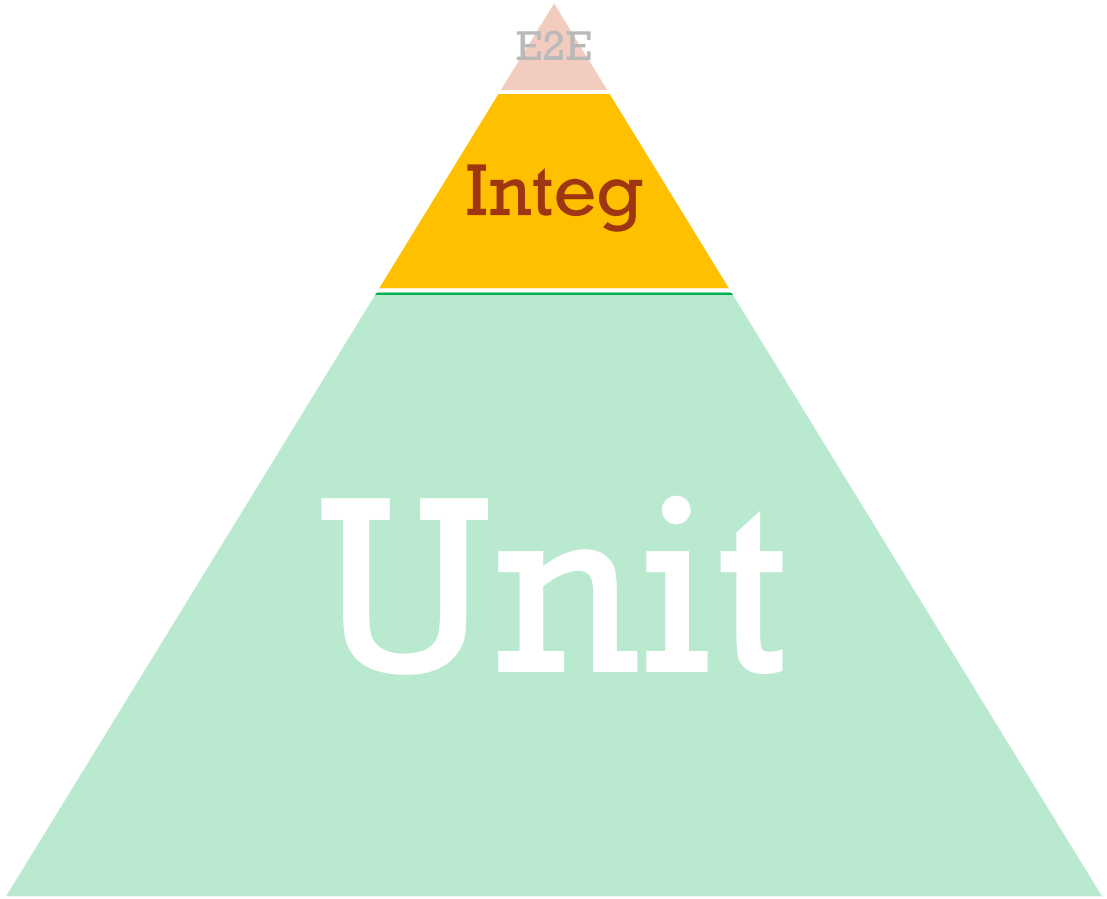
Source: components/QnA/HCPInfo.js



Test: tests /components/HCPInfo.test.js

```
describe('tHCPInfo', () => {
    it('Workplace name should be visible', () => {

        const renderResult = render(<HCPInfo data={data}
            cur-longLabel={longLabel} />);
        expect(renderResult.getByText(data.mainActivity.workplace.name))
            .toBeInTheDocument();
    });

    it.each([
        ...Object.entries(testAddressInfo),
        ['city', 'Natick'],
        ['county', 'MA']
    ])
        ('Workplace address should contain appropriate information "%s"',
            (key, expected) => {

                render(<HCPInfo data={data} cur-longLabel={longLabel} />);

                const elWorkplace = screen.getByTestId('data-workplace');
                expect(elWorkplace).toHaveTextContent(expected);

            });
});
```

RECAP-INTEGRATION TESTING

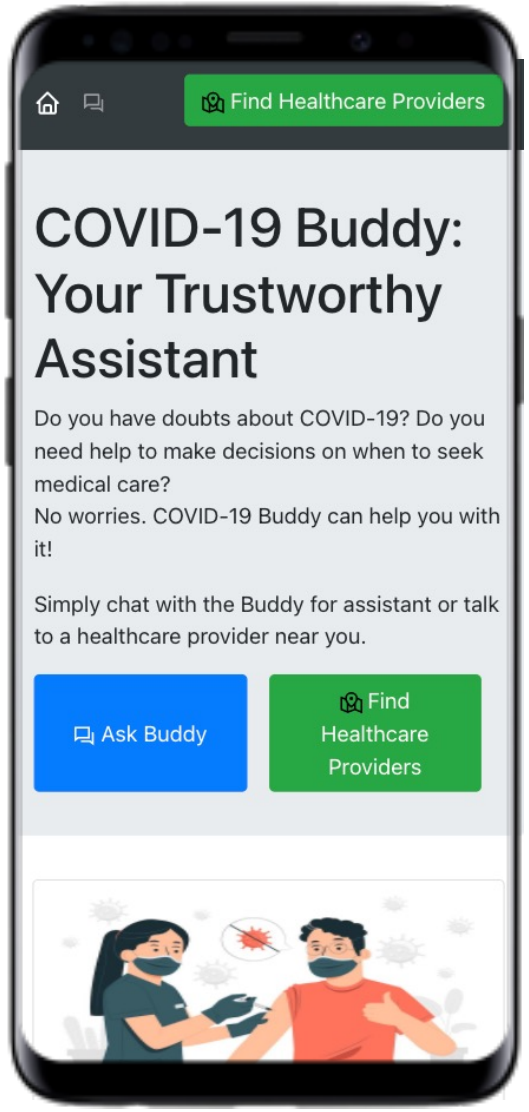


- Process of testing the interface between two software units or modules
- Focus on determining the correctness of the interface
  - The purpose of the integration testing is to expose faults in the interaction between integrated units
- **More costly than unit testing** but less expensive than E2E testing
  - There should be around 20% of tests added to this level

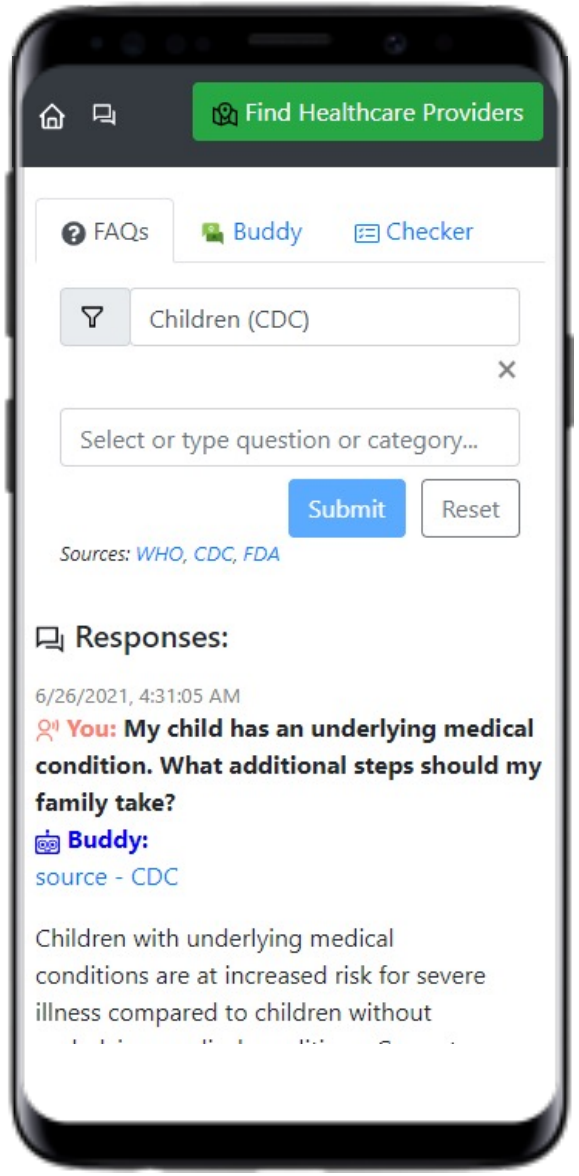
INTEG TEST CASES

Ensure that after clicking the ‘Ask Buddy’ icon in the Navigation bar,

- The pathname is updated to **‘/askbuddy’**
- The app is re-rendered to show the **Buddy page**



Source: App.js



Test: __tests__/App.test.js

```
it('click "Ask Buddy" in navbar should navigate to the Buddy view',
  async () => {
    const history = createMemoryHistory()
    render(
      <Router history={history}>
        <App />
      </Router>,
    );

    const elNavAskBuddy = await screen.findByTestId('nav-item-askbuddy');
    userEvent.click(elNavAskBuddy);

    // Location should be updated
    expect(history.location.pathname).toEqual('/askbuddy');

    // View should be updated to the Ask Buddy mode
    expect(screen.queryByTestId('qna-container')).toBeInTheDocument();
  });
```

HOW DID WE DO SO FAR?

CODE COVERAGE COLLECTION

To collect code coverage in jest, using the command below*

% yarn test --coverage --watchAll

**Note: the '—watchAll' option is required to temporary workaround an open issue as describe below (as of August 13th, 2021)*

https://stackoverflow.com/questions/57451028/jest-finds-tests-but-doesnt-collect-coverage

CODE COVERAGE RESULTS (CLI)

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	38.31	27.15	36.73	38.18	
src	58.06	45	61.54	58.06	
App.js	69.23	56.25	72.73	69.23	49-55,68-73,94,118,140,156-157,164-166
index.js	0	100	100	0	9-20
reportWebVitals.js	0	0	0	0	1-8
src/components/HCPMap	5.71	0	0	5.88	
HCLSDK.js	5	0	0	5	12-80
HCPMap.js	6.67	100	0	7.14	14-56
src/components/Landing	72.73	100	50	72.73	
CardBody.js	100	100	100	100	
Landing.js	66.67	100	33.33	66.67	25,57-58
src/components/Layout	83.33	50	50	83.33	
Layout.js	83.33	50	50	83.33	23
src/components/QnA	37.84	29.35	34.09	38.12	
Buddy.js	25	0	20	25	16,24-80,111
FAQ.js	41.18	50	40	41.18	25,30,59-90,100-102,105,117-118,125-126,135-136,148-156,166
HCPCard.js	5.56	0	0	5.88	17-60
HCPInfo.js	72.22	66.67	66.67	75	44-48
QnA.js	36.17	31.82	37.5	36.96	51-52,60-70,90-141,154-164
Response.js	78.57	57.14	50	78.57	13,19,90
Suggestion.js	11.76	0	0	11.76	20-102
src/services	32.17	12	40	30.63	
HCLSDKService.js	23.53	0	28.57	22.22	64-75,124-148,157-168,180-291,305-340
QnAService.js	100	100	100	100	
src/store	100	100	100	100	
hcp-context.js	100	100	100	100	

CODE COVERAGE RESULTS (REPORT)

All files

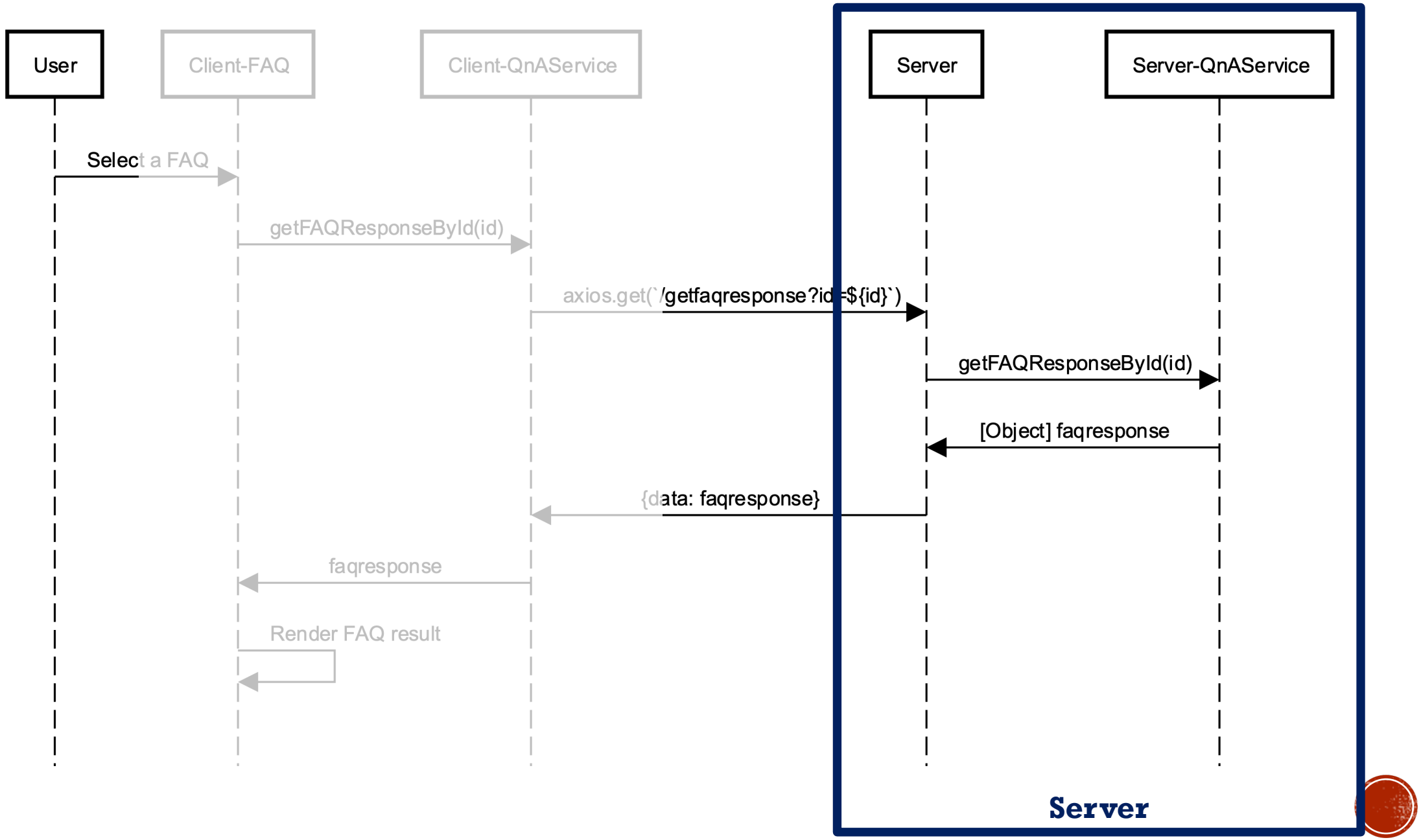
38.31% Statements `159/415` **27.15%** Branches `41/151` **36.73%** Functions `36/98` **38.18%** Lines `155/406`

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
src	58.06%	36/62	45%	9/20	61.54%	8/13	58.06%	36/62
src/components/HCPMap	5.71%	2/35	0%	0/10	0%	0/10	5.88%	2/34
src/components/Landing	72.73%	8/11	100%	0/0	50%	2/4	72.73%	8/11
src/components/Layout	83.33%	5/6	50%	2/4	50%	1/2	83.33%	5/6
src/components/QnA	37.84%	70/185	29.35%	27/92	34.09%	15/44	38.12%	69/181
src/services	32.17%	37/115	12%	3/25	40%	10/25	30.63%	34/111
src/store	100%	1/1	100%	0/0	100%	0/0	100%	1/1

SERVER-SIDE TESTING

FAQ Workflow



Source: server/Utils.js

```
/**
 * zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]
 * @param {function} a function callback that takes in xs and ys
 * @param {[Number]} xs
 * @param {[Number]} ys
 * @returns
 */
const zipWith = (f, xs, ys) => {
    const ny = ys.length;
    return (xs.length <= ny ? xs : xs.slice(0, ny))
        .map((x, i) => f(x, ys[i]));
}

module.exports = {
    zipWith,
    dotProduct
};
```

Test: __tests__/server/Utils.test.js

```
it('zipWith should return an appropriate output', () => {

    const myFcn = (a, b) => [a, b];

    // --- When xs and ys has the same number of elements
    let xs = [1, 2, 3];
    let ys = [4, 5, 6];

    const actSameElems = zipWith(myFcn, xs, ys);
    expect(actSameElems).toEqual([
        [1, 4],
        [2, 5],
        [3, 6]
    ]);

    // --- When a number of Elements in xs of LESS than a number of element of ys
    xs = [1, 2];
    ys = [4, 5, 6];

    const actXsLessThanYs = zipWith(myFcn, xs, ys);
    expect(actXsLessThanYs).toEqual([
        [1, 4],
        [2, 5]
    ]);

    // --- When a number of Elements in xs of GREATER than a number of element of ys
    xs = [1, 2, 3];
    ys = [4];

    const actXsGtYs = zipWith(myFcn, xs, ys);
    expect(actXsGtYs).toEqual([
        [1, 4]
    ]);
```

To run test

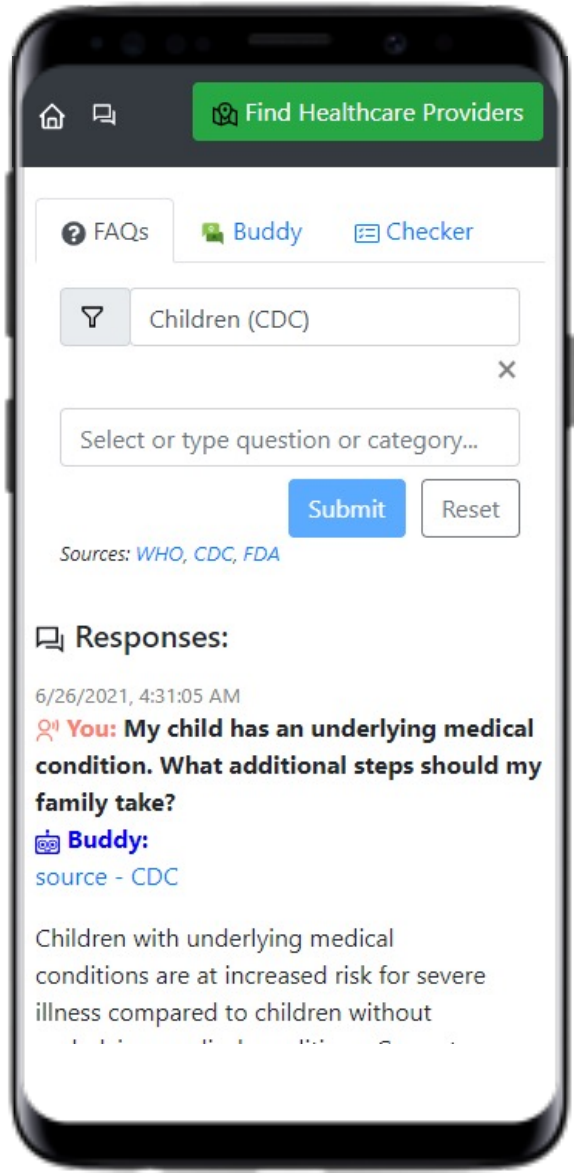
% cd <prj_root>
% npm run test-server

WHAT IS OUR CODE COVERAGE (SERVER)?

% npm run test-server -- --coverage

```
> covid-buddy@1.0.0 test-server
> jest __tests__/server "--coverage"

 PASS  __tests__/server/Utils.test.js
  tUtils
    ✓ zipWith should return an appropriate output (3 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   54.55 |    33.33 |   33.33 |      75 |
 Utils.js |   54.55 |    33.33 |   33.33 |      75 | 22-24
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.294 s, estimated 1 s
Ran all test suites matching /__tests__\/server/i.
```

END-TO-END TESTING



E2E TEST CASES

- Open a production app, then
  - Select a FAQ
  - click Submit
  - Verify that an **appropriate response** retrieved from the server is **rendered properly**

- Ensure that this design case work properly **across browsers** (chromium, webkit, firefox) and **devices** (desktop, mobile devices)

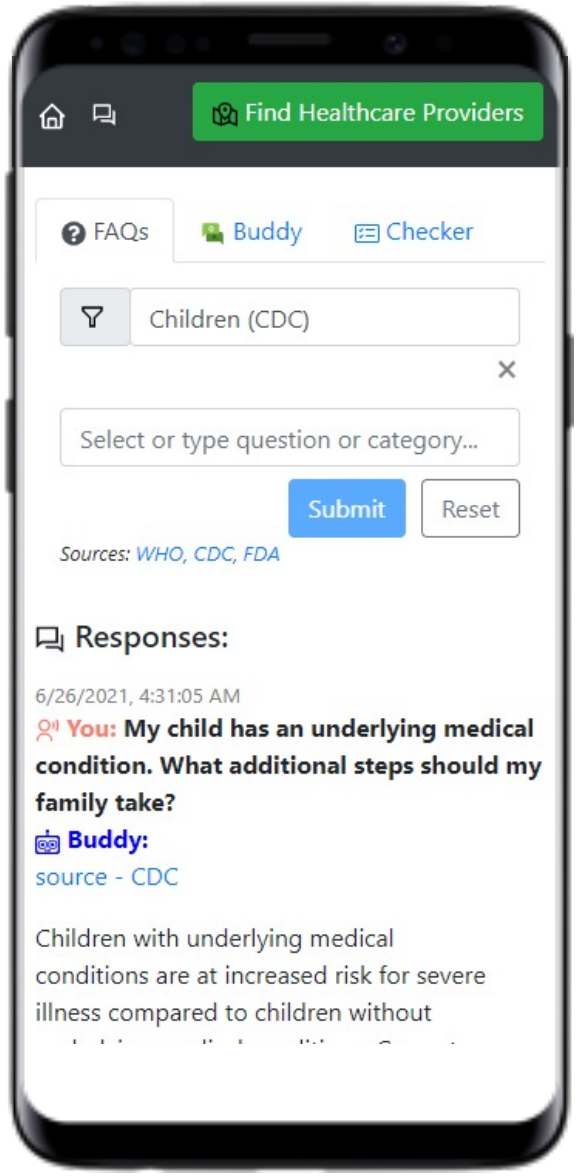
To run E2E tests, use the command below

```
% npm run test-e2e
```

Test: __tests__/e2e/prodApp.test.js

```
const deviceList = [
    'Galaxy S8',
    'iPad Mini landscape',
    'iPhone 12',
    'Pixel 2 XL',
    'Desktop Safari',
    'Desktop Chrome',
    'Dekstop Edge',
    'Desktop Firefox',
];

describe.each([
    [chromium.name(), chromium],
    [firefox.name(), firefox],
    [webkit.name(), webkit],
])('test on %p', (_browserName, browserType) => {

    let newBrowser;

    beforeAll(async () => {
        newBrowser = await browserType.launch();
    });

    afterAll(async () => {
        await newBrowser.close();
    });
```

```
let context
try {
    context = await newBrowser.newContext({
        ...devices[curDeviceName],

        // Required when clicking Submit
        geolocation: { longitude: 48.858455, latitude: 2.294474 },
        permissions: ['geolocation']
    });
} catch (e) {
    console.log(`Skip test in "${_browserName}" `);
    return
}

const page = await context.newPage();
await page.goto('http://localhost:8080');

await expect(page).toHaveSelector('[data-testid="nav-item-brand"]');

const selector = '[data-testid="nav-item-askbuddy"]';
await page.click(selector);
await expect(page).toHaveSelector('[data-testid="qna-container"]');

const selectorResContainer = '[data-testid="response-container"]';
const elResContainerBefore = await page.$(selectorResContainer);
const elResContainerContentBefore = await elResContainerBefore.textContent();
expect(elResContainerContentBefore).toBeFalsy();
```

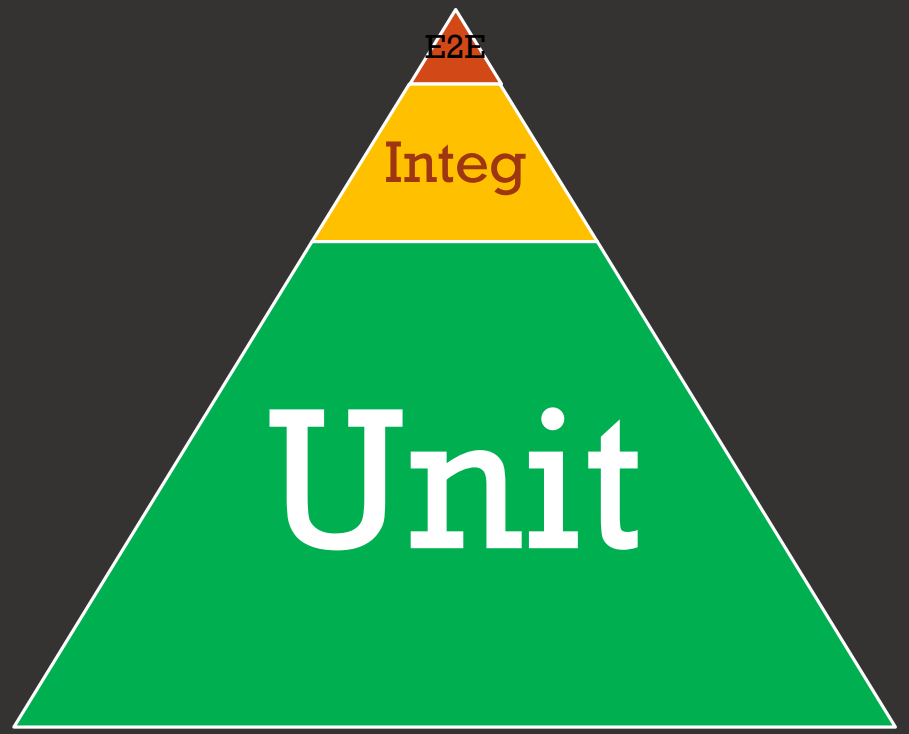



```
// Click [placeholder="Select or type question or category..."]
await page.click('[placeholder="Select or type question or category..."]');

// Click text=What can we do so that other diseases like COVID-19 do not affect us in future?
await page.click('#faq-typeahead-item-0');
const element = await page.$('input[placeholder="Select or type question or category..."]');

const elemValue = await element.inputValue();
expect(elemValue).toBeTruthy();

// Click text=Submit
await page.click('text=Submit');

// Poll until the response shows up
await page.waitForSelector('[data-testid="response-item"]');

const elResContainerAfter = await page.$(selectorResContainer);
const elResContainerContentAfter = await elResContainerAfter.textContent();
expect(elResContainerContentAfter).toBeTruthy();

await page.close();
```

These commands to interact with a browser can be **automatically generated** through the **'recording'** feature that is available after invoking **'await page.pause()'** ☺

TAKEAWAYS

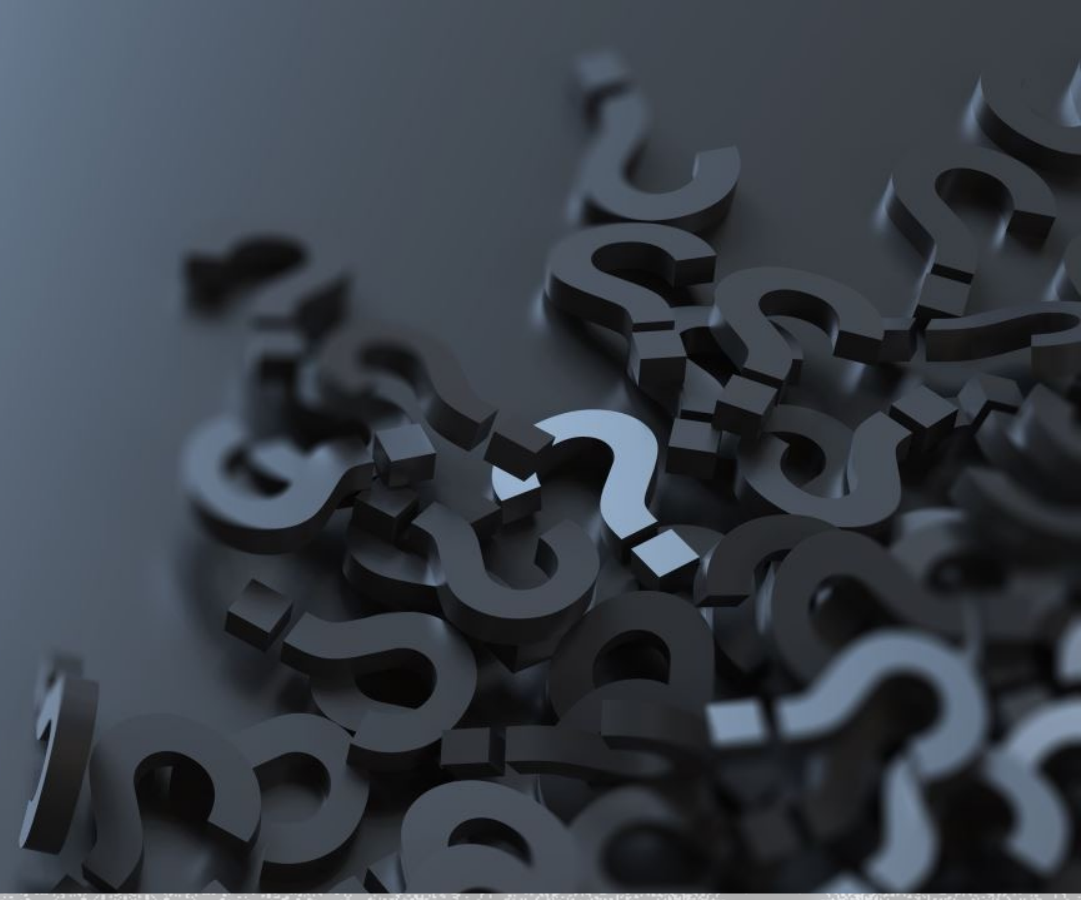
- Use **TESTABILTIY** as a lens of your **design**
- Adhere to **Test Pyramid** when developing automated tests for your project




THAT'S IT! HAVE FUN TESTING
IN YOUR NEXT PROJECT ☺

RESOURCES

- GitHub Project: https://github.com/nongnoochr/covid-buddy
  - This presentation is available in artifacts/intro_automatedtesting.pptx
- https://jestjs.io/
- https://playwright.dev/



QUESTIONS?