What Is End-to-End Testing?

End-to-end testing, or E2E for short, tests real-world scenarios from the user's perspective. Examples of E2E testing scenarios include testing the registration and login process or testing the checkout for an e-commerce site.

To better understand how end-to-end testing works, let's take a look at a real-world example from the freeCodeCamp codebase. This example uses Playwright, a popular end-to-end testing framework developed by Microsoft.

Inside the freeCodeCamp codebase, there is a directory called `e2e` which holds tests for donations, certifications, exams and more. Inside the directory, there is a `donate-page-donor.spec.ts` file which has tests for the authenticated donor supporter page.

The first step is to navigate to the donation page as a demo user who is already a supporter:

```
test.beforeEach(async ({ page }) => {
  execSync("node ./tools/scripts/seed/seed-demo-user --set-true isDonating");
  await page.goto("/donate");
});
```

Since there are multiple tests in this file, the `beforeEach` hook will run before each of the tests.

Once the user is on the donation page, they will see a thank you message with suggestions on how to support freeCodeCamp further. Here are just a few of the tests to check for some of the text found on the donation page:

```
test("should render the donate page correctly", async ({ page }) => {
  await expect(
    page.getByText("Thank you for your continued support")
  ).toBeVisible();

  await expect(
    page.getByText(
      "Your contributions are crucial in creating resources that empower millions of
people to learn new skills and support their families."
    )
  ).toBeVisible();

  ...
});
```

There are also tests to check that the donor has a supporter link in the menu bar, as well as a special stylized border around their avatar to indicate they are a supporter:

```
test("The menu should have a supporters link", async ({ page }) => {
  const menuButton = page.getByTestId("header-menu-button");
  const menu = page.getByTestId("header-menu");

  await expect(menuButton).toBeVisible();
  await menuButton.click();

  await expect(menu).toBeVisible();

  await expect(page.getByRole("link", { name: "Supporters" })).toBeVisible();
});

test("The Avatar should have a special border for donors", async ({ page }) => {
  const container = page.locator(".avatar-container");
  await expect(container).toHaveClass("avatar-container gold-border");
});
```

By having these types of detailed tests, you can increase the test coverage for your application beyond just unit tests. End-to-end tests also help ensure that your application behaves correctly, and is predictable for users. Examples of other end-to-end testing tools include Cypress, Selenium, and Puppeteer.

While end-to-end testing is important, it is expensive because it is time-consuming to set up, design, and maintain. E2E tests are great for critical user flows, while unit tests are great for testing small units in the application.

Questions

Which of the following is NOT an example of a commonly used E2E testing tool?

Cypress

Selenium

Agile

Correct!

Puppeteer

Which of the following is a popular end-to-end testing framework developed by Microsoft?

Playwright

Correct!

Java

Angular

JUnit

Which of the following is a commonly used test scenario for end-to-end tests?

Testing the validity of CSS code.

Testing the performance for an application.

Testing the validity of HTML code.

Testing the user login process.

Correct!