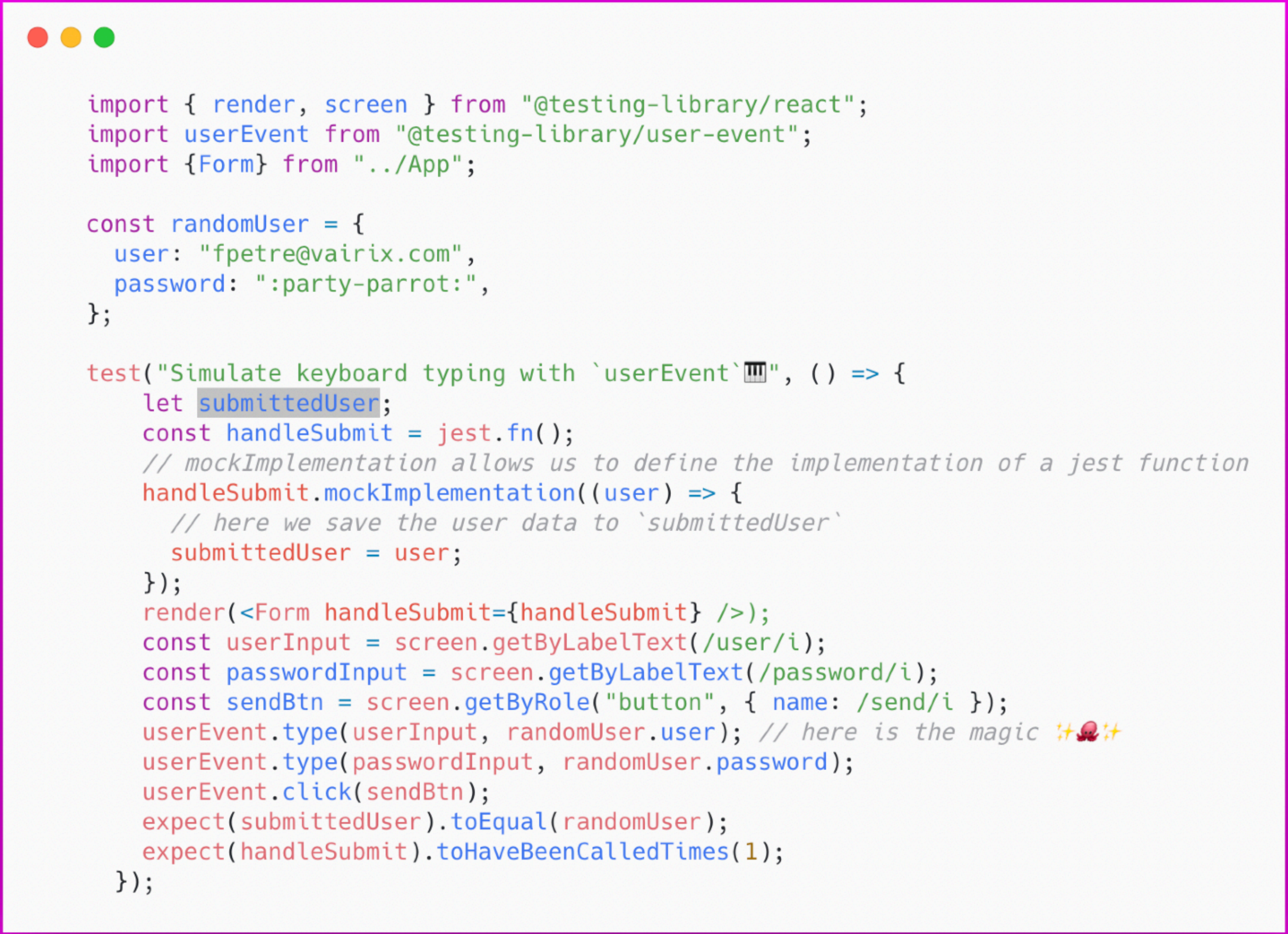
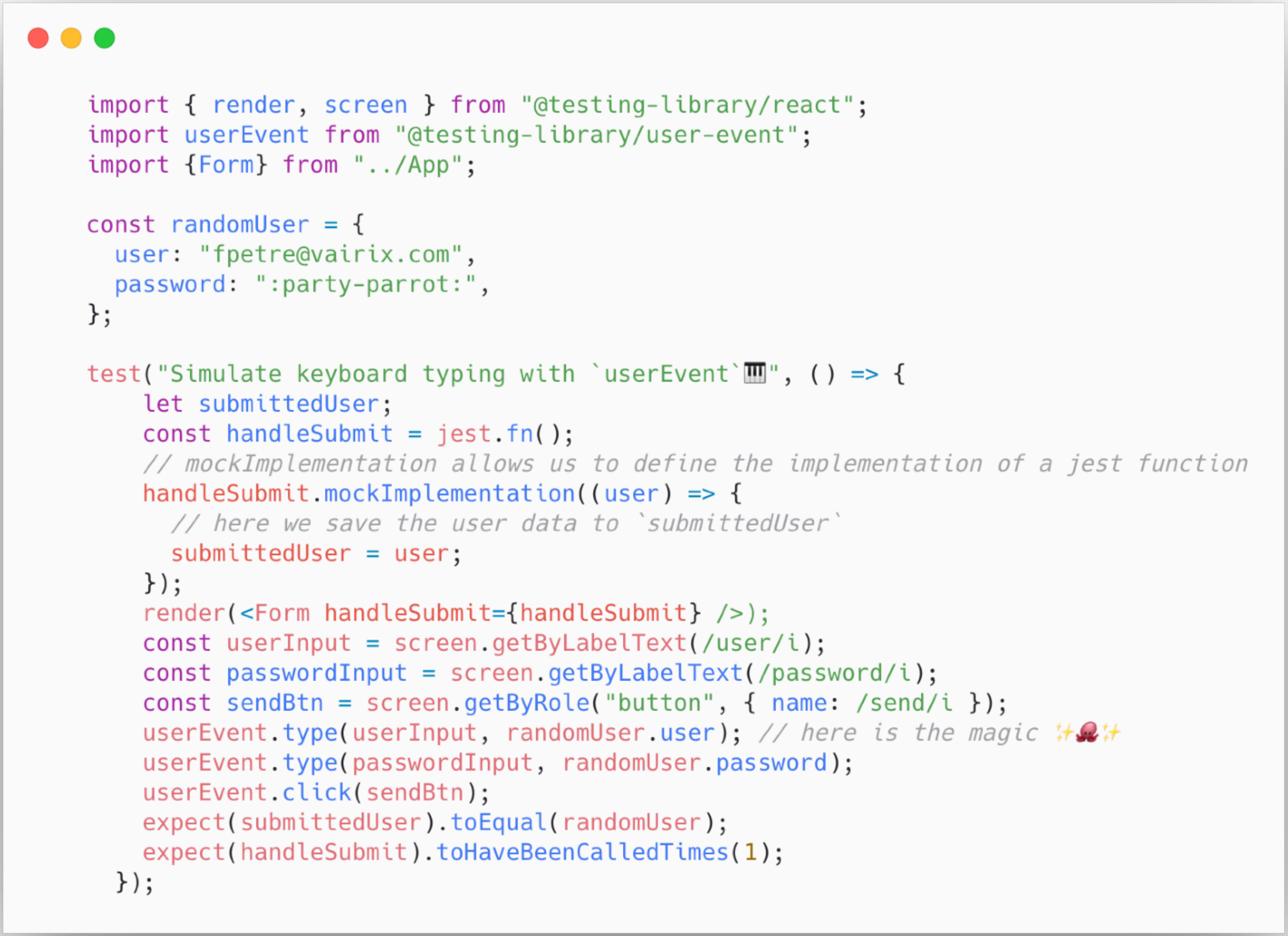


```
import { render, screen } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import {Form} from "../App";

const randomUser = {
  user: "fpetre@vairix.com",
  password: ":party-parrot:",
};

test("Simulate keyboard typing with `userEvent`🎹", () => {
    let submittedUser;
    const handleSubmit = jest.fn();
    // mockImplementation allows us to define the implementation of a jest function
    handleSubmit.mockImplementation((user) => {
      // here we save the user data to `submittedUser`
      submittedUser = user;
    });
    render(<Form handleSubmit={handleSubmit} />);
    const userInput = screen.getByLabelText(/user/i);
    const passwordInput = screen.getByLabelText(/password/i);
    const sendBtn = screen.getByRole("button", { name: /send/i });
    userEvent.type(userInput, randomUser.user); // here is the magic ✨🐙✨
    userEvent.type(passwordInput, randomUser.password);
    userEvent.click(sendBtn);
    expect(submittedUser).toEqual(randomUser);
    expect(handleSubmit).toHaveBeenCalledTimes(1);
  });
```

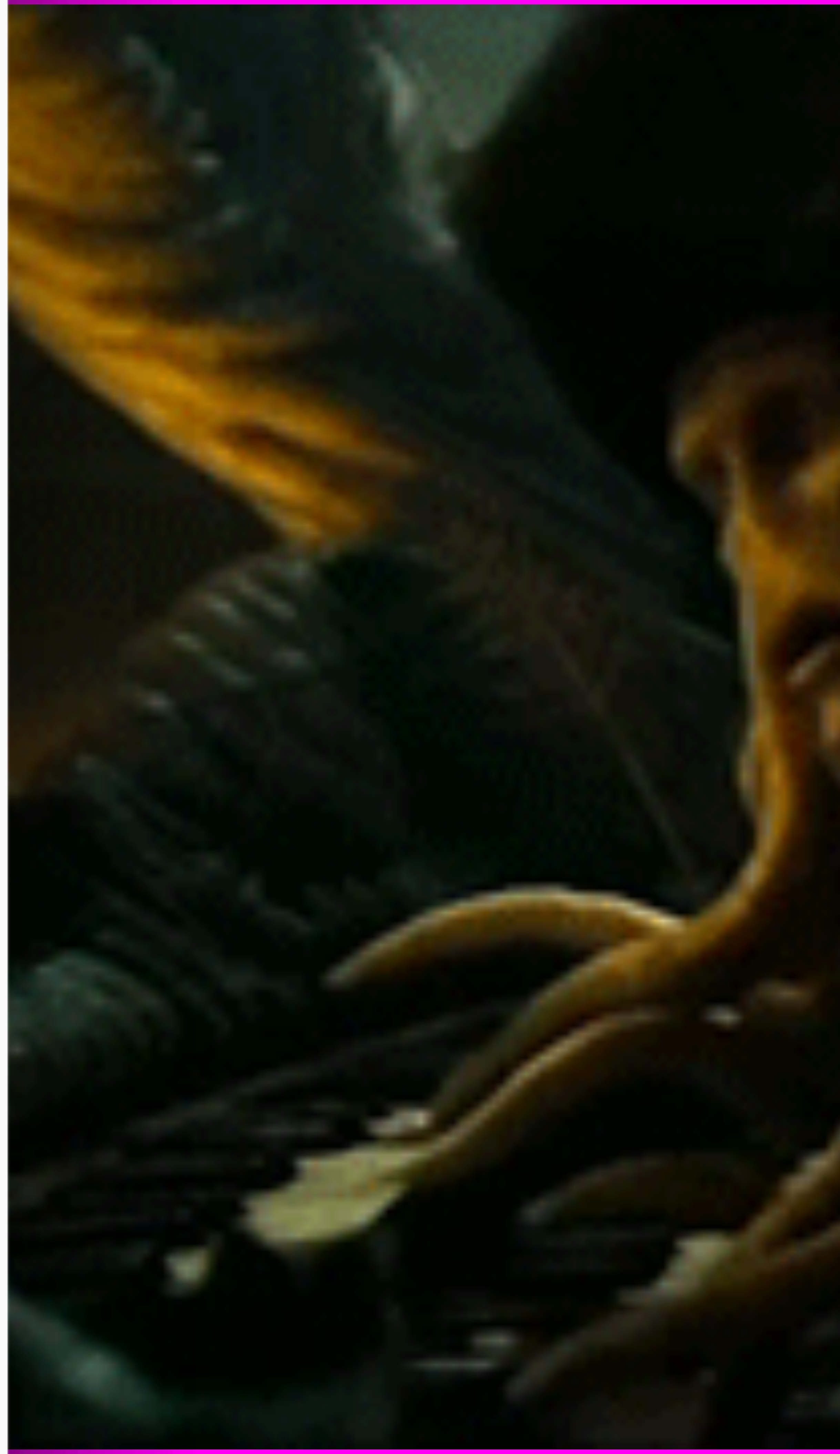


```javascript
import { render, screen } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import {Form} from "../App";

const randomUser = {
  user: "fpetre@vairix.com",
  password: ":party-parrot:",
};

test("Simulate keyboard typing with `userEvent`🎹", () => {
    let submittedUser;
    const handleSubmit = jest.fn();
    // mockImplementation allows us to define the implementation of a jest function
    handleSubmit.mockImplementation((user) => {
      // here we save the user data to `submittedUser`
      submittedUser = user;
    });
    render(<Form handleSubmit={handleSubmit} />);
    const userInput = screen.getByLabelText(/user/i);
    const passwordInput = screen.getByLabelText(/password/i);
    const sendBtn = screen.getByRole("button", { name: /send/i });
    userEvent.type(userInput, randomUser.user); // here is the magic ✨🐙✨
    userEvent.type(passwordInput, randomUser.password);
    userEvent.click(sendBtn);
    expect(submittedUser).toEqual(randomUser);
    expect(handleSubmit).toHaveBeenCalledTimes(1);
  });
```

Let’s type something 🐙

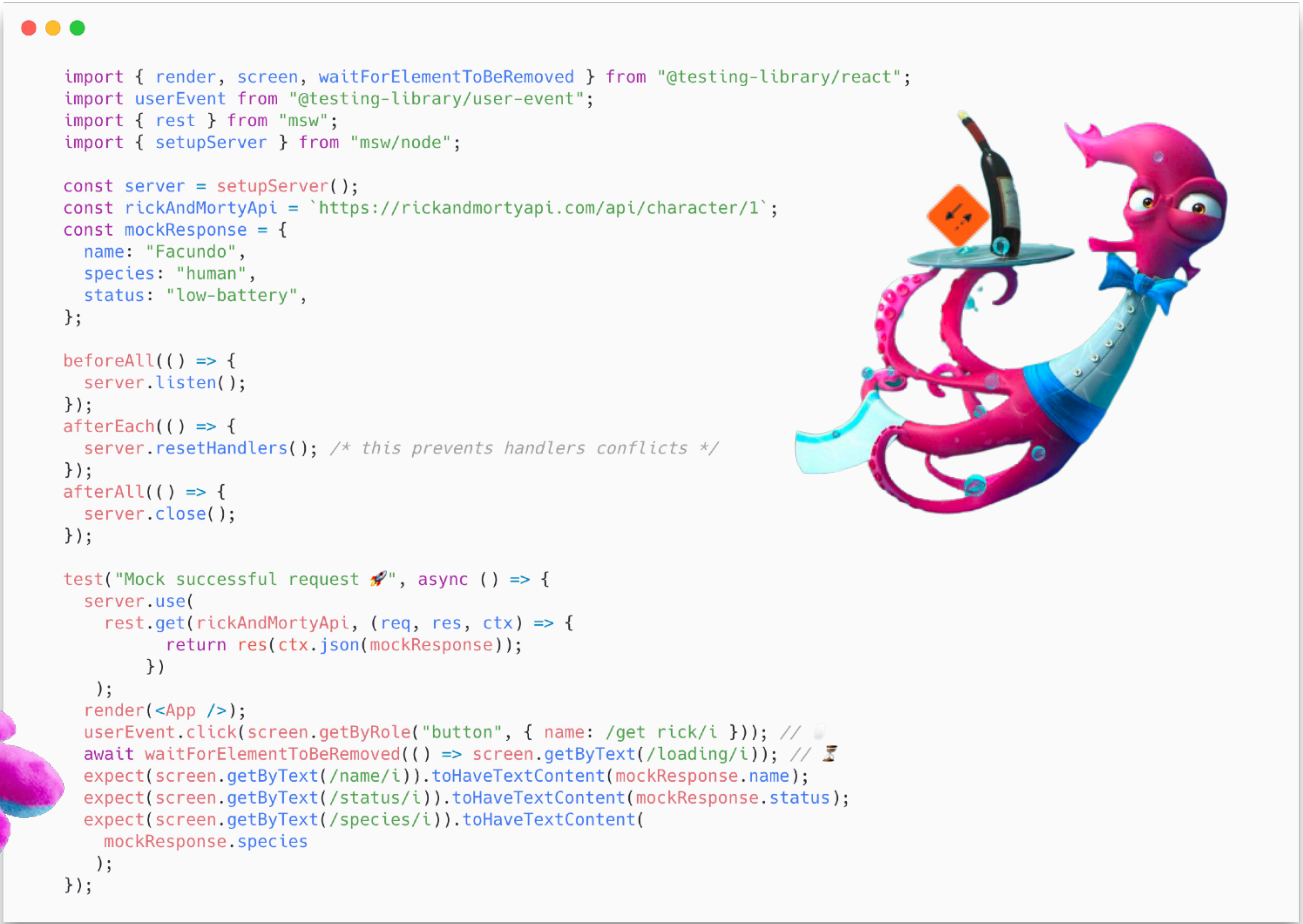
Let’s type something 🐙

```
import { render, screen } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import {Form} from "../App";

const randomUser = {
  user: "fpetre@vairix.com",
  password: ":party-parrot:",
};

test("Simulate keyboard typing with `userEvent`🎹", () => {
    let submittedUser;
    const handleSubmit = jest.fn();
    // mockImplementation allows us to define the implementation of a jest function
    handleSubmit.mockImplementation((user) => {
      // here we save the user data to `submittedUser`
      submittedUser = user;
    });
    render(<Form handleSubmit={handleSubmit} />);
    const userInput = screen.getByLabelText(/user/i);
    const passwordInput = screen.getByLabelText(/password/i);
    const sendBtn = screen.getByRole("button", { name: /send/i });
    userEvent.type(userInput, randomUser.user); // here is the magic ✨🐙✨
    userEvent.type(passwordInput, randomUser.password);
    userEvent.click(sendBtn);
    expect(submittedUser).toEqual(randomUser);
    expect(handleSubmit).toHaveBeenCalledTimes(1);
  });
```



HTTP mocking with MSW

(Happy Path)



```
import { render, screen, waitForElementToBeRemoved } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import { rest } from "msw";
import { setupServer } from "msw/node";

const server = setupServer();
const rickAndMortyApi = `https://rickandmortyapi.com/api/character/1`;
const mockResponse = {
  name: "Facundo",
  species: "human",
  status: "low-battery",
};

beforeAll(() => {
  server.listen();
});
afterEach(() => {
  server.resetHandlers(); /* this prevents handlers conflicts */
});
afterAll(() => {
  server.close();
});

test("Mock successful request 🚀", async () => {
  server.use(
    rest.get(rickAndMortyApi, (req, res, ctx) => {
          return res(ctx.json(mockResponse));
        })
   );
  render(<App />);
  userEvent.click(screen.getByRole("button", { name: /get rick/i })); // 🖱️
  await waitForElementToBeRemoved(() => screen.getByText(/loading/i)); // ⏳
  expect(screen.getByText(/name/i)).toHaveTextContent(mockResponse.name);
  expect(screen.getByText(/status/i)).toHaveTextContent(mockResponse.status);
  expect(screen.getByText(/species/i)).toHaveTextContent(
    mockResponse.species
   );
});
```