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
   );
});
```

HTTP mocking with MSW

(Unhappy 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 message = `R.I.P 🪦`;

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

test(`Mock request failure 🧨💥`, async () => {
  server.use(
    rest.get(rickAndMortyApi, (req, res, ctx) => {
      return res(ctx.delay(1000), ctx.status(500), ctx.json({ message }));
    })
  );
  render(<App />);
  userEvent.click(screen.getByRole("button", { name: /get rick/i }));
  await waitForElementToBeRemoved(() => screen.getByText(/loading/i), {
    timeout: 2000,
  });
  expect(screen.getByRole("alert")).toHaveTextContent(message);
});
```