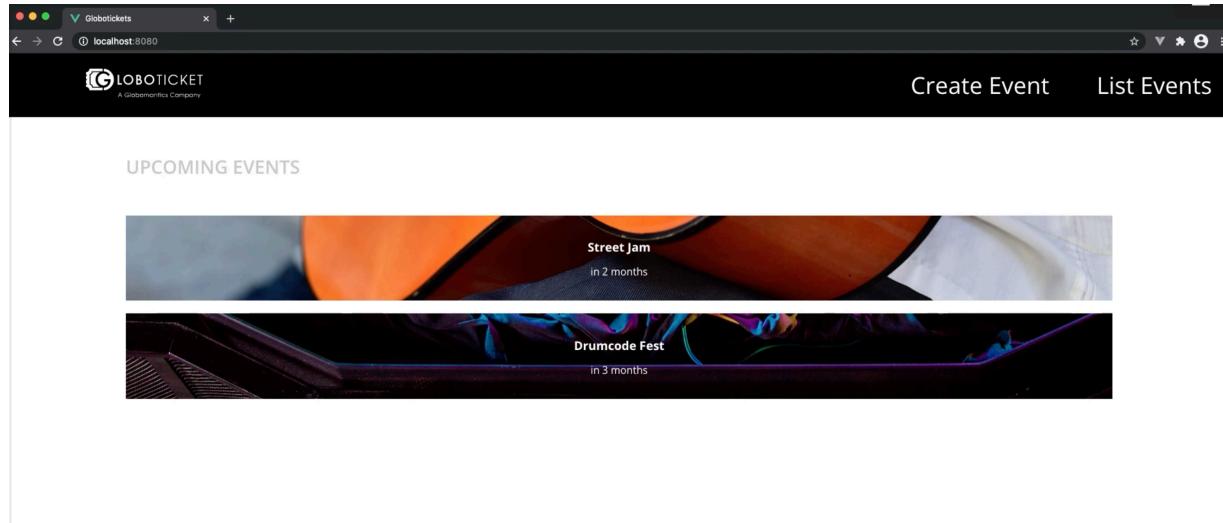
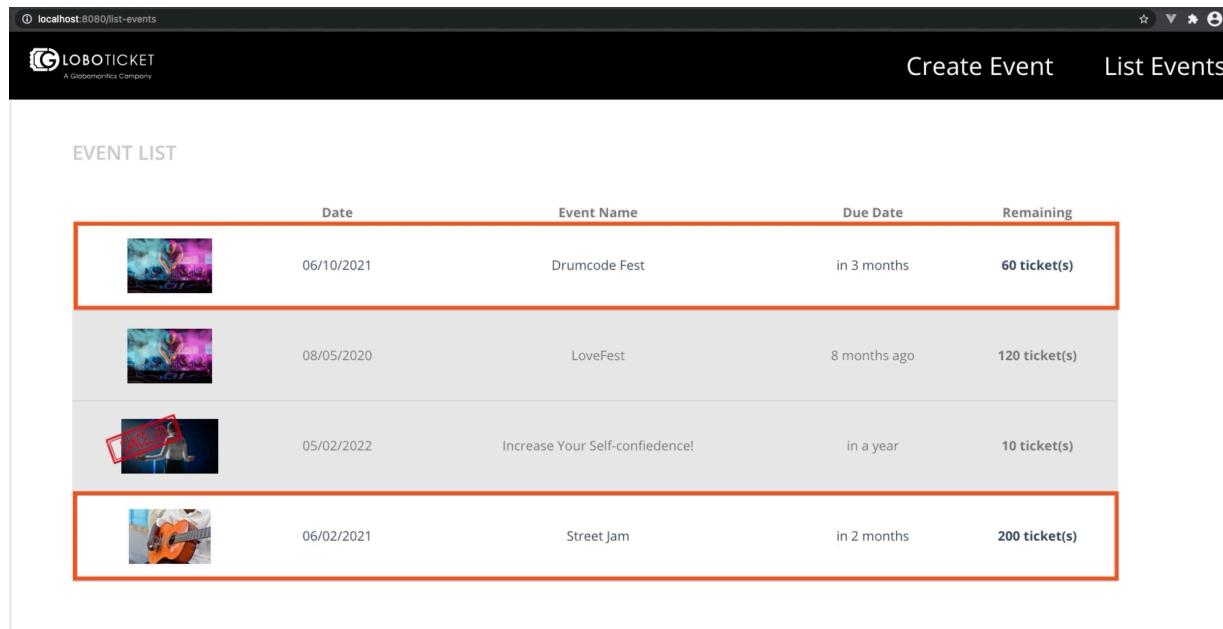


Cypress version2

Application:



A screenshot of a web browser window titled "Globotickets" at "localhost:8080". The page features a dark header with the "LOBOTICKET" logo and "A Globomantics Company" text, along with "Create Event" and "List Events" buttons. Below the header, the text "UPCOMING EVENTS" is displayed. Two event cards are shown: "Street Jam" (in 2 months) and "Drumcode Fest" (in 3 months). Both cards include a thumbnail image and the event name.



A screenshot of a web browser window titled "localhost:8080/list-events". The interface is identical to the previous screenshot, with the "LOBOTICKET" logo, "Create Event" and "List Events" buttons, and the "EVENT LIST" heading. The event list table has columns for Date, Event Name, Due Date, and Remaining tickets. Four events are listed: "Drumcode Fest" (06/10/2021, due in 3 months, 60 tickets), "LoveFest" (08/05/2020, due 8 months ago, 120 tickets), "Increase Your Self-confidence!" (05/02/2022, due in a year, 10 tickets), and "Street Jam" (06/02/2021, due in 2 months, 200 tickets). The first and fourth rows are highlighted with a red border.

Date	Event Name	Due Date	Remaining
06/10/2021	Drumcode Fest	in 3 months	60 ticket(s)
08/05/2020	LoveFest	8 months ago	120 ticket(s)
05/02/2022	Increase Your Self-confidence!	in a year	10 ticket(s)
06/02/2021	Street Jam	in 2 months	200 ticket(s)

CREATE EVENT

Event Name *
Alexander Lemtov Live
Event name is a required field.

Event Venue *
Sherwood Event Hall, WA

Event Date *
Month: 12 | Day: 00 | Year: 2023

Event Time *
12:00 a.m.

Ticket Quantity *
142

Additional Notes
Register now! Dress in orange...

Cover Image

Event Cover 1
 Event Cover 2
 Event Cover 3

CREATE **CANCEL**

Cypress commands are asynchronous.

```
it('test', () => {
  let username = undefined;

  cy.visit('http://localhost:8080')
  cy.get('#username')
    .then(($el) => {
      username = $el.text()
    });

  if (username) {
    cy.contains(username).click();
  } else {
    cy.contains('MyProfile').click();
  }
});
```

```
it('test', () => {
  let username = undefined;

  cy.visit('http://localhost:8080');
  cy.get('#username')
    .then(($el) => {
      username = $el.text();

      if (username) {
        cy.contains(username).click();
      } else {
        cy.contains('My Profile').click();
      }
    });
});
```

Exploring Retry-ability

```
cy.get('.main-list li') // command  
  .should('have.length', 3) // assertion
```

Retry only works with assertion

Intercepting HTTP Requests

If a server-side implementation is not ready

If you intend only to test a presentation layer

```
cy.intercept(url, routeMatcher, routeHandler).as('alias');  
  
cy.wait('@alias');
```

The difference between `cy.stub` and `cy.intercept` is that:

- `cy.stub` can be used for stubbing functions to record their usage, or control their behavior.
- `cy.intercept` can be used for intercepting network requests to return

mock values.

For example, to stub a method of an object, you can call the following in Cypress:

```
const user = {  
    // Get method for getting a user based on an ID  
    get(id) { ... }  
};  
  
cy.stub(user, 'get')  
    .withArgs('age')  
    .returns(30);
```

The above code example will stub the user.get method with an argument of age, and it forces the function to return 30. On the other hand, you can intercept network requests in the following way:

```
cy.intercept('/api', { fixtures: 'response.json' });
```

This will intercept any requests made to /api and forces it to respond with the contents of the response.json file.

#pluralsight