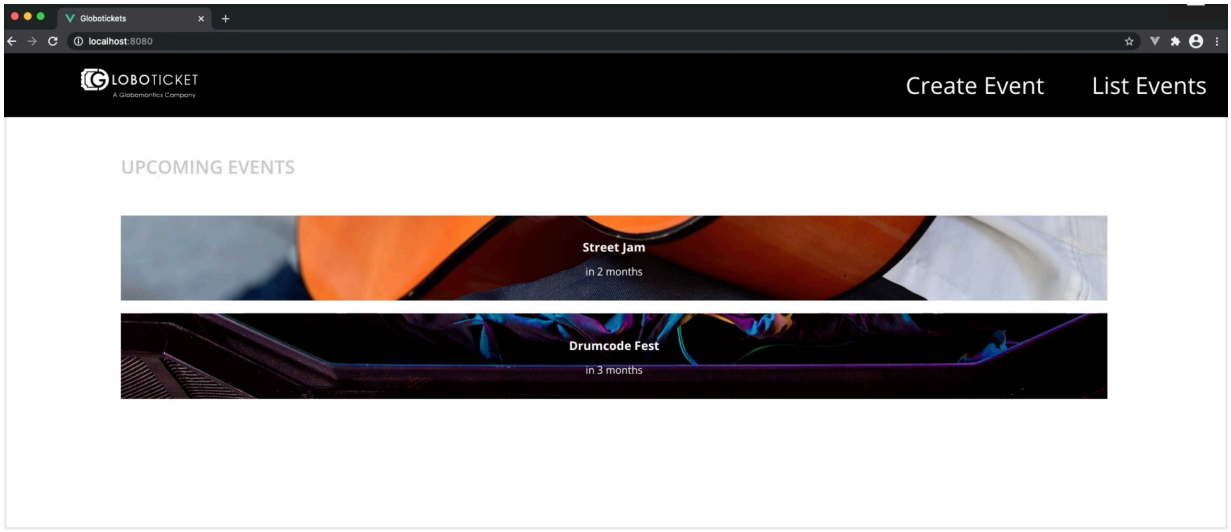
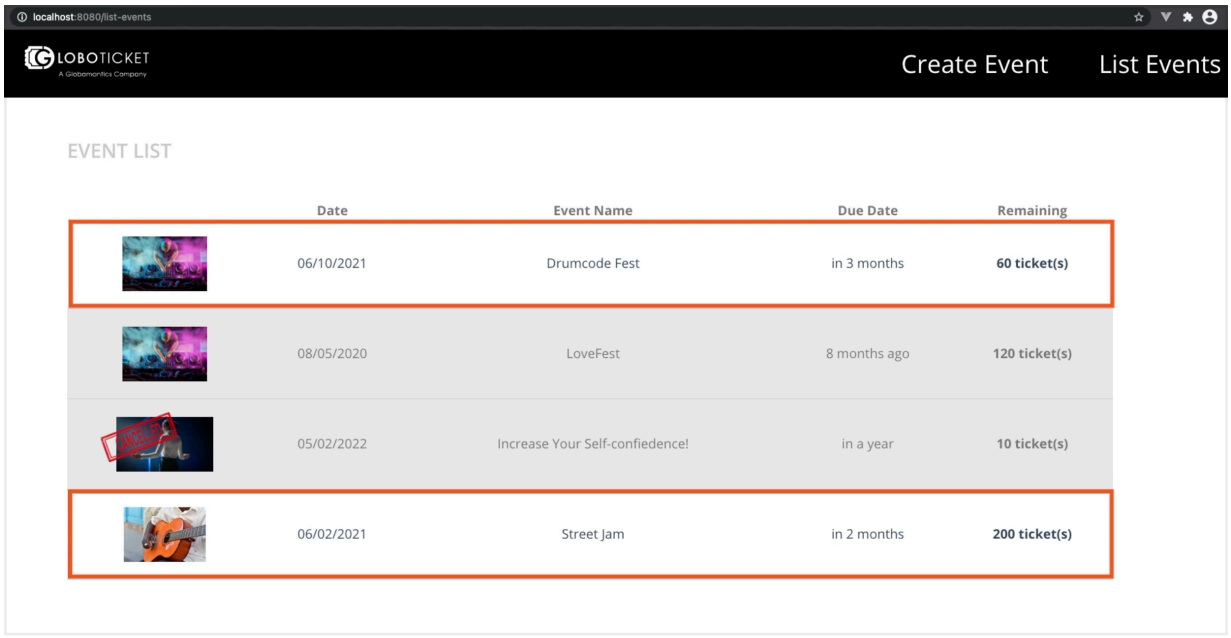
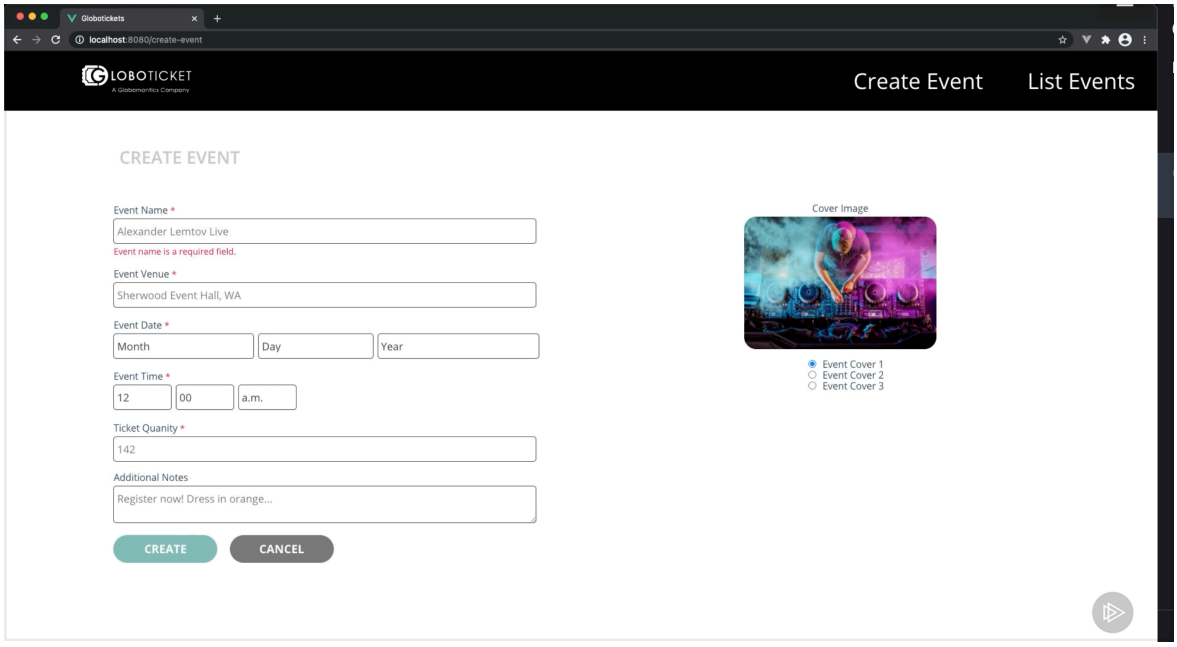
Cypress version2

Application:







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


```
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