**Tests are written by considering GET API**

Request- Get: /api/v1/festivals

Response:

[

{

"name": "string",

"bands": [

{

"name": "string",

"recordLabel": "string"

}

]

}

]

Assuming only the Get api requirement (not considering POST,PUT and DELETE requests). Will build following tests in API tests in Postman.

**Test case#1:**

**Objective:** Check the response code of below Get api is 200

GET /api/v1/festivals

**Test case #2:**

**Objective:** Verify that response is less than 200 ms

**Postman script:**

pm.test("Response time is less than 200ms", function () {

pm.expect(pm.response.responseTime).to.be.below(200);

});

**Test case#3:**

**Objective**: Send GET request /api/v1/festivals and verify that body is present bands attributes are present

const jsonData = pm.response.json();

pm.test('Has data', function() { pm.expect(jsonData).to.have.property(bands); });

**Test case#4:**

**Objective**: Verify that GET response festival name is not null and bands is not null

var jsonData = JSON.parse(responseBody);

tests["nameExist"] = jsonData.name !== null ;

for (var i = 0; i<jsonData.bands.length;i++){

test[“bandnameExist”]=jsonData.bands[i].name!==null;

test[“recordlabelExist”]=jsonData.bands[i].recordLabel!==null;

}

**Test case #5:**

**Objective**: Verify the name and bands (name, recordlabel) are type string

var jsonData = JSON.parse(responseBody);

pm.expect(typeOf jsonData.name==='string').to.be.true

for (var i = 0; i<jsonData.bands.length;i++){

pm.expect(typeOf jsonData.bands[i].name==='string').to.be.true

pm.expect(typeOf jsonData.bands[i].recordLabel==='string').to.be.true

}

**Test case#6(Negative test):**

**Objective**: Turn off the server or make server down

Send the below GET response, make sure status code 503 is received.

**Website Tests:**

**Test case#1:**

**Objective**: Verify that Festival name and Bands field is displayed on website

**Steps**:

1. Navigate to website<url>
2. Verify that Festival name and Bands field is displayed on website

**Expected Results:**

Festival name and Bands field should be displayed on the website.

**Test case#2:**

Objective: Verify user is relevant Bands selected based on the festival name selection

**Steps**:

1. Navigate to website<url>
2. Select Festival name
3. Verify related festival bands only displayed in the festival band field

**Expected Results:**

Only associated Festival bands should be displayed

**Mobile Tests:**

**Test case#1:**

**Objective**: Verify that Festival name and Bands field is displayed on Mobile app

**Steps**:

1. Launch mobile app
2. Verify that Festival name and Bands field is displayed on website

**Expected Results:**

Festival name and Bands field should be displayed on the mobile app screen.

**Test case#2:**

Objective: Verify user is relevant Bands selected based on the festival name selection

**Steps**:

1. Navigate to website<url>
2. Select Festival name
3. Verify related festival bands only displayed in the festival band field

**Expected Results:**

Only associated Festival bands should be displayed