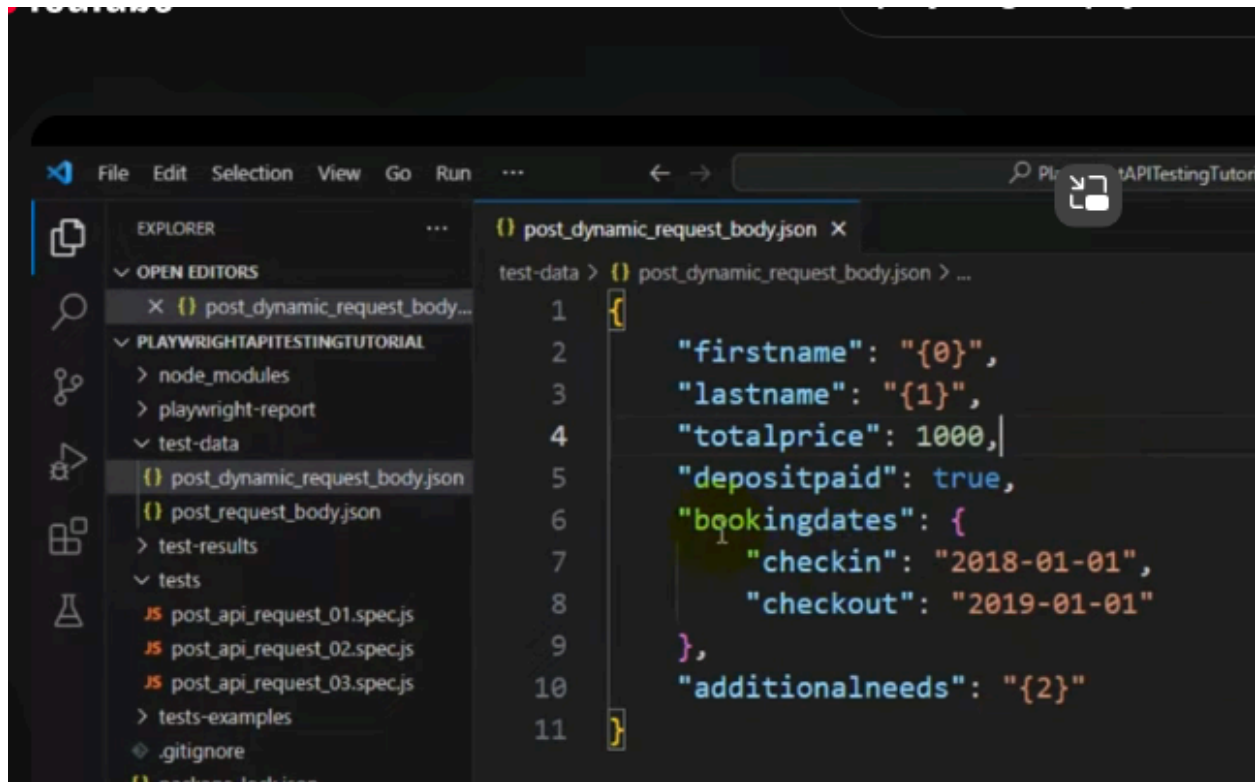


baka-pwapi-5

POST API Request using Dynamic JSON File

we need to give {1}, {2} as index numbers inside json file.

json data-



```
1  {
2    "firstname": "{0}",
3    "lastname": "{1}",
4    "totalprice": 1000,
5    "depositpaid": true,
6    "bookingdates": {
7      "checkin": "2018-01-01",
8      "checkout": "2019-01-01"
9    },
10   "additionalneeds": "{2}"
11 }
```

```
1  {
2    "firstname": "{0}",
3    "lastname": "{1}",
4    "totalprice": 1000,
5    "depositpaid": true,
6    "bookingdates": {
7      "checkin": "2018-01-01",
8      "checkout": "2019-01-01"
9    },
10   "additionalneeds": "{2}"
11 }
```

```
1 //common utility functions.
2
3 /**
4  * This code defines a utility function stringFormat in
5  * JavaScript/TypeScript, which mimics a common string
6  * formatting pattern used in other languages like C# or Python.
7
8  ✓ Function Purpose
9  The function replaces placeholders in a string
10 (like {0}, {1}, etc.) with corresponding values from the args array.
11
12 ♦ str:
13 The base string with placeholders like {0}, {1}, etc.
14
15 ♦ ...args:
16 The values that will replace the placeholders.
17
18 The values that will replace the placeholders.
19
20 ♦ str.replace(/{{(\d+)}}/g, ...):
21 A regular expression that finds all substrings
22 that match {number} format.
23
24 {{(\d+)}}: Captures one or more digits inside curly braces.
25
26 g: Global flag to replace all instances.
```

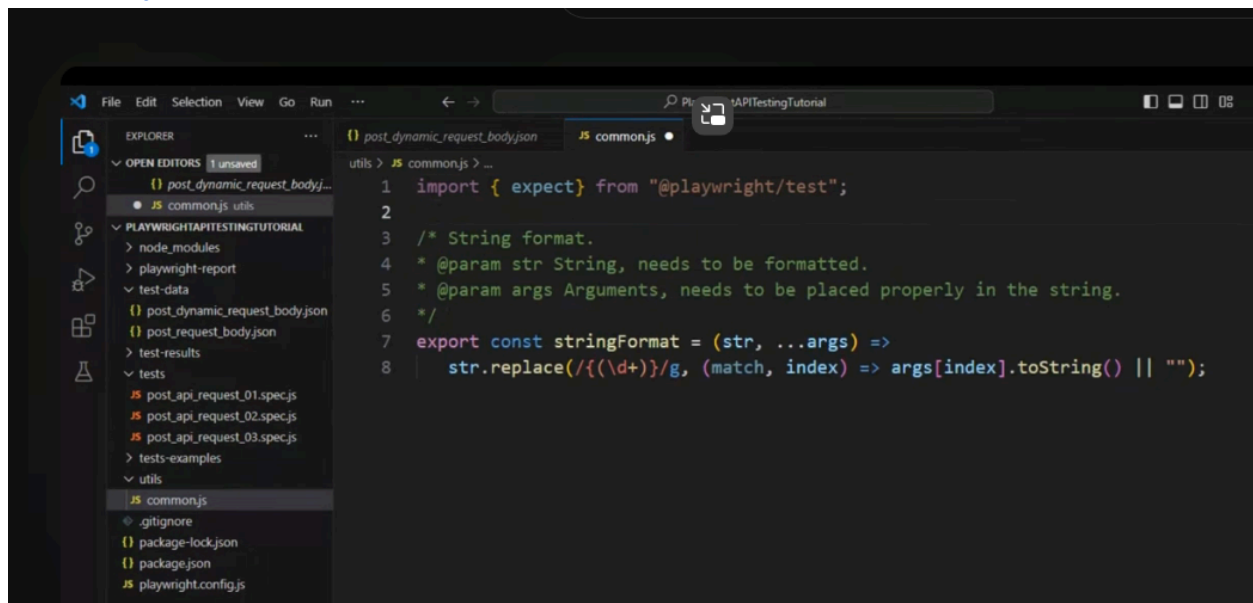
```

1  * (match, index) => args[index].toString() || "";
2
3  For each placeholder found:
4
5  index is the captured number inside {}.
6
7  It replaces the placeholder with the corresponding argument value converted to a string.
8
9  If the argument is undefined or null, it falls back to an empty string.
10
11 examples-
12
13 stringFormat("Hello, {0}! You have {1} messages.", "Alice", 5);
14 // Output: "Hello, Alice! You have 5 messages."
15
16 stringFormat("This is {0}, and this is {2}.", "X", "Y", "Z");
17 // Output: "This is X, and this is Z."
18
19
20 stringFormat("Missing {0} and {1}", "onlyOne");
21 // Output: "Missing onlyOne and "

```

codesnap.dev

[Commons.js](#) file –



```
1 */
2
3 import { expect } from "@playwright/test";
4
5 /* String format.
6 * @param str String, needs to be formatted.
7 * @param args Arguments, needs to be placed properly in the string.
8 */
9 export const stringFormat = (str, ...args) =>
10   str.replace(/{\d+}/g, (match, index) => args[index].toString() || "");
```

```

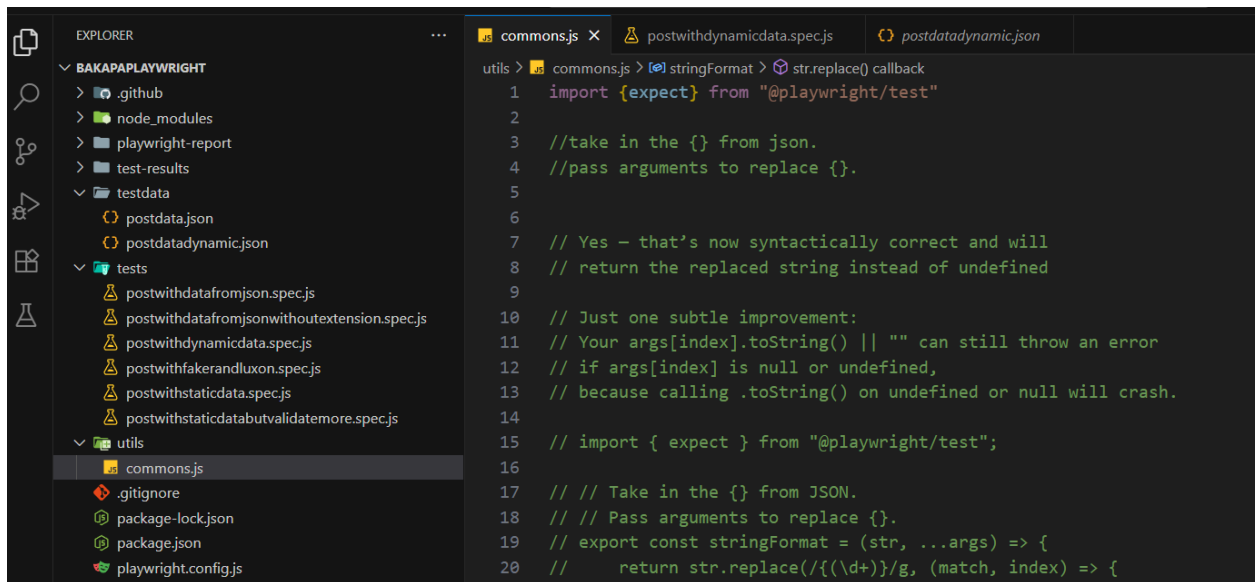
1 //
2 const { test, expect } = require("@playwright/test");
3 import { faker } from "@faker-js/faker";
4 const { DateTime } = require("luxon");
5 var dynamicPostRequest = require("../test-data/dynamicrequestbody.json");
6 import { stringFormat } from "../utils/commons";
7
8 test("Create Post api request using dynamic JSON file in playwright", async ({
9   request,
10 }) => {
11
12
13   // const dynamicRequestBody=stringFormat(JSON.stringify(dynamicPostRequest),
14   // "testers talk cypress", "testers talk js", "banana")
15
16   //call the util function first.
17   var updatedRequestBody = stringFormat(
18     JSON.stringify(dynamicPostRequest),
19     "testers talk cypress",
20     "testers talk javascript",
21     "apple"
22   );
23
24   // create post api request using playwright
25   const postAPIResponse = await request.post("/booking", {
26     //convert the string received to json object.
27     //use json.parse and pass the string.
28     data: JSON.parse(updatedRequestBody),
29   });
30
31   // validate status code
32   console.log(await postAPIResponse.json());
33
34   expect(postAPIResponse.ok()).toBeTruthy();
35   expect(postAPIResponse.status()).toBe(200);
36
37   // validate api response json obj
38   const postAPIResponseBody = await postAPIResponse.json();
39
40   expect(postAPIResponseBody.booking).toHaveProperty("firstname", "testers talk cypress");
41   expect(postAPIResponseBody.booking).toHaveProperty("lastname", "testers talk javascript");
42
43   // validate api response nested json obj
44   expect(postAPIResponseBody.booking.bookingdates).toHaveProperty(
45     "checkin",
46     "2018-01-01"
47   );
48   expect(postAPIResponseBody.booking.bookingdates).toHaveProperty(
49     "checkout",
50     "2019-01-01"
51   );
52 });

```

codesnap.dev

My version of code as bakapa should also work as per chat gpt-

[commons.js](#) util file –



```
1 import {expect} from "@playwright/test"
2
3 //take in the {} from json.
4 //pass arguments to replace {}.
5
6
7 // Yes – that's now syntactically correct and will
8 // return the replaced string instead of undefined
9
10 // Just one subtle improvement:
11 // Your args[index].toString() || "" can still throw an error
12 // if args[index] is null or undefined,
13 // because calling .toString() on undefined or null will crash.
14
15 // import { expect } from "@playwright/test";
16
17 // // Take in the {} from JSON.
18 // // Pass arguments to replace {}.
19 // export const stringFormat = (str, ...args) => {
20 //     return str.replace(/{\d+}/g, (match, index) => {
```

```
20 //     return str.replace(/{\d+}/g, (match, index) => {
21 //         return args[index] != null ? String(args[index]) : "";
22 //     });
23 // };
24
25 // This way:
26
27 // If the placeholder's argument is missing, it becomes an empty string.
28
29 // No TypeError from .toString() on undefined.
30
31
32
```

```
32
33 export const stringFormat=(str, ...args)=>{
34     return str.replace(/{\d+}/g, (match, index)=> {
35         return args[index].toString() || "" //if we dont write this we get undefined during run time for
36         //the values.
37     })
38 }
39
```

Extra function with return on both lines–

```

39
40
41 //extra function with return everywhere.
42 export const stringFormat1=(str, ...args)=>{
43     return str.replace(/{{(\d+)}}/g, (match, index)=> {
44         return args[index].toString() || ""
45     })
46 }

```

```

46 }
47
48
49 //this is also right
50 export const stringFormat2 = (str, ...args) =>
51     str.replace(/{{(\d+)}}/g, (match, index) => args[index]??.toString() || "");
52
53
54 //this is also right
55 export const stringFormat3 = (str, ...args) =>
56     str.replace(/{{(\d+)}}/g, (match, index) => {
57         return args[index]??.toString() || "";
58     });
59
60

```

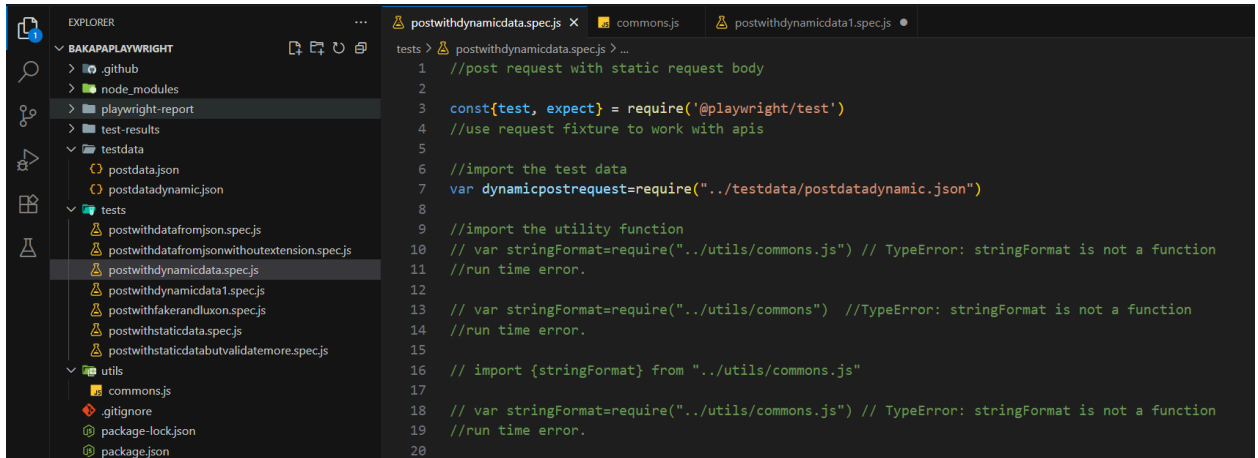
Test data--

```

1  {
2    "firstname": "{0}",
3    "lastname": "{1}",
4    "totalprice": 1000,
5    "depositpaid": true,
6    "bookingdates": {
7      "checkin": "{2}",
8      "checkout": "{3}"
9    },
10   "additionalneeds": "{4}"
11 }
12

```

Normal test file --



```
EXPLORER
BAKAPAPLAYWRIGHT
  .github
  node_modules
  playwright-report
  test-results
  testdata
    postdata.json
    postdatadynamic.json
  tests
    postwithdatafromjson.spec.js
    postwithdatafromjsonwithoutextension.spec.js
    postwithdynamicdata.spec.js
    postwithdynamicdata1.spec.js
    postwithfakerandluxon.spec.js
    postwithstaticdata.spec.js
    postwithstaticdatabutvalidatemore.spec.js
  utils
    commons.js
    .gitignore
    package-lock.json
    package.json

tests > postwithdynamicdata.spec.js > ...
1 //post request with static request body
2
3 const {test, expect} = require('@playwright/test')
4 //use request fixture to work with apis
5
6 //import the test data
7 var dynamicpostrequest=require("../testdata/postdatadynamic.json")
8
9 //import the utility function
10 // var stringFormat=require("../utils/commons.js") // TypeError: stringFormat is not a function
11 //run time error.
12
13 // var stringFormat=require("../utils/commons.js") //TypeError: stringFormat is not a function
14 //run time error.
15
16 // import {stringFormat} from "../utils/commons.js"
17
18 // var stringFormat=require("../utils/commons.js") // TypeError: stringFormat is not a function
19 //run time error.
20
```

```
20
21 //give with full extension and it works.
22 import {stringFormat} from "../utils/commons.js"
23
24 test('post with dynamic run time data', async({request}) => {
25
26     //first generate the data using function.
27     //this will generate string.
28     var updatedrequestbody=stringFormat(JSON.stringify(dynamicpostrequest),
29     "karan", "john", "2025-08-10", "2025-08-15", "wrestling")
30
```

```
30
31 console.log("Updated request body:", updatedrequestbody);
32
33     const postapiresponse=await request.post("/booking",{
34
35         //convert the string to json format.
36         //use json.parse and pass the string.
37         data:JSON.parse(updatedrequestbody) //SyntaxError: Unexpected token u in JSON at position 0
38         //run time error. because the string format is not returning anything.
39     // In JavaScript, String.prototype.replace() returns the new
40     // string but doesn't change str in place,
41     // so without returning it, your function always returns undefined.
42     //var updatedrequestbody = stringFormat(...);
43     //var updatedrequestbody = undefined;
44     //JSON.parse(undefined)
```

```

44 //JSON.parse(undefined)
45 //SyntaxError: Unexpected token u in JSON at position 0
46 //because "undefined" starts with "u".
47     });
48
49     //get the response in json format
50     const respinjson=await postapiresponse.json()
51     console.log(respinjson)
52

```

```

52
53     //validate status code and message
54     //status message like created ok etc.
55     expect(postapiresponse.ok()).toBeTruthy()
56     expect(postapiresponse.status()).toBe(200)
57
58     //validate the response fields
59     expect(respinjson.booking).toHaveProperty("firstname", "karan" )
60     expect(respinjson.booking).toHaveProperty("lastname", "john" )
61     expect(respinjson.booking).toHaveProperty("additionalneeds", "wrestling" )
62
63     //validate the response in nested json
64     expect(respinjson.booking.bookingdates).toHaveProperty("checkin", "2025-08-10" )
65     expect(respinjson.booking.bookingdates).toHaveProperty("checkout", "2025-08-15" )
66

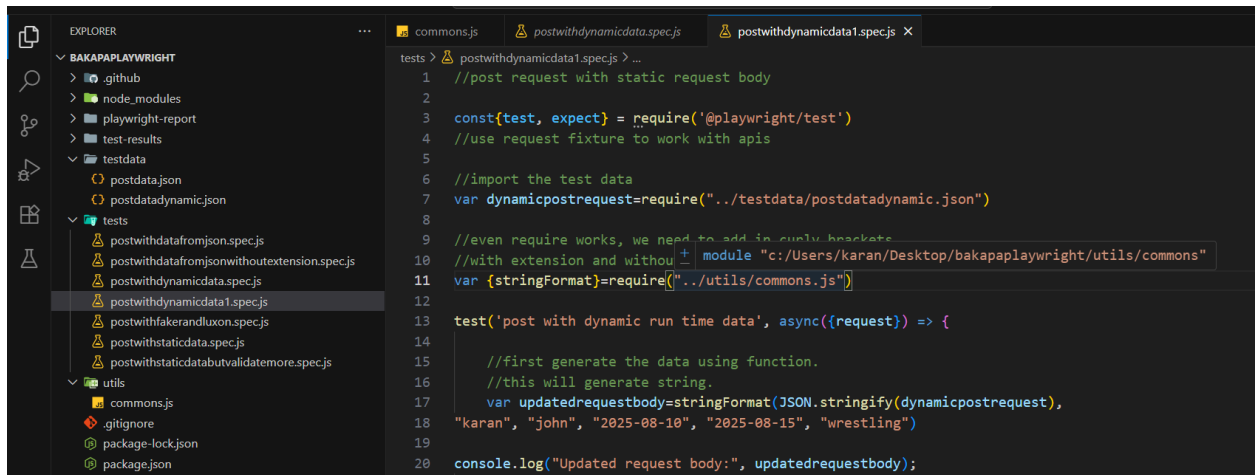
```

```

66
67     });
68
69

```

With require also it works and with extension—



```
1 //post request with static request body
2
3 const {test, expect} = require('@playwright/test')
4 //use request fixture to work with apis
5
6 //import the test data
7 var dynamicpostrequest=require("../testdata/postdatadynamic.json")
8
9 //even require works, we need to add in ourly brackets
10 //with extension and without module "c:/Users/karan/Desktop/bakapaplaywright/utlis/commons"
11 var {stringFormat}=require("../utlis/commons.js")
12
13 test('post with dynamic run time data', async({request}) => {
14
15     //first generate the data using function.
16     //this will generate string.
17     var updatedrequestbody=stringFormat(JSON.stringify(dynamicpostrequest),
18     "karan", "john", "2025-08-10", "2025-08-15", "wrestling")
19
20     console.log("Updated request body:", updatedrequestbody);
```

```
20 console.log("Updated request body:", updatedrequestbody);
21
22 const postapiresponse=await request.post("/booking",{
23
24     //convert the string to json format.
25     //use json.parse and pass the string.
26     data:JSON.parse(updatedrequestbody) //SyntaxError: Unexpected token u in JSON at position 0
27     //run time error. because the string format is not returning anything.
28     // In JavaScript, String.prototype.replace() returns the new
29     // string but doesn't change str in place,
30     // so without returning it, your function always returns undefined.
31     //var updatedrequestbody = stringFormat(...);
32     //var updatedrequestbody = undefined;
33     //JSON.parse(undefined)
34     //SyntaxError: Unexpected token u in JSON at position 0
35     //because "undefined" starts with "u".
36     });
37
```

```
36     });
37
38     //get the response in json format
39     const respinjson=await postapiresponse.json()
40     console.log(respinjson)
41
42     //validate status code and message
43     //status message like created ok etc.
44     expect(postapiresponse.ok()).toBeTruthy()
45     expect(postapiresponse.status()).toBe(200)
46
47     //validate the response fields
48     expect(respinjson.booking).toHaveProperty("firstname", "karan" )
49     expect(respinjson.booking).toHaveProperty("lastname", "john" )
50     expect(respinjson.booking).toHaveProperty("additionalneeds", "wrestling" )
51
```

```

51
52 //validate the response in nested json
53 expect(respinjson.booking.bookingdates).toHaveProperty("checkin", "2025-08-10" )
54 expect(respinjson.booking.bookingdates).toHaveProperty("checkout", "2025-08-15" )
55
56 });
57
58

```

Without extension also works—
Using require.

```

... postwithdynamicdata1.spec.js postwithdynamicdata2.spec.js X
tests > postwithdynamicdata2.spec.js > ...
1 //post request with static request body
2
3 const{test, expect} = require('@playwright/test')
4 //use request fixture to work with apis
5
6 //import the test data
7 var dynamicpostrequest=require("../testdata/postdatadynamic.json")
8
9 //even require works, we need to add in curly brackets.
10 //with extension and without extension it works.
11 var {stringFormat}=require("../utils/commons")
12
13 test('post with dynamic run time data', async({request}) => {
14
15

```

```

14
15 //first generate the data using function.
16 //this will generate string.
17 var updatedrequestbody=stringFormat(JSON.stringify(dynamicpostrequest),
18 "karan", "john", "2025-08-10", "2025-08-15", "wrestling")
19
20 console.log("Updated request body:", updatedrequestbody);
21
22 const postapiresponse=await request.post("/booking",{
23
24 //convert the string to json format.
25 //use json.parse and pass the string.
26 data:JSON.parse(updatedrequestbody) //SyntaxError: Unexpected token u in JSON at position 0
27 //run time error. because the string format is not returning anything.
28 // In JavaScript, String.prototype.replace() returns the new
29 // string but doesn't change str in place,
30 // so without returning it, your function always returns undefined.
31 //var updatedrequestbody = stringFormat(...);
32 //var updatedrequestbody = undefined;

```

```

32 //var updatedrequestbody = undefined;
33 //JSON.parse(undefined)
34 //SyntaxError: Unexpected token u in JSON at position 0
35 //because "undefined" starts with "u".
36     });
37
38     //get the response in json format
39     const respinjson=await postapiresponse.json()
40     console.log(respinjson)
41

```

```

41
42     //validate status code and message
43     //status message like created ok etc.
44     expect(postapiresponse.ok()).toBeTruthy()
45     expect(postapiresponse.status()).toBe(200)
46
47     //validate the response fields
48     expect(respinjson.booking).toHaveProperty("firstname", "karan" )
49     expect(respinjson.booking).toHaveProperty("lastname", "john" )
50     expect(respinjson.booking).toHaveProperty("additionalneeds", "wrestling" )
51

```

```

50     expect(respinjson.booking).toHaveProperty("additionalneeds", "wrestling" )
51
52     //validate the response in nested json
53     expect(respinjson.booking.bookingdates).toHaveProperty("checkin", "2025-08-10" )
54     expect(respinjson.booking.bookingdates).toHaveProperty("checkout", "2025-08-15" )
55
56 });
57
58

```