

Booking API Testing

Tested by: Akib Rahman

Base URL: <https://restful-booker.herokuapp.com>

Environment Variables:

All variables	
E Environment	
base_url	https://restful-booker.herokuapp.com
booking_id	4416
token_id	92ce615df902b5f
firstName	Elliot
lastName	Jast

Test Steps:

1. Create Booking:

- Method: **POST**
- URL Format: Base_Url/booking/
- Expected Status Code: **200 OK**
- Actual Status Code: **200 OK**

i. Body:

```
POST {{base_url}}/booking

Params Authorization Headers (9) Body Scripts Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2   "firstname" : "{{firstName}}",
3   "lastname" : "{{lastName}}",
4   "totalprice" : 111,
5   "depositpaid" : true,
6   "bookingdates" : {
7     "checkin" : "2018-01-01",
8     "checkout" : "2019-01-01"
9   },
10  "additionalneeds" : "Breakfast"
11 }
```

ii. Pre-Request Scripts:

```
Pre-request • 1 //Generating random names using Javascript
2
Post-response • 3 var firstName = pm.variables.replaceIn("{{$randomFirstName}}")
4 pm.environment.set("firstName",firstName)
5
6 var lastName = pm.variables.replaceIn("{{$randomLastName}}")
7 pm.environment.set("lastName",lastName)
```

iii. Response:

```
Body Cookies Headers (11) Test Results (1/1) 200 OK • 288 ms • 949 B
Pretty Raw Preview Visualize JSON
1 {
2   "bookingid": 4416,
3   "booking": {
4     "firstname": "Elliot",
5     "lastname": "Jast",
6     "totalprice": 111,
7     "depositpaid": true,
8     "bookingdates": {
9       "checkin": "2018-01-01",
10      "checkout": "2019-01-01"
11    },
12    "additionalneeds": "Breakfast"
13  }
14 }
```

iv. Post-Response Scripts and Result:

```
Pre-request • 1 //Creating booking id variable in environment
2 var jsonResponse = pm.response.json()
Post-response • 3 pm.environment.set("booking_id",jsonResponse.bookingid)
4
5 //Checking the status code
6 pm.test("Status code is 200", function () {
7   pm.response.to.have.status(200);
8 });

Body Cookies Headers (11) Test Results (1/1) 200 OK • 288 ms • 949 B
All Passed Skipped Failed
PASS Status code is 200
```

2. Get Booking Informations:

- Method: **GET**
- URL Format: Base_Url/booking/id
- Expected Status Code: **200 OK**
- Actual Status Code: **200 OK**

i. URL and Response:

The screenshot shows a REST client interface with a GET request to the URL `{{base_url}}/booking/{{booking_id}}`. The response is a JSON object with the following structure:

```
1  {
2    "firstname": "Jaunita",
3    "lastname": "Daniel",
4    "totalprice": 111,
5    "depositpaid": true,
6    "bookingdates": {
7      "checkin": "2018-01-01",
8      "checkout": "2019-01-01"
9    },
10   "additionalneeds": "Breakfast"
11 }
```

ii. Post-Response Scripts and Test Result:

The screenshot shows the 'Scripts' tab in a REST client. The 'Post-response' section contains the following scripts:

```
1  var jsonData = pm.response.json()
2
3  //Validating the names
4  pm.test("First Name Validation", function () {
5    pm.expect(jsonData.firstname).to.eql(pm.environment.get("firstName"))
6  })
7
8  pm.test("Last Name Validation", function () {
9    pm.expect(jsonData.lastname).to.eql(pm.environment.get("lastName"))
10 })
11
12 //Validating the status code
13 pm.test("Status code is 200", function () {
14   pm.response.to.have.status(200);
15 });
```

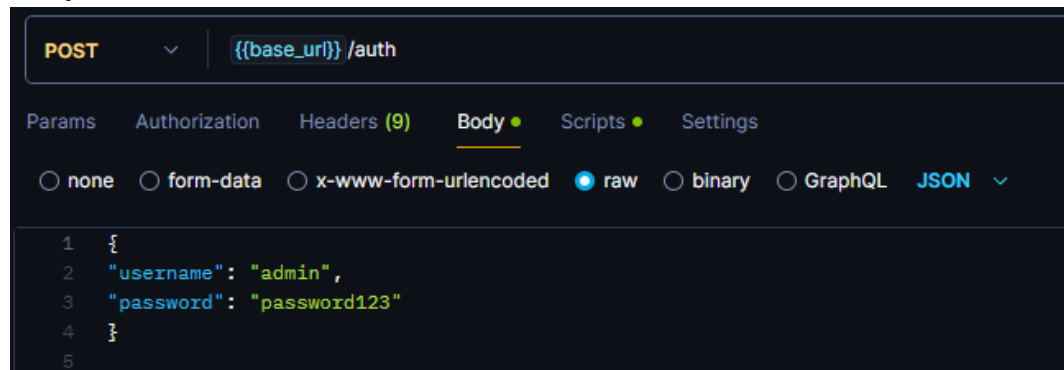
The 'Test Results' section shows the following results:

- PASS First Name Validation
- PASS Last Name Validation
- PASS Status code is 200

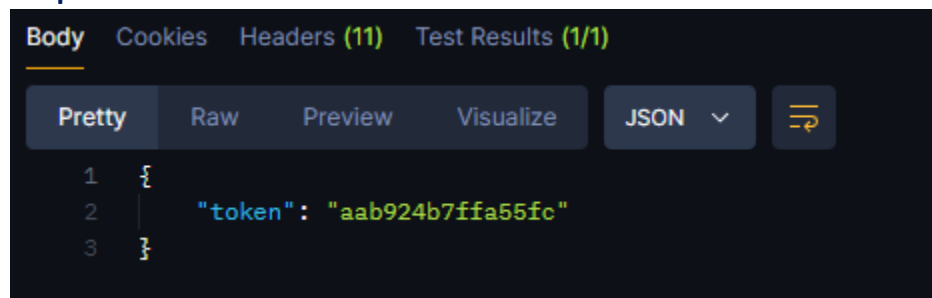
3. Creating Token:

- **Method:** POST
- **URL Format:** Base_Url/auth
- **Expected Status Code:** 200 OK
- **Actual Status Code:** 200 OK

i. Body:



ii. Response:



iii. Post-Response Scripts and Test Result:

The screenshot shows the Postman interface with the 'Scripts' tab selected. The 'Post-response' section contains the following code:

```
1 //Saving token data into environment
2 var token_data = pm.response.json()
3 pm.environment.set("token_id",token_data.token)
4
5 //Validating the status code
6 pm.test("Status code is 200", function () {
7     pm.response.to.have.status(200);
8 });
```

Below the scripts, the 'Test Results (1/1)' tab is active, showing a '200 OK' status and a duration of '1189 ms'. The test result is 'PASS' for the test 'Status code is 200'.

4. Update Booking Data:

- Method: **PUT**
- URL Format: Base_Url/booking/id
- Expected Status Code: **200 OK**
- Actual Status Code: **200 OK**

i. Header:

The screenshot shows the Postman interface with the 'Headers' tab selected. The request method is 'PUT' and the URL is '{{base_url}}/booking/{{booking_id}}'. The 'Headers' section shows a table with the following data:

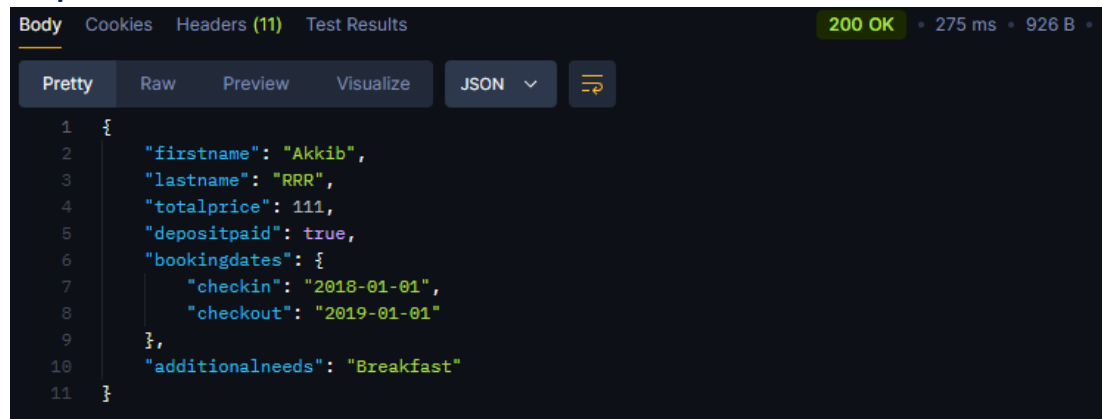
	Key	Value	Description
<input checked="" type="checkbox"/>	Cookie	token= {{token_id}}	
	Key	Value	Description

ii. Body:

The screenshot shows the Postman interface with the 'Body' tab selected. The body is a JSON object with the following structure:

```
1 {
2   "firstname" : "Akkib",
3   "lastname"  : "RRR",
4   "totalprice": 111,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin" : "2018-01-01",
8     "checkout" : "2019-01-01"
9   },
10  "additionalneeds" : "Breakfast"
11 }
```

iii. **Response:**



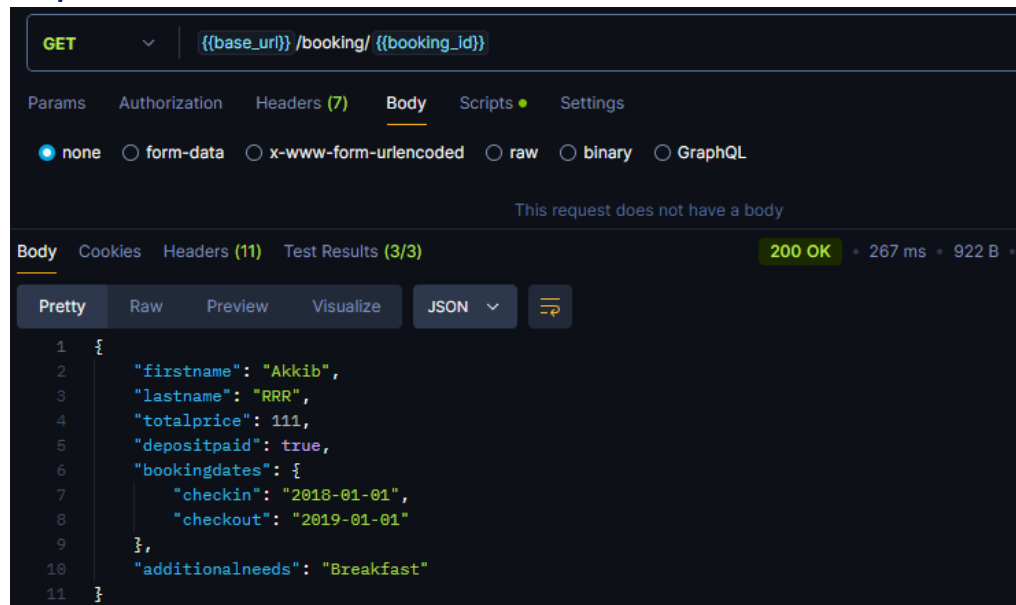
The screenshot shows a REST client interface with the 'Body' tab selected. The response status is '200 OK' with a response time of 275 ms and a size of 926 B. The JSON body is formatted as follows:

```
1 {
2   "firstname": "Akkib",
3   "lastname": "RRR",
4   "totalprice": 111,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin": "2018-01-01",
8     "checkout": "2019-01-01"
9   },
10  "additionalneeds": "Breakfast"
11 }
```

5. **Verify The Updates:**

- **Method:** GET
- **URL Format:** Base_Url/booking/id
- **Expected Status Code:** 200 OK
- **Actual Status Code:** 200 OK

i. **Response:**



The screenshot shows a REST client interface for a GET request. The URL is set to `{{base_url}} /booking/ {{booking_id}}`. The 'Body' tab is selected, and the response status is '200 OK' with a response time of 267 ms and a size of 922 B. The JSON body is formatted as follows:

```
1 {
2   "firstname": "Akkib",
3   "lastname": "RRR",
4   "totalprice": 111,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin": "2018-01-01",
8     "checkout": "2019-01-01"
9   },
10  "additionalneeds": "Breakfast"
11 }
```

ii. Post-Response Scripts and Test Result:

The screenshot shows the Postman interface for a GET request to `{{base_url}}/booking/{{booking_id}}`. The **Scripts** tab is active, displaying the following Post-response script:

```
1 var jsonData = pm.response.json()
2
3 //Validating the names update
4 pm.test("First Name Validation", function () {
5     pm.expect(jsonData.firstname).to.eql("Akkib")
6 })
7
8 pm.test("Last Name Validation", function () {
9     pm.expect(jsonData.lastname).to.eql("RRR")
10 })
11
12 //Validating the status code
13 pm.test("Status code is 200", function () {
14     pm.response.to.have.status(200);
15 });
```

The **Test Results (3/3)** tab is also active, showing three passed tests:

- PASS** First Name Validation
- PASS** Last Name Validation
- PASS** Status code is 200

The overall response status is **200 OK** with a response time of 267 ms.

6. Delete The Booking:

- **Method:** DELETE
- **URL Format:** Bas_Url/booking/id
- **Expected Status Code:** 200 CREATED
- **Actual Status Code:** 200 CREATED

i. Header:

The screenshot shows the Postman interface for a DELETE request to `{{base_url}}/booking/{{booking_id}}`. The **Headers (8)** tab is active, displaying the following headers:

Key	Value	Description
<input checked="" type="checkbox"/> Cookie	token= {{token_id}}	
Key	Value	Description

There are 7 hidden headers.

ii. **Post Response and Test Result:**

The screenshot displays the Postman interface with the 'Scripts' tab selected. The 'Post-response' section contains a test script:

```
1 pm.test("Status code is 201", function () {  
2     pm.response.to.have.status(201);  
3 });
```

Below the script, the 'Test Results (1/1)' section shows a single result:

- 201 Created** • 272 ms • 757 B •

At the bottom, a summary bar indicates the overall status:

- PASS** Status code is 201