

## Task#2:

Given the API specification documentation attached above, create an API test automation suite using any tool of your choice to cover all positive and negative scenarios of the CreateBooking endpoint specified in the API documentation.

(API testing Code is Shared to GitHub Repository)

[https://github.com/chamodya-layaji97/QA\\_Engineer\\_PracticalTest.git](https://github.com/chamodya-layaji97/QA_Engineer_PracticalTest.git)

### 1.Get Token Test

```
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});

// Verify the response body contains a token
pm.test("Response body contains a token", function () {
  pm.response.to.have.jsonBody('token');
});

pm.test("Request body contains username property", function () {
  var requestBody = JSON.parse(pm.request.body.raw);
  pm.expect(requestBody).to.have.property('username');
});

pm.test("Request body contains password property", function () {
  var requestBody = JSON.parse(pm.request.body.raw);
  pm.expect(requestBody).to.have.property('password');
});
```

The screenshot shows the Postman application interface. At the top, the request is a POST to `https://restful-booker.herokuapp.com/auth`. The 'Tests' tab is active, displaying the following JavaScript code:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 // Verify the response body contains a token
5 pm.test("Response body contains a token", function () {
6   pm.response.to.have.jsonBody('token');
7 });
```

Below the code, the 'Test Results (4/4)' section shows four passed tests:

- PASS Status code is 200
- PASS Response body contains a token
- PASS Request body contains username property
- PASS Request body contains password property

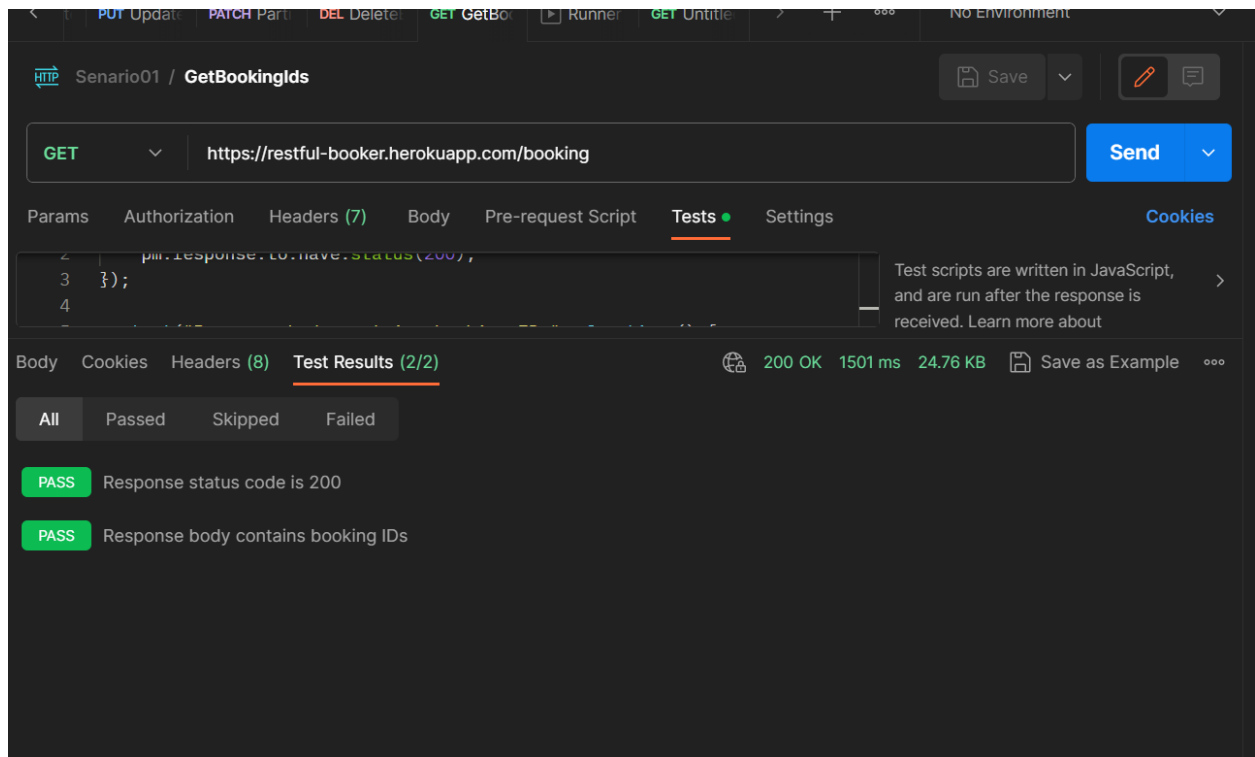
The status bar at the bottom indicates a 200 OK response, 1010 ms execution time, and 271 B body size.

## 2. GetBookingIds

```
pm.test("Response status code is 200", function () {
  pm.response.to.have.status(200);
});

pm.test("Response body contains booking IDs", function () {
  var responseBody = pm.response.json();
  pm.expect(responseBody).to.be.an('array').that.is.not.empty;

  for (var i = 0; i < responseBody.length; i++) {
    pm.expect(responseBody[i]).to.have.property('bookingid');
    pm.expect(responseBody[i].bookingid).to.be.a('number');
  }
});
```



### 3. GetBooking

```
pm.test("Response status code is 200", function () {
  pm.response.to.have.status(200);
});

pm.test("Response body has correct fields", function () {
  var responseBody = pm.response.json();

  pm.expect(responseBody).to.have.property("firstname");
  pm.expect(responseBody).to.have.property("lastname");
  pm.expect(responseBody).to.have.property("totalprice");
  pm.expect(responseBody).to.have.property("depositpaid");
  pm.expect(responseBody).to.have.property("bookingdates");
  pm.expect(responseBody).to.have.property("additionalneeds");

  pm.expect(responseBody.firstname).to.be.a("string");
  pm.expect(responseBody.lastname).to.be.a("string");
  pm.expect(responseBody.totalprice).to.be.a("number");
  pm.expect(responseBody.depositpaid).to.be.a("boolean");
});
```

```

pm.expect(responseBody.bookingdates).to.be.an("object");
pm.expect(responseBody.additionalneeds).to.be.a("string");
});

```

Scenario01 / GetBooking

GET https://restful-booker.herokuapp.com/booking/2

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

```

1 pm.test("Response status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Response body has correct fields", function () {
6   var responseBody = pm.response.json();
7
8   pm.expect(responseBody).to.have.property("firstname");
9   pm.expect(responseBody).to.have.property("lastname");

```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about [tests scripts](#)

Snippets

- [Get an environment variable](#)
- [Get a global variable](#)

Body Cookies Headers (8) Test Results (2/2)

200 OK 1082 ms 415 B Save as Example

All Passed Skipped Failed

PASS Response status code is 200

PASS Response body has correct fields

## 4.CreateBooking

```

pm.test("Response status code is 200", function () {
  pm.response.to.have.status(200);
});

pm.test("Response body has booking ID and details", function () {
  var responseBody = pm.response.json();

  pm.expect(responseBody).to.have.property("bookingid");
  pm.expect(responseBody.bookingid).to.be.a("number");
  pm.expect(responseBody).to.have.property("booking").that.is.an("object");

```

```

    var booking = responseBody.booking;
    pm.expect(booking).to.have.property("firstname").that.is.a("string");
    pm.expect(booking).to.have.property("lastname").that.is.a("string");
    pm.expect(booking).to.have.property("totalprice").that.is.a("number");
    pm.expect(booking).to.have.property("depositpaid").that.is.a("boolean");
    pm.expect(booking).to.have.property("bookingdates").that.is.an("object");
    pm.expect(booking).to.have.property("additionalneeds").that.is.a("string");

    var bookingDates = booking.bookingdates;
    pm.expect(bookingDates).to.have.property("checkin").that.is.a("string");
    pm.expect(bookingDates).to.have.property("checkout").that.is.a("string");
  });

  // Positive Scenario - Successful Booking Creation
  pm.test("Create Booking - Successful", function () {
    // Verify the response status code is 200
    pm.response.to.have.status(200);

    // Verify the response body contains the booking ID and details
    pm.expect(pm.response.json()).to.have.property("bookingid");
    pm.expect(pm.response.json().bookingid).to.be.a("number");
    pm.expect(pm.response.json()).to.have.property("booking").that.is.an("object");
  });

  // Negative Scenario - Missing Required Field
  pm.test("Create Booking - Missing Required Field", function () {
    // Verify the response status code is 400
    pm.response.to.have.status(400);

    // Verify the response body contains the error message for the missing field
    pm.expect(pm.response.json().error).to.equal("Missing required field(s)");
  });

  // Negative Scenario - Invalid Data Type
  pm.test("Create Booking - Invalid Data Type", function () {
    // Verify the response status code is 400
    pm.response.to.have.status(400);

    // Verify the response body contains the error message for the invalid data type
    pm.expect(pm.response.json().error).to.equal("Invalid data type");
  });

```

```
// Negative Scenario - Invalid Date Format
pm.test("Create Booking - Invalid Date Format", function () {
  // Verify the response status code is 400
  pm.response.to.have.status(400);

  // Verify the response body contains the error message for the invalid date format
  pm.expect(pm.response.json().error).to.equal("Invalid date format");
});
```

The screenshot displays the TMOSS REST client interface. On the left, a sidebar shows a collection named 'Scenario01' containing several API endpoints, with 'POST CreateBooking' selected. The main panel shows a POST request to 'https://restful-booker.herokuapp.com/booking' with the 'Content-Type' header set to 'application/json'. Below the request details, the 'Test Results (3/6)' section is visible, showing a summary of test outcomes.

Result	Test Description
PASS	Response status code is 200
PASS	Response body has booking ID and details
PASS	Create Booking - Successful
FAIL	Create Booking - Missing Required Field   AssertionError: expected response to have status code 400 but got 200
FAIL	Create Booking - Invalid Data Type   AssertionError: expected response to have status code 400 but got 200
FAIL	Create Booking - Invalid Date Format   AssertionError: expected response to have status code 400 but got 200