API Testing Strategy and Execution

Project: Test Plan for RESTful API Testing – Restful Booker

Version: 1.0 (Initial Draft)

Created By: Dilan Meegoda

Date: 18th March 2025

1 Introduction

This document outlines the API testing strategy and test plan to verify the Restful-booker API. The primary goal is to evaluate the correct functionality, integration, end to end workflows and negative test scenarios of the listed API. Also, this document contains the test execution results of test run records from executed test cases, screenshots and logs from Postman test runs and response time analysis records.

By following this structured approach, the test plan ensures that API is tested for reliability stability and performance with attached screenshots of test execution.

2 Scope of Testing

2.1 In-Scope

The following API functions will be tested

- Authentication - Validate the token-based authentication and security.
- Booking Management - Validate following operations:
    - GET - Get booking details
    - POST - Create a booking
    - PUT - Complete update of the booking
    - PATCH - Partial update of the booking
    - DELETE - Remove booking details
- Health Check - Verify the API up and running
- Errors and status codes - Validate proper error and status codes for invalid data
- Performance Testing - Calculate the response time and API stability under load.

2.2 Out of Scope

The following aspects will not be tested under API test plan.

- Database validations
- UI Testing
- Third party integrations (Only the given API will be tested)

3 Test Objectives

The objective of this test plan is to validate the functionality, reliability, security and performance of the restful broker API. The testing will ensure that:

1. All API endpoints function as expected.

2. All API responses are accurate and consistent.
3. Error handling work as expected.
4. Authentication and authorization are enforced.
5. API can maintain stability under load.
6. All the defects and improvements are documented.

4 Test Strategy

4.1 Testing Approach

The test approach defines a methodology to be followed and continue for validating Restful booker API, this approach ensures its functional accuracy, integration, conciseness, security and performance. Test approach include manual, automated and performance testing techniques to achieve the complete API validation.

4.2 Test Levels

4.2.1 Functional Testing

The primary focus of this test level is to ensure API endpoints behave as expected. This approach contains:

- Validating GET, POST, PUT, PATCH and DELETE methods.
- Verify the request response structures:
    - Headers – Content type, authorization tokens
    - Payload – Request body, parameters
- Validate correct status code
    - 200 for successful request
    - 201 for resource successful creation
    - 400 for bad request due to invalid request
    - 401 for unauthorized access or invalid authentication.
    - 404 for accessing nonexistent resource.
- Executing tests via Postman for validation and verification

4.2.2 Data Validation and Error Handling

Data validation ensures that API returns accurate, structured, and consistent data and error handling ensure the failure control.

Data Validation Approach:

- Validate correct data is retuned for different booking scenarios
- Validate API response format against expected JSON schema

- Verify API responses structure against missing, incorrect and unexpected request parameters.
- Validate API response consistency across multiple test executions.

Error Handling Verification:

- Validate error messages are meaningful and aligned with HTTP status codes
- Validate API behavior for missing, incorrect, and unexpected inputs.

4.2.3 Integration Testing

Integration testing ensure that multiple API endpoints work together and maintain state consistency.

Integration Testing Objectives:

- Validate dependencies and interactions between multiple API endpoints.
- Ensure data consistency when transitioning between different API calls.

Integration Test Scenarios:

- End to End Workflow Validation
    - Authenticate (POST /auth) to generate an access token.
    - Create a booking (POST /booking) and store the returned booking ID.
    - Retrieve the created booking (GET /booking/{id}) and verify details.
    - Update the booking (PUT /booking/{id}) and confirm changes.
    - Delete the booking (DELETE /booking/{id}) and verify removal.
- State Consistency Testing
    - Ensure data remains consistent across multiple requests.
- Authentication & Authorization Impact
    - Create, update, or delete a booking without authentication and should return 401 status code
- Error Handling & Dependency Failures
    - Send a GET request to retrieve non existing booking data (GET /booking/978654) and should return 404 status code

4.2.4 Security Testing

Security is an important aspect in API testing by ensuring that the API protect data, block unauthorized access and follow security best practices.

Security Testing Objectives:

- Validate the authentication and authorization
- Ensure only the authenticated users can access protected resources
- Prevent unauthorized modifications

Security Test Scenarios:

- Verify that the "/auth" endpoint correctly generates tokens.
- Ensure valid authentication tokens are required for modifying booking data
  - Attempt to create/update/delete a booking without a token should return 401 status code.
  - An expired token should return 403 status code
- Verify that API does not expose sensitive user data

4.2.5 Performance Testing

Performance testing verify the API stability, responsiveness and scalable over different load conditions

Testing Objectives:

- Measure API response time under different load conditions like different concurrent user values.
- Verify the API can handle concurrent users efficiently without performance impact
- Identify potential bottlenecks and performance failures under stress tests

Type of Performance Tests:

- Load testing by sending 50,100, 500 user requests for https://restful-booker.herokuapp.com/booking endpoint.
- Stress testing by gradually increasing the requests until API fails to response or until it degrades the response time.
- Spike testing by increasing requests per second without observing any performance impacts.

Performance Metrics to Capture:

- Response Time
- Throughput
- Error Rate
- Latency

4.2.6 Test Execution and Reporting Approach

Test Execution Process:

- Postman Collection Execution
- JMeter tool for performance testing
- Error logging and debugging

Reporting Approach:

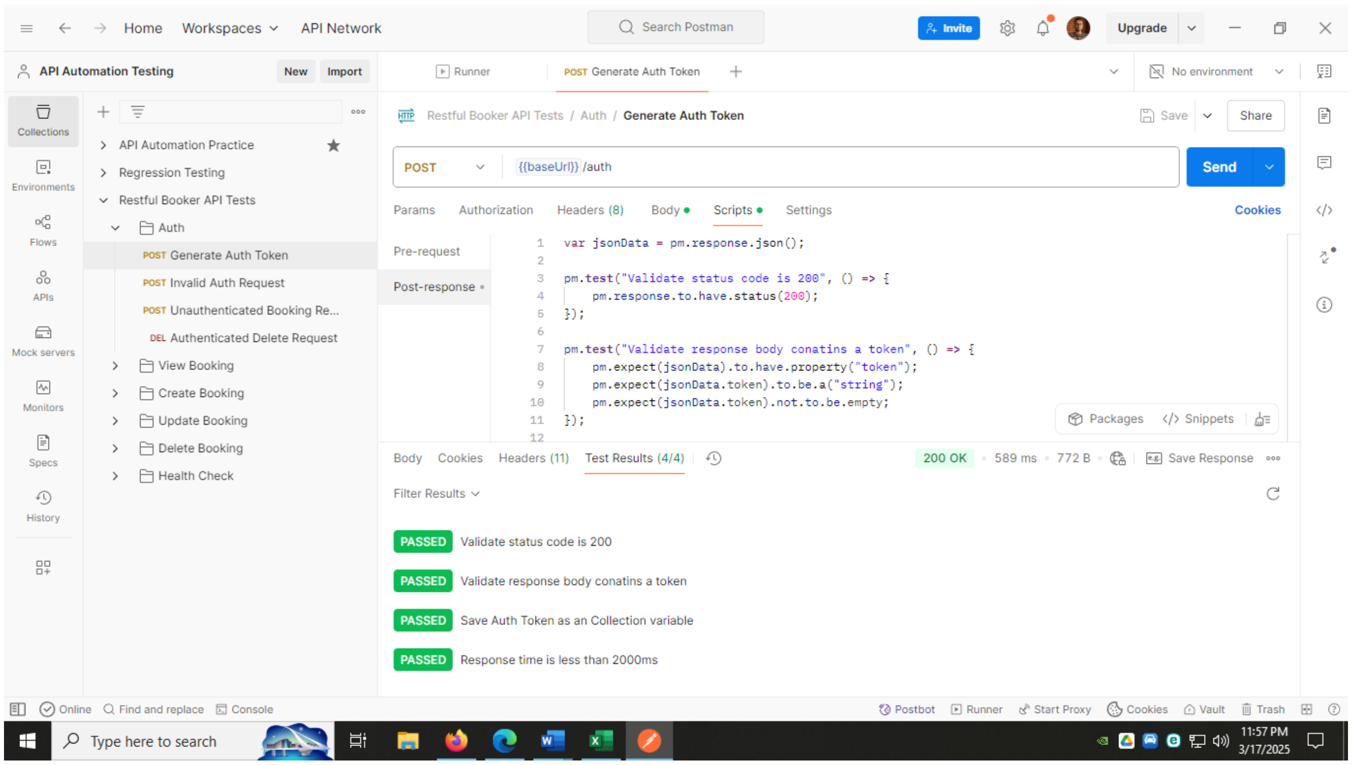
- API Response Validation
- Defect Logging
- Performance Metrics Report
- Test Summary Report

5 API Test Cases

5.1 Authentication Test Cases (https://restful-booker.herokuapp.com/auth)

Test Case ID	**API-TC-001**
Test Scenario	Verify authentication token generation with valid credentials.
API Endpoint	https://restful-booker.herokuapp.com/auth
Request Type	POST
Priority	P1 (High Priority)
Preconditions	API should be available and accept requests
Test Data	Username: admin Password: password123
Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. In the request body add valid credentials {"username": "admin", "password": "password123"} 4. Click Send button 5. Validate the response status code 6. Validate the response body content 7. Validate the response time
Expected Result	API should return 200 OK response. Response body should contain the generate token Response token should be a string type The response time should be less than 2000ms

Actual Result	The API returned 200 OK response status. The response body contained a "token" property with a non-empty value. The token was successfully stored as a collection variable in Postman. The response time was 1.40s (1400ms), which is within the acceptable limit of less than 2000ms. All Postman tests passed successfully.
Test Status	Passed



POST {{baseUrl}} /auth

```
var jsonData = pm.response.json();

pm.test("Validate status code is 200", () => {
    pm.response.to.have.status(200);
});

pm.test("Validate response body conatins a token", () => {
    pm.expect(jsonData).to.have.property("token");
    pm.expect(jsonData.token).to.be.a("string");
    pm.expect(jsonData.token).not.to.be.empty;
});
```

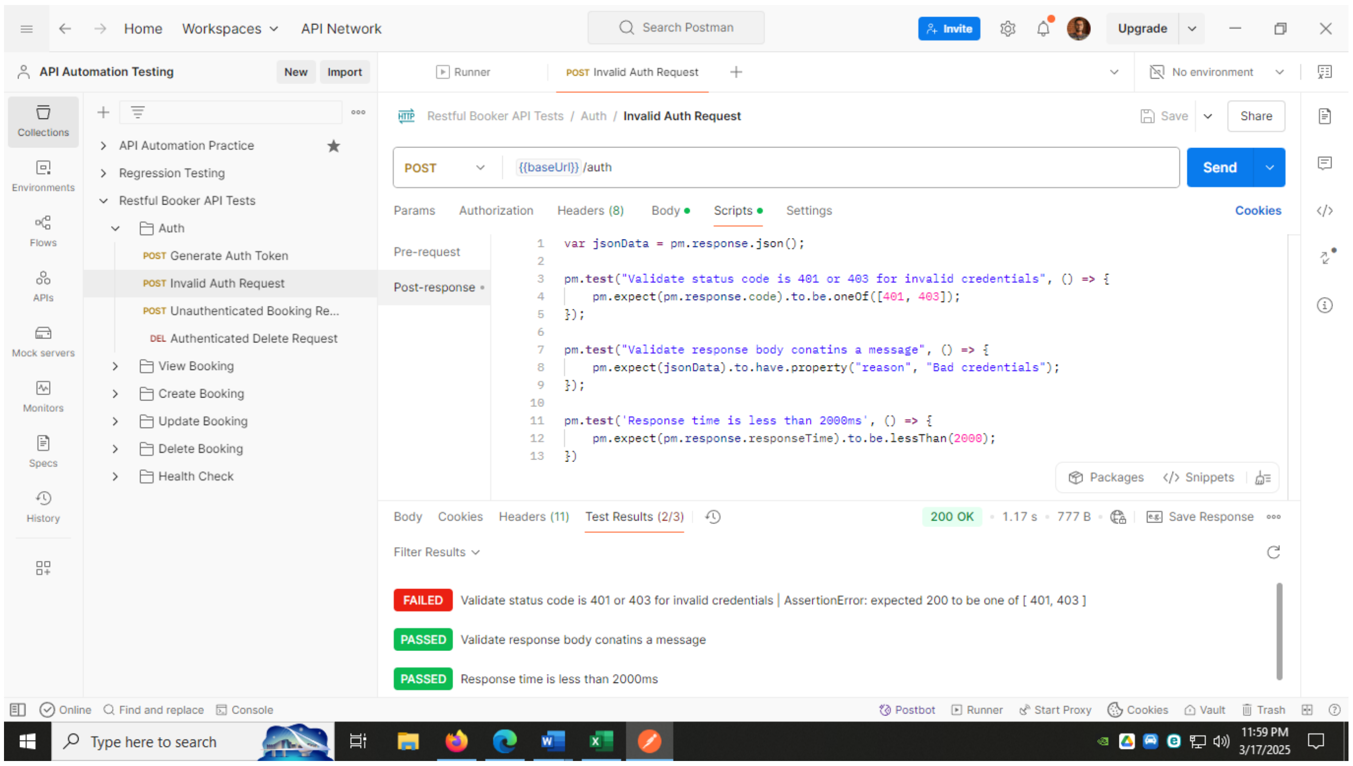
200 OK · 589 ms · 772 B

PASSED Validate status code is 200
PASSED Validate response body conatins a token
PASSED Save Auth Token as an Collection variable
PASSED Response time is less than 2000ms

Figure 1: Execution proof for API-TC-001 (Authentication Token Generation)

Test Case ID	**API-TC-002**
Test Scenario	Verify authentication token will not be generated with invalid credentials.
API Endpoint	https://restful-booker.herokuapp.com/auth
Request Type	POST
Priority	P1 (High Priority)
Preconditions	API should be available and accept requests

Test Data	Username: test Password: test123
Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. In the request body add valid credentials {"username": "test", "password": "test123"} 4. Click Send button 5. Validate the response status code 6. Validate the response body content 7. Validate the response time
Expected Result	API should return 401 or 403 status code without generated token on response body Appropriate error message should contain on the response body The response time should be less than 2000ms
Actual Result	API returned 200 OK with error message {“reason": "Bad credentials”} The response time was 1.05s (1050ms), which is within the acceptable limit of less than 2000ms. One out of three Postman test cases failed, and two test cases passed.
Test Status	Failed

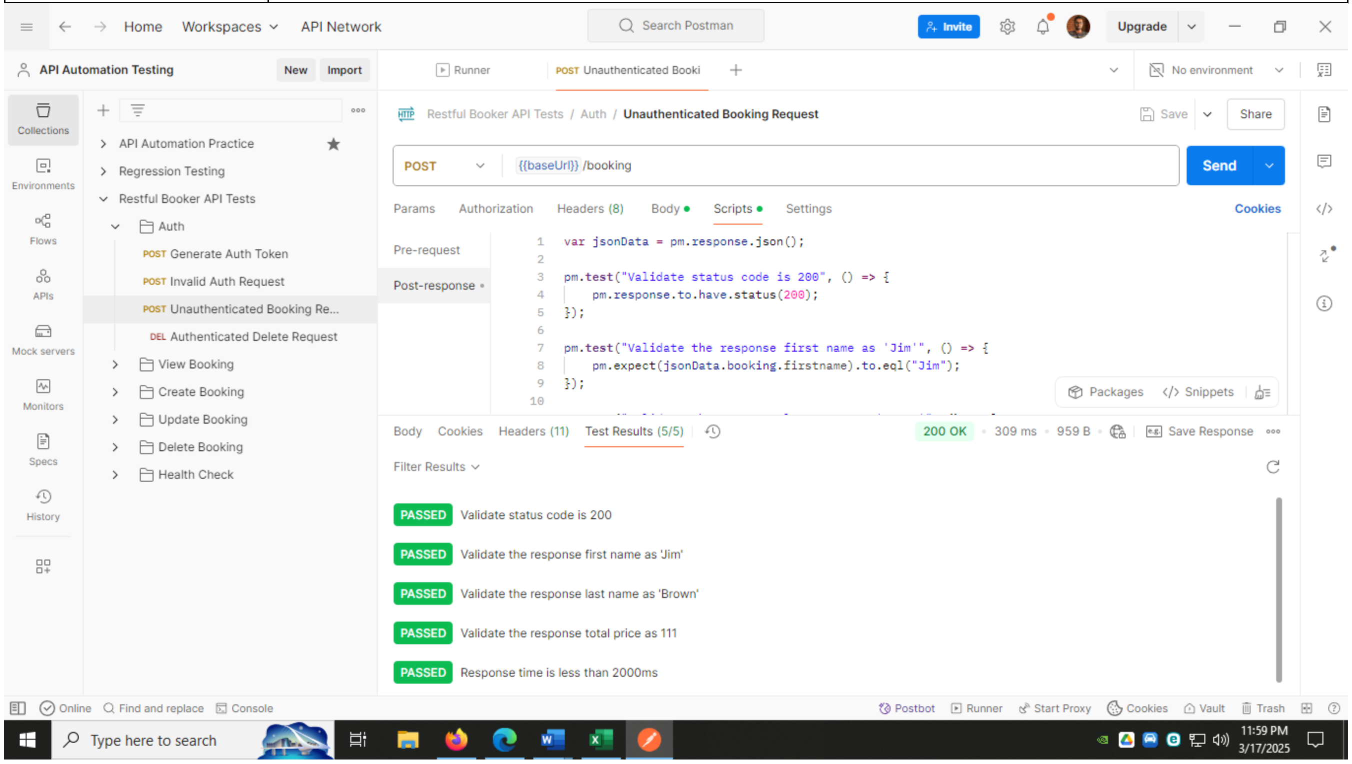


```
var jsonData = pm.response.json();

pm.test("Validate status code is 401 or 403 for invalid credentials", () => {
    pm.expect(pm.response.code).to.be.oneOf([401, 403]);
});

pm.test("Validate response body conatins a message", () => {
    pm.expect(jsonData).to.have.property("reason", "Bad credentials");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

```
FAILED  Validate status code is 401 or 403 for invalid credentials | AssertionError: expected 200 to be one of [ 401, 403 ]
PASSED  Validate response body conatins a message
PASSED  Response time is less than 2000ms
```

Figure 2: Execution proof for API-TC-002 (Invalid Auth Request)

Test Case ID	**API-TC-003**
Test Scenario	Verify that an authentication token is not required to create Booking
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	POST
Priority	P2 (Medium Priority)
Preconditions	API should be available and accept requests
Test Data	First Name: Jim Last Name: Brown Total Price: 111 Deposit Paid: true Booking Dates: 2018-01-01 and 2018-02-01 Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. In the request body add booking data without authentication token 4. Click Send button 5. Validate the response body content and status code
Expected Result	API should return 200 OK or 201 Created status code
Actual Result	API returned 200 OK and response body first name, last name and total price data matches with test data. The response time was 257ms, which is within the acceptable limit
Test Status	Passed



POST {{baseUrl}} /booking

```
var jsonData = pm.response.json();

pm.test("Validate status code is 200", () => {
    pm.response.to.have.status(200);
});

pm.test("Validate the response first name as 'Jim'", () => {
    pm.expect(jsonData.booking.firstname).to.eql("Jim");
});
```

Test Results (5/5) — 200 OK · 309 ms · 959 B

PASSED Validate status code is 200

PASSED Validate the response first name as 'Jim'

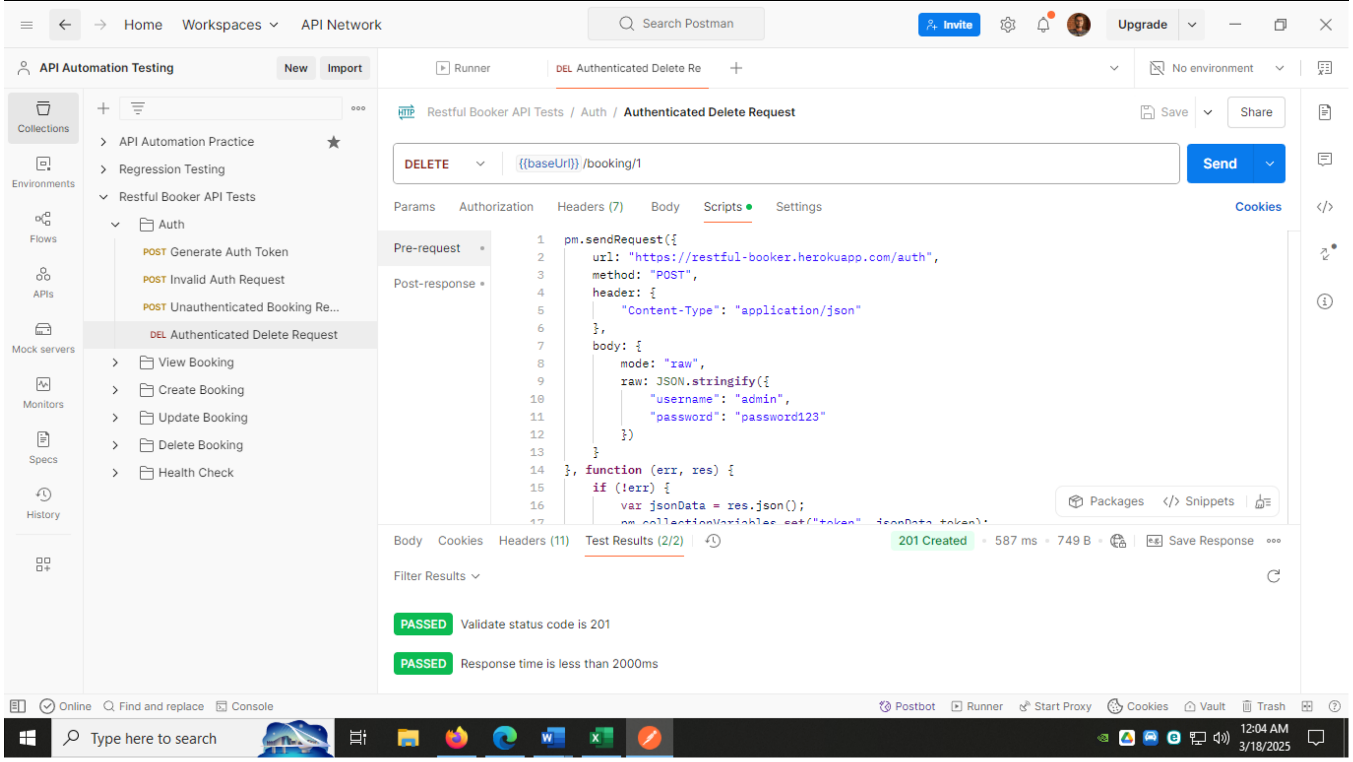
PASSED Validate the response last name as 'Brown'

PASSED Validate the response total price as 111

PASSED Response time is less than 2000ms

Figure 3: Execution proof for API-TC-003 (Unauthenticated Booking Request)

Test Case ID	**API-TC-004**
Test Scenario	Verify that a booking can only be deleted using a valid authentication token
API Endpoint	https://restful-booker.herokuapp.com/booking/1
Request Type	DELETE
Priority	P1 (High Priority)
Preconditions	API should be available and accept requests
Test Data	Authentication Token: Valid token should be obtained
Test Steps	1. Open Postman 2. Generate an authentication token using correct API endpoint • The Pre-request Script will automatically send a POST request to obtain a valid token. 3. Send a DELETE request to API endpoint without authentication token or invalid authentication token 4. Validate that the API returns status 403 Forbidden 5. Send a DELETE request to API endpoint with valid authentication token 6. Validate that the API returns 201 Created by confirming deletion.
Expected Result	API should return 201 Created responses when deleting with a valid authentication token. API should return 403 Forbidden responses when deleting without a valid authentication token.
Actual Result	API returned 201 Created with valid authentication token and returned 403 Forbidden without valid authentication token
Test Status	Passed



DELETE {{baseUrl}} /booking/1

```
pm.sendRequest({
    url: "https://restful-booker.herokuapp.com/auth",
    method: "POST",
    header: {
        "Content-Type": "application/json"
    },
    body: {
        mode: "raw",
        raw: JSON.stringify({
            "username": "admin",
            "password": "password123"
        })
    }
}, function (err, res) {
    if (!err) {
        var jsonData = res.json();
        pm.collectionVariables.set("token", jsonData.token);
```

201 Created · 587 ms · 749 B

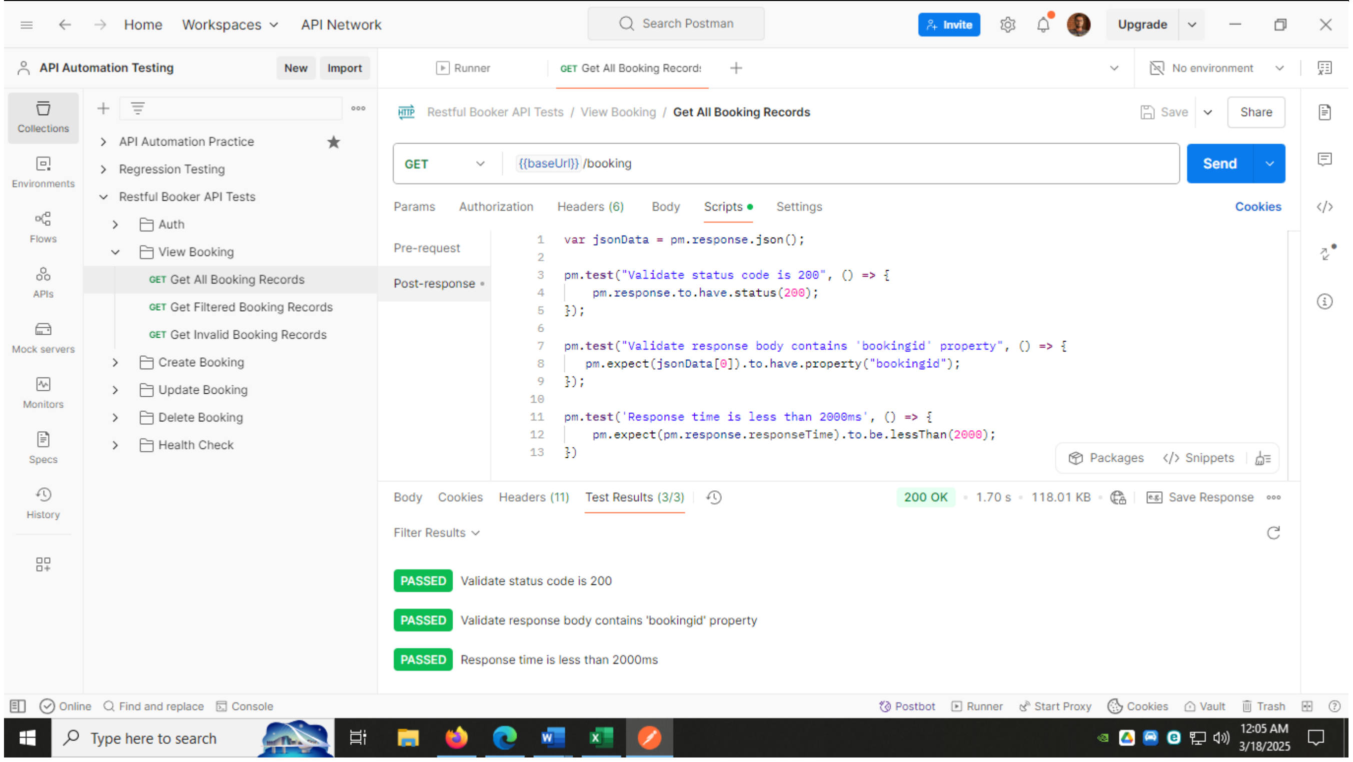
PASSED Validate status code is 201

PASSED Response time is less than 2000ms

Figure 4: Execution proof for API-TC-004 (Authenticated Delete Request)

5.2 Booking Test Cases (https://restful-booker.herokuapp.com/booking)

Test Case ID	**API-TC-005**
Test Scenario	Verify that all booking IDs can be retrieved successfully.
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	GET
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests.
Test Data	N/A
Test Steps	1. Open Postman 2. Create a GET request to API endpoint 3. Click Send button. 4. Validate that the response contains a list of booking IDs in JSON format 5. Validate that the API returns 200 OK.
Expected Result	API should return 200 OK responses status The response should contain a list of booking IDs in JSON format.
Actual Result	API returned 200 OK status code with "bookingid" property on response body
Test Status	Passed



GET {{baseUrl}} /booking

```
var jsonData = pm.response.json();

pm.test("Validate status code is 200", () => {
    pm.response.to.have.status(200);
});

pm.test("Validate response body contains 'bookingid' property", () => {
   pm.expect(jsonData[0]).to.have.property("bookingid");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

200 OK · 1.70 s · 118.01 KB

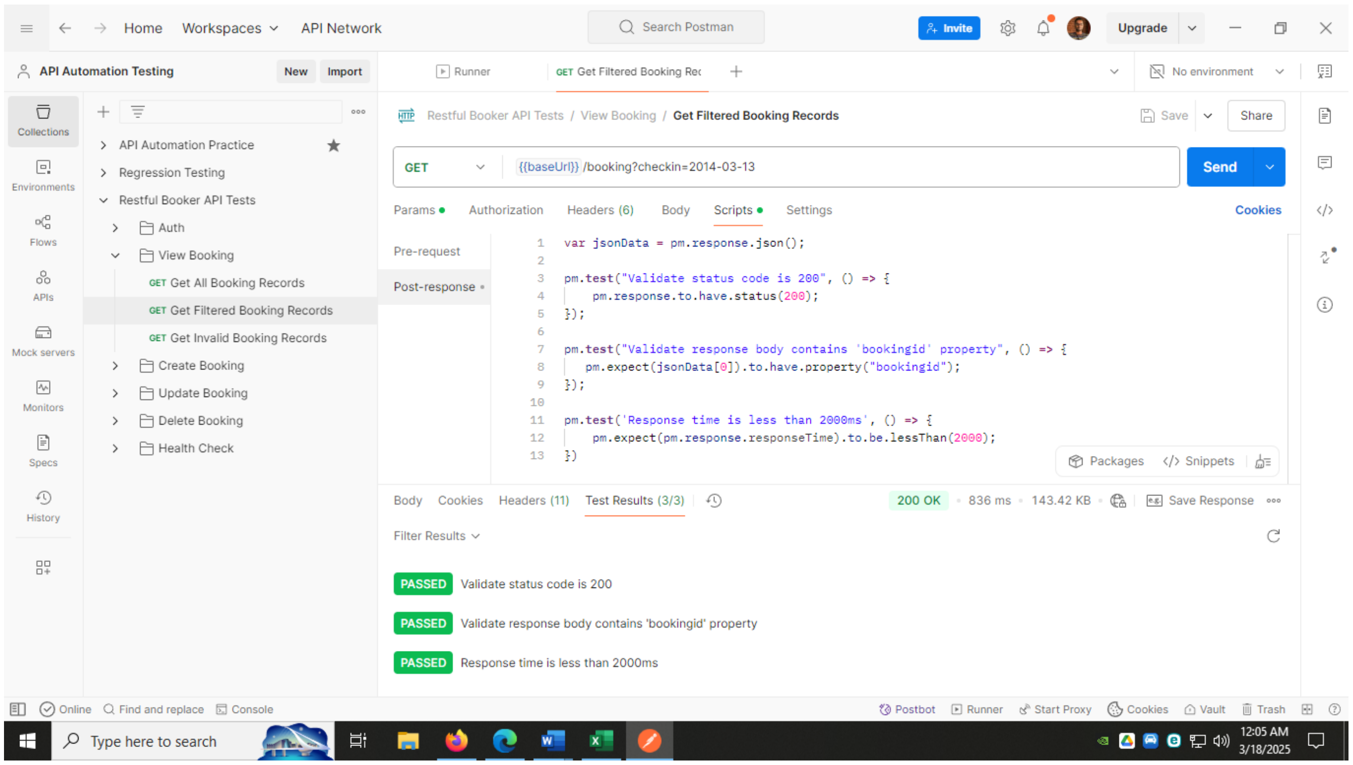
PASSED Validate status code is 200

PASSED Validate response body contains 'bookingid' property

PASSED Response time is less than 2000ms

Figure 5: Execution proof for API-TC-005 (Get All Booking Records)

Test Case ID	**API-TC-006**
Test Scenario	Verify that selected booking IDs can be retrieved successfully.
API Endpoint	https://restful-booker.herokuapp.com/booking?checkin=2014-03-13
Request Type	GET
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests with query parameters
Test Data	Check in Date: 2014-03-13
Test Steps	1. Open Postman 2. Create a GET request to API endpoint with query parameter of checking date 3. Click Send button. 4. Validate that the response contains booking IDs filtered by check-in date 5. Validate that the API returns 200 OK
Expected Result	API should return 200 OK responses status The response should contain a filtered list of booking IDs in JSON format.
Actual Result	API returned 200 OK status code with “bookingid” property on response body
Test Status	Passed



GET {{baseUrl}} /booking?checkin=2014-03-13

```
var jsonData = pm.response.json();

pm.test("Validate status code is 200", () => {
    pm.response.to.have.status(200);
});

pm.test("Validate response body contains 'bookingid' property", () => {
   pm.expect(jsonData[0]).to.have.property("bookingid");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

200 OK · 836 ms · 143.42 KB

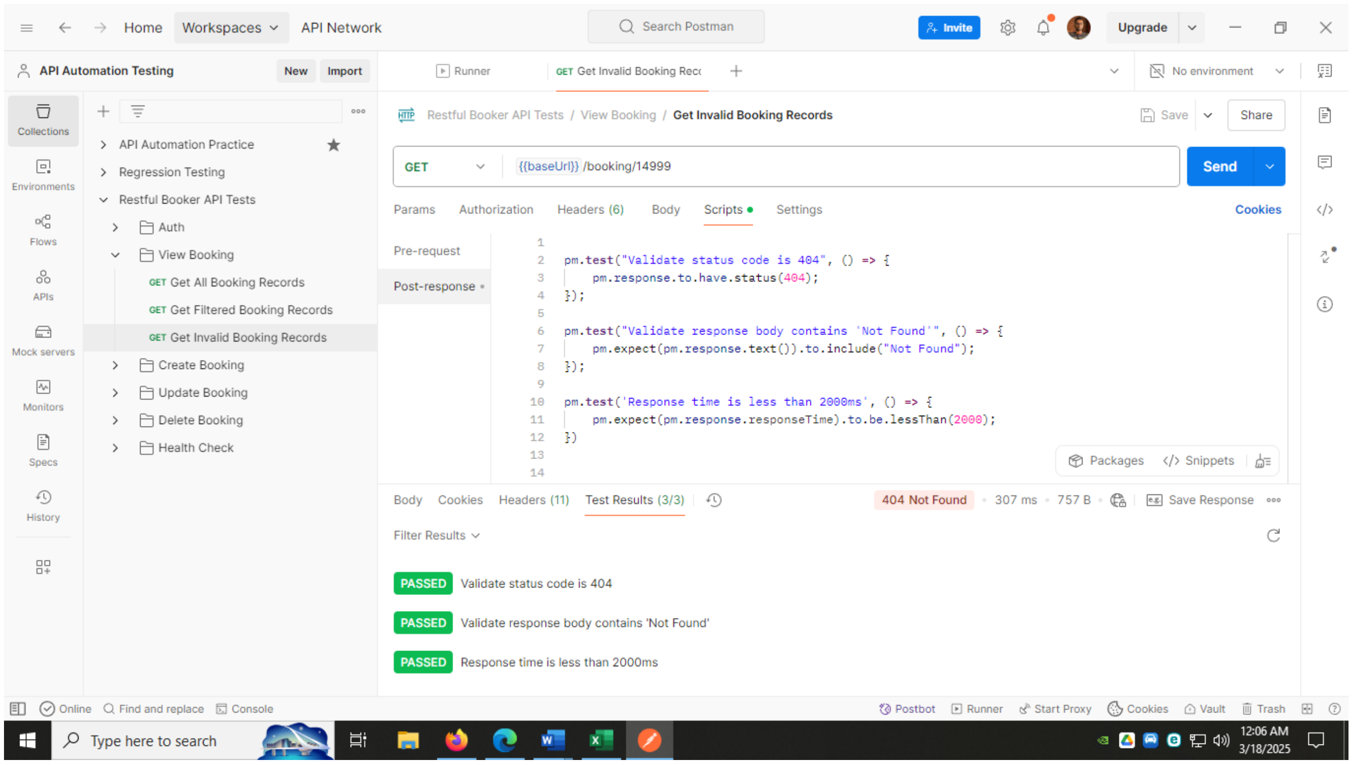
PASSED Validate status code is 200

PASSED Validate response body contains 'bookingid' property

PASSED Response time is less than 2000ms

Figure 6: Execution proof for API-TC-006 (Get Filtered Booking Records)

Test Case ID	**API-TC-007**
Test Scenario	Verify that the API returns an appropriate response when retrieving booking details with an invalid booking ID
API Endpoint	https://restful-booker.herokuapp.com/booking/199
Request Type	GET
Priority	P2 (Medium Priority)
Preconditions	API should be available and accept requests to retrieve booking details.
Test Data	Booking ID: 199
Test Steps	1. Open Postman 2. Create a GET request to API endpoint with an invalid booking ID 3. Click Send button. 4. Validate that the response contains Not Found text instead booking details 5. Validate that the API returns 404 Not Found
Expected Result	API should return 404 Not Found The response body should contain an "Not Found" indicating that the booking ID does not exist.
Actual Result	API returned 404 Not found status code with "Not Found" text on response body
Test Status	Passed



Restful Booker API Tests / View Booking / Get Invalid Booking Records

GET {{baseUrl}} /booking/14999

```
pm.test("Validate status code is 404", () => {
    pm.response.to.have.status(404);
});

pm.test("Validate response body contains 'Not Found'", () => {
    pm.expect(pm.response.text()).to.include("Not Found");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

Body Cookies Headers (11) Test Results (3/3) — 404 Not Found · 307 ms · 757 B

PASSED Validate status code is 404

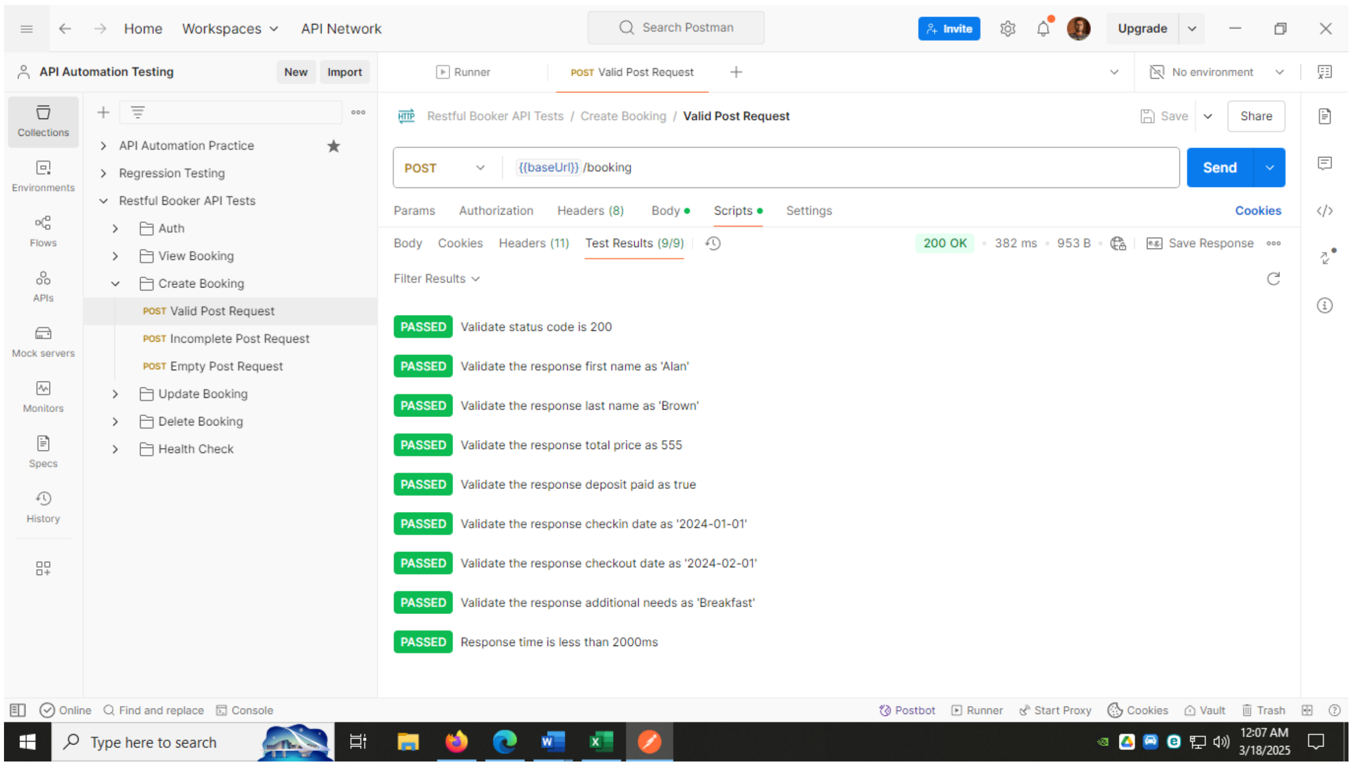
PASSED Validate response body contains 'Not Found'

PASSED Response time is less than 2000ms

Figure 7: Execution proof for API-TC-007 (Get Invalid Booking Record)

5.3 Create Booking Test Cases (https://restful-booker.herokuapp.com/booking)

Test Case ID	**API-TC-008**
Test Scenario	Verify that a booking can be created successfully with valid data.
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	POST
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests.
Test Data	First Name: Alan Last Name: Brown Total Price: 555 Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. Set Content type to application/json in the request header section as key value pair. 4. Add a valid JSON request body with all required fields 5. Click Send button. 6. Validate that the response to contain correct status code, response body and response structure
Expected Result	API should return 200 OK responses status The response body should contain the correct data, correct booking ID and correct response structure
Actual Result	API returned 200 OK status code The API response body content data matches the request test data set.
Test Status	Passed



POST {{baseUrl}} /booking

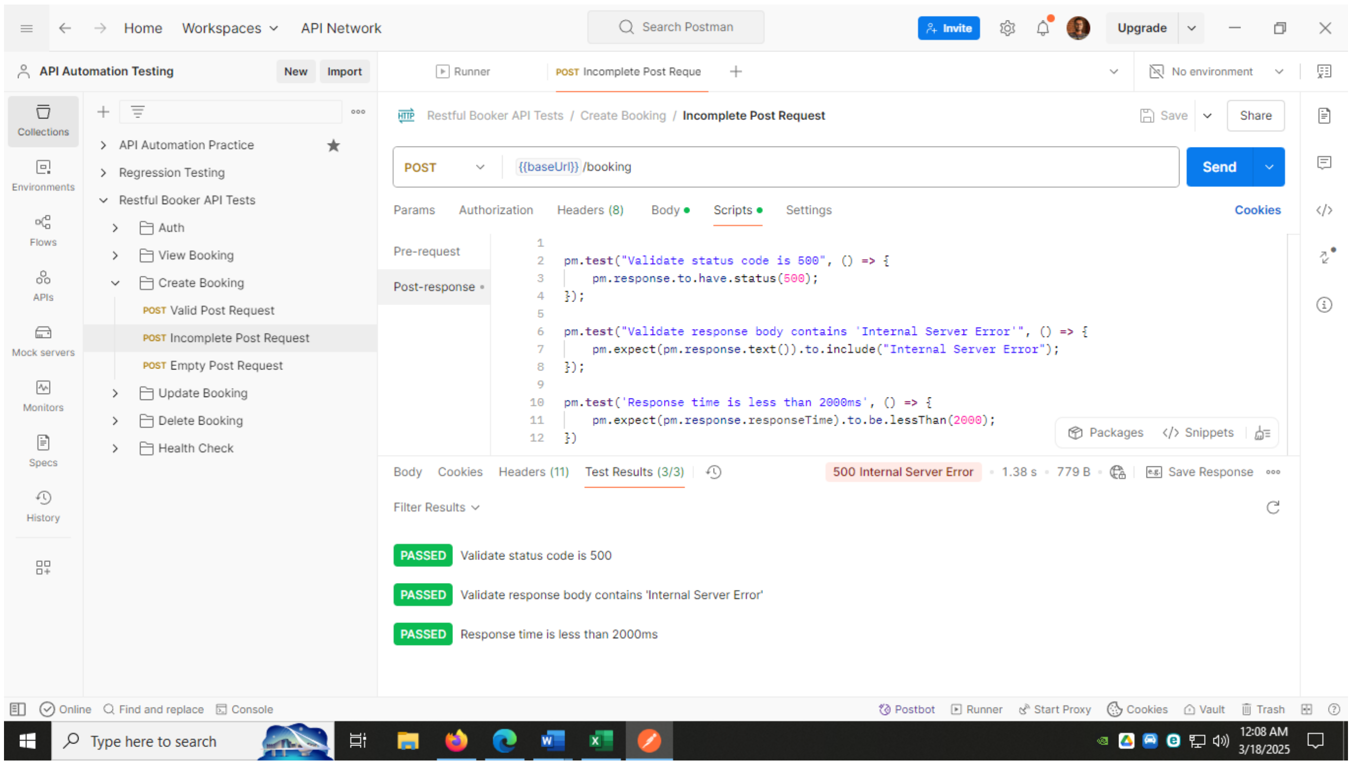
200 OK · 382 ms · 953 B

PASSED Validate status code is 200
PASSED Validate the response first name as 'Alan'
PASSED Validate the response last name as 'Brown'
PASSED Validate the response total price as 555
PASSED Validate the response deposit paid as true
PASSED Validate the response checkin date as '2024-01-01'
PASSED Validate the response checkout date as '2024-02-01'
PASSED Validate the response additional needs as 'Breakfast'
PASSED Response time is less than 2000ms

Figure 8: Execution proof for API-TC-008 (Create Valid Booking Record)

Test Case ID	**API-TC-009**
Test Scenario	Verify that a booking cannot be created when required fields are completely missing
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	POST
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests.
Test Data	Total Price: 555 Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a GET request to API endpoint 3. Set Content type to application/json in the request header section as key value pair. 4. Add a valid JSON request body with missing required fields (First name and Last name) 5. Click Send button. 6. Validate that the response status code and response body content
Expected Result	API should return 500 Internal Server Error, indicating an issue with missing required keys in the request body.

Actual Result	API returned 500 status code The API response body contained Internal Server Error message
Test Status	Passed



Restful Booker API Tests / Create Booking / Incomplete Post Request

POST {{baseUrl}} /booking

```
pm.test("Validate status code is 500", () => {
    pm.response.to.have.status(500);
});

pm.test("Validate response body contains 'Internal Server Error'", () => {
    pm.expect(pm.response.text()).to.include("Internal Server Error");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

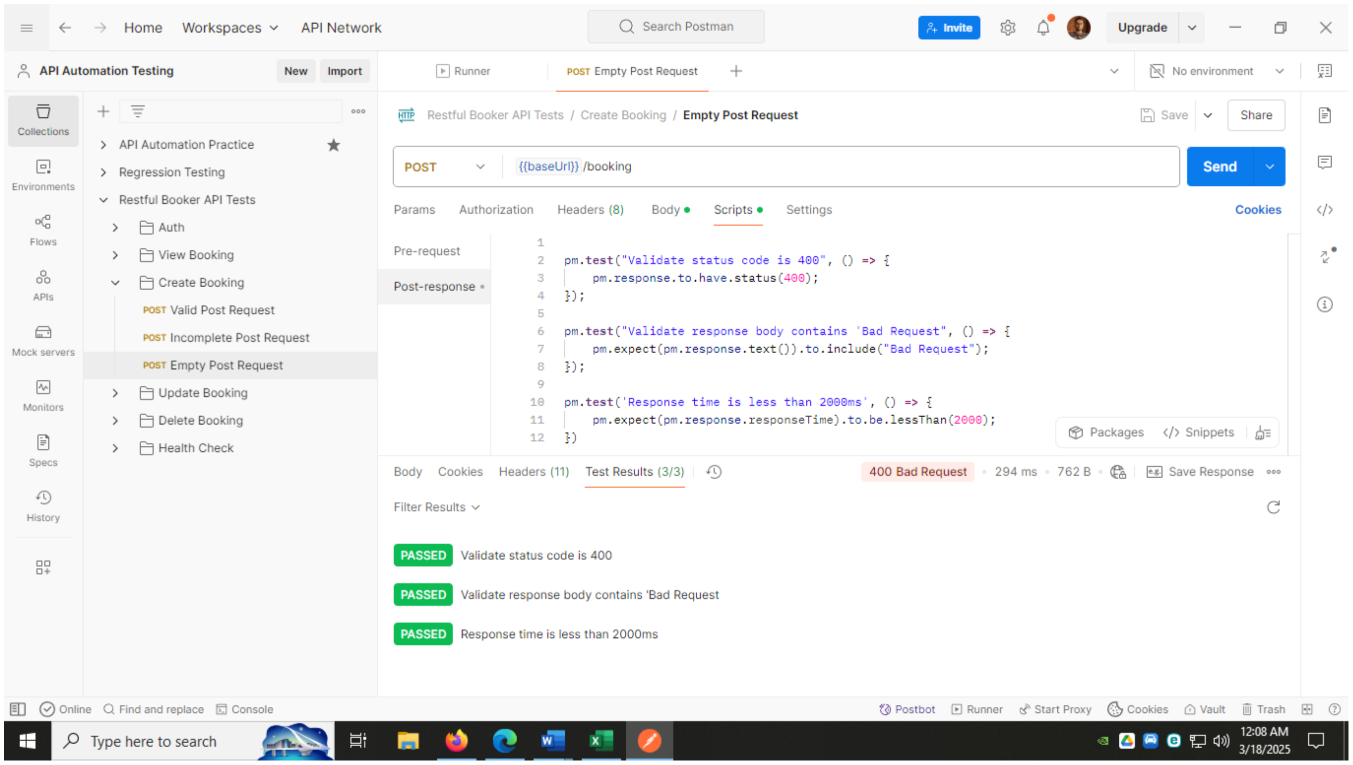
Test Results (3/3) — 500 Internal Server Error · 1.38 s · 779 B

```
PASSED  Validate status code is 500
PASSED  Validate response body contains 'Internal Server Error'
PASSED  Response time is less than 2000ms
```

Figure 9: Execution proof for API-TC-009 (Incomplete POST Request)

Test Case ID	**API-TC-010**
Test Scenario	Verify that a booking cannot be created when a required field has an empty value
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	POST
Priority	P2 (Medium Priority)
Preconditions	API should be available and accept booking retrieval requests.
Test Data	First Name: null Last Name: Bell Total Price: 555 Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. Set Content type to application/json in the request header section as key value pair.

	4. Add a valid JSON request body with an empty value for first name 5. Click Send button. 6. Validate that the response status code and response body content
Expected Result	API should return 400 Bad request, indicating that request is not correct.
Actual Result	API returned 400 status code The API response body contained Bad Request message
Test Status	Passed



POST {{baseUrl}} /booking

```
pm.test("Validate status code is 400", () => {
    pm.response.to.have.status(400);
});

pm.test("Validate response body contains 'Bad Request", () => {
    pm.expect(pm.response.text()).to.include("Bad Request");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

Test Results (3/3) · 400 Bad Request · 294 ms · 762 B

```
PASSED  Validate status code is 400
PASSED  Validate response body contains 'Bad Request
PASSED  Response time is less than 2000ms
```

Figure 10: Execution proof for API-TC-010 (Empty POST Request)

Test Case ID	**API-TC-011**
Test Scenario	Verify Booking Cannot Be Created with Invalid Data Types
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	POST
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests.
Test Data	First Name: Rupet Last Name: Marton Total Price: text Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Breakfast

Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. Set Content type to application/json in the request header section as key value pair. 4. Add a valid JSON request body where total price as a string instead of a number 5. Click Send button. 6. Validate that the response status code and response body content
Expected Result	API should return 400 Bad Request, indicating that total price must be a number
Actual Result	To Be Tested
Test Status	To Be Tested

Test Case ID	**API-TC-012**
Test Scenario	Verify Booking Cannot Be Created with an Invalid Date Range
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	POST
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests.
Test Data	First Name: Rupet Last Name: Marton Total Price: text Deposit Paid: true Booking Dates: 2025-01-01 and 2019-01-01 Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. Set Content type to application/json in the request header section as key value pair. 4. Add a valid JSON request body where check in date is later than checkout date 5. Click Send button. 6. Validate that the response status code and response body content
Expected Result	API should return 400 Bad Request, indicating that indicating that the check-in date must be before the check-out date.
Actual Result	To Be Tested
Test Status	To Be Tested

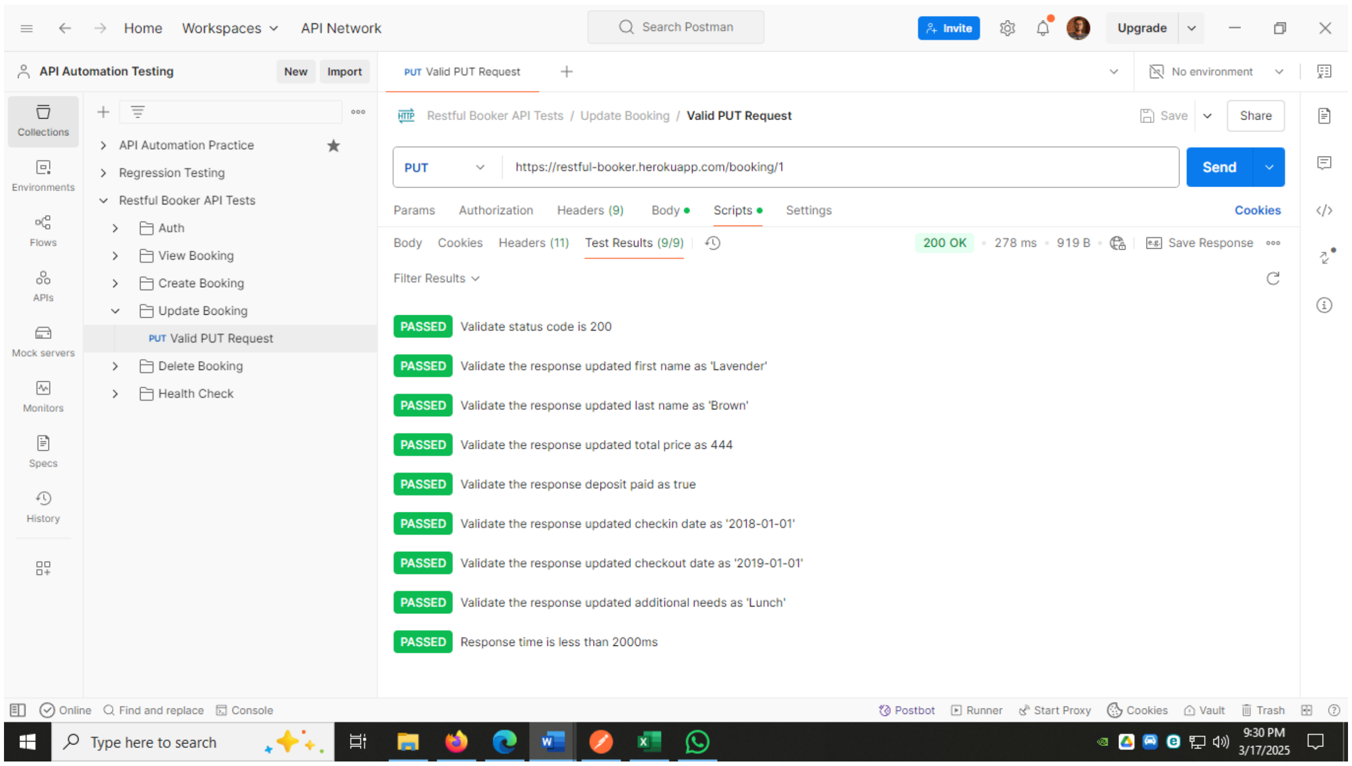
Test Case ID	**API-TC-013**
Test Scenario	Verify Booking Cannot Be Created with an Invalid Date Format
API Endpoint	https://restful-booker.herokuapp.com/booking
Request Type	POST
Priority	P1 (High Priority)

Preconditions	API should be available and accept booking retrieval requests.
Test Data	First Name: Rupet Last Name: Marton Total Price: text Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a POST request to API endpoint 3. Set Content type to application/json in the request header section as key value pair. 4. Add a valid JSON request body where total price as a string instead of a number 5. Click Send button. 6. Validate that the response status code and response body content
Expected Result	API should return 400 Bad Request, indicating that total price must be a number
Actual Result	To Be Tested
Test Status	To Be Tested

5.4 Update Booking Test Cases (https://restful-booker.herokuapp.com/booking/:id)

Test Case ID	**API-TC-014**
Test Scenario	Verify Booking Can be updated with valid data
API Endpoint	https://restful-booker.herokuapp.com/booking/1
Request Type	PUT
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests A valid booking ID must exist in the system. A valid authentication token must be generated and included in the request
Test Data	First Name: Lavender Last Name: Brown Total Price: 444 Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Lunch
Test Steps	1. Open Postman 2. Create a PUT request to API endpoint 3. Generate a valid authentication token by sending a request to the login/authentication API 4. Store the authentication token received in the response 5. Set Headers • Cookie: Token 6. Add a valid JSON request body with updated data 7. Click Send button.

	8. Validate that the response status code and response body content
Expected Result	API should return 200 OK, updated booking details should match the request
Actual Result	API returned 200 OK status code The API response body content data matches the request test data set.
Test Status	Passed



Restful Booker API Tests / Update Booking / Valid PUT Request

PUT https://restful-booker.herokuapp.com/booking/1

200 OK · 278 ms · 919 B

PASSED Validate status code is 200
PASSED Validate the response updated first name as 'Lavender'
PASSED Validate the response updated last name as 'Brown'
PASSED Validate the response updated total price as 444
PASSED Validate the response deposit paid as true
PASSED Validate the response updated checkin date as '2018-01-01'
PASSED Validate the response updated checkout date as '2019-01-01'
PASSED Validate the response updated additional needs as 'Lunch'
PASSED Response time is less than 2000ms

Figure 11: Execution proof for API-TC-014 (Valid PUT Request)

Test Case ID	**API-TC-015**
Test Scenario	Verify Successfully Update Only Specific Fields While Keeping Others Unchanged
API Endpoint	https://restful-booker.herokuapp.com/booking/:id
Request Type	PUT
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests A valid booking ID must exist in the system. A valid authentication token must be generated and included in the request
Test Data	First Name: Lavender Last Name: Brown Total Price: 666 Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01

	Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a PUT request to API endpoint 3. Generate a valid authentication token by sending a request to the login/authentication API 4. Store the authentication token received in the response 5. Set Headers 6. Content-Type: application/json 7. Cookie: Token 8. Add a valid JSON request body with only updating the total price 9. Click Send button. 10. Validate that the response status code and response body content
Expected Result	API should return 200 OK, updated total price data should be reflected while keeping other fields unchanged
Actual Result	To Be Tested
Test Status	To Be Tested

Test Case ID	**API-TC-016**
Test Scenario	Verify that updating a non-existent booking ID returns a 404 Not Found response.
API Endpoint	https://restful-booker.herokuapp.com/booking/:invalidid
Request Type	PUT
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests A valid booking ID must not exist in the system. A valid authentication token must be generated and included in the request
Test Data	Booking ID: 0000 (Invalid) First Name: Lavender Last Name: Brown Total Price: 444 Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Dinner
Test Steps	1. Open Postman 2. Create a PUT request to API endpoint 3. Generate a valid authentication token by sending a request to the login/authentication API 4. Store the authentication token received in the response 5. Set Headers • Content-Type: application/json • Cookie: Token 6. Add a valid JSON request body with updated data 7. Click Send button. 8. Validate that the response status code and response body content

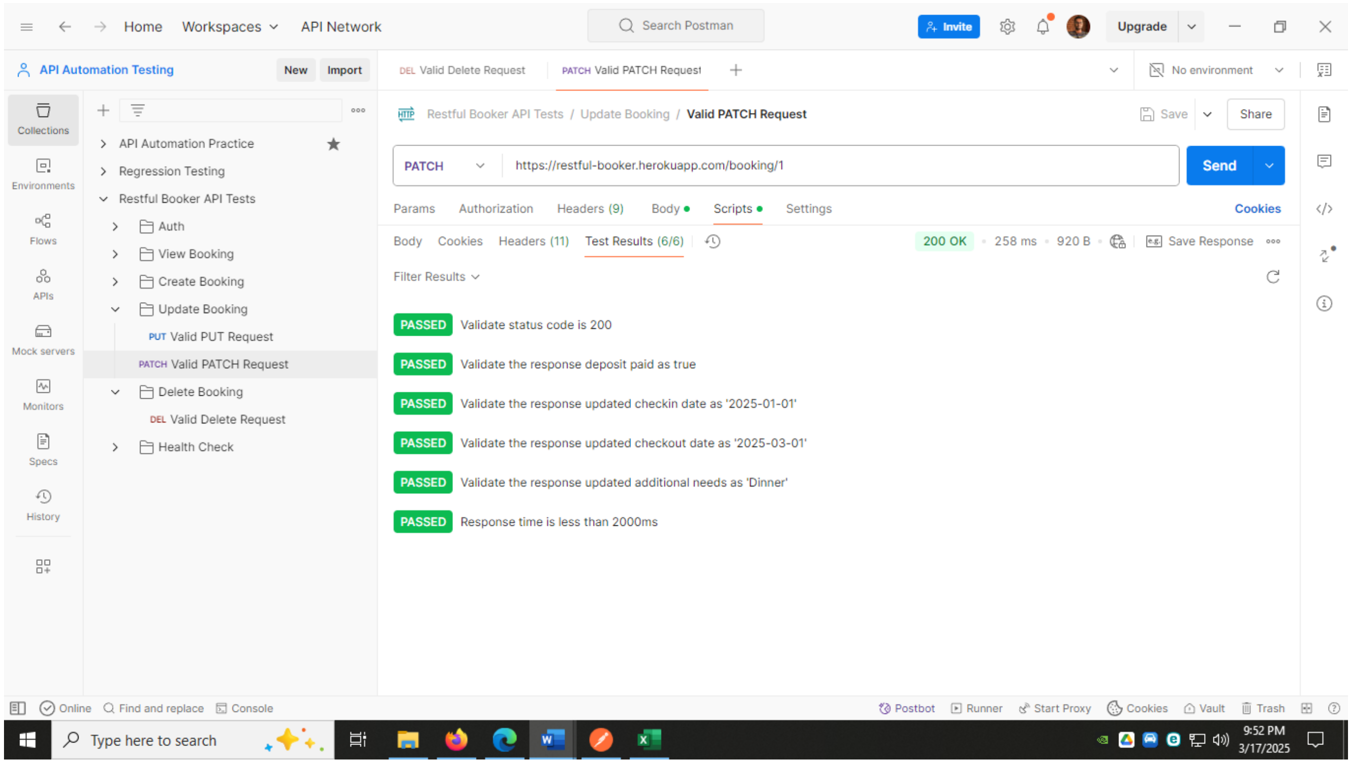
Expected Result	API should return 404 Not found, indicating that the requested booking ID does not exist
Actual Result	To Be Tested
Test Status	To Be Tested

Test Case ID	**API-TC-017**
Test Scenario	Verify that an updating existing booking cannot be done when required fields are completely missing
API Endpoint	https://restful-booker.herokuapp.com/booking/:id
Request Type	PUT
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests A valid booking ID must exist in the system. A valid authentication token must be generated and included in the request
Test Data	Total Price: 555 Deposit Paid: true Booking Dates: 2024-01-01 and 2024-02-01 Additional Needs: Breakfast
Test Steps	1. Open Postman 2. Create a PUT request to API endpoint 3. API should be available and accept booking retrieval requests 4. A valid booking ID must exist in the system. 5. A valid authentication token must be generated and included in the request Set Content type to application/json in the request header section as key value pair. 6. Add a valid JSON request body with missing required fields (First name and Last name) 7. Click Send button. 8. Validate that the response status code and response body content
Expected Result	API should return 400 Bad request, indicating an issue with missing required key and values in the request body.
Actual Result	To Be Tested
Test Status	To Be Tested

5.5 Partial Update Booking Test Cases (https://restful-booker.herokuapp.com/booking/:id)

Test Case ID	**API-TC-018**
Test Scenario	Verify that specific fields in an existing booking can be updated while keeping others unchanged.
API Endpoint	https://restful-booker.herokuapp.com/booking/:id
Request Type	PATCH
Priority	P1 (High Priority)

Preconditions	API should be available and accept booking retrieval requests A valid booking ID must exist in the system. A valid authentication token must be generated and included in the request
Test Data	Check in date: 2025-01-01 Check out date: 2025-03-01 Additional Needs: Dinner
Test Steps	1. Open Postman 2. Create a PATCH request to API endpoint 3. Generate a valid authentication token by sending a request to the login/authentication API 4. Store the authentication token received in the response 5. Set Headers • Cookie: Token 6. Add a valid JSON request body with updated Last Name and Total price data 7. Click Send button. 8. Validate that the response status code and response body content
Expected Result	API should return 200 OK, updated fields data should be visible while other fields unchanged.
Actual Result	API returned 200 OK status code The API response body content data matches the request test data set.
Test Status	Passed



PATCH https://restful-booker.herokuapp.com/booking/1

200 OK · 258 ms · 920 B

PASSED Validate status code is 200

PASSED Validate the response deposit paid as true

PASSED Validate the response updated checkin date as '2025-01-01'

PASSED Validate the response updated checkout date as '2025-03-01'

PASSED Validate the response updated additional needs as 'Dinner'

PASSED Response time is less than 2000ms

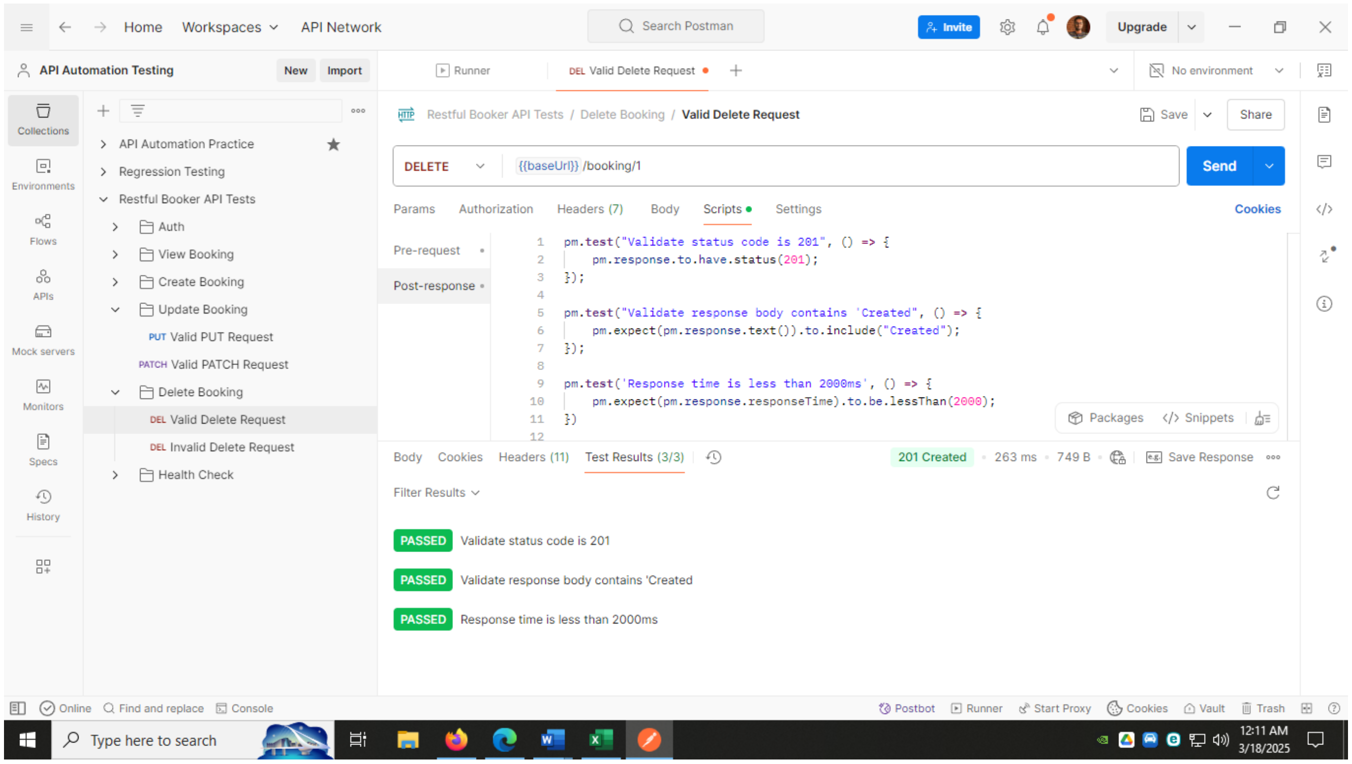
Figure 12: Execution proof for API-TC-018 (Valid PATCH Request)

Test Case ID	**API-TC-019**
Test Scenario	Verify that updating a booking ID that does not exist returns an appropriate error.
API Endpoint	https://restful-booker.herokuapp.com/booking/:invalidid
Request Type	PATCH
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests Booking ID must not exist in the system. A valid authentication token must be generated and included in the request
Test Data	Booking ID: 9999 Last Name: Brown Total Price: 444
Test Steps	1. Open Postman 2. Create a PATCH request to API endpoint 3. Generate a valid authentication token by sending a request to the login/authentication API 4. Store the authentication token received in the response 5. Set Headers • Content-Type: application/json • Cookie: Token 6. Add a valid JSON request body with updated Last Name and Total price data 7. Click Send button. 8. Validate that the response status code and response body content
Expected Result	API should return 404 Not found, indicating that the requested booking ID does not exists.
Actual Result	To Be Tested
Test Status	To Be Tested

5.6 Delete Booking Test Cases (https://restful-booker.herokuapp.com/booking/1)

Test Case ID	**API-TC-020**
Test Scenario	Verify that a booking can be successfully deleted using a valid token and an existing booking ID
API Endpoint	https://restful-booker.herokuapp.com/booking/1
Request Type	DELETE
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests Booking ID must exist in the system. A valid authentication token must be generated and included in the request
Test Data	Booking ID: 1
Test Steps	1. Open Postman 2. Create a DELETE request to API endpoint 3. Generate a valid authentication token by sending a request to the login/authentication API

	4. Store the authentication token received in the response 5. Set Headers: • Cookie: Token 6. Click Send button. 7. Validate that the response status code and response body content
Expected Result	API should return 201 Created status code Response body contains a confirmation message “Created”
Actual Result	API returned 201status code The API response body contained Created message
Test Status	Passed


Restful Booker API Tests / Delete Booking / Valid Delete Request

DELETE {{baseUrl}} /booking/1

```
pm.test("Validate status code is 201", () => {
    pm.response.to.have.status(201);
});

pm.test("Validate response body contains 'Created", () => {
    pm.expect(pm.response.text()).to.include("Created");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

Test Results (3/3) — 201 Created · 263 ms · 749 B

PASSED Validate status code is 201

PASSED Validate response body contains 'Created

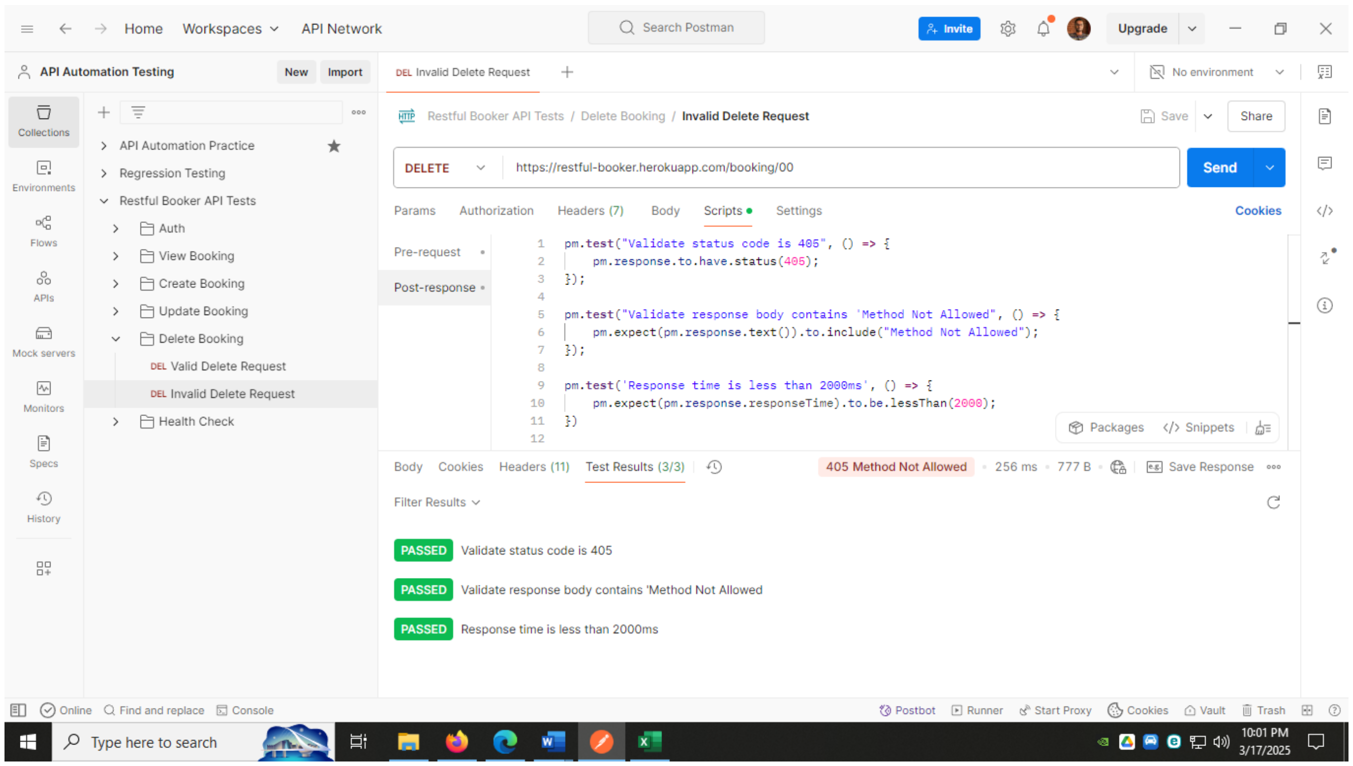
PASSED Response time is less than 2000ms

Figure 13: Execution proof for API-TC-020 (Valid DELETE Request)

Test Case ID	**API-TC-021**
Test Scenario	Verify that a booking cannot be deleted if an invalid or missing authentication token is used.
API Endpoint	https://restful-booker.herokuapp.com/booking/1
Request Type	DELETE
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests Booking ID must exist in the system.
Test Data	Booking ID: 1

Test Steps	1. Open Postman 2. Create a DELETE request to API endpoint 3. Set Headers • Content-Type: application/json 4. Send a DELETE request without an authentication token 5. Validate that the response status code and response body content
Expected Result	API should return 403 Forbidden status code Response body contains an error message “Forbidden”
Actual Result	To Be Tested
Test Status	To Be Tested

Test Case ID	**API-TC-022**
Test Scenario	Verify that an attempt to delete a booking without specifying a valid ID result in an error.
API Endpoint	https://restful-booker.herokuapp.com/booking/00
Request Type	DELETE
Priority	P1 (High Priority)
Preconditions	API should be available and accept booking retrieval requests Booking ID must exist in the system.
Test Data	N/A
Test Steps	1. Open Postman 2. Create a DELETE request to API endpoint 3. Generate a valid authentication token by sending a request to the login/authentication API 4. Set Headers • Cookie: Token 5. Send a DELETE request without an authentication token 6. Validate that the response status code and response body content
Expected Result	API should return 405 Method Not Allowed status code Response body contains an error message “Method Not Allowed”
Actual Result	API returned 201status code The API response body contained “Method Not Allowed” message
Test Status	Passed



DELETE https://restful-booker.herokuapp.com/booking/00

```
pm.test("Validate status code is 405", () => {
    pm.response.to.have.status(405);
});

pm.test("Validate response body contains 'Method Not Allowed", () => {
    pm.expect(pm.response.text()).to.include("Method Not Allowed");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

405 Method Not Allowed · 256 ms · 777 B

PASSED Validate status code is 405

PASSED Validate response body contains 'Method Not Allowed

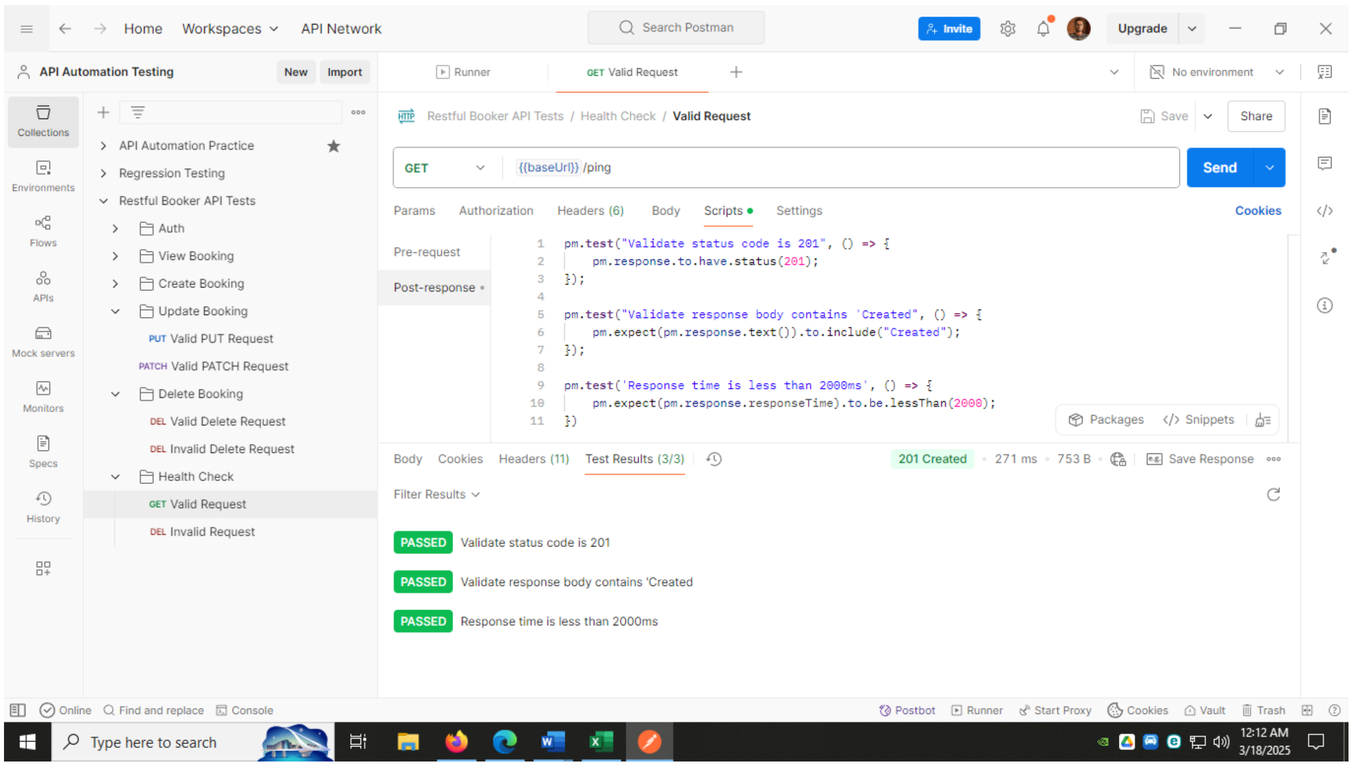
PASSED Response time is less than 2000ms

10:01 PM 3/17/2025

Figure 14: Execution proof for API-TC-022 (Invalid DELETE Request)

5.7 Health Check Test Cases (https://restful-booker.herokuapp.com/ping)

Test Case ID	**API-TC-023**
Test Scenario	Verify API is reachable and returns a success response
API Endpoint	https://restful-booker.herokuapp.com/ping
Request Type	GET
Priority	P1 (High Priority)
Preconditions	API server should be running and accessible.
Test Data	N/A
Test Steps	1. Open Postman 2. Create a GET request to API endpoint 3. Click Send button. 4. Validate that the response status code and response body content 5. Validate the response time within the acceptable limit.
Expected Result	API should return 201 Created status code Response body contains a confirmation message “Created” The response time should be within acceptable limits.
Actual Result	API returned 201status code The API response body contained “Created” message
Test Status	Passed



GET {{baseUrl}} /ping

```
pm.test("Validate status code is 201", () => {
    pm.response.to.have.status(201);
});

pm.test("Validate response body contains 'Created", () => {
    pm.expect(pm.response.text()).to.include("Created");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

201 Created · 271 ms · 753 B

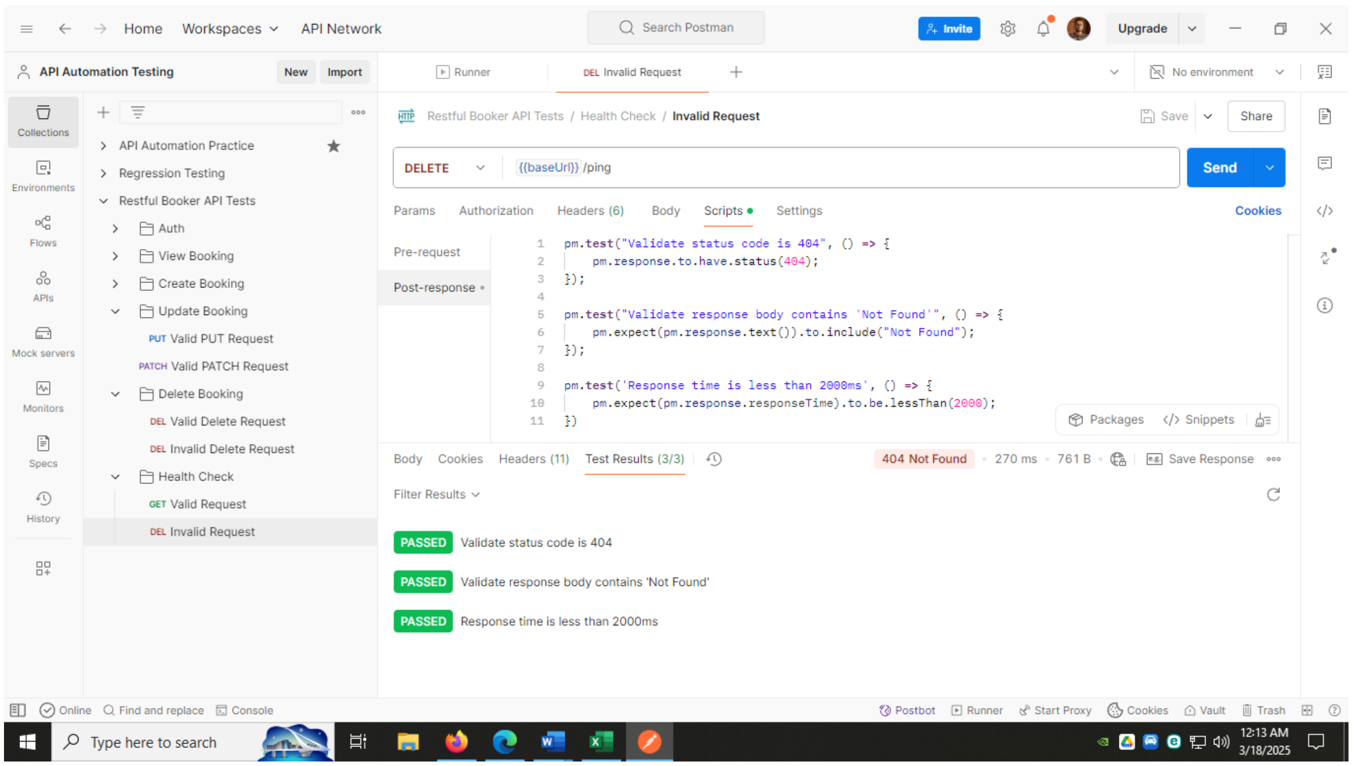
PASSED Validate status code is 201

PASSED Validate response body contains 'Created

PASSED Response time is less than 2000ms

Figure 15: Execution proof for API-TC-023 (Valid GET Request)

Test Case ID	**API-TC-024**
Test Scenario	Verify that the API rejects requests made with unsupported HTTP methods
API Endpoint	https://restful-booker.herokuapp.com/ping
Request Type	DELETE
Priority	P1 (High Priority)
Preconditions	API server should be running and accessible.
Test Data	N/A
Test Steps	1. Open Postman 2. Create a DELETE request to API endpoint 3. Click Send button. 4. Validate that the response status code and response body content
Expected Result	API should return 404 status code Response body contains a confirmation message “Not Found” The response time should be within acceptable limits.
Actual Result	API returned 404status code The API response body contained “Not Found” message
Test Status	Passed



DELETE {{baseUrl}} /ping

```
pm.test("Validate status code is 404", () => {
    pm.response.to.have.status(404);
});

pm.test("Validate response body contains 'Not Found'", () => {
    pm.expect(pm.response.text()).to.include("Not Found");
});

pm.test('Response time is less than 2000ms', () => {
    pm.expect(pm.response.responseTime).to.be.lessThan(2000);
})
```

404 Not Found · 270 ms · 761 B

PASSED Validate status code is 404

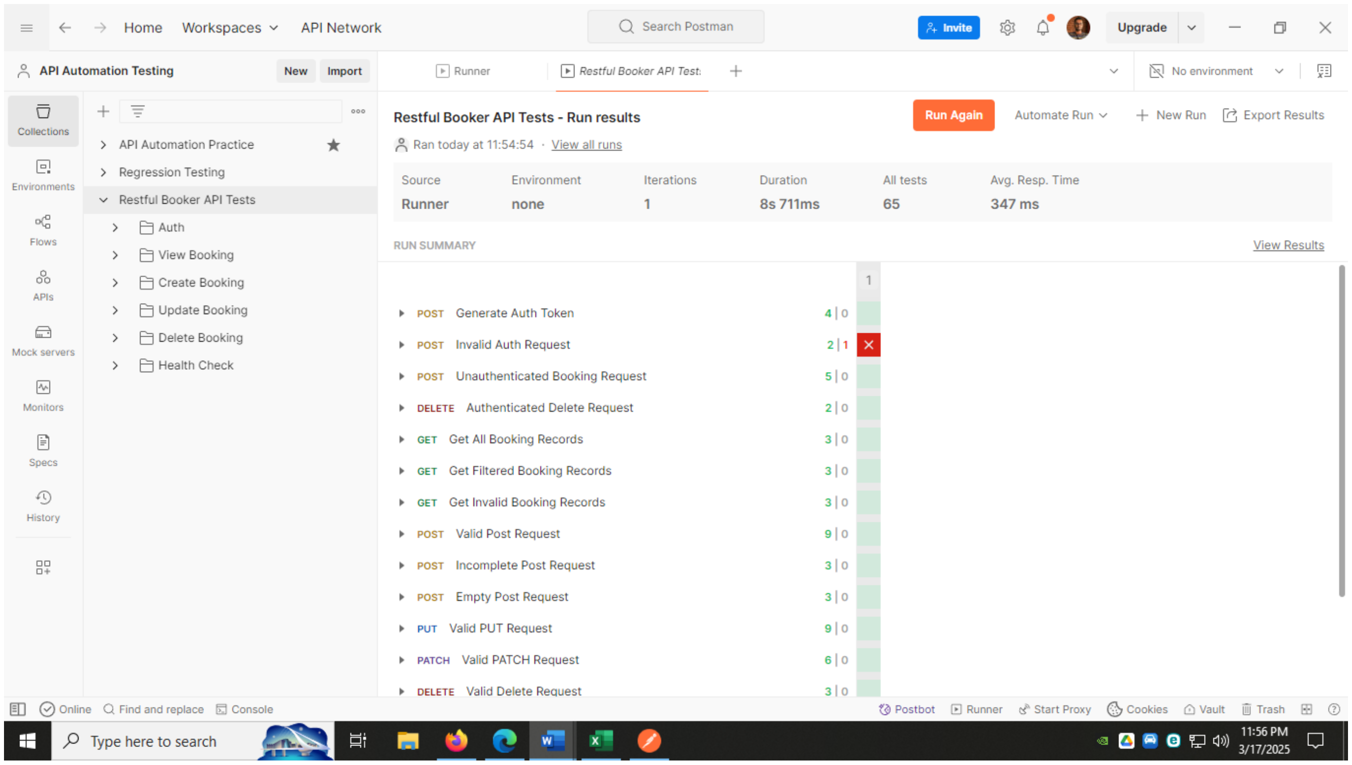
PASSED Validate response body contains 'Not Found'

PASSED Response time is less than 2000ms

Figure 16: Execution proof for API-TC-023 (Invalid Request)

Test Case ID	**API-TC-025**
Test Scenario	Verify that the API correctly rejects requests to an incorrect endpoint
API Endpoint	https://restful-booker.herokuapp.com/pingtest
Request Type	GET
Priority	P1 (High Priority)
Preconditions	API server should be running and accessible.
Test Data	N/A
Test Steps	1. Open Postman 2. Create a GET request to API endpoint 3. Click Send button. 4. Validate that the response status code and response body content
Expected Result	API should return 404 Not found status code Response body contains a confirmation message “Not Found”
Actual Result	To Be Tested
Test Status	To Be Tested

6. Test Collection Execution Results



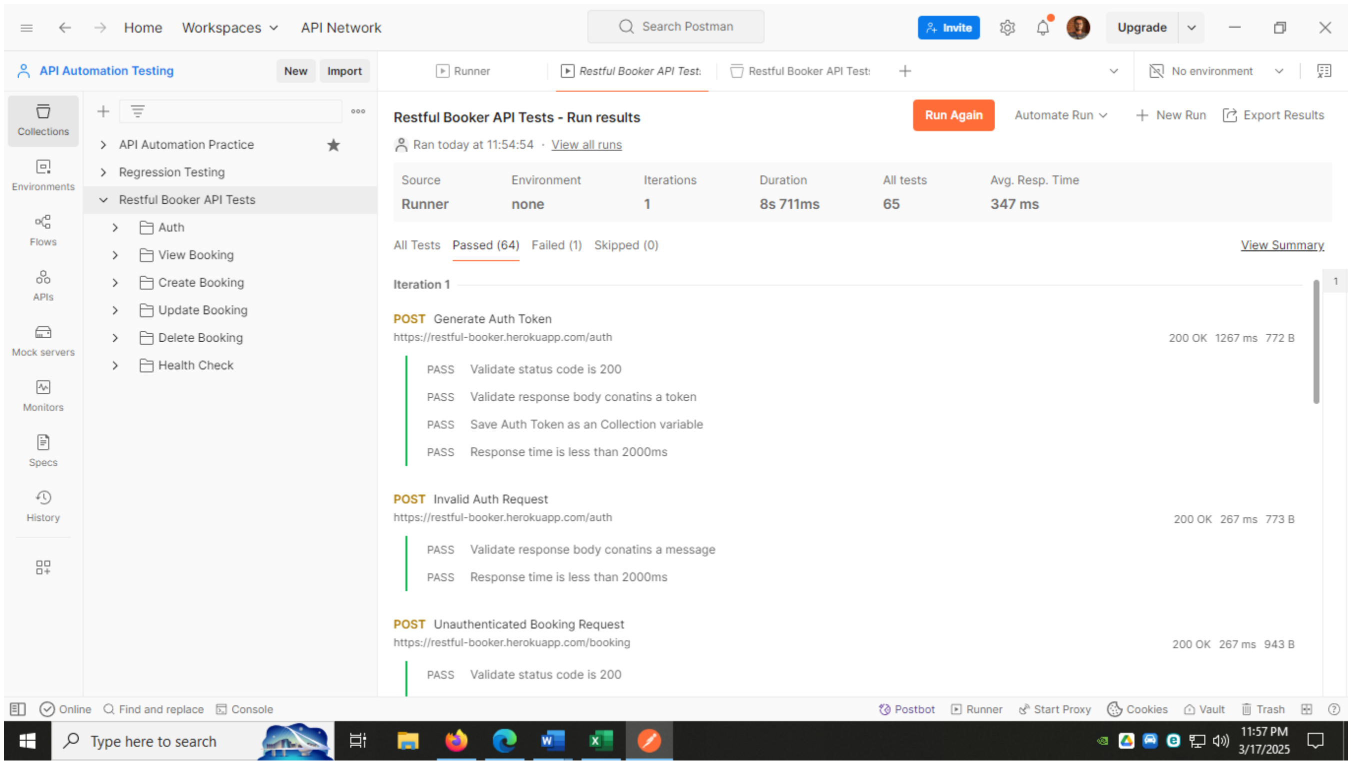
Restful Booker API Tests - Run results

Ran today at 11:54:54 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	8s 711ms	65	347 ms

RUN SUMMARY — View Results

Method	Request	Passed | Failed	1
POST	Generate Auth Token	4 | 0	🟩
POST	Invalid Auth Request	2 | 1	🟥
POST	Unauthenticated Booking Request	5 | 0	🟩
DELETE	Authenticated Delete Request	2 | 0	🟩
GET	Get All Booking Records	3 | 0	🟩
GET	Get Filtered Booking Records	3 | 0	🟩
GET	Get Invalid Booking Records	3 | 0	🟩
POST	Valid Post Request	9 | 0	🟩
POST	Incomplete Post Request	3 | 0	🟩
POST	Empty Post Request	3 | 0	🟩
PUT	Valid PUT Request	9 | 0	🟩
PATCH	Valid PATCH Request	6 | 0	🟩
DELETE	Valid Delete Request	3 | 0	🟩



Restful Booker API Tests - Run results

Ran today at 11:54:54 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	8s 711ms	65	347 ms

All Tests   Passed (64)   Failed (1)   Skipped (0)   View Summary

Iteration 1

POST Generate Auth Token
https://restful-booker.herokuapp.com/auth — 200 OK 1267 ms 772 B

- PASS Validate status code is 200
- PASS Validate response body conatins a token
- PASS Save Auth Token as an Collection variable
- PASS Response time is less than 2000ms

POST Invalid Auth Request
https://restful-booker.herokuapp.com/auth — 200 OK 267 ms 773 B

- PASS Validate response body conatins a message
- PASS Response time is less than 2000ms

POST Unauthenticated Booking Request
https://restful-booker.herokuapp.com/booking — 200 OK 267 ms 943 B

- PASS Validate status code is 200

This section presents the results of the API collection test execution, highlighting passed, failed, and skipped tests. It provides insights into API stability and performance over multiple runs. Out of 65 tests, 64 passed, while 1 failed due to unexpected response codes. The average response time varied between 300-400ms, indicating stable but slightly fluctuating API performance.

Above postman collection is available here for reuse in automation and regression testing.

7. Test Deliverables

The following test artifacts will be delivered as part of API testing process ensure the robustness and reliability of the API under test:

1. **Test Plan Document -** Outlines the scope, objectives, testing strategy, and approach for API validation.
2. **Test Strategy Document** - Defines the level of testing, testing techniques and tools.
3. **Test Case Document -** A structured set of test cases covering functional validation, authentication handling, response validations, and error handling for given API endpoints.
4. **Test Execution Results -** A record of executed test cases along with their pass/fail status, response times, and API behavior verification.
5. **Postman API Collection -** A structured API request collection with test scripts validating API responses, status codes, and performance metrics.
6. **Defect Report & Issues Summary -** A report contains summary if the defects, usability concerns and suggested improvements.

8. Test Completeness

To ensure comprehensive testing and meet defined test objectives, the following exit criteria are considered for API validation:

1. All test cases for authentication, booking creation, retrieval, and deletion are executed, covering functional and negative test scenarios.
2. Identified defects and API inconsistencies are logged, documented, and reported for resolution.
3. All critical and high-severity defects related to authentication failures, incorrect status codes, and unexpected API behavior are fixed, retested and verified.
4. API responses are validated to ensure they match the expected request data, ensuring data integrity across API transactions.
5. API performance is tested to ensure response times remain within acceptable limits (<2000ms) for optimized system performance.
6. API security is validated by checking that unauthorized access is restricted, and authentication tokens work as expected.

9. Proposed Improvements for Shift Left Testing in API Validation

In order to enhance the efficacy and reliability of API testing, it is recommended to implement shift left testing approach. This will help to identify defects earlier in the SDLC which will reduce the cost, and also improve the software quality.

1. Automated token validation

Implemented pre request script in Postman to dynamically generate and store the authentication token. This approach helped to run each test with a new token. This approach helped to reduce manual work of generating authentication token and save on each API test before run.

2. API Response Structure Validation

Ensured that API response contained the required fields like "bookingid", "firtsname" and "lastname". This helps to adhere to the expected structure from the early stages of the SDLC.

3. Performance Testing Integration

By response time validation, it ensured that the API response adhere with performance benchmarks. But it needs to conduct a separate performance testing with tools like JMeter. This early-stage verification helps to get an idea about the performance of the APIs in advance.

4. Mock APIs for early development collaborations

By using mock APIs to simulate real API endpoints in early stages of SDLC allows to start early API testing. This will reduce the testing bottlenecks in later stages of SDLC

5. Integration of CI/CD for continuous testing

By automating API testing and integrating CI/CD pipeline will help to run API tests on continuous testing environment like running API tests after each code change happens in test environment.