Instructions

In this document you will find two tasks. After completing the tasks follow these steps:

1. Create a private repository/ies in Git Hub (<https://github.com/>)
2. Grant repository access to these emails:
   1. alexander.atanasov@nextbasket.com
   2. desislav.ivanov@nextbasket.com
   3. petar.uzunov@nextbasket.com
3. Upload both tasks to the repository/ies
4. Add project description, technologies used and other valuable information to the README file

Tasks

Task 1

On this test store: [https://teststoreforsouthafri.nextbasket.shop/](https://teststoreforsouthafri.nextbasket.shop/" \t "_blank" \o "https://teststoreforsouthafri.nextbasket.shop/)

1. Create an automated test that would order the first in stock non-promo product (use South Africa as Country and Alberton as City)
2. Create a automated test verifying that the "50% off" label is visible
3. Create 10 BDD test cases for checkout (no need to automate these, just add them to a text file inside the project)
4. Describe at least two bugs (BDD Steps, add them to a text file inside the project)
5. Answer
6. package Assessment;
7. import java.time.Duration;
8. import org.openqa.selenium.By;
9. import org.openqa.selenium.WebDriver;
10. import org.openqa.selenium.WebElement;
11. import org.openqa.selenium.chrome.ChromeDriver;
12. import org.openqa.selenium.support.ui.WebDriverWait;
13. import org.testng.annotations.Test;
14. import io.github.bonigarcia.wdm.WebDriverManager;
15. public class OrderNonPromoProduct {
16. WebDriver driver;
17. *@Test*(priority=0)
18. public void setup() {
19. WebDriverManager.*chromedriver*().setup();
20. driver = new ChromeDriver();
21. driver.manage().window().maximize();
22. WebDriverWait wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));
23. }
24. *@Test*(priority=1)
25. public void orderProduct() {
26. driver.get("https://teststoreforsouthafri.nextbasket.shop/");
27. WebElement product = driver.findElement(By.*xpath*("//\*[@id=\"root\"]/div/main/div/div/div/div[1]/div[1]/a/img"));
28. product.click();
29. WebElement addToBasket = driver.findElement(By.*linkText*("Add to basket"));
30. addToBasket.click();
31. WebElement bIcon = driver.findElement(By.*xpath*("//\*[@id=\"headlessui-popover-button-:rgr:\"]/button/div/div/svg"));
32. bIcon.click();
33. WebElement goToPaymentBtn = driver.findElement(By.*className*("stWcf \_2FTCz \_1zffG Ck0CN ll85s -xvsH"));
34. goToPaymentBtn.click();
35. // WebElement emailTextbox = driver.findElement(By.name("email"));
36. // emailTextbox.sendKeys("bumar1209@gmail.com");
37. //
38. // WebElement guestBtn = driver.findElement(By.xpath("//\*[@id=\"root\"]/div/div/div[3]/div[1]/form/div/div[2]/button/p"));
39. // guestBtn.click();
40. //
41. // WebElement fname = driver.findElement(By.name("shippingAddress.firstName"));
42. // fname.sendKeys("Umar");
43. //
44. // WebElement lname = driver.findElement(By.name("shippingAddress.lastName"));
45. // lname.sendKeys("Farooq");
46. //
47. // WebElement phone = driver.findElement(By.name("shippingAddress.phone"));
48. // phone.sendKeys("+923076235676");
49. WebElement country = driver.findElement(By.*id*("headlessui-combobox-input-:rh4:"));
50. country.sendKeys("South Africa");
51. WebElement city = driver.findElement(By.*id*("headlessui-combobox-input-:rh7:"));
52. city.sendKeys("Alberton");
53. WebElement confirmBtn = driver.findElement(By.*xpath*("//\*[@id=\"root\"]/div/div/div[3]/div[2]/div/div/div/button/p"));
54. confirmBtn.click();
55. driver.quit();
56. }

2. Answer

*@Test*(priority=2)

public void promoVisible() {

driver.get("https://teststoreforsouthafri.nextbasket.shop/");

// Check for the "50% off" label

WebElement promoLabel = wait.until(ExpectedConditions.visibilityOfElementLocated(By.*xpath*("//\*[text()='50% off']")));

// Verify the label is displayed

if (promoLabel.isDisplayed()) {

System.***out***.println("The '50% off' label is visible.");

} else {

System.***out***.println("The '50% off' label is not visible.");

}

driver.quit();

}

3. Answer

1. Scenario: Successful Checkout with Valid Information

Given the user has added products to the cart

When the user proceeds to checkout

And the user enters valid shipping information

And the user selects a valid payment method

And the user places the order

Then the order should be successfully placed

1. Scenario: Checkout with Invalid Payment Information

Given the user has added products to the cart

When the user proceeds to checkout

And the user enters valid shipping information

And the user enters invalid payment information

And the user tries to place the order

Then the user should see an error message regarding payment

1. Scenario: Checkout with Missing Shipping Information

Given the user has added products to the cart

When the user proceeds to checkout

And the user leaves the shipping information blank

And the user tries to place the order

Then the user should see an error message regarding shipping information

1. Scenario: Checkout with Invalid Shipping Address

Given the user has added products to the cart

When the user proceeds to checkout

And the user enters an invalid shipping address

And the user tries to place the order

Then the user should see an error message regarding the shipping address

1. Scenario: Checkout with Expired Credit Card

Given the user has added products to the cart

When the user proceeds to checkout

And the user enters valid shipping information

And the user enters an expired credit card

And the user tries to place the order

Then the user should see an error message regarding the credit card

1. Scenario: Checkout with Special Instructions

Given the user has added products to the cart

When the user proceeds to checkout

And the user enters valid shipping information

And the user adds special instructions for delivery

And the user places the order

Then the order should be successfully placed with special instructions noted

1. Scenario: Checkout with Promo Code

Given the user has added products to the cart

When the user proceeds to checkout

And the user enters a valid promo code

And the user enters valid shipping information

And the user places the order

Then the order should be successfully placed with the promo discount applied

1. Scenario: Checkout with Guest Account

Given the user has added products to the cart

When the user proceeds to checkout

And the user selects the guest checkout option

And the user enters valid shipping and payment information

And the user places the order

Then the order should be successfully placed

1. Scenario: Checkout with Registered Account

Given the user has added products to the cart

When the user proceeds to checkout

And the user logs in to their account

And the user enters valid shipping and payment information

And the user places the order

Then the order should be successfully placed

1. Scenario: Checkout with Out-of-Stock Product

Given the user has added products to the cart

When the user proceeds to checkout

And the user enters valid shipping and payment information

And the user tries to place the order

Then the user should see an error message regarding the out-of-stock product

4th Answer

1. Scenario: Product Not Added to Cart

Given the user is on the product page

When the user clicks the "Add to Cart" button

Then the product should be added to the cart

But the product is not added to the cart

And the cart remains empty

1. Scenario: Incorrect Total Price Calculation

Given the user has added multiple products to the cart

And the user proceeds to checkout

When the user reviews the order summary

Then the total price should be the sum of the product prices

But the total price is incorrectly calculated

And the user is charged an incorrect amount

Task 2

<https://www.postman.com/galactic-moon-789755/workspace/live-practice-projects-for-the-api-testing/request/611814-92259b3e-eb3e-4790-83d5-dbb185ff8606>

Create a fork of the above postman project.

Create a collection that:

1. Creates a new booking

<https://restful-booker.herokuapp.com/booking>

Body:

{

    "firstname" : "Umar",

    "lastname" : "Farooq",

    "totalprice" : 199,

    "depositpaid" : true,

    "bookingdates" : {

        "checkin" : "2018-01-05",

        "checkout" : "2019-01-05"

    },

    "additionalneeds" : "Breakfast"

Scripts:

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

pm.test("Verify booking response values", function() {

    var jsonData = pm.response.json();

    pm.expect(jsonData.booking.firstname).to.eql("Umar");

    pm.expect(jsonData.booking.lastname).to.eql("Farooq");

    pm.expect(jsonData.booking.totalprice).to.eql(199);

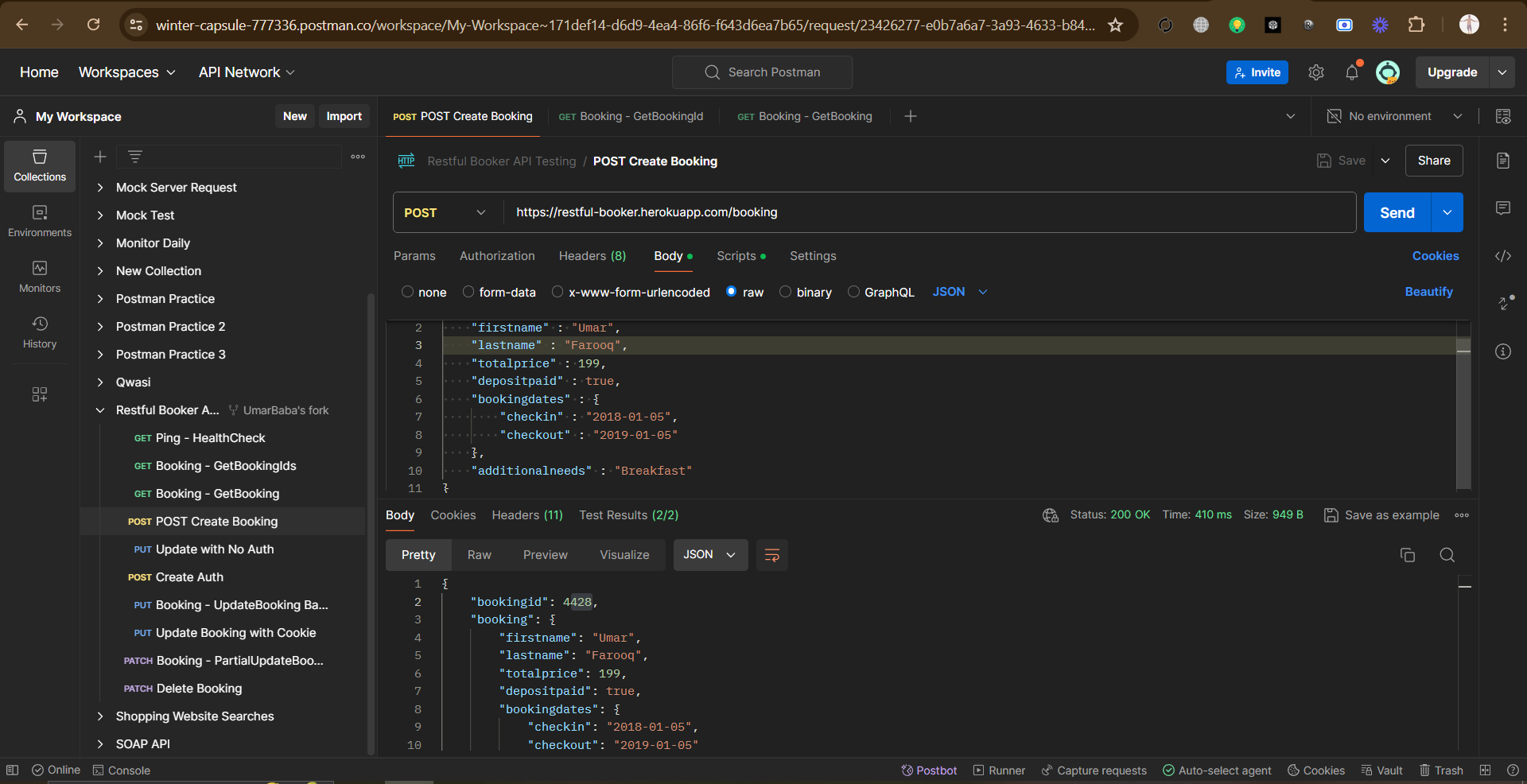
    pm.expect(jsonData.booking.depositpaid).to.eql(true);

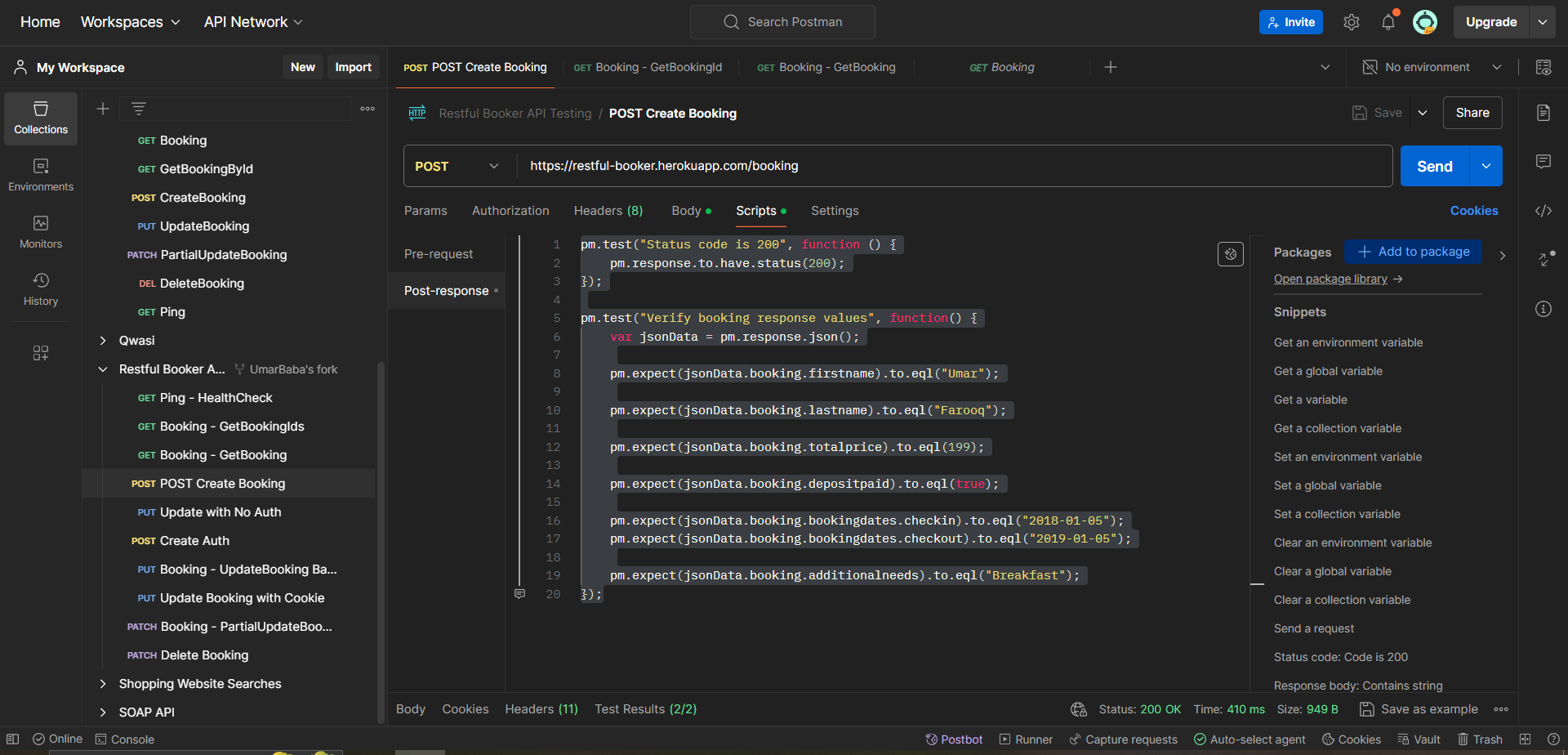
    pm.expect(jsonData.booking.bookingdates.checkin).to.eql("2018-01-05");

    pm.expect(jsonData.booking.bookingdates.checkout).to.eql("2019-01-05");

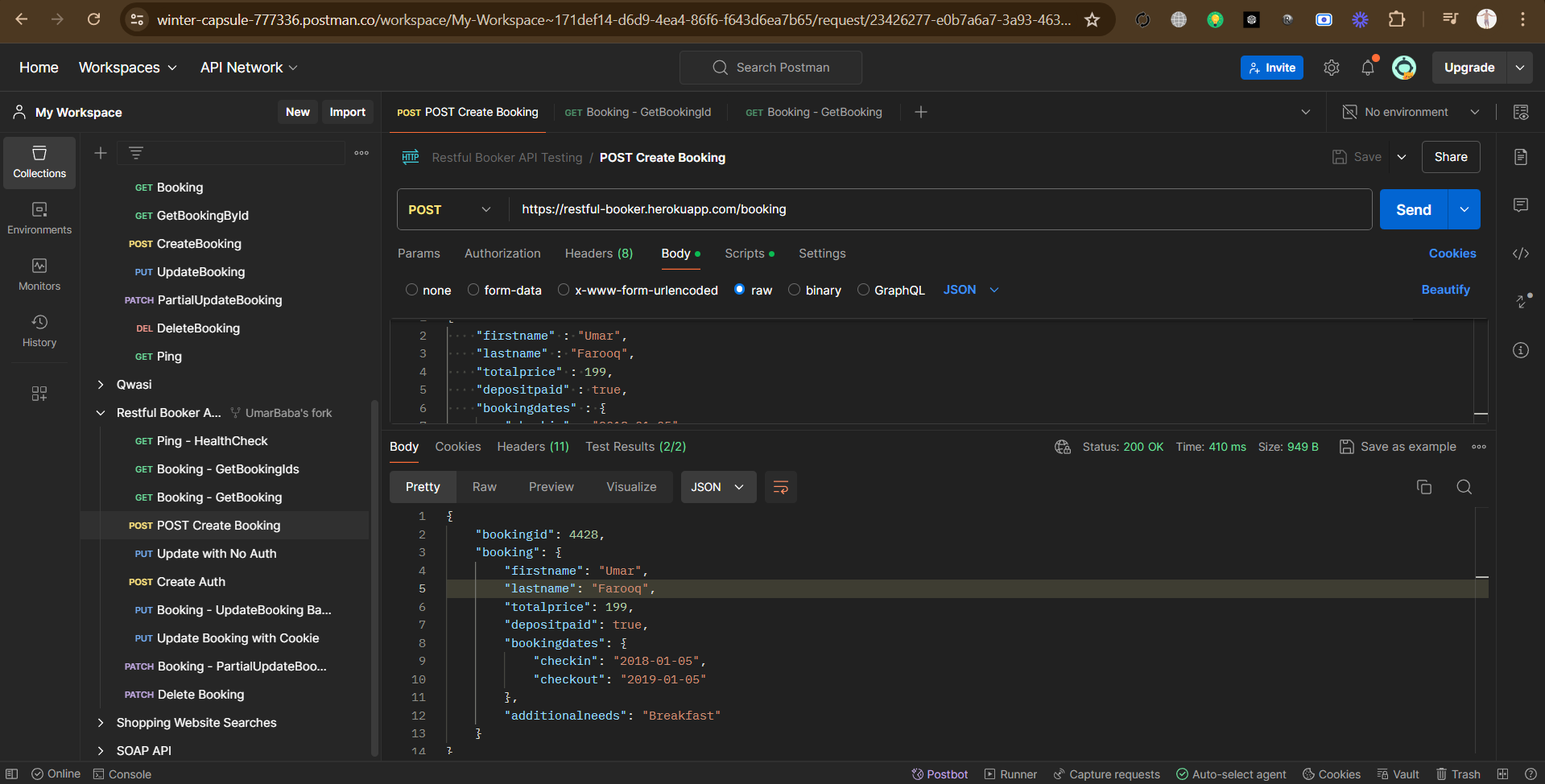
    pm.expect(jsonData.booking.additionalneeds).to.eql("Breakfast");

});





1. Verifies each of the values in the response



1. Verifies that the id of the new booking is present in the response of GetBookingIds

<https://winter-capsule-777336.postman.co/workspace/My-Workspace~171def14-d6d9-4ea4-86f6-f643d6ea7b65/request/23426277-e0b7a6a7-3a93-4633-b844-300bc69dd4a2?tab=body>