No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

1	1. Verify the user can add an existing product ID to the kit.	1. The user selects POST request with URL + api/v1/kits/6/products
		
		2. The user sends a request by using an existing product ID in the path params and request body.
		
		3. The user adds existing products to the kit on the response.
		
		4. The user gets Status: 200 OK on the response.
			Path params: id=6
			
			{
			"productsList": [
			{
			"id": 3,
			"quantity": 3
			}
			]
			}
				- Added existing products to the kit
				
				- Status: 200 OK
					- Added existing products to the kit
					
					- Status: 200 OK
						Passed	

2	Verify the user can add same existing product to the same kit	1. The user selects POST request with URL + "api/v1/kits/6/products"
		
		2. The user adds existing products to the kit in the response.
		
		3. Verify the use receives 200 ok status
		
		4. Verify the products are added to the kit=6
			Path params: id=6
			
			{
			"productsList": [
			{
			"id": 3,
			"quantity": 3
			}
			]
			}
				- Added same existing products to the same kit
				
				- Status: 200 OK
					- Added existing same product to the same kit
					
					- Status: 200 OK
						Passed	

3	verify the user can add multiple products to a kit	1. The user selects POST request with URL + "api/v1/kits/6/products"
		
		2. The user adds existing products to the kit in the response.
		
		3. Verify the use receives 200 ok status
		
		4. Verify the products are added to the kit=6
			path parameters: id=6
			
			{
			"productsList": [
			{
			"id": 1,
			"quantity": 2
			},
			{
			"id": 6,
			"quantity": 2
			}
			]
			}
				- Added multiple products to the kit
				
				- Status: 200 OK
					- Added multiple products to the kit
					
					- Status: 200 OK
						Passed	

4	verify the user cannot add empty body request products to a kit	1. The user selects POST request with URL + "api/v1/kits/4/products"
		
		2. Enter "[]" into the request body field
		
		4. Verify the response code is 400 Bad request.
		
		5. Verify no products are added to the kit
			{
			"productsList": []
			}
				- empty body request cannot be placed into the request body
				
				- 400 bad request
					- 200 ok status
					
					- empty body request is acceted into the body format but does not add to product count
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-38

5	Verify the user cannot add 0 products to the kit	1. The user selects POST request with URL + "api/v1/kits/4/products"
		
		2. Enter in the request body under productId="0"
		
		3. Enter in the request body under the quantity=1
		
		4. Verify the response code is 400 Bad request.
		
		5. Verify no products are added to the kit
			{
			"productsList": [
			{
			"id": "0",
			"quantity": 1
			}]
			}
				- Zero products ID cannot be added to the kit
				
				- 1 quantity cannot be added to the kit
				
				- Status 400 bad request
					- 200 OK
					
					- 0 products ID are added to kit
					
					- 1 quantity is added to kit
					
					{
					"id": 4,
					"name": "Pizza",
					"productsList": [
					{
					"id": 19,
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-10

6	Verify the user can add 1 product to the kit	1. The user selects POST request with URL + "api/v1/kits/3/products"
		
		2. Enter in the request body under productId="1"
		
		3. Enter in the request body under the quantity="1"
		
		4. Verify the response code is 200 OK.
		
		5. Verify productId "1" "Orange Juice" added to the kit 3
			{
			"productsList": [
			{
			"id": 1,
			"quantity": 1
			}
			]
			}
				- "1 orange Juice" is added to KitID=3
				
				- "1" quantity is added to KitID=3
				
				- Status: 200 OK
					- Added 1 orange Juice to the kitID=3
					
					- Added 1 quantity to kitID=3
					
					- Status: 200 OK
						Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

7	Verify the user can add 2 products to the kit	1. The user selects POST request with URL + "api/v1/kits/2/products"
		
		2. Enter in the body request two seperate productsID="1", "2"
		
		3. Enter in the request body under the quantity="1"
		
		4. Verify the response code is 200 OK.
		
		5. Verify 2 seperate products are added to kitID=2
			{
			"productsList": [
			{
			"id": 1,
			"quantity": 1
			},
			{
			"id": 2,
			"quantity": 1
			}
			]
			}
				- Added 2 seperate products to the kitID=2
				
				- Status: 200 OK
					- Added 1 orange juice to kit ID=2
					
					- added 1 mountain dew to kitID=2
					
					- 200 OK status
						Passed	

8	Verify the user can add 29 different products to an empty kit (seperate kits)	1. Create a new KIT#7 POST request and enter the URL: + "/api/v1/kits"
		
		2. Enter this into the body request to create the new Kit#7
		
		{
		"name": "New Kit",
		"cardId": 1
		}
		
		Then once the new kit is created:
		
		1. Make POST request and enter the URL + "/api/v1/kits/7/products"
		
		2. Enter 29 seperate products into the request body.
		
		3.. Verify the response status code 200 OK
		
		4. Verify 29 seperate products can be entered into the kitID=7
			{
			"productsList": [
			{ "id": 1, "quantity": 1 },
			{ "id": 2, "quantity": 1 },
			{ "id": 3, "quantity": 1 },
			{ "id": 4, "quantity": 1 },
			{ "id": 5, "quantity": 1 },
			{ "id": 6, "quantity": 1 },
			{ "id": 7, "quantity": 1 },
			{ "id": 8, "quantity": 1 },
			{ "id": 9, "quantity": 1 },
			{ "id": 10, "quantity": 1 },
			{ "id": 11, "quantity": 1 },
			{ "id": 12, "quantity": 1 },
			{ "id": 13, "quantity": 1 },
			{ "id": 14, "quantity": 1 },
			{ "id": 15, "quantity": 1 },
			{ "id": 16, "quantity": 1 },
			{ "id": 17, "quantity": 1 },
			{ "id": 18, "quantity": 1 },
			{ "id": 19, "quantity": 1 },
			{ "id": 20, "quantity": 1 },
			{ "id": 21, "quantity": 1 }
				- Added 29 seperate products to the kitID=7
				
				- Status: 200 OK
					- 200 OK
					
					{
					"id": 7,
					"name": "New Kit",
					"productsList": [
					{
					"id": 1,
					"name": "Orange Juice - Cold-Pressed, No Added Sugar, Preservative Free",
					"price": 2,
					"weight": 473,
					"units": "ml",
					"quantity": 1
					},
					{
					"id": 2,
					"name": "Mountain Dew Soft Drink",
					"price": 1,
					"weight": 1,
					"units": "l"
						Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

9	Verify the user can add 30 different products to an empty kit (seperate kits)	1. Create a new KIT#8 POST request and enter the URL: + "/api/v1/kits"
		
		2. Enter this into the body request to create the new Kit#8
		
		{
		"name": "New Kit",
		"cardId": 1
		}
		
		Then once the new kit is created:
		
		1. Make POST request and enter the URL + "/api/v1/kits/8/products"
		
		2. Enter 30 seperate products into the request body.
		
		3.. Verify the response status code 200 OK
		
		4. Verify 29 seperate products can be entered into the kitID=8
			{
			"productsList": [
			{ "id": 1, "quantity": 1 },
			{ "id": 2, "quantity": 1 },
			{ "id": 3, "quantity": 1 },
			{ "id": 4, "quantity": 1 },
			{ "id": 5, "quantity": 1 },
			{ "id": 6, "quantity": 1 },
			{ "id": 7, "quantity": 1 },
			{ "id": 8, "quantity": 1 },
			{ "id": 9, "quantity": 1 },
			{ "id": 10, "quantity": 1 },
			{ "id": 11, "quantity": 1 },
			{ "id": 12, "quantity": 1 },
			{ "id": 13, "quantity": 1 },
			{ "id": 14, "quantity": 1 },
			{ "id": 15, "quantity": 1 },
			{ "id": 16, "quantity": 1 },
			{ "id": 17, "quantity": 1 },
			{ "id": 18, "quantity": 1 },
			{ "id": 19, "quantity": 1 },
			{ "id": 20, "quantity": 1 },
			{ "id": 21, "quantity": 1 },
			{ "id": 22, "quantity": 1 },
			{ "id": 23, "quantity": 1 },
			{ "id": 24, "quantity": 1 },
			{ "id": 25, "quantity": 1 },
			{ "id": 26, "quantity": 1 },
			{ "id": 27, "quantity": 1 },
			{ "id": 28, "quantity": 1 },
			{ "id": 29, "quantity": 1 },
			{ "id": 30, "quantity": 1 }
			]
			}
				- Added 30 products to the kit
				
				- Status: 200 OK
					- 200 OK
					
					{
					"id": 8,
					"name": "New Kit",
					"productsList": [
					{
					"id": 1,
					"name": "Orange Juice - Cold-Pressed, No Added Sugar, Preservative Free",
					"price": 2,
					"weight": 473,
					"units": "ml",
					"quantity": 1
					},
					{
					"id": 2,
					"name": "Mountain Dew Soft Drink",
					"price": 1,
					"weight": 1,
					"units": "l",
					"quantity": 1
					},
					{
					"id": 3,
					"name": "Pepsi Soft Drink",
					"price": 2,
					"weight": 1,
					"units": "l",
					"quantity": 1
					},
					{
					"id": 4,
					"name": "Sprite Soft Drink",
					"price": 1,
					"weight": 900,
					"units": "ml",
						Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

10	Verify the user cannot add 31 different products to an empty kit (seperate kits)	1. Create a new KIT#9 POST request and enter the URL: + "/api/v1/kits"
		
		2. Enter this into the body request to create the new Kit#9
		
		{
		"name": "New Kit",
		"cardId": 1
		}
		
		Then once the new kit is created:
		
		1. Make POST request and enter the URL + "/api/v1/kits/9/products"
		
		2. Enter 30 seperate products into the request body.
		
		3.. Verify the response status code 200 OK
		
		4. Verify 29 seperate products can be entered into the kitID=9
			{
			"productsList": [
			{ "id": 1, "quantity": 1 },
			{ "id": 2, "quantity": 1 },
			{ "id": 3, "quantity": 1 },
			{ "id": 4, "quantity": 1 },
			{ "id": 5, "quantity": 1 },
			{ "id": 6, "quantity": 1 },
			{ "id": 7, "quantity": 1 },
			{ "id": 8, "quantity": 1 },
			{ "id": 9, "quantity": 1 },
			{ "id": 10, "quantity": 1 },
			{ "id": 11, "quantity": 1 },
			{ "id": 12, "quantity": 1 },
			{ "id": 13, "quantity": 1 },
			{ "id": 14, "quantity": 1 },
			{ "id": 15, "quantity": 1 },
			{ "id": 16, "quantity": 1 },
			{ "id": 17, "quantity": 1 },
			{ "id": 18, "quantity": 1 },
			{ "id": 19, "quantity": 1 },
			{ "id": 20, "quantity": 1 },
			{ "id": 21, "quantity": 1 },
			{ "id": 22, "quantity": 1 },
			{ "id": 23, "quantity": 1 },
			{ "id": 24, "quantity": 1 },
			{ "id": 25, "quantity": 1 },
			{ "id": 26, "quantity": 1 },
			{ "id": 27, "quantity": 1 },
			{ "id": 28, "quantity": 1 },
			{ "id": 29, "quantity": 1 },
			{ "id": 30, "quantity": 1 },
			{ "id": 31, "quantity": 1 }
			]
			}
				- Unable to add 31 products to the kit
				
				- Status: 400 bad request
					- Unable to add 31 products to the kit
					
					- Status: 400 bad request
						Passed	

11	Verify the user cannot add non existent products=ID to the kit	1. Create a new POST request and enter the URL: + "/api/v1/kits/2/products"
		
		2. Enter into the request body: id=100
		
		3. Verify the response status code 400 bad request
		
		4. Verify the request cannot accept non existent productID into the kit
			{
			"productsList": [
			{
			"id": "100",
			"quantity": 1
			}
			]
			}
				- 400 bad request
				
				- unable to add nonexistent productId to the kit
					- 200 ok status
					
					- id=100 not added to kit but the associated quantity= 1added
					
					{
					"id": 2,
					"name": "For movies and series",
					"productsList": [
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-14

12	Verify the user cannot add products to a non existent KIT ID endpoint	1. Create a new POST request and enter the URL: + "/api/v1/kits/100/products"
		
		3. Verify the response status code 404 not found
			path param = 100
			
			{
			"productsList": [
			{
			"id": 1,
			"quantity": 2
			},
			{
			"id": 2,
			"quantity": 2
			}
			]
			}
				- 404 Not Found	- 404 not found	Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

13	Verify the user cannot add products using invalid KitID's in the path parameter	1. Create a new POST request using the URL: + "/api/v1/kits/abc/products"
		
		2. Verify the response status code 404 not found
			{
			"productsList": [
			{
			"id": 1,
			"quantity": 2
			}
			]
			}
				- 404 Not Found	- 500 internal server error
					
					<!DOCTYPE html>
					<html lang="en">
					<head>
					<meta charset="utf-8">
					<title>Error</title>
					</head>
					<body>
					<pre>Internal Server Error</pre>
					</body>
					</html>
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-15

14	Verify the user cannot add products using missing KitID format for the prodcut ID endpoint	1. Create a new POST request and enter the URL: + "/api/v1/kits//products"
		
		2. Verify the response status code 404 not found
			{
			"productsList": [
			{
			"id": 1,
			"quantity": 2
			},
			{
			"id": 2,
			"quantity": 2
			}
			]
			}
				- 404 Not Found	- 404 not found
					
					<!DOCTYPE html>
					<html lang="en">
					
					<head>
					<meta charset="utf-8">
					<title>Error</title>
					</head>
					
					<body>
					<pre>Cannot POST /api/v1/kits//products</pre>
					</body>
					
					</html>
						Passed	

15	Verify the request body ID is structured correctly	1. Create a new POST request and enter the URL: + "/api/v1/kits/3/products"
		
		2. Verify the response status code 200 OK
		
		3. Verify the structure format is accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": 2
			},
			{
			"id": 6,
			"quantity": 2
			}
			]
			}
				- Correct structure format is input
				
				- Status 200 OK
					- Correct structure format passed
					
					- Status 200 OK
						Passed	

16	Verify the request body ID is structured correctly	1. Create a new POST request and enter the URL: + "/api/v1/kits/3/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify the incorrect structure format is not accepted
			{
			productsList: [
			{
			id: 1
			quantity: 2,
			},
			{
			id: 6,
			quantity: 2
			},
			]
				- Syntax errors have been input
				
				- 400 Bad request
					{
					"code": 400,
					"message": "Expected property name or '}' in JSON at position 5"
					}
						Passed	

17	Verify the user cannot add products with **commas** in the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/5/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify commas in the ID cannot be accepted
			{
			"productsList": [
			{
			"id": "1,",
			"quantity": 1
			}
			
			]
			}
				- Invalid data input using commas shall not be accepted
				
				- 400 Bad request
					-500 internal server error
					
					<!DOCTYPE html>
					<html lang="en">
					
					<head>
					<meta charset="utf-8">
					<title>Error</title>
					</head>
					
					<body>
					<pre>Internal Server Error</pre>
					</body>
					
					</html>
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-16

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

18	Verify the user cannot add products with **dashes** in the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/2/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify dashes in the ID cannot be accepted
			{
			"productsList": [
			{
			"id": "1-",
			"quantity": 1
			}
			
			]
			}
				- Invalid data input using dashes shall not be accepted
				
				- 400 Bad request
					- 500 internal server error
					
					<!DOCTYPE html>
					<html lang="en">
					
					<head>
					<meta charset="utf-8">
					<title>Error</title>
					</head>
					
					<body>
					<pre>Internal Server Error</pre>
					</body>
					
					</html>
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-17

19	Verify the user cannot add products with **decimals** in the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/3/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify decimals in the ID cannot be accepted
			{
			"productsList": [
			{
			"id": ".1",
			"quantity": 1
			}
			
			]
			}
				- Invalid data input using decimals shall not be accepted
				
				- 400 Bad request
					- 500 internal server error
					
					<!DOCTYPE html>
					<html lang="en">
					
					<head>
					<meta charset="utf-8">
					<title>Error</title>
					</head>
					
					<body>
					<pre>Internal Server Error</pre>
					</body>
					
					</html>
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-18

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

20	Verify the user cannot add products with **negative** ID numbers	1. Create a new POST request and enter the URL: + "/api/v1/kits/4/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify negative numbers in the ID cannot be accepted
			{
			"productsList": [
			{
			"id": "-1",
			"quantity": 1
			}
			
			]
			}
				- Invalid data input using negative numbers shall not be accepted
				
				- 400 Bad request
					- 200 OK status
					
					{
					"id": 4,
					"name": "Pizza",
					"productsList": [
					{
					"id": 19,
					"name": "Salami",
					"price": 1,
					"weight": 190,
					"units": "g",
					"quantity": 1
					},
					{
					"id": 20,
					"name": "Bacon",
					"price": 4,
					"weight": 500,
					"units": "g",
					"quantity": 1
					},
					{
					"id": 21,
					"name": "Pastrami",
					"price": 2,
					"weight": 200,
					"units": "g",
					"quantity": 1
					},
					{
					"id": 22,
					"name": "Pepperoni",
					"price": 5,
					"weight": 500,
					"units": "g",
					"quantity": 1
					},
					{
					"id": 71,
					"name": "Processed
					Cheese Slices",
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-19

21	Verify the user cannot add products with **Latin Letters** to the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/5/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify Latin Letters in the ID cannot be accepted
			{
			"productsList": [
			{
			"id": "1A",
			"quantity": 1
			}
			
			]
				- Invalid data input using latin letters shall not be accepted
				
				- 400 Bad request
					- 500 Internal Server Error
					
					<!DOCTYPE html>
					<html lang="en">
					
					<head>
					<meta charset="utf-8">
					<title>Error</title>
					</head>
					
					<body>
					<pre>Internal Server Error</pre>
					</body>
					
					</html>
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-20

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
22	Verify the user cannot add products with **Non- Latin Letters** to the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/6/products" 2. Verify the response status code 400 bad request 3. Verify Non-Latin Letters in the ID cannot be accepted	{ "productsList": [{ "id": "1Ж", "quantity": 1 }] }	- Invalid data input using Non latin letters shall not be accepted - 400 Bad request	- 500 internal server error <!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8"> <title>Error</title> </head> <body> <pre>Internal Server Error</pre> </body> </html>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-21
23	Verify the user cannot add products with **special characters** to the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/2/products" 2. Verify the response status code 400 bad request 3. Verify special characters Letters in the ID cannot be accepted	{ "productsList": [{ "id": "1#", "quantity": 1 }] }	- Invalid data input using special characters shall not be accepted - 400 Bad request	- 500 internal server error <!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8"> <title>Error</title> </head> <body> <pre>Internal Server Error</pre> </body> </html>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-22
24	Verify the user cannot add products with **leading spaces** to the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/3/products" 2. Verify the response status code 400 bad request 3. Verify leading spaces in the ID cannot be accepted	{ "productsList": [{ "id": " 1", "quantity": 1 }] }	- Invalid data input using leading spaces with numbers cannot be accepted - 400 Bad request	- 200 OK status { "id": 3, "name": "Tastes of Paris", "productsList": [{ "id": 61, "name": "Baguette French Recipe", "price": 3, "weight": 350, "units": "g", "quantity": 1	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-24
25	Verify the user cannot add products with **empty string** to the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/4/products" 2. Verify the response status code 400 bad request 3. Verify empty strings in the ID cannot be accepted	{ "productsList": [{ "id": "", "quantity": 1 }] }	- Invalid data input using empty string not accepted - 400 Bad request	- 500 internal server error <!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8"> <title>Error</title> </head> <body> <pre>Internal Server Error</pre> </body> </html>	Failed	https://dgsalsbury1.atlassian.net/html/browse/S4PT-25

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

26	Verify the user cannot add products with **missing parameters** to the ID	1. Create a new POST request and enter the URL: + "/api/v1/kits/5/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify missing parameters in the ID cannot be accepted
			{
			"productsList": [
			{
			"quantity": 2
			}
			]
			}
				- Invalid data input using missing parameters shall not be accepted
				
				- 400 Bad request
					- 200 OK
					
					{
					"id": 5,
					"name": "Pasta",
					"productsList": [
					{
					"id": 15,
					"name": "Vienna Sausages",
					"price": 3,
					"weight": 350,
					"units": "g"
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-26

27	Verify the user cannot add commas to the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/5/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify commas in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": "2,"
			}
			]
			}
				- Invalid comma data input into quantity field not accepted
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"2022,\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-27

28	Verify the user cannot add products with dashes in the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/6/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify dashes in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": "2-"
			}
			]
			}
				- Invalid dashes data input into quantity field not accepted
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"1022-\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-28

29	Verify the user cannot add products with decimals in the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/2/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify decimals in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": "0.2"
			}
			]
			}
				- Invalid decimals data input into quantity field not accepted
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"2720.2\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-29

30	Verify the user cannot add products with negative numbers in the quantity field	1. Create a new POST request and enter the URL: + "/api/v1/kits/3/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify negative numbers in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": "-1"
			}
			]
			}
				- Invalid negative numbers data input into quantity field not accepted
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"191-1\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-30

31	Verify the user cannot add products with Latin Letters to the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/4/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify Latin Letters in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": "A1"
			}
			]
			}
				- Invalid Latin Letter data input into quantity field not accepted
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"151A1\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-31

32	Verify the user cannot add products with Non- Latin Letters to the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/5/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify Latin Letters in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": "一二"
			}
			]
			}
				- Invalid non Latin Letter data input into quantity field not accepted
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"151一二\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-32

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

33	Verify the user cannot add products with special characters to the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/6/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify Latin Letters in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": "1#"
			}
			]
			}
				- Invalid special characters data input into quantity field cannot be added
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"1011#\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-33

34	Verify the user cannot add products with leading spaces to the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/2/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify Latin Letters in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": " 1"
			}
			]
			}
				- Invalid leading spaces data input into quantity field is not accepted
				
				- 400 bad request
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"271 1\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-35

35	Verify the user cannot add products with empty string to the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/3/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify Latin Letters in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1,
			"quantity": ""
			}
			]
			}
				- Invalid empty string data input into quantity should not be accepted
				
				- 400 bad request
					- 200 OK status
					
					- products count is noted as having "191" when requirements state no more than 30 should be allowed.
					
					{
					"id": 3,
					"name": "Tastes of Paris",
					"productsList": [
					{
					"id": 61,
					"name": "Baguette French
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-36

36	Verify the user cannot add products with missing parameters to the quantity	1. Create a new POST request and enter the URL: + "/api/v1/kits/5/products"
		
		2. Verify the response status code 400 bad request
		
		3. Verify Latin Letters in the quantity cannot be accepted
			{
			"productsList": [
			{
			"id": 1
			}
			]
			}
				- 400 bad request
				
				- Missing quantity parameter creating incorrect structure shall not be accepted
					- 500 internal server error
					
					{
					"code": 500,
					"message": "invalid input syntax for integer: \"NaN\""
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-37

37	Verify the name of the service "Fast Delivery" is correct	1. Create a new POST request and enter the URL: + "/fast-delivery/v3.1.1/calculate-delivery.xml" 2. Verify the response status code 200 OK 3. Verify the response name="fast delivery"	<InputModel> <productsCount>2</productsCount> <productsWeight>5.115</productsWeight> <deliveryTime>20</deliveryTime> </InputModel>	<response name="Fast Delivery" - 200 OK	<response name="Fast Delivery" - 200 OK	Passed	
38	Verfy the "ToBeDeliveredTime" is (25-30 min) for the "Fast Delivery" service is correct	1. Create a new POST request and enter the URL: + "/fast-delivery/v3.1.1/calculate-delivery.xml" 2. Verify the response status code 200 OK 3. Verify the response ="tobedeliveredtime" min=25 max=30	<InputModel> <productsCount>2</productsCount> <productsWeight>2.5</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- 200 OK - tobedelivered response time (25-30)	- 200 OK <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime>	Passed	
39	Verify **negative** numbers cannot be calculated into the delivery time response field for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime= "-8" 3. Verify the response status code is 400 bad request 4. Verify the response parameter isItPossibleToDeliver = False	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>-8</deliveryTime> </InputModel>	- 400 bad request - isItPossibleToDeliver=False	- 200 OK - <response name="Fast Delivery" />	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-44
40	Verify **decimal** numbers cannot be calculated into the delivery time response field for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime= "8.75" 3. Verify the response status code is 400 bad request 4. Verify the response parameter isItPossibleToDeliver = False	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>8.75</deliveryTime> </InputModel>	- 400 bad request - isItPossibleToDeliver=False	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-45

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
41	Verify **latin** letters cannot be calculated into the delivery time response field for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime= "A" 3. Verify the response status code is 400 bad request 4. Verify the response parameter isItPossibleToDeliver = False	`<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>A</deliveryTime> </InputModel>`	- 400 bad request - isItPossibleToDeliver=False	- 200 OK `<response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>`	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-46
42	Verify **non latin** letters cannot be calculated into the delivery time response field for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime= "Ж" 3. Verify the response status code is 400 bad request 4. Verify the response parameter isItPossibleToDeliver = False	`<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>Ж</deliveryTime> </InputModel>`	- 400 bad request - isItPossibleToDeliver=False	- 200 OK `<response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>`	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-47
43	Verify **symbols** cannot be calculated into the delivery time response field for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime= "#" 3. Verify the response status code is 400 bad request 4. Verify the response parameter isItPossibleToDeliver = False	`<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>#</deliveryTime> </InputModel>`	- 400 bad request - isItPossibleToDeliver=False	- 200 OK `<response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>`	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-48
44	Verify the operating hours of 0 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=0 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = False	`<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>0</deliveryTime> </InputModel>`	- isItPossibleToDeliver="false" in the response body - 200 OK	- 200 OK `<response name="Fast Delivery"/>`	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-41
45	Verify the operating hours of 1 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=1 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = False	`<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>1</deliveryTime> </InputModel>`	- isItPossibleToDeliver="false" in the response body - 200 OK	- 200 OK `<response name="Fast Delivery"/>`	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-42
46	Verify the operating hours of 2 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=2 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = False	`<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>2</deliveryTime> </InputModel>`	- isItPossibleToDeliver="false" in the response body - 200 OK	- 200 OK `<response name="Fast Delivery"/>`	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-43

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
47	Verify the operating hours of 6 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=6 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = False 5. Verify the toBeDeliveredTime <min>25</min> <max>30</max>	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>6</deliveryTime> </InputModel>	- isItPossibleToDeliver="false" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	- <response name="Fast Delivery" /> - 200 OK	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-9
48	Verify the operating hours of 7 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=7 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = true 5. Verify the toBeDeliveredTime <min>25</min> <max>30</max>	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	Passed	
49	Verify the operating hours of 8 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=8 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = true 5. Verify the toBeDeliveredTime <min>25</min> <max>30</max>	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	Passed	
50	Verify the operating hours of 20 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=20 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = true 5. Verify the toBeDeliveredTime <min>25</min> <max>30</max>	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>20</deliveryTime> </InputModel>	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	Passed	
51	Verify the operating hours of 21 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=21 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = true 5. Verify the toBeDeliveredTime <min>25</min> <max>30</max>	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>21</deliveryTime> </InputModel>	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	- isItPossibleToDeliver="true" in the response body - Verify toBeDeliveredTime 25-30 min/max - 200 OK	Passed	
52	Verify the operating hours of 22 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=22 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = False 5. Verify the toBeDeliveredTime <min>25</min> <max>30</max>	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>22</deliveryTime> </InputModel>	- isItPossibleToDeliver="false" - ToBeDeliveredTime=25-30 - 200 OK	- <response name="Fast Delivery" /> - 200 ok	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-1

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
53	Verify the operating hours of 23 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=23 3. Verify the response status code is 200 OK 4. Verify the response parameter isItPossibleToDeliver = False 5. Verify the toBeDeliveredTime <min>25</min> <max>30</max>	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>23</deliveryTime> </InputModel>	- isItPossibleToDeliver="false" - ToBeDeliveredTime=25-30 - 200 OK	- <response name="Fast Delivery" /> - 200 ok	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-39
54	Verify the operating hours of 24 hours for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter delivertime=24 3. Verify the response status code is 400 bad request 4. Verify the response parameter isItPossibleToDeliver = False	<InputModel> <productsCount>3</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>24</deliveryTime> </InputModel>	- isItPossibleToDeliver="false" - 400 bad request	- <response name="Fast Delivery" /> - 200 ok	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-40
55	Verify product count of 0 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=0 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 7 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>0</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>10</deliveryTime> </InputModel>	- Host delivery cost = 7 - Client delivery cost = 0 - 400 bad request	- hostDeliveryCost="3" - clientDeliveryCost="0" - 200 OK status	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-2
56	Verify product count of 1 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=1 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 status ok	- Host delivery cost = 3 - Client delivery cost = 0 - 200 status ok	Passed	
57	Verify product count of 2 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=2 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>2</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 status ok	- Host delivery cost = 3 - Client delivery cost = 0 - 200 status ok	Passed	
58	Verify product count of 6 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=6 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>6</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
59	Verify product count of 7 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=7 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>7</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 status OK	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	Passed	
60	Verify product count of 8 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=8 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>8</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 200 OK - Host delivery cost = 6 - Client delivery cost = 0	- 200 OK - Host delivery cost = 6 - Client delivery cost = 0	Passed	
61	Verify product count of 7 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=7 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>7</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 200 OK - Host delivery cost = 6 - Client delivery cost = 0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="6" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Passed	
62	Verify product count of 8 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=8 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>8</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	Passed	
63	Verify product count of 9 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=9 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>9</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	Passed	
64	Verify product count of 13 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=13 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>13</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
65	Verify product count of 14 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=14 3. Verify the response status code is 200 OK status 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>14</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	- Host delivery cost = 6 - Client delivery cost = 0 - 200 status ok	Passed	
66	Verify product count of 15 for fast delivery service	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=15 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 7 5. Verify the response parameter clientdeliverycost= 9	<InputModel> <productsCount>15</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 7 - Client delivery cost = 9 - 200 OK	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="6" clientDeliveryCost="6"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-5
67	Verify order weight of 0.0 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=0.0 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>0.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	- hostDeliveryCost="3" - ClientDeliveryCost=3 - 200 OK status	Passed	
68	Verify order weight of 0.1 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=0.1 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>0.1</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	Passed	
69	Verify order weight of 2.4 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=2.4 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>2.4</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	- hostDeliveryCost="3" - 200 OK status	Passed	
70	Verify order weight of 2.5 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=2.5 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>2.5</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 3 - Client delivery cost = 0 - 200 OK status	- hostDeliveryCost="3" - 200 OK status	Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
71	Verify order weight of 2.6 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=2.6 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK	- hostDeliveryCost="6" - clientDeliveryCost="0" - 200 OK status	Passed	
72	Verify order weight of 2.5 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=2.5 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>9</productsCount> <productsWeight>2.5</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 -200 OK	- Host delivery cost = 6 - Client delivery cost = 0 -200 OK status	Passed	
73	Verify order weight of 2.6 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=2.6 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>9</productsCount> <productsWeight>2.6</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	Passed	
74	Verify order weight of 2.7 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=2.7 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>9</productsCount> <productsWeight>2.7</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	Passed	
75	Verify order weight of 5.9 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=5.9 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>9</productsCount> <productsWeight>5.9</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	Passed	
76	Verify order weight of 6.0 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=6.0 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 6 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>9</productsCount> <productsWeight>6.0</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	- Host delivery cost = 6 - Client delivery cost = 0 - 200 OK status	Passed	

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
77	Verify order weight of 6.1 for fast delivery	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight=6.1 3. Verify the response status code is 200 OK 4. Verify the response parameter hostdeliverycost= 7 5. Verify the response parameter clientdeliverycost= 9	<InputModel> <productsCount>9</productsCount> <productsWeight>6.1</productsWeight> <deliveryTime>8</deliveryTime> </InputModel>	- Host delivery cost = 7 - Client delivery cost = 9 - 200 OK	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="6" clientDeliveryCost="6"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-8
78	Verfiy **decimal numbers** cannot be calculated into the products count request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount=1.5 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1.5</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-49
79	Verfiy **negativie numbers** cannot be calculated into the products count request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount="-1" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>-1</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-50
80	Verfiy **latin letters** cannot be calculated into the products count request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount="A" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>A</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-51
81	Verfiy **non-latin letters** cannot be calculated into the products count request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount="Ж" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>Ж</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-52
82	Verfiy **symbols** cannot be calculated into the products count request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount="#" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>#</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-53

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report
83	Verfiy an **empty string** cannot be calculated into the products count request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsCount="" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount></productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-54
84	Verfiy **negative numbers** cannot be calculated into the products weight request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight="-1" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>-1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-55
85	Verfiy **latin letters** cannot be calculated into the products weight request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight="A" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>A</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-56
86	Verfiy **non- latin letters** cannot be calculated into the products weight request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight="Ж" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>Ж</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-57
87	Verfiy **symbols** cannot be calculated into the products weight request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight="Ж" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight>#</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-58
88	Verfiy an **empty string** cannot be calculated into the products weight request body	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter productsWeight="" 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 3 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <productsWeight></productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=3 - clientdeliverycost=0	- 200 OK <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-59

No	Test Cases	Test Steps	Test Data (Request Body)	Expected Result	Actual Result	Status	Link to Bug Report

89	Verfiy **missing parameter string** cannot be accepted into the body request for operatingHours	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml
		
		2. In the request body, remove the parameter for operatingHours
		
		3. Verify the response status code is 400 bad request
		
		4. Verify isItPossibleToDelivery=false in the response body
			<InputModel>
			  <productsCount>1</productsCount>
			  <productsWeight>2.0</productsWeight>
			 </InputModel>
				- 400 bad request
				
				- isitpossibletodeliver=False
					- 500 internal server error
					
					{
					  "code": 500,
					  "message": "Cannot read properties of undefined (reading '0')"
					}
						Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-60

90	Verfiy **missing parameter string** cannot be accepted into the body request for productsCount	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, remove the parameter productsCount 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 7 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsWeight>1.0</productsWeight> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=7 - clientdeliverycost=0	- 500 internal server error {"code":500,"message":"Cannot read properties of undefined (reading '0')"}	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-61
91	Verfiy **missing parameter string** cannot be accepted into the body request for productsWeight	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, removethe parameter productsWeight 3. Verify the response status code is 400 bad request 4. Verify the response parameter hostdeliverycost= 7 5. Verify the response parameter clientdeliverycost= 0	<InputModel> <productsCount>1</productsCount> <deliveryTime>7</deliveryTime> </InputModel>	- 400 bad request - hostdelivercost=7 - clientdeliverycost=0	- 500 internal server error {"code":500,"message":"Cannot read properties of undefined (reading '0')"}	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-62
92	Verfiy **hours:minutes** format cannot be accepted into the body request for DeliveryTime	1. Send a POST request to URL + /fast-delivery/v3.1.1/calculate-delivery.xml 2. In the request body, use the parameter deliverytime="11:30" 3. Verify the response status code is 400 bad request 4. Verify isitpossibletodeliver=False in the response body	<InputModel> <productsCount>1</productsCount> <productsWeight>1.0</productsWeight> <deliveryTime>7:30</deliveryTime> </InputModel>	- 400 bad request - isitpossibletodeliver=False	- 200 OK internal server error <response name="Fast Delivery" isItPossibleToDeliver="true" hostDeliveryCost="3" clientDeliveryCost="0"> <toBeDeliveredTime> <min>25</min> <max>30</max> </toBeDeliveredTime> </response>	Failed	https://dgsalsbury1.atlassian.net/browse/S4PT-63

Test Closure:	Column 1	Column 2
	Value	Rate
Number of test cases	92	100%
Number of passes	38	41.30%%
Number of fails	54	58.70%