G10-1355 / API Testing / API Testing - Lolita Fedorishina

[POST] New User Registration

Priority	Severity	Behavior	Type
High	-	Not set	Other

Layer	Is Flaky	Is Muted	Automation
API	No	No	Manual

Status	Milestone
Actual	Release 09.06.2025

Preconditions

Upload script to the Scripts: Pre-conditions for the whole collection

Parameters

Name	Value
base_url	https://qa.demoshopping.ru
{{username}}	use script
{{password}	use script
Content-Type	application/json
key	use script

Steps to reproduce

#	Step	Action	Expected Result
1	Set [POST] request to {{{base_url}}/register		

2	Fill in Body (raw → JSON)	{
		"username": "{{username}}",
		"password": "{{password}}"
		}
		
		Variables {{username}} and {{password}} should be active, if not, check the presence of the script in Environment/Collection variables and script in Scripts Pre-Request
		
		let testMode = "positive"; // можно менять на "negative"
		
		function getRandomFrom(str) {
		return str.charAt(Math.floor(Math.random() * str.length));
		}
		
		function generateValidUsername(length) {	

#	Step	Action	Expected Result

```
const chars =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567
89_';
let result = '';
for (let i = 0; i < length; i++) {
 result += getRandomFrom(chars);
 }
return result;
}


function generateInvalidUsername(caseType) {
const cyrillic = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя';
const special = '!@#$%^&*()-=+,.';
switch (caseType) {
case "tooShort": return generateValidUsername(2);
case "tooLong": return generateValidUsername(16);
case "invalidChars": return getRandomFrom(cyrillic) +
getRandomFrom(special) + getRandomFrom(cyrillic);
default: return "??";
 }
}


function generateValidPassword(length) {
const letters =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
const digits = '0123456789';
const all = letters + digits;
let password = getRandomFrom(letters) +
getRandomFrom(digits);
for (let i = 2; i < length; i++) {
 password += getRandomFrom(all);
 }
return password;
}


function generateInvalidPassword(caseType) {
const letters = 'abcdefghijklmnopqrstuvwxyz';
const digits = '0123456789';
switch (caseType) {
case "tooShort": return generateValidPassword(7);
case "noDigits": return letters.repeat(8).substring(0, 8);
case "noLetters": return digits.repeat(8).substring(0, 8);
default: return "1234567";
```

#	Step	Action	Expected Result

			}
			}
			
			let username = "";
			let password = "";
			
			if (testMode === "positive") {
			username = generateValidUsername(Math.floor(Math.random() * 13) + 3);
			password = generateValidPassword(Math.floor(Math.random() * 5) + 8);
			} else {
			const usernameCases = ["tooShort", "tooLong", "invalidChars"];
			const passwordCases = ["tooShort", "noDigits", "noLetters"];
			username = generateInvalidUsername(usernameCases[Math.floor(Math.random() * usernameCases.length)]);
			password = generateInvalidPassword(passwordCases[Math.floor(Math.random() * passwordCases.length)]);
			}
			
			// Сохраняем для регистрации и логина
			pm.collectionVariables.set("username", username);
			pm.collectionVariables.set("password", password);

3	Fill in Scripts → Post-response		Script to check
			
			1. the Test Request for correct status answer and time
			2. response is correct
			
			const status = pm.response.code;
			let jsonData = {};
			try {
			jsonData = pm.response.json();
			} catch (e) {
			console.warn("❗ JSON не разобрался:", e);
			}
			
			// Проверка на допустимые статусы
			pm.test("Check if status is 200 or 400", function () {
			pm.expect([200, 400]).to.include(status);
			});

#	Step	Action	Expected Result

```
// === Статус 200 ===
if (status === 200) {
 pm.test("✅ Status 200: Success message present", function
() {
 pm.expect(jsonData).to.have.property("message");
 pm.expect(jsonData.message).to.include("успешно");
 });


// Проверка на наличие токена, если он вдруг появится
if (jsonData.token) {
 pm.test("✅ Optional: Token is present", function () {
 pm.collectionVariables.set("key", jsonData.token);
 pm.expect(jsonData).to.have.property("token");
 });
 } else {
 console.info("ℹ️ Токен не возвращается при регистрации — это
нормально.");
 }
}


// === Статус 400 ===
else if (status === 400) {
 pm.test("⚠️ Status 400: Invalid user data", function () {
 pm.response.to.have.status(400);
 });
}


// === Другие статусы ===
else {
 pm.test("❌ Unexpected status code: " + status, function ()
{
 pm.expect.fail("Received unexpected status: " + status);
 });
}
```

4	Send request		Status code 200 and message about new user registration is present

Attachments

script for postman collection (pre-request).rtf

G10-1356 / API Testing / API Testing - Lolita Fedorishina

[GET] Return a list of all products

Priority	Severity	Behavior	Type
High	-	Not set	Other

Layer	Is Flaky	Is Muted	Automation
API	No	No	Manual

Status	Milestone
Actual	Release 09.06.2025

Preconditions

Upload script to the Scripts: Pre-conditions for the whole collection

Parameters

Name	Value
base_url	https://qa.demoshopping.ru
Content-Type	application/json
key	script

Steps to reproduce

#	Step	Action	Expected Result
1	Set [GET] request to {{base_url}}/products		{{base_url}} is not red

2	Fill in Body (raw → JSON)		[
			{
			"id": 0,
			"name": "string",
			"description": "string",
			"price": 0,
			"category": "string",
			"manufacturer": "string",
			"imageUrl": "string",
			"freeShipping": true
			}
			]

3	Fill in Scripts → Post-response		Script to check 1. the Test Request for correct status answer and time 2. response is correct 3. creation of variable for next tests pm.test("Status 200: Successful request", function () {

#	Step	Action	Expected Result

			pm.response.to.have.status(200);
			});
			
			pm.test("Response is an array of products", function () {
			const jsonData = pm.response.json();
			pm.expect(jsonData).to.be.an("array");
			});
			
			const products = pm.response.json();
			
			if (products.length > 0) {
			const randomProduct = products[Math.floor(Math.random() * products.length)];
			pm.collectionVariables.set("test_product_id", randomProduct.product_id);
			console.log("Random productId set:", randomProduct.product_id);
			} else {
			console.warn("Product list is empty!");
			}

4	Send request		Status code 200 and the list of products is returned

Attachments

script for postman collection (pre-request).rtf

G10-1357 / API Testing / API Testing - Lolita Fedorishina

[PATCH] Update amount of products in the user's cart

Priority	Severity	Behavior	Type
High	-	Not set	Other
Layer	**Is Flaky**	**Is Muted**	**Automation**
API	No	No	Manual
Status	**Milestone**		
Actual	Release 09.06.2025		

Preconditions

Upload script to the Scripts: Pre-conditions for the whole collection

Parameters

Name	Value
base_url	https://qa.demoshopping.ru
cartItemId	use script
test_quantity	use script
Content-Type	application/json
authorization key	use script

Steps to reproduce

#	Step	Action	Expected Result
			All variables are not red If it is red: 1. For url check Environment should be QA 2. for cartItemId check POST request Adding products to user's cart and Scripts Post-response. The following script should present
1	Set [PATCH] request {{base_url}}/cart/{{cartItemId}}		
2	Fill in Scripts → Pre-response		To add random number of items to the product in the cart

```
try {
 const cart = pm.response.json();
const addedProductId =
pm.collectionVariables.get("test_product_id");

 const addedItem = cart.find(item => item.product_id ==
addedProductId);

 if (addedItem && addedItem.cart_item_id) {
  pm.collectionVariables.set("cartItemId",
addedItem.cart_item_id);
  console.log("✅ cartItemId saved:",
addedItem.cart_item_id);
  } else {
  console.warn("⚠️ Product not found in cart:",
addedProductId);
  }
} catch (e) {
  console.error("❌ Failed to parse cart JSON:", e);
}
```

```
const qty = Math.floor(Math.random() * 3) + 2; // 0,1,2
+ 2 → 2..4
pm.collectionVariables.set("test_quantity", qty);
console.log("📦 Quantity to update (2–4):", qty);
```

#	Step	Action	Expected Result

3	Fill in Body (raw → JSON)		{
			"quantity": {{test_quantity}}
			}

4	Fill in Scripts → Post-response		Script to check
			
			1. the Test Request for correct status answer and time
			2. response is correct
			
			const status = pm.response.code;
			
			if (status === 200) {
			pm.test("✅ Status 200: Quantity updated", function () {
			pm.response.to.have.status(200);
			});
			} else if (status === 400) {
			pm.test("❌ Status 400: Bad request", function () {
			pm.expect.fail("Expected 200, but got 400 — check body or path.");
			});
			} else if (status === 401) {
			pm.test("❌ Status 401: Unauthorized", function () {
			pm.expect.fail("Expected 200, but got 401 — token missing or invalid.");
			});
			} else if (status === 404) {
			pm.test("❌ Status 404: Cart item not found", function () {
			pm.expect.fail("Expected 200, but got 404 — cartItemId is wrong or expired.");
			});
			} else if (status === 500) {
			pm.test("❌ Status 500: Server error", function () {
			pm.expect.fail("Expected 200, but got 500 — backend issue.");
			});
			} else {
			pm.test(`❌ Unexpected status: ${status}`, function () {
			pm.expect.fail(`Expected 200, but got unexpected status: ${status}`);
			});
			}

5	Send the request		Test Response should have green status code 200 and message that quantity updated

Attachments

cartItemId and test_quantity.rtf

G10-1358 / API Testing / API Testing - Lolita Fedorishina

[PUT] Full update of the product by ID

Priority	Severity	Behavior	Type
High	-	Not set	Other

Layer	Is Flaky	Is Muted	Automation
API	No	No	Manual

Status	Milestone
Actual	-

Preconditions

Upload script to the Scripts: Pre-conditions for the whole collection

Parameters

Name	Value
base_url	https://qa.demoshopping.ru
product_id	use script
Content-Type	application/json
Authorization key	use script

Steps to reproduce

#	Step	Action	Expected Result
1	Set PUT request with {{base_url}}/products/id/{{product_id}}		All variables are not red

2	Fill in the Body (raw → JSON)		<pre>{
			"name": "string",
			"description": "string",
			"price": 0,
			"category": "string",
			"manufacturer": "string",
			"imageUrl": "string",
			"freeShipping": true
			}</pre>

3	Fill in Scripts → Post-response		Script to check

#	Step	Action	Expected Result

			1. the Test Request for correct status answer and time
			2. response is correct
			
			`const status = pm.response.code;`
			
			`if (status === 200) {`
			`pm.test("✅ Status 200 (OK): Product updated successfully", function () {`
			`pm.response.to.have.status(200);`
			`});`
			`} else if (status === 400) {`
			`pm.test("❌ Status 400 (Bad Request)", function () {`
			`pm.expect.fail("Expected 200, but got 400 Bad Request – invalid or missing data.");`
			`});`
			`} else if (status === 403) {`
			`pm.test("❌ Status 403 (Forbidden)", function () {`
			`pm.expect.fail("Expected 200, but got 403 Forbidden – you don't have permission to update this product.");`
			`});`
			`} else if (status === 404) {`
			`pm.test("❌ Status 404 (Not Found)", function () {`
			`pm.expect.fail("Expected 200, but got 404 – product with this ID not found.");`
			`});`
			`} else {`
			`pm.test(`❌ Unexpected status code: ${status}`, function () {`
			`pm.expect.fail(`Expected 200, but got unexpected status: ${status}`);`
			`});`
			`}`

4	Send the request		Test Response should have green status code 200 and message that product was updated

Attachments

product_id.rtf

G10-1359 / API Testing / API Testing - Lolita Fedorishina

[DELETE] Delete user by ID

Priority	Severity	Behavior	Type
Not set	-	Not set	Other

Layer	Is Flaky	Is Muted	Automation
Not set	No	No	Manual

Status	Milestone
Actual	-

Preconditions

Upload script to the Scripts: Pre-conditions for the whole collection

Parameters

Name	Value
base_url	https://qa.demoshopping.ru
delete_user_id	use script
Content-Type	application/json
Authorization key	use script

Steps to reproduce

#	Step	Action	Expected Result
1	Set DELETE with {{base_url}}/users/{{delete_user_id}}		All variables are not red

2	Fill in Scripts → Post-response		Script to check
			1. the Test Request for correct status answer and time
			2. response is correct
			`pm.test("Status 200: User deleted or 404: Not found", function () { pm.expect([200, 404]).to.include(pm.response.code); });`
			`console.log("Tried to delete user_id:", pm.environment.get("delete_user_id"));`

3	Send the request		Test Response should have an answer that the user was deleted

Attachments

delete_user_id.rtf