TEST CASE DEVELOPMENT

BACKEND (C# APIs): User Authentication

ID	Title	Description	Entry	Expected Result
U1	Login with Valid Credentials	Send a POST request with a valid username and password	{"username": "testuser", "password": "password"}	200 OK with a valid JWT token and successful authentication message.
U2	Test with incorrect username and password	Test error when submitting an incorrect username and password.	{"username": "testuser2", "password": "password2"}	401 Unauthorized with error message "Invalid Credentials"
U3	Login with Valid Username and Empty Password	Test with valid username and empty password field	{"username": "testuser", "password": ""}	400 Bad Request indicating that the "password" field is required.
U4	Login with Valid Password and Empty Username	Test with valid password and empty username.	{"username": "", "password": "password"}	400 Bad Request indicating that the "username" field is required.
U5	Login with Empty Fields	Test with both fields empty.	{"username": "", "password": ""}	400 Bad Request indicating that both fields are required.
U6	Login with Missing Fields	Test sending a request with a missing field.	{"username": "testuser"}	400 Bad Request indicating that the "password" field is missing.
U7	Login with Invalid Payload	Test sending an invalid request.	{username: testuser, password: 123}	400 Bad Request indicating an invalid request format (incorrect JSON structure).

BACKEND (C# APIs): Products

PRODUCT CREATION (POST - api/Product)				
ID	**Title**	**Description**	**Entry**	**Expected Result**
P01	Create a product with valid data	Create a product with valid data.	{"id": 1, "name": "Product", "price": 120}	201 Created with the object of the created product..
P02	Create a product with duplicate ID	Attempt to create a product with an existing ID.	{"id": 1, "name": "Product2", "price": 30}	409 Conflict indicating that the ID already exists.
P03	Create a product with id = 0	Test if an ID equal to 0 is allowed.	{"id": 0, "name": "Product", "price": 120}	400 Bad Request indicating that the ID must be greater than 0.
P04	Create a product with negative id	Attempt to create a product with negative ID.	{"id": -1, "name": "Product", "price": 120}	400 Bad Request indicating that the ID is invalid.
P05	Create a product without a name field	Test omitting the ‘name’ field.	{"id": 1, "price": 120}	400 Bad Request indicating that the ‘name’ field is required.
P06	Create a product with negative price	Attempt to create a product with negative price.	{"id": 1, "name": "Product", "price": -30}	400 Bad Request indicating that the price cannot be negative.
P07	Create a product with price as string	Attempt to create a product with price in incorrect format	{"id": 0, "name": "Product", "price": "free"}	400 Bad Request indicating that the price should be numeric.
P08	Create a product with empty fields	Send a product with all fields empty.	{"id": " ", "name": " ", "price": " "}	400 Bad Request indicating that the fields cannot be empty.

P09	Create a product with duplicate id and name	Attempt to create a product with duplicate id and name.	{"id": 1, "name": "Product", "price": 50}	409 Conflict indicating duplicate ID and name.
READING PRODUCTS (GET - api/Product - api/Product/{id})				
ID	**Title**	**Description**	**Entry**	**Expected Result**
P010	Get list of products	Get complete list of products.	N/A	200 OK with a list of products.
P011	Get an existing product by id	Get an existing product by ID.	id = 1	200 OK with product details.
P012	Get a product that does not exist	Search for a non-existent product.	id = 999	404 Not Found indicating that the product does not exist.
P013	Get a product with negative id	Attempt to get a product using a negative ID.	id = -1	400 Bad Request indicating that the ID is invalid.
P014	Get a product with non-numeric id	Try to get a product using a non-numeric ID.	id = "abc"	400 Bad Request indicating that the ID must be numeric.
P015	Get a product with id in 0	Try searching for a product with ID equal to 0.	id = 0	400 Bad Request indicating that the ID is invalid.
PRODUCT UPDATE (PUT - api/Product/{id})				
ID	**Title**	**Description**	**Entry**	**Expected Result**
P016	Update an existing product	Update an existing product	id = 1, {"id": 1, "name": "Object",	200 OK with success message and the

	existing product	with valid data.	Object ", "price": 1500}	product updated.
P017	Update a non-existing product	Attempt to update a product that does not exist.	id = 999, {"id": 999, "name": "Tablet", "price": 500}	404 Not Found indicating that the product does not exist.
P018	Update a product with duplicate id	Attempt to update a product by assigning it a duplicate ID.	id = 2, {"id": 1, "name": "Duplicate", "price": 200}	409 Conflict indicating that the ID already exists in another product.
P019	Update a product with negative id	Attempt to update a product using a negative ID.	id = 1, {"id": -1, "name": "Test", "price": 100}	400 Bad Request indicating that the ID is invalid.
P20	Update a product eliminating required fields	Attempt to update a product by eliminating the ‘name’ field.	id = 1, {"price": 1200}	400 Bad Request indicating that the ‘name’ field is required.
P21	Update with mismatched URL and payload ID	Update a product with a different ID in the URL and in the body.	id = 2, {‘id’: 3, ‘name’: ‘Mismatch’, ‘price’: 200}	400 Bad Request indicating that the ID in the URL and in the body do not match.

PRODUCT DELETE (DELETE - api/Product/{id})

ID	Title	Description	Entry	Expected Result
P022	Delete an existing product	Delete an existing product.	id = 1	200 OK with a success message indicating that the product was deleted.
P023	Delete a non-existing product	Attempt to delete a product that does not exist.	id = 999	404 Not Found indicating that the product does not exist.
P024	Delete a product with	Attempt to delete a product using a	id = -2	400 Bad Request indicating that the ID

	negative id	using a negative ID.		is invalid.
P025	Attempt to delete a product with id = 0	Test if it is allowed to delete a product with ID equal to 0.	id = 0	400 Bad Request indicating that the ID is invalid.
P026	Delete a product with non-numeric id	Attempt to delete a product with a non-numeric ID.	id = ‘abc’	400 Bad Request indicating that the ID must be numeric.
P027	Delete multiple products at once	Attempt to delete multiple products at once (not allowed).	id = [1, 2, 3]	400 Bad Request indicating that the operation is not allowed for multiple IDs.
BACKEND (C# APIs): Orders				
ORDERS CREATION (POST - api/Order)				
ID	**Title**	**Description**	**Entry**	**Expected Result**
O01	Create an order with valid data	Create an order with valid data.	{ "id": 1, "productName": "Laptop", "quantity": 3, "status": "Pending" }	201 Created with success message and order details created.
O02	Create order with quantity = 0	Attempt to create an order with quantity equal to 0.	{ "id": 2, "productName": "Mouse", "quantity": 0, "status": "Pending" }	400 Bad Request indicating that the quantity must be greater than 0.
O03	Create order with empty status	Attempt to create an order with empty ‘status’ field.	{ "id": 3, "productName": "Keyboard", "quantity": 2, "status": "" }	400 Bad Request indicating that the ‘status’ field is required.
O04	Create order with empty product name	Attempt to create an order with empty product name.	{ "id": 4, "productName": "", "quantity": 2, "status": "Pending" }	400 Bad Request indicating that the field ‘productName’ is required.
			{ "id": 1,	

O05	Create an order with duplicate ID	Attempt to create an order with an existing ID.	"productName": "Monitor", "quantity": 1, "status": "Pending" }	409 Conflict indicating that the ID already exists.

READING ORDERS (GET - api/Order - api/Order/{id})

ID	Title	Description	Entry	Expected Result
O06	Get the list of all orders.	Get the complete list of all existing orders.	N/A	200 OK with an array of existing orders.
O07	Get list when no orders exist.	Get the list when no orders exist.	N/A	200 OK with an empty array ([]).
O08	Get an order by an existing ID	Get an order by its existing ID.	id: 1	200 OK with details of the corresponding order.
O09	Get an order with non-existing ID	Attempt to get an order with an ID that does not exist.	id: 999	404 Not Found indicating that the order does not exist.
O010	Get an order with negative ID	Attempt to obtain an order using a negative ID.	id: -1	400 Bad Request indicating that the ID is invalid.
O011	Get an order with non-numeric ID	Attempt to get an order with a non-numeric ID.	id = "abc"	400 Bad Request indicating that the ID must be numeric.
O012	Get an order with id = 0	Attempt to get an order with ID equal to 0.	id = 0	400 Bad Request indicating that the ID cannot be equal to 0.

ORDER UPDATE (PUT - api/Order/{id})

ID	Title	Description	Entry	Expected Result
O013	Update an existing order with valid data.	Update an existing order with valid data.	id: 1 { "id": 1, "productName": "Monitor", "quantity": 2, "status": "Shipped" }	200 OK with success message and order details updated.
O014	Update order with quantity = 0.	Attempt to update an order with quantity equal to 0.	id: 1 { "id": 1, "productName": "Monitor", "quantity": 0, "status": "Pending" }	400 Bad Request indicating that the quantity must be greater than 0.
O015	Update order with empty status	Attempt to update an order with empty ‘status’ field.	id: 1 { "id": 1, "productName": "Monitor", "quantity": 2, "status": "" }	400 Bad Request indicating that the ‘status’ field is required.
O016	Update non-existent order.	Attempt to update an order that does not exist.	id: 999 { "id": 999, "productName": "Tablet", "quantity": 1, "status": "Pending" }	404 Not Found indicating that the order does not exist.

ORDER DELETE (DELETE - api/Order/{id})

ID	Title	Description	Entry	Expected Result
O017	Delete an existing order	Delete an existing order.	id = 1	OK with success message indicating that the order was deleted.
O018	Delete a non-existent order	Attempt to delete an order that does not exist.	id = 999	404 Not Found indicating that the order does not exist.
O019	Delete an order with negative ID	Attempt to delete an order with a negative ID.	id = -1	400 Bad Request indicating that the ID is invalid.

O020	Delete an order with non-numeric ID	Attempt to delete an order using a non-numeric ID.	id = ‘abc’	400 Bad Request indicating that the ID must be numeric.
O021	Delete an order with id = 0	Attempt to delete an order with ID equal to 0.	id = 0	400 Bad Request indicating that the ID cannot be equal to 0.

FRONTEND (ReactJS)

ID	Title	Description	Steps	Expected Result
U01	Login with valid credentials	The user tries to log in with a valid username and password.	1. Go to login. 2. Enter a valid username and password. 3. Click on the button.	The user is successfully logged in, a token is generated and the user is redirected to the Dashboard.
U02	Login with valid credentials	The user tries to log in with an invalid username and password.	1. Go to login. 2. Enter a valid username and password. 3. Click the button.	The user cannot log in, and an error message appears.
P01	View the list of products	The user navigates to the Products page to see the available products.	1. Go to Products. 2. Verify that there is a list of available products.	The user can see the products added to the list.
O01	View the list of orders	The user navigates to the Orders page to see their past orders.	1. Go to Orders. 2. Verify that there is a list of the user's orders	The user can see his orders added to the list.
R01	Check if the page is responsive	Ensure that the website is responsive and adapts to various screen sizes and devices.	1. Access each page address on responsive computers. 2. Check that the page adapts to each screen.	The user can access the site from any device, making the design adaptable and intuitive.