PART 2: TEST CASE DEVELOPMENT

Instructions for use of the table:

- ***ID:** Unique identifier of the test case, useful for tracking.*
- ***Test Case Title**: Brief description of the purpose of the test case.*
- ***Description:** More detailed explanation about what is being tested and why it is relevant.*
- ***Steps to Execute:** Detailed instructions on how to carry out the test.*
- ***Expected Result:** What should happen if the system works correctly.*

BACKEND (C# APIS):

ID	Test Case Title	Description	Steps to Execute	Expected Result
TC001	Login with Valid Credentials	Verify that users can successfully log in with valid credentials.	1. Send a POST request to /api/User/login with valid credentials.	The API responds with a valid authentication token and status code 200.
TC002	Login with Invalid Credentials	Validate that the system rejects login with incorrect credentials.	1. Send a POST request to /api/User/login with invalid credentials.	The API responds with an error message and status code 401.
TC003	Login with both fields empty.	Validate that the system does not authorize login with empty fields.	1. Send a POST request to /api/User/login without placing any credentials.	The API responds with an error message and status code 401.
TC004	Login with empty user field.	Validate that the system does not authorize login with an empty user field.	1. Send a POST request to /api/User/login without placing the user, but placing the password.	The API responds with an error message and status code 401.
TC005	Login with empty password field.	Validate that the system does not authorize login with an empty password field.	1. Send a POST request to /api/User/login without putting the password, but putting the user.	The API responds with an error message and status code 401.
TC006	Create Product	Verify that the API allows to create a product with valid data.	1. Send a POST request to /api/Product with all required data.	The API responds with a success message and a 201 status code.
TC007	Consult Product List	Verify that the API allows consulting the list of products correctly.	1. Send a GET request to /api/Product.	The API responds with the complete list of products and a 200 status code.
TC008	Create Product with Existing ID	Validate that the API properly handles the creation of a product when the ID already exists.	1. Send a POST request to /api/Product using an ID that already exists.	The API responds with an error message and a 409 status code, indicating that the ID already exists.
TC009	Browse Product by ID	Verify that the API allows querying a	1. Send a GET request to	The API responds with the requested product

		specific product using its ID.	/api/Product with a specific ID.	details and a 200 status code.
TC010	Edit Product by ID	Verify that the API allows editing a specific product using its ID.	1. Send a PUT request to /api/Product with a specific ID and new data to update the product.	The API responds with a success message and a status code 200.
TC011	Delete Product by ID	Verify that the API allows you to delete a specific product using its ID.	1. Send a DELETE request /api/Product with a specific ID.	The API responds with a success message and a status code 200.
TC012	Product Upgrade with Nonexistent ID	Test system behavior when attempting to update a product that does not exist.	1. Send a PUT request to /api/Product with a product ID that does not exist.	The API responds with an error message and 404 status code.
TC013	Elimination of Product with Inexistent ID	Verify the behavior of the system when an attempt is made to delete a product that does not exist.	1. Send a PUT request to /api/Product with a product ID that does not exist.	The API responds with an error message and 404 status code.
TC014	View all orders	Verify that the API correctly returns all existing orders.	1. Send a GET request to /api/Order without additional parameters.	The API responds with a complete list of commands and a status code 200.
TC015	Search for an order using an existing ID.	Verify that the API returns a specific order when a valid ID is provided.	1. Send a GET request to the /api/Order with a valid order ID as parameter.	The API responds with the details of the specific order and a 200 status code.
TC016	Search an order for a non-existent ID.	Test the behavior of the API when requesting an order with an ID that does not exist.	1. Send a GET request to /api/Order with an order ID that does not exist.	The API responds with an error message and a 404 status code.
TC017	Search order with invalid ID	Validate that the API properly handles the request when an invalid ID (e.g., non-numeric) is provided.	1. Send a GET request to /api/Order with an invalid order ID.	The API responds with an error message and a status code 400 or 422.
TC018	Create order with valid data	Verify that the API allows to create an order with all valid data.	1. Send a POST request to /api/Order with valid values.	The API responds with a success message and a 201 status code, and the order is successfully saved.
TC019	Create order with empty productName	Validate that the API properly handles the creation of an order when the productName is empty.	1. Send a POST request to /api/Order with empty productName in the body.	The API responds with an error message and a status code 400 or 422.
TC020	Create order with quantity at 0	Verify that the API rejects the creation of an order when quantity	1. Send a POST request to /api/Order	The API responds with an error message and a status code 400 or 422.

		is equal to 0, since it does not make sense to create an order without products.	with quantity equal to 0 in the body.	
TC021	Create order with negative quantity	Validate that the API rejects the creation of an order when quantity is negative.	1. Send a POST request to /api/Order with negative quantity in the body.	The API responds with an error message and a status code 400 or 422.
TC022	Create order with empty status	Verify that the API properly handles the creation of an order when status is empty.	1. Send a POST request to /api/Order with empty status in the body.	The API responds with an error message and a status code 400 or 422.
TC023	Create order with duplicate ID	Validate that the API rejects the creation of an order when the id already exists.	1. Send a POST request to /api/Order with an id that already exists.	The API responds with an error message and a 409 status code, indicating ID duplication.
TC024	Create order with all empty fields	Validate that the API rejects the creation of an order when all fields are empty.	1. Send a POST request to /api/Order with all empty fields in the body.	The API responds with an error message and a status code 400 or 422.
TC025	Create order with incorrect data type	Verify that the API properly handles the creation of an order when sending an incorrect data type (e.g., quantity as string).	1. Send a POST request to /api/Order with an incorrect data type in one of the fields.	The API responds with an error message and a status code 400 or 422.
TC026	Update order with valid data (without changing the ID)	Verify that the API allows updating an existing order with all valid data except the id, which must not be editable.	1. Send a PUT request to /api/Order/{id} with a body that includes productName, quantity, and status with valid values.	The API responds with a success message and a status code 200 or 204, and the order is successfully updated in the database, without changing the id.
TC027	Update the ID of an order	Verify that the API does not allow to modify the id of an existing order.	1. Send a PUT request to /api/Order/{id} with a body that includes a different id than the current one.	The API responds with an error message and a status code 400 or 409, indicating that the id is not editable.
TC028	Update order with empty productName	Validate that the API properly handles the update of an order when the productName is empty.	1. Send a PUT request to /api/Order/{id} with empty productName in the body.	The API responds with an error message and a status code 400 or 422.
TC029	Update order with quantity at 0	Verify that the API rejects the update of an order when quantity equals 0	1. Send a PUT request to /api/Order/{id} with quantity at 0 in the body.	The API responds with an error message and a status code 400 or 422, indicating that quantity cannot be 0.
TC030	Update order with negative quantity	Validate that the API rejects the update of	1. Send a PUT request to /api/Order/{id} with	The API responds with an error message and a status code 400 or 422.

		an order when quantity is negative.	negative quantity on the body.	
TC031	Update the order with an empty status	Verify that the API properly handles the update of an order when status is empty.	1. Send a PUT request to /api/Order/{id} with empty status in the body.	The API responds with an error message and a status code 400 or 422.
TC032	Update order with non-existent ID	Test the behavior of the API when trying to update an order with an ID that does not exist.	1. Send a PUT request to /api/Order/{id} with an id that does not exist.	The API responds with an error message and a 404 status code.
TC033	Update order with all fields empty	Validate that the API rejects the update of an order when all fields are empty.	1. Send a PUT request to /api/Order/{id} with all empty fields in the body.	The API responds with an error message and a status code 400 or 422.
TC034	Update order with incorrect data type	Verify that the API properly handles the update of an order when an incorrect data type is sent (e.g., quantity as string).	1. Send a PUT request to /api/Order/{id} with an incorrect data type in one of the fields.	The API responds with an error message and a status code 400 or 422.
TC035	Delete order with existing ID	Verify that the API allows deleting an existing order with a valid order ID.	1. Send a DELETE request to /api/Order/{id} with a valid id in the URL.	The API responds with a success message and a status code 200 or 204, and the order is deleted.
TC036	Delete order with non-existent ID	Verify that the API properly handles the request when trying to delete an order with an ID that does not exist.	1. Send a DELETE request to /api/Order/{id} with an id that does not exist.	The API responds with an error message and a 404 status code, indicating that the command was not found.
TC037	Delete order with non-numeric ID	Verify that the API handles the request properly when trying to delete an order with a non-numeric ID.	1. Send a DELETE request to /api/Order/{id} with a non-numeric id in the URL (e.g., a string).	The API responds with an error message and a status code 400, indicating an invalid ID format.
TC038	Delete order with negative ID	Verify that the API properly handles the request when trying to delete an order with a negative ID	1. Send a DELETE request to /api/Order/{id} with a negative id in the URL.	The API responds with an error message and a status code 400, indicating an invalid ID format.
TC039	Delete order without authorization	Verify that the API properly handles the request when an attempt is made to delete an order without proper authorization.	1. Send a DELETE request to /api/Order/{id} without a valid authorization token or credentials.	The API responds with an error message and a 401 status code, indicating lack of authorization.

Frontend (ReactJS)

ID	Test Case Title	Description	Steps to Execute	Expected Result
TC001	Login with Valid Credentials	Verify that users can successfully log in with valid credentials.	1. Navigate to the login page. 2. Enter a valid username and password. 3. Click on “Login”.	The user is redirected to the home page and a welcome message is displayed.
TC002	Login with Invalid Credentials	Validate that the system rejects login with incorrect credentials.	1. Navigate to the login page. 2. Enter an invalid username and password. 3. Click on “Login”.	An error message appears indicating that the credentials are incorrect.
TC003	Viewing the Product List	Verify that the product list is loaded correctly and shows updated information.	1. Navigate to the product page. 2. Verify that all available products are correctly displayed with their information.	The product list is successfully loaded and displays the correct information.
TC004	User Interface Responsibility	Ensure that the user interface fits correctly on different devices.	1. Accessing the application from a mobile device. 2. Navigate through the different sections of the application. 3. Verify the visualization on desktop, tablet and mobile.	The interface adapts adequately to different screen sizes.