TEST EXECUTION AND REPORTING

User Authentication (AU)- BACKEND (C# APIs):			
ID	**TITLE**	**RESULT**	**STATUS**
AU01	Verify login with valid credentials	PASS	200 OK response with a valid JWT token and a success message.
AU02	Login attempt with wrong credentials	PASS	401 Unauthorized response with message “Incorrect credentials”.
AU03	Login with empty fields	FAIL	Responde 401 Unauthorized instead of 400 Bad Request
AU04	Login with correct username and empty password field.	FAIL	Responde 401 Unauthorized instead of 400 Bad Request
AU05	Login with correct password and empty username field	FAIL	Responde 401 Unauthorized instead of 400 Bad Request
AU06	Login attempt with invalid JSON formatting	FAIL	Responde 401 Unauthorized instead of 400 Bad Request

BACKEND (C# APIs): Product Management: (Create Product - POST)			
ID	**TITLE**	**RESULT**	**STATUS**

PC01	Create product with valid credentials	PASS	Response 201 Created with the data of the created product.
PC02	Create product with negative price	FAIL	Response 201 created, instead 400 Bad Request. Negative price allowed, validation for price missing.
PC03	Create a product without the “name” field	FAIL	Response 201 created, 400 Bad Reques. Empty name allowed, validation for name field missing
PC04	Attempting to create a product ith a duplicate ID	FAIL	Response 201 created instead 409 Conflict response. Missing validation for duplicate entries
PC05	Create a product with empty fields	PASS	400 confilct response with the message "Fiileds cannot be empty"
PC06	Create a product without the “price” field	PASS	Response 400 Bad Request indicating that the “price” field is obligatory.
PC07	Create product with incorrectly formatted price	PASS	Response 400 Bad Request indicating that the “price” field is obligatory.
BACKEND (C# APIs): Product Management: (Reding Products - GET)			
ID	**TITLE**	**RESULT**	**STATUS**
PC08	Get an existing product by ID	Pass	Response returned 200 OK. The product with the specified ID was GET successfully.

PC09	Get product list	Pass	Response returned 200 OK. The product list was retrieved and displayed correctly.
PC10	Get a product with invalid ID	Pass	Response returned 400 Bad Request. The server correctly handled the request with an invalid product ID by returning an appropriate error.
PC11	Get non-existing product	Pass	Response returned 404 Not Found. The server correctly identified the non-existing product and returned the expected error response.
PC12	Get a product with negative id	Fail	Response returned 200 OK instead 400 Not Found. Missing validation for negative product IDs, as these should not be accepted.
BACKEND (C# APIs): Product Management: (Update Products - PUT)			
ID	**TITLE**	**RESULT**	**STATUS**
PC13	Update an existing product	Fail	204 No Content. Response returned 204 No Content. The existing product was updated successfully with no issues.
PC14	Update non-existent product	Pass	Response returned 404 Not Found. The server correctly identified that the product to be updated does not exist.
PC15	Update product with negative price	Fail	Response returned 204 No Content instead 400 Bad Request. Missing validation for negative price values, which should not be accepted.

PC16	Update a product with negative ID	Fail	Response returned 204 No Content 400 Bad Request. Missing validation for negative product IDs, which should not be allowed.
PC17	Update a product with missing fields	Fail	Response returned 204 No Content instead of 400 Bad Request. Missing validation for required fields when updating a product.

BACKEND (C# APIs): Product Management: (Delete Products - DELETE)			
ID	**TITLE**	**RESULT**	**STATUS**
PC18	Delete an existing product	Fail	Response returned 204 No Content. The product was successfully deleted
PC19	Delete a non-existing product	Pass	esponse returned 404 Not Found. The server correctly identified that the product to be deleted does not exist.
PC20	Delete a product without specifying an ID	Pass	Response returned 400 Bad Request. The request failed due to the absence of a specified product ID, as expected.
PC21	Delete a product with invalid negative ID	Fail	Response returned 204 No Content instead of Bad Request. Missing validation for negative IDs, which should not be accepted in delete requests.
PC22	Delete a product with ID set to zero	Fail	Response returned 204 No Content instead of 400 Bad Request. Missing validation for IDs set to zero, which are invalid for delete requests.

BACKEND (C# APIs): Order Processing: (Create Order- POST)			
ID	**TITLE**	**RESULT**	**STATUS**
OC01	Create order with valid data	Pass	Response returned 201 Created. The order was successfully created with valid data, and the response includes the order ID.
OC02	Create an order ith zero quantity	Fail	Response returned 201 Created instead of 400 Bad Request. Missing validation for zero quantities, which should trigger a 400 Bad Request or similar error.
OC03	Create order with missing status	Fail	Response returned 201 Created instead of 400 Bad Request. Missing validation for the required status field, which should not allow order creation with incomplete data.
OC04	Create order with Invalid data types in the fields	Pass	Response returned 400 Bad Request. The system correctly rejected the order due to invalid data types in the input fields.
OC05	Create order with an existing order ID	Fail	Response returned 201 Created instead of 409 Conflict. Missing validation to detect duplicate order IDs, which should result in a 409 Conflict or similar error.
BACKEND (C# APIs): Order Processing: (Reding Order- GET)			
ID	**TITLE**	**RESULT**	**STATUS**
OC06	Get an existing Order by ID	Pass	Response returned 200 OK. The existing order was successfully retrieved, and the response includes the correct order details.
OC07	Get Order list	Pass	Response returned 200 OK. The list of orders was successfully retrieved, displaying all available orders in the

			displaying all available orders in the system.
OC08	Get an order with invalid ID	Pass	Response returned 400 Bad Request. The system correctly rejected the request for an invalid order ID, maintaining proper validation rules.
OC09	Get non-existing order	Pass	Response returned 404 Not Found. The system correctly indicated that the requested order does not exist.
OC10	Get an order with negative id	Fail	Response returned 200 OK instead of 400 Not Found. Missing validation for negative order IDs, which should return a 400 Bad Request or 404 Not Found.
OC11	Get an order with id=0	Fail	Response returned 200 OK instead of 400 Not Found. Missing validation for order IDs set to 0, which should return a 400 Bad Request or similar error.
BACKEND (C# APIs): Order Processing: (Updated Order- PUT)			
ID	**TITLE**	**RESULT**	**STATUS**
OC12	Update an existing order	Fail	Response 204 No Content instead of 2OO OK
OC13	Update non-existent order	Pass	Response returned 404 Not Found. The system correctly indicated that the order does not exist, ensuring proper validation for non-existing orders.
OC14	Update an order with negative quantity	Fail	Response returned 204 No Content instead of 400 Bad Request. Missing validation for orders with negative quantities

OC15	Update an order with empty status fields	Fail	esponse returned 204 No Content instead of 400 Bad Request. Missing validation for orders with empty status fields

BACKEND (C# APIs): Order Processing: (Delete Order- DELETE)			
ID	**TITLE**	**RESULT**	**STATUS**
OC16	Delete an existing order	Fail	Response 204 No Content instead of 2OO OK
OC17	Delete a non-existing order	Pass	Response returned 404 Not Found. The system correctly indicated that the order does not exist
OC18	Delete an order without specifying an ID	Pass	Response returned 400 Bad Request. The system correctly rejected the request due to a missing order ID, maintaining required validation checks.
OC19	Delete an order with invalid negative ID	Fail	Response returned 204 No Content instead of 400 Bad Request. Missing validation for invalid negative order IDs
OC20	Delete an order with ID set to zero	Fail	Response returned 204 No Content instead of 400 Bad Request. Missing validation for order IDs set to 0.

FRONTEND (ReactJS)			
ID	**TITLE**	**DESCRIPTION**	**RESULT EXPECTED**

U1	Successful login with correct credentials	Successful login with correct credentials	The user can log in with the credentials "testuser" and "password," and a token is generated. However, the system does not redirect to the dashboard.
U2	Login attempt with incorrect credentials	Login attempt with incorrect credentials	The user cannot log in with incorrect credentials, and a message is displayed indicating that the login has failed.
U3	View product list	Ensure the product list is loaded correctly when visiting the products page.	When the user navigates to the products section, they can view a list of products created in the backend.
U4	View order list	Ensure the user can view their list of placed orders.	When accessing the orders section, the user can view a list of orders by product, which have been created in the backend.
U5	Test how the UI adapts to different devices. (Responsive)	Test how the page layout adjusts to various screen sizes (mobile, tablet, desktop).	The page has no design, so it is not possible to evaluate its responsiveness on devices with different screen sizes.