PART 2: TEST CASE DEVELOPMENT

User Authentication (AU)- BACKEND (C# APIs):

ID	TITLE	DESCRIPTION	ENTRY	RESULT EXPECTED
AU01	Verify login with valid credentials	A POST request is made with valid login credentials.	{"username": "testuser", "password": "password"}	200 OK response with a valid JWT token and a success message.
AU02	Login attempt with wrong credentials	Attempting to log in with incorrect credentials	{"username": "user_erroneo", "password": "wrongPassword"}	401 Unauthorized response with message “Incorrect credentials”.
AU03	Login with empty fields	Test with username and password fields empty	{"username": "", "password": ""}	Response 400 Bad Request with a message that both fields are required.
AU04	Login with correct username and empty password field.	Test login with correct username and empty password field	{"username": "testuser", "password": ""}	Response 400 Bad Request with a message that the "password" field is required.
AU05	Login with correct password and empty username field	Test login with correct password and empty username field	{"username": "", "password": "password"}	Response 400 Bad Request with a message that the "username" field is required.
AU06	Login attempt with invalid JSON formatting	Test when sending poorly structured data.	{"username": "testuser", "password": "$$$"}	Response 400 Bad Request with a menssage that an invalid request format (incorrect JSON structure).

BACKEND (C# APIs): Product Management: (Create Product - POST)

ID	TITLE	DESCRIPTION	ENTRY	RESULT EXPECTED
PC01	Create product with valid credentials	Verify that a product can be created with correct data.	{"id": 1, name": "Product one", "price": 10}	Response **201** Created with the data of the created product.
PC02	Create product with negative price	Check the behavior of the system when trying to create a product with a negative price.	{"id": 2, name": "Product Two", "price": -40}	Response **400** Bad Request indicating that the price cannot be negative.
	Create a product without	Verify that the system returns an error if the	{"id": 3	Response 400 Bad Request

PC03	Create a product without the “name” field	returns an error if the product name is missing.	{"id": 2, "price": 10}	Response 400 Bad Request indicating that the “name” field is obligatory.
PC04	Attempting to create a product with a duplicate ID	Verify that a product cannot be created with an ID that already exists.	{"id": 1, name": "Product Four", "price": 150}	409 Conflict response with the message “ID already exists”.
PC05	Create a product with empty fields	Verify that a product cannot be created with empty fields	{"id": "", "name": "", "price": ""}	400 confilct response with the message "Fiileds cannot be empty"
PC06	Create a product without the “price” field	Verify that the system returns an error if the price is missing.	{"id": 3, name": "Product five", "price": }	Response 400 Bad Request indicating that the “price” field is obligatory.
PC07	Create product with incorrectly formatted price	Check that only valid formats are allowed for the price.	{"id": 5, name": "Product six", "price": %%% }	Response 400 Bad Request indicating “Invalid price format”.

BACKEND (C# APIs): Product Management: (Reding Products - GET)

ID	TITLE	DESCRIPTION	ENTRY	RESULT EXPECTED
PC08	Get an existing product by ID	Get a product using a valid ID.	i:3,	Response 200 OK with product details.
PC09	Get product list	Get the product list complete	N/A	Response 200 OK with the product list.
PC10	Get a product with invalid ID	Attempt to obtain a product with a non-numeric ID.	i: abc,	400 Bad Request indicating that the ID must be numeric.
PC11	Get non-existing product	Attempting to get a product that does not exist in the liist.	i: 800,	Response 404 Not Found with a messege that the product does not exist.
PC12	Get a product with negative id	Attempting to get a product with a ID negative	i:-10,	Response 400 Not Found with a messege that the ID is invalid

BACKEND (C# APIs): Product Management: (Update Products - PUT)				
ID	**TITLE**	**DESCRIPTION**	**ENTRY**	**RESULT EXPECTED**
PC13	Update an existing product	Verify that an existing product can be updated with a valid ID.	id: 5, {"id": 1, name": "Order one", "price": 244}	Response 200 OK with confirmation of updated data.
PC14	Update non-existent product	Verify that you cannot update a product that does not exist.	id: 200, {"id": 22, name": "Order five", "price": 1500}	Response 404 Not Found with a message indicating “Product not found”.
PC15	Update product with negative price	Verify that negative prices are not allowed to be assigned to the product.	id: 200, {"id": 22, name": "Order five", "price": -1500}	Response 400 Bad Request indicating “Price cannot be negative”.
PC16	Update a product with negative ID	Verify that updating a product with a negative ID is not allowed.	id: 200, {"id": -200, 'name": "Order six", "price": 300}	Response 400 Bad Request indicating “ID cannot be negative”.
PC17	Update a product with missing fields	Verify that it is not allowed to update a product by omitting obligatory fields.	id = 3, {"price": 120}	Response 400 Bad Request with a message indicanting that that the ‘name’ field is require.

BACKEND (C# APIs): Product Management: (Delete Products - DELETE)				
ID	**TITLE**	**DESCRIPTION**	**ENTRY**	**RESULT EXPECTED**
PC18	Delete an existing product	Confirm that an existing product can be successfully removed from the system.	id: 20,	Response 200 OK with a confirmation message indicating that the product has been deleted.
PC19	Delete a non-existing product	Ensure that the system responds correctly when an attempt is made to remove a product that does not exist.	id: 1000,	Response 404 Not Found with a message indicating the product was not found.
PC20	Delete a product without specifying an ID	Test how the system reacts when the ID is omitted from the delete request.	id: ,	400 Bad Request with an error indicating that the ID is required.
PC21	Delete a product with invalid negative ID	Test the behavior when trying to delete a product using a negative number as the ID.	id: -2,	Response 400 Bad Request with a message stating the ID provided is invalid.
	Delete a product with ID	Validate that products		400 Bad Request indicating the

PC22	Delete a product with ID set to zero	validate that products with an ID equal to 0 cannot be removed.	id: 0,	400 Bad Request indicating the provided ID cannot be zero.

BACKEND (C# APIs): Order Processing: (Create Order- POST)

ID	TITLE	DESCRIPTION	ENTRY	RESULT EXPECTED
OC01	Create order with valid data	Verify that an order can be created with correct data.	{ "id": 1, roductName": "Product one", "quantity": 5, "status": "Done" }	201 Created with a success message and details of the created order.
OC02	Create an order with zero quantity	Ensure the system handles attempts to place an order with a quantity set to 0.	{ "id": 1, roductName": "Product Two", "quantity": 0, "status": "Pending" }	Response 400 Bad Request with a message indicating the quantity must be greater than zero.
OC03	Create order with missing status	Validate system behavior when the "status" field is left empty.	{ "id": 6, roductName": "Product one", "quantity": 20, "status": "" }	Response 400 Bad Request stating the status field is required.
OC04	Create order with Invalid ata types in the fields	Ensure proper error handling when invalid data types (e.g., strings for quantity) are sent.	{ "id": 1, roductName": "Product Five", "quantity": "two", "status": "Done" }	400 Bad Request indicating the fields must have valid data types.
OC05	Create order with an existing order ID	Validate that the system prevents the creation of a new order using an already existing order ID.	{ "id": 1, roductName": "Product Nine", "quantity": 27, "status": "In progress" }	409 Conflict indicating the order ID is already in use.

BACKEND (C# APIs): Order Processing: (Reding Order- GET)

ID	TITLE	DESCRIPTION	ENTRY	RESULT EXPECTED
OC06	Get an existing Order by ID	Get a order using a valid ID.	i:1,	Response 200 OK with order details created.
OC07	Get Order list	Get the order list complete	N/A	Response 200 OK with an array of existing orders.
OC08	Get an order with invalid ID	Attempt to get a order ith a non-numeric ID.	i:abc,	400 Bad Request indicating that the ID must be numeric.

OC09	Get non-existing order	Attempting to get a order that does not exist in the liist.	i:80,	Response 404 Not Found with a messege that the order does not exist.
OC10	Get an order with negative id	Attempting to get a order with a ID negative	i:-10,	Response 400 Not Found with a messege that the ID is invalid
OC11	Get an order with id=0	Attempting to get a order with a ID = 0	i:0,	Response 400 Not Found with a messege that the ID cannot be equal to 0.

BACKEND (C# APIs): Order Processing: (Updated Order- PUT)

ID	TITLE	DESCRIPTION	ENTRY	RESULT EXPECTED
OC12	pdate an existing order	Verify that an existing order can be updated with a valid ID.	id: 1, { "id": 1, roductName": "Product Six", "quantity": 10, "status": "In progress" }	Response 200 OK with confirmation of order details updated.
OC13	Update non-existent order	Verify that you cannot update an order that does not exist.	d: 1000, { "id": 1002, roductName": "Product Six", "quantity": 10, "status": "In progress" }	Response 404 Not Found with a message indicating “Order not found”.
OC14	Update an order with negative quantity	Verify that negative quantity are not allowed to be ssigned to the order.	id: 1, { "id": 1, roductName": "Product one", "quantity": -5, "status": "In progress" }	Response 400 Bad Request indicating “Quantity cannot be negative”.
OC15	Update an order with empty status fields	Verify that it is not allowed to update a order by omitting obligatory fields.	id: 1, { "id": 1, roductName": "Product Six", "quantity": 10, "status": "" }	Response 400 Bad Request with a message indicanting that that the ‘status’ field is require.

BACKEND (C# APIs): Order Processing: (Delete Order- DELETE)

ID	TITLE	DESCRIPTION	ENTRY	RESULT EXPECTED
OC16	Delete an existing order	Confirm that an existing order can be successfully removed from the system.	id:1,	Response 200 OK with a confirmation message indicating that the order has been deleted.
OC17	Delete a non-existing order	Ensure that the system responds correctly when an attempt is made to remove a order that	id: 1000,	Response 404 Not Found with a message indicating the order was not found.

		does not exist.		
OC18	Delete an order without specifying an ID	Test how the system reacts when the ID is omitted from the delete request.	id: ,	400 Bad Request with an error indicating that the ID is required.
OC19	Delete an order with invalid negative ID	Test the behavior when trying to delete an order using a negative number as the ID.	id: -2,	Response 400 Bad Request with a message stating the ID provided is invalid.
OC20	Delete an order with ID set to zero	Validate that orders with an ID equal to 0 cannot be removed.	id: 0,	400 Bad Request indicating the provided ID cannot be zero.

FRONTEND (ReactJS)

ID	TITLE	DESCRIPTION	STEPS	RESULT EXPECTED
U1	Successful login with correct credentials	Successful login with correct credentials	. Open the login page. 2. Enter a valid username and assword. 3. Click the 'Login' button."	User is logged in successfully and redirected to the Daskboard.
U2	Login attempt with incorrect credentials	Login attempt with incorrect credentials	1. Open the login page. 2. Enter a valid username and an incorrect password. 3. Click in 'Login'."	The user is unable to log in, and an error message is displayed.
U3	View product list	Ensure the product list is loaded correctly when visiting the products page.	Open the 'Products' page. 2. Verify the products are listed.	The product list is displayed with accurate product information.
U4	View order list	Ensure the user can view their list of placed orders.	Go to the 'Orders' page. 2. Check that a list of orders is displayed.	The list of orders is shown correctly with details
U5	Test how the UI adapts to different devices. (Responsive)	Test how the page layout adjusts to various screen sizes (mobile, tablet, desktop).	1. Open the app on a mobile, tablet, and desktop. 2. Check the page layout on each device.	The page layout adjusts appropriately for each device type.