Franklin Camacho
Chicks Group Inc.

TEST CASE DEVELOPMENT

This document outlines the test cases created to verify the functionality and reliability of the QA challenge. These test cases were crafted following industry-standard quality assurance practices to guarantee thorough coverage of the established requirements. Their main objective is to uncover potential issues, validate adherence to acceptance criteria, and provide a reliable framework for informed decision-making throughout the development process.

Given the need for clear and efficient documentation, the test cases in this report are presented in a tabular format. Below are the key advantages of using a table to organize test cases:

Advantages of Using a Table Format for Test Cases

- **Clarity and Organization:** Tables present information in a structured and concise manner, making it easier to review and navigate through multiple test cases.
- **Standardization:** Encourages consistency across test cases, facilitating better communication among team members and stakeholders.
- **Quick Reference:** Enables testers and developers to quickly locate specific test cases and understand their purpose, steps, and expected results without unnecessary detail.
- **Tool Compatibility:** A tabular format aligns with many test management tools, such as TestRail, JIRA, or even Excel, streamlining integration and execution.
- **Scalability:** Handles large volumes of test cases effectively, making it suitable for both small and complex testing projects.
- **Recommendation Insight:** Tables are ideal for collaborative environments where quick execution and iteration are required. Their compact format allows teams to maintain focus and ensure progress without losing critical details.

Franklin Camacho
Chicks Group Inc.

Instructions for Using the Table Format

To ensure consistency and clarity, the table format for test cases follows a structured approach. Each column is designed to capture specific information critical for executing and understanding the test cases. Below is a breakdown of the elements in the table and their purpose:

- **ID:** A unique identifier assigned to each test case. This helps track and reference specific cases during execution or review.
- **Test Case Title:** A brief and descriptive title that summarizes the purpose of the test case.
- **Description:** A detailed explanation of what is being tested, including its relevance to the overall functionality or requirement.
- **Steps to Execute:** Step-by-step instructions that guide the tester on how to perform the test accurately.
- **Expected Result:** The anticipated outcome if the system functions as intended. This serves as the benchmark for validating the test results.

By following these guidelines, the table becomes a reliable resource for documenting and executing test cases effectively, ensuring that all relevant details are captured and easy to follow.

Backend (C# APIS):

ID	Test Case Title	Description	Steps to Execute	Expected Result
BE-TC-Login001	Login with Valid Credentials	This test case verifies that the system allows users to log in successfully when valid credentials are provided.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include valid credentials in the request body, such as: json {"username": "testuser", "password": "password" } 3. Send the request and capture the API response.	- The API responds with: - A valid authentication token in the response body: json {"token": "sampletoken" } - A status code of 200 OK. - The response time is less than 200 ms.
BE-TC-Login002	Invalid username with valid password	Verify that the system returns an error when an invalid username is provided with a valid password.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include invalid username and valid password in the request body, such as: {"username": "wrongusername", "password": "password"} 3. Send the request and	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.

			capture the API response.	
BE-TC-Login003	Valid username with invalid password	Verify that the system returns an error when a valid username is provided with an invalid password.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include valid username and invalid password in the request body, such as: {"username": "testuser", "password": "wrongpassword"} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login004	Both username and password invalid	Verify that the system returns an error when both username and password are invalid.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include invalid username and invalid password in the request body, such as: {"username": "wrongusername", "password": "wrongpassword"} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login005	Both fields empty	Verify that the system returns an error when both fields are empty.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include empty values for both username and password in the request body. {"username": "", "password": ""} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login006	Empty username, valid password	Verify that the system returns an error when the username is empty but the password is valid.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include an empty username and a valid password in the request body, such as: {"username": "", "password": "password"} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login007	Empty username, invalid password	Verify that the system returns an error when the username is empty and the password is invalid.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include an empty username and an invalid password in the request body, such as: {"username": "", "password": "wrongpassword"} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login008	Valid username,	Verify that the	1. Prepare a POST request to	- The API responds with:

	empty password	system returns an error when the username is valid but the password is empty.	the /api/User/login endpoint. 2. Include a valid username and an empty password in the request body, such as: {"username": "testuser", "password": ""} 3. Send the request and capture the API response.	- Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login009	Invalid username, empty password	Verify that the system returns an error when the username is invalid and the password is empty.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include an invalid username and an empty password in the request body, such as: {"username": "wrongusername", "password": ""} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login010	Password with special characters and normal username	Verify that the system allows login with special characters in the password.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include valid username and a password with special characters in the request body, such as: {"username": "testuser", "password": "password!@#"} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms. - The response time is less than 200 ms.
BE-TC-Login011	Username with special characters and normal password	Verify that the system allows login with special characters in the username.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include a username with special characters and valid password in the request body, such as: {"username": "testuser!@#", "password": "password"} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login012	Username with uppercase letters and normal password	Verify that the system allows login with uppercase letters in the username.	1. Prepare a POST request to the /api/User/login endpoint. 2. Include a username with uppercase letters and valid password in the request body, such as: {"username": "TestUser", "password": "password"} 3. Send the request and capture the API response.	- The API responds with: - Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Login013	Password with	Verify that the	1. Prepare a POST request to	- The API responds with:

	uppercase letters and normal username	system allows login with uppercase letters in the password.	the /api/User/login endpoint. 2. Include a valid username and a password with uppercase letters in the request body, such as: {"username": "testuser", "password": "Password"} 3. Send the request and capture the API response.	- Status code 401 Unauthorized. - No token in the response body. - The response time is less than 200 ms.
BE-TC-Prod001	Create Product	Verify that the API allows creating a product with valid data.	1. Prepare a POST request to the /api/Product endpoint. 2. Include the required data in the request body: {"id": 0, "name": "Product1", "price": 1}. 3. Send the request.	- The API responds with: - The same request body in the response: {"id": 0, "name": "Product1", "price": 1}. - Status code 201 Created. - The response time is less than 200 ms.
BE-TC-Prod002	Consult Product List	Verify that the API allows consulting the list of products correctly.	1. Prepare a GET request to the /api/Product endpoint. 2. Send the request and capture the API response.	- The API responds with: - The complete list of products in the response body. - Status code 200 OK. - The response time is less than 200 ms.
BE-TC-Prod003	Create Product with Existing ID	Validate that the API handles product creation when the ID already exists.	1. Prepare a POST request to the /api/Product endpoint. 2. Use an existing product ID in the request body: {"id": 1, "name": "Product1", "price": 1}. 3. Send the request.	- The API responds with: - An error message in the response body, such as {"error": "Product ID already exists"}. - Status code 409 Conflict. - The response time is less than 200 ms.
BE-TC-Prod004	Browse Product by ID	Verify that the API allows querying a specific product using its ID.	1. Prepare a GET request to the /api/Product/{id} endpoint. 2. Replace {id} with a valid product ID, e.g., /api/Product/1. 3. Send the request.	- The API responds with: - The product details in the response body: {"id": 1, "name": "Product1", "price": 1}. - Status code 200 OK. - The response time is less than 200 ms.
BE-TC-Prod005	Edit Product by ID	Verify that the API allows editing a specific product using its ID.	1. Prepare a PUT request to the /api/Product/{id} endpoint. 2. Replace {id} with a valid product ID. 3. Include updated data in the request body: {"name": "UpdatedProduct", "price": 2}.	- The API responds with: - A success message in the response body, such as {"message": "Product updated successfully"}. - Status code 200 OK. - The response time is

			4. Send the request.	less than 200 ms.
BE-TCProd006	Delete Product by ID	Verify that the API allows deleting a specific product using its ID.	1. Prepare a DELETE request to the /api/Product/{id} endpoint. 2. Replace {id} with a valid product ID, e.g., /api/Product/1. 3. Send the request.	- The API responds with: - A success message in the response body, such as {"message": "Product deleted successfully"}. - Status code 200 OK. - The response time is less than 200 ms.
BE-TC-Prod007	Product Update with Nonexistent ID	Test system behavior when attempting to update a product that does not exist.	1. Prepare a PUT request to the /api/Product/{id} endpoint. 2. Replace {id} with a nonexistent product ID, e.g., /api/Product/999. 3. Include updated data: {"name": "NonExistent", "price": 3}. 4. Send the request.	- The API responds with: - An error message in the response body, such as {"error": "Product not found"}. - Status code 404 Not Found. - The response time is less than 200 ms.
BE-TC-Prod008	Elimination of Product with Nonexistent ID	Verify the behavior of the system when an attempt is made to delete a product that does not exist.	1. Prepare a DELETE request to the /api/Product/{id} endpoint. 2. Replace {id} with a nonexistent product ID, e.g., /api/Product/999. 3. Send the request and capture the response.	- The API responds with: - An error message in the response body, such as {"error": "Product not found"}. - Status code 404 Not Found. - The response time is less than 200 ms.
BE-TC-Prod009	Create Product with Missing Data	Verify that the API returns an error when creating a product with missing required data.	1. Prepare a POST request to the /api/Product endpoint. 2. Include incomplete data in the request body, e.g., {"id": 0, "name": "Product1"} (missing price). 3. Send the request.	- The API responds with: - An error message in the response body, such as {"error": "Price is required"}. - Status code 400 Bad Request. - The response time is less than 200 ms.
BE-TC-Prod010	Create Product with Invalid Data	Verify that the API returns an error when creating a product with invalid data (e.g., negative price).	1. Prepare a POST request to the /api/Product endpoint. 2. Include invalid data in the request body, e.g., {"id": 0, "name": "Product1", "price": -1}. 3. Send the request.	- The API responds with: - An error message in the response body, such as {"error": "Invalid price"}. - Status code 400 Bad Request. - The response time is less than 200 ms.
BE-TC-Prod011	Product Price Update	Verify that the API correctly updates the price of a	1. Prepare a PUT request to the /api/Product/{id} endpoint. 2. Replace {id} with a valid product ID.	- The API responds with: - A success message in the response body, such as {"message": "Product

		product when using the PUT method.	3. Include updated price in the request body: {"price": 5}. 4. Send the request.	updated successfully"}. - Status code 200 OK. - The response time is less than 200 ms.
BE-TC-Prod012	Product Deletion with Invalid ID	Test system behavior when attempting to delete a product using an invalid ID format.	1. Prepare a DELETE request to the /api/Product/{id} endpoint. 2. Replace {id} with an invalid ID, e.g., /api/Product/abc. 3. Send the request.	- The API responds with: - An error message in the response body, such as {"error": "Invalid product ID"}. - Status code 400 Bad Request. - The response time is less than 200 ms.
BE-TC-Prod013	Empty Product List	Verify that the API returns an empty list when there are no products in the database.	1. Prepare a GET request to the /api/Product endpoint. 2. Ensure the product database is empty. 3. Send the request.	- The API responds with: - An empty list in the response body: []. - Status code 200 OK. - The response time is less than 200 ms.
BE-TC-Orders001	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 orders - The status code is 200 OK - The response time is less than 200 ms.
BE-TC-Orders002	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 /api/Order with a valid order ID as a parameter.	- The API responds with the details of the specific order - The status code is 200 OK - The response time is less than 200 ms.
BE-TC-Orders003	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 - The status code is 404 Not Found - The response time is less than 200 ms.
BE-TC-Orders004	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 - The status code is 400 or 422 - The response time is less than 200 ms.

BE-TC-Orders005	Create order with valid data	Verify that the API allows creating an order with all valid data.	1. Send a POST request to /api/Order with the following request body: { "id": 1, "productName": "Sample Product", "quantity": 5, "status": "Pending" }	- The API responds with a success message - The status code is 201 Created - The order is successfully saved - The response time is less than 200 ms.
BE-TC-Orders006	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 the following request body: { "id": 2, "productName": "", "quantity": 5, "status": "Pending" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders007	Create order with quantity at 0	Verify that the API rejects the creation of an order when quantity is equal to 0, since it does not make sense to create an order without products.	1. Send a POST request to /api/Order with the following request body: { "id": 3, "productName": "Sample Product", "quantity": 0, "status": "Pending" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders008	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 the following request body: { "id": 4, "productName": "Sample Product", "quantity": -1, "status": "Pending" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders009	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 the following request body: { "id": 5, "productName": "Sample Product", "quantity": 5, "status": "" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders010	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 the following request body: { "id": 6, "productName": "Sample Product", "quantity": 5, "status": "Pending" }	- The API responds with an error message - The status code is 409 Conflict - The response time is less than 200 ms.
BE-TC-Orders011	Create order with all empty fields	Validate that the API rejects the creation of an order when all	1. Send a POST request to /api/Order with the following request body: { "id": 7, "productName": "",	- The API responds with an error message - The status code is 400 or 422

		fields are empty.	"quantity": 0, "status": "" }	- The response time is less than 200 ms.
BE-TC-Orders012	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 the following request body: { "id": 8, "productName": "Sample Product", "quantity": "Five", "status": "Pending" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders013	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 the following request body: { "id": 1, "productName": "Updated Product", "quantity": 10, "status": "Shipped" }	- The API responds with a success message - The status code is 200 or 204 - The order is successfully updated, without changing the ID - The response time is less than 200 ms.
BE-TC-Orders014	Update the ID of an order	Verify that the API does not allow modifying the ID of an existing order.	1. Send a PUT request to /api/Order/{id} with the following request body: { "id": 2, "productName": "New Product", "quantity": 15, "status": "Pending" }	- The API responds with an error message - The status code is 400 or 409 - The response time is less than 200 ms.
BE-TC-Orders015	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 the following request body: { "id": 1, "productName": "", "quantity": 10, "status": "Shipped" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders016	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 the following request body: { "id": 1, "productName": "Updated Product", "quantity": 0, "status": "Shipped" }	The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders017	Update order with negative quantity	Validate that the API rejects the update of an order when quantity is negative.	1. Send a PUT request to /api/Order/{id} with the following request body: { "id": 1, "productName": "Updated Product", "quantity": -5, "status": "Shipped" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders018	Update the order	Verify that the	1. Send a PUT request to	- The API responds with

	with an empty status	API properly handles the update of an order when status is empty.	/api/Order/{id} with the following request body: { "id": 1, "productName": "Updated Product", "quantity": 10, "status": "" }	an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders019	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 and the following request body: { "id": 999, "productName": "Non-existent Product", "quantity": 1, "status": "Pending" }	- The API responds with an error message - The status code is 404 Not Found - The response time is less than 200 ms.
BE-TC-Orders020	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 the following request body: { "id": 1, "productName": "", "quantity": 0, "status": "" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders021	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 the following request body: { "id": 1, "productName": "Updated Product", "quantity": "Ten", "status": "Shipped" }	- The API responds with an error message - The status code is 400 or 422 - The response time is less than 200 ms.
BE-TC-Orders022	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 the following request body: { "id": 1, "productName": "Sample Product", "quantity": 5, "status": "Pending" }	- The API responds with a success message - The status code is 200 or 204 - The order is successfully deleted - The response time is less than 200 ms.

Franklin Camacho
Chicks Group Inc.

Frontend (ReactJS):

ID	Test Case Title	Description	Steps to Execute	Expected Result
FE-TC-Login001	Login with Valid Credentials	This test case verifies that the system allows users to log in successfully when valid credentials are provided.	1. Open the login page. 2. Enter valid credentials (username: "testuser", password: "password"). 3. Click the "Login" button.	- The system shows the message: "Logged in with token: sampletoken". - The user is redirected to the dashboard. - The page loads successfully within 2 seconds.
FE-TC-Login002	Invalid username with valid password	Verify that the system returns an error when an invalid username is provided with a valid password.	1. Open the login page. 2. Enter invalid username (username: "wrongusername") and valid password (password: "password"). 3. Click the "Login" button.	- The system shows an error message: "Invalid credentials". - No token is displayed on the screen. - The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login003	Valid username with invalid password	Verify that the system returns an error when a valid username is provided with an invalid password.	1. Open the login page. 2. Enter valid username (username: "testuser") and invalid password (password: "wrongpassword"). 3. Click the "Login" button.	- The system shows an error message: "Invalid credentials". - No token is displayed on the screen. - The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login004	Both username and password invalid	Verify that the system returns an error when both username and password are invalid.	1. Open the login page. 2. Enter invalid username (username: "wrongusername") and invalid password (password: "wrongpassword"). 3. Click the "Login" button.	- The system shows an error message: "Invalid credentials". - No token is displayed on the screen. - The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login005	Both fields empty	Verify that the system returns an error when both fields are empty.	1. Open the login page. 2. Leave the username and password fields empty. 3. Click the "Login" button.	- The system shows an error message: "Please fill in both fields". - No token is displayed on the screen. - The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login006	Empty username, valid password	Verify that the system returns an error when the username is empty but the	1. Open the login page. 2. Leave the username field empty and enter valid password (password: "password").	- The system shows an error message: "Please enter a username". - No token is displayed on the screen.

		password is valid.	3. Click the "Login" button.	- The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login007	Empty username, invalid password	Verify that the system returns an error when the username is empty and the password is invalid.	1. Open the login page. 2. Leave the username field empty and enter invalid password (password: "wrongpassword"). 3. Click the "Login" button.	- The system shows an error message: "Please enter a username". - No token is displayed on the screen. - The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login008	Valid username, empty password	Verify that the system returns an error when the username is valid but the password is empty.	1. Open the login page. 2. Enter valid username (username: "testuser") and leave the password field empty. 3. Click the "Login" button.	- The system shows an error message: "Please enter a password". - No token is displayed on the screen. - The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login009	Invalid username, empty password	Verify that the system returns an error when the username is invalid and the password is empty.	1. Open the login page. 2. Enter invalid username (username: "wrongusername") and leave the password field empty. 3. Click the "Login" button.	- The system shows an error message: "Invalid credentials". - No token is displayed on the screen. - The page reloads after 1-2 seconds, remaining on the login page.
FE-TC-Login010	Password with special characters and normal username	Verify that the system allows login with special characters in the password.	1. Open the login page. 2. Enter valid username (username: "testuser") and a password with special characters (password: "password!@#"). 3. Click the "Login" button.	- The system shows the message: "Logged in with token: sampletoken". - The user is redirected to the dashboard. - The page loads successfully within 2 seconds.
FE-TC-Login011	Username with special characters and normal password	Verify that the system allows login with special characters in the username.	1. Open the login page. 2. Enter a username with special characters (username: "testuser!@#") and valid password (password: "password"). 3. Click the "Login" button.	- The system shows the message: "Logged in with token: sampletoken". - The user is redirected to the dashboard. - The page loads successfully within 2 seconds.
FE-TC-Login012	Username with uppercase letters and normal	Verify that the system allows login with uppercase	1. Open the login page. 2. Enter a username with uppercase letters (username: "TestUser") and valid password	- The system shows the message: "Logged in with token: sampletoken". - The user is redirected to

	password	letters in the username.	(password: "password"). 3. Click the "Login" button.	the dashboard. - The page loads successfully within 2 seconds.
FE-TC-Login013	Password with uppercase letters and normal username	Verify that the system allows login with uppercase letters in the password.	1. Open the login page. 2. Enter valid username (username: "testuser") and a password with uppercase letters (password: "Password"). 3. Click the "Login" button.	- The system shows the message: "Logged in with token: sampletoken". - The user is redirected to the dashboard. - The page loads successfully within 2 seconds.
FE-TC-DB001	Validate correct access to the Dashboard page after a successful login	Validate that the user can access the Dashboard after logging in successfully.	1. Click on the "Dashboard" option in the menu list.	The link redirects to the Dashboard page.
FE-TC-DB002	Validate correct redirection from the Products page to the Dashboard page, after a successful login	Verify that clicking "Dashboard" redirects from the Products page after login.	1. Access the Products page. 2. Click on the "Dashboard" option in the menu list.	The link redirects to the Dashboard page from the Products page.
FE-TC-DB003	Validate correct redirection from the Orders page to the Dashboard page, after a successful login	Ensure that clicking "Dashboard" redirects from the Orders page after login.	1. Access the Orders page. 2. Click on the "Dashboard" option in the menu list.	The link redirects to the Dashboard page from the Orders page.
FE-TC-Prod001	Validate correct access to the Products page after a successful login	Ensure that the user can access the Products page after logging in successfully.	1. Click on the "Products" option in the menu list.	The link redirects to the Products page.
FE-TC-Prod002	Validate correct redirection from the Dashboard page to the Products page, after a successful login	Verify that clicking "Products" redirects from the Dashboard page after login.	1. Access the Dashboard page. 2. Click on the "Products" option in the menu list.	The link redirects to the Products page from the Dashboard page.
FE-TC-Prod003	Validate correct redirection from the Orders page	Ensure that clicking "Products"	1. Access the Orders page. 2. Click on the "Products" option in the menu list.	The link redirects to the Products page from the Orders page.

	to the Products page, after a successful login	redirects from the Orders page after login.		
FE-TC-Prod004	Validate that products are listed and displayed correctly on the Products page	Check if products are listed correctly on the Products page.	1. Click on the "Products" option in the menu list.	The link redirects to the Products page, and the created products are listed correctly.
FE-TC-Prod005	Validate that products are listed correctly after being updated from the API	Ensure products are listed correctly after updates from the API.	1. Click on the "Products" option in the menu list.	The link redirects to the Products page, and the updated products are listed correctly.
FE-TC-Prod006	Validate that deleted products are not shown on the Products page	Ensure deleted products are removed from the Products page.	1. Click on the "Products" option in the menu list.	The link redirects to the Products page, and deleted products are not shown in the list.
FE-TC-Prod007	Validate correct display of products after creation, update, and deletion	Check if the created, updated, and deleted products are listed correctly.	1. Click on the "Products" option in the menu list.	The link redirects to the Products page, and the created, updated, and deleted products are listed correctly.
FE-TC-Orders001	Validate correct access to the Orders page after a successful login	Ensure that the user can access the Orders page after logging in successfully.	1. Click on the "Orders" option in the menu list.	The link redirects to the Orders page.
FE-TC-Orders002	Validate correct redirection from the Dashboard page to the Orders page, after a successful login	Ensure redirection from the Dashboard page to Orders page after login.	1. Access the Dashboard page. 2. Click on the "Orders" option in the menu list.	The link redirects to the Orders page from the Dashboard page.
FE-TC-Orders003	Validate correct redirection from the Products page to the Orders page, after a successful login	Ensure redirection from the Products page to Orders page after login.	1. Access the Products page. 2. Click on the "Orders" option in the menu list.	The link redirects to the Orders page from the Products page.
FE-TC-Orders004	Validate that orders are	Verify that orders are	1. Click on the "Orders" option in the menu list.	The link redirects to the Orders page, and created

	created and displayed correctly on the Orders page	correctly displayed on the Orders page.		orders are displayed correctly.
FE-TC-Orders005	Validate that updated orders are displayed correctly on the Orders page	Check if updated orders are shown correctly on the Orders page.	1. Click on the "Orders" option in the menu list.	The link redirects to the Orders page, and updated orders are displayed correctly.
FE-TC-Orders006	Validate correct display of orders after creation, update, and deletion	Ensure created, updated, and deleted orders are listed correctly.	1. Click on the "Orders" option in the menu list.	The link redirects to the Orders page, and created and updated orders are displayed correctly.
FE-TC-Acc001	Validate the correct functionality of the website across different browsers	Ensure the website works correctly across Google Chrome, Microsoft Edge, and Opera.	1. Open the website in Google Chrome. 2. Open the website in Microsoft Edge. 3. Open the website in Opera.	The website loads correctly in all browsers.
FE-TC-Perf001	Verify performance of the Login page using Lighthouse tool in DevTools	Check the performance of the Login page on both mobile and desktop.	1. Open the Login page. 2. Open DevTools and navigate to Lighthouse. 3. Select "Desktop" or "Mobile" mode and analyze the page load.	Lighthouse successfully analyzes the page and provides performance results.
FE-TC-Perf002	Verify performance of the Dashboard page using Lighthouse tool in DevTools	Check the performance of the Dashboard page on both mobile and desktop.	1. Open the Dashboard page. 2. Open DevTools and navigate to Lighthouse. 3. Select "Desktop" or "Mobile" mode and analyze the page load.	Lighthouse successfully analyzes the page and provides performance results.
FE-TC-Perf003	Verify performance of the Products page using Lighthouse tool in DevTools	Check the performance of the Products page on both mobile and desktop.	1. Open the Products page. 2. Open DevTools and navigate to Lighthouse. 3. Select "Desktop" or "Mobile" mode and analyze the page load.	Lighthouse successfully analyzes the page and provides performance results.
FE-TC-Perf004	Verify performance of the Orders page using Lighthouse tool in DevTools.	Check the performance of the Orders page on both mobile and desktop.	1. Open the Orders page. 2. Open DevTools and navigate to Lighthouse. 3. Select "Desktop" or "Mobile" mode and analyze the page load.	Lighthouse successfully analyzes the page and provides performance results.