**Objective of the Test Plan**

The primary objective of the test plan is to verify that the functionalities of the SUT API (React Application) for managing Login, Order, and Product operate optimally and meet user requirements. This includes login, creation, update, deletion, and verification of Orders and Products, providing the user with an effective tool for making purchases efficiently.  
This will ensure the quality and reliability of the main system functionalities, improving the end-user experience. These tests will help identify and mitigate defects that could impact user experience and business processes.

**Specific Behaviors to Validate for Login:**

1. Correct Authentication: Verify that valid credentials allow the user to access the system seamlessly.
2. Handling Incorrect Credentials: Ensure the system provides clear and precise error messages when invalid credentials are entered.

**Specific Behaviors to Validate for Product:**

1. Product Creation: Products must be created correctly and stored in the system.
2. Product Retrieval: Created products must be displayed in a list accessible to the user.
3. Product Update: Products must allow easy and efficient modification of their information.
4. Product Deletion: Products must be deleted correctly and should no longer appear in the system after deletion.

**Specific Behaviors to Validate for Order:**

Although the technical test does not reference a CRUD, it is implemented in the API documentation. Therefore, it is essential to develop a set of tests as follows:

1. Order Creation: Orders must be created correctly and stored in the system.
2. Order Retrieval: Created orders must be displayed in a list accessible to the user.
3. Order Update: Orders must allow easy and efficient modification of their information.
4. Order Deletion: Orders must be deleted correctly and should no longer appear in the system after deletion.

These behaviors are fundamental to ensuring a consistent user experience aligned with functional expectations when using the API.

**FUNCTIONAL TEST PLAN (API TESTING LEVEL)**

**SUMMARY:**

**Level Test Plan / Feature:**

1. **User Authentication:**
   * Verify user login with valid credentials.
   * Verify error message for invalid credentials.
   * Test token generation and expiration.
2. **Product Management:**
   * Create, read, update, and delete (CRUD) operations for the product.
   * Verify error handling for invalid product data.
   * Test search and filter functionalities.
3. **Order Processing:**
   * Create an order and verify order details.
   * Test order cancellation and status updates.
   * Verify error handling for invalid order operations.
4. **ReactJS (User Authentication & Dashboard):**
   * Verify login and logout functionalities.
   * Test user dashboard data display and refresh.
   * Check UI responsiveness and accessibility.
5. **ReactJS (Product Listing & Order Processing):**
   * Verify product listing display and filtering.
   * Test order creation, update, and cancellation workflows.
   * Check UI responsiveness and accessibility.

**TARGET:**

Ensure the correct validation of the basic functionalities related to the management of Login, Product, and Order, utilizing test automation through APIs using Postman.

**ASSUMPTIONS:**

* It is assumed that the test environment has constant and reliable internet access to make requests to the SUT API.
* It is assumed that the user already has active and valid authentication (such as access tokens) required to interact with the API.
* It is assumed that the names, parameters, and values used in the tests (such as Product names or IDs) are correct and meet the API requirements.
* It is assumed that the SUT server is operational, and the API responds according to official documentation, without crashes or unexpected behavior.
* It is assumed that the API always returns appropriate HTTP status codes (e.g., 200 for success, 404 for not found, etc.).
* It is assumed that no changes are expected in the structure of endpoints, parameters, or response formats of the API during the test period.
* It is assumed that the API documentation is accurate and reflects the current behavior of the API, including endpoints and expected responses.

**ASSUMPTIONS: Login**

* It is assumed that the username and password fields only accept alphabetic characters (a-z, A-Z) and do not allow numeric or special characters.
* It is assumed that the username and password fields only accept 8 characters.

**SCOPE:**

The QA team will carry out a comprehensive validation of the correct functioning of the functionalities related to Login, Product, and Order in the SUT. This process will ensure compliance with functional requirements and established user experience standards. The validation will cover the following key functionalities:

**Login:**

1. **Logging In:**
   * Validate successful login.

**Product:**

1. **Product Creation:**
   * Validate that products are created correctly and adequately stored in the system.
2. **Product Retrieval:**
   * Validate that created products are displayed in a list accessible to the user.
3. **Product Update:**
   * Validate that products allow modification of their information, ensuring that changes are correctly reflected in the system and accessible to the user.
4. **Product Deletion:**
   * Validate that products can be deleted correctly from the system, ensuring they are removed from the product list and cannot be accessed or interacted with after deletion.

**Key Validations for Orders:**

1. **Order Creation:**
   * Validate that orders are created correctly and adequately stored in the system.
2. **Order Retrieval:**
   * Validate that created orders are displayed in a list accessible to the user.
3. **Order Update:**
   * Validate that orders allow modification of their information, ensuring that changes are

**Technical Validations via API:**

QA will perform exhaustive tests using API methods to ensure compliance with the defined acceptance criteria. The tests will include:

1. **Login:**
   * Verify that the user logs in successfully.
2. **POST (Creation):**
   * Verify that Products are created correctly with a provided id, name, and price.
   * Verify that Orders are created correctly with a provided id, product Name, quantity, and status.
   * Validate that API responses return a status code of 200 and a body containing the data of the created Product and Order.
3. **PUT (Update):**
   * Ensure that Products and Orders can be updated correctly (parameters such as id, name, price).
   * Validate that API responses return a status code of 200 and a body reflecting the changes made.
4. **DELETE (Deletion):**
   * Ensure that Products and Orders are deleted correctly upon performing a DELETE request.
   * Verify that API responses return a status code of 200 and an empty body.
5. **GET (Query):**
   * Validate that created Products and updated Orders can be queried, and the response returns a status code of 200 with accurate data.
   * Validate that the created or updated information is correct.
6. **RISKS:**

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Unable to save the SUT API token | Send the token via email. Generate the token using cookies. |
|  |
|  |
| Lack of knowledge about the API documentation | Ensure the UI is linked to all the requests available in the API. |  |

**RESOURCES/ITEMS:**

Key resources and elements for API testing:

1. **Postman**
   * Main tool used for API test execution.
2. **SUT API Documentation**
   * Fundamental guide to correctly understanding and using the SUT API.
3. **Test Plan**
   * Planning document detailing the necessary tests and their strategic approach.
4. **API Testing**
   * Test Cases (TC) and Steps are generated to ensure comprehensive test coverage. These tests are also considered for future automation.

**Structure for Execution in Postman:**

* **Collections (Test Execution):**
  + Each folder represents a set of tests aligned with a specific acceptance criterion.
  + **1 Folder = 1 Acceptance Criterion (AC) = 1 Test Set** (For 4 acceptance criteria, there will be 4 folders).
* **Definition of Test Cases:**
  + **1 Folder = 1 Test Case**
  + Each test case contains the required requests and action steps.
* **Mapping Between Elements:**
  + **1 Request = 1 Action Step**
  + **1 Test = 1 Expected Outcome**

**ENVIRONMENT CONFIGURATION:**

1. Tests will be performed in the SUT environment.
2. Tests will be executed on the SUT APIs.

**Test Coverage Plan**

**Backend (C# APIs)**

1. **User Authentication**
   * TC1: Validate successful login by entering Username and Password correctly.
   * TC2: Validate that cannot log in by leaving the Username field empty and the Password correct.
   * TC3: Validate Unable to log in using correct Username and empty Password.
   * TC4: Validate Unable to log in leaving all fields empty.
   * TC5: Validate Unable to log in using correct Username and incorrect Password.
   * TC6: Validate Unable to log in using incorrect username and correct password.
   * TC7: Validate Unable to log in when using Incorrect data.
   * TC8: Validate Unable to log in leaving Username empty and Password incorrect.
   * TC9: Validate Unable to log in using Incorrect Username and Empty Password.
   * TC10: Validate successful session close.
2. **Product Management**

**Product Creation**

* + TC11: Validate that you can successfully create a product.
  + TC12: Validate not being able to create a Product using invalid id.
  + TC13: Validate not being able to create a Product using an invalid name.
  + TC14: Validate not being able to create a Product using an invalid price.
  + TC15: Validate that a Product cannot be created using a negative price.
  + TC16: Validate not being able to create a Product using all invalid data.
  + TC17: Validate not being able to create a Product leaving the id field Null.
  + TC18: Validate not being able to create a Product leaving the name field Null.
  + TC19: Validate not being able to create a Product leaving the price field Null.
  + TC20: Validate not being able to create a Product leaving all fields Null.
  + TC21: Validate not being able to create a Product using a previously used id.

**Retrieve Product**

* + TC22: Validate that can successfully obtain all the products from the list.
  + TC23: Validate the ability to obtain a specific Product by ID.
  + TC24: Validate unable to get Product using incorrect id.
  + TC25: Validate not being able to obtain Product leaving id Null.

**Update Product**

* + TC26: Validate that can successfully update your product.
  + TC27: Validate not being able to update Product using a non-existent product id.
  + TC28: Validate cannot update Product when ID is Null.
  + TC29: Validate unable to update Product using invalid id.
  + TC30: Validate unable to update Product using invalid name.
  + TC31: Validate not being able to update Product when price has negative values.
  + TC32: Validate not being able to update Product by entering all incorrect values.
  + TC33: Validate not being able to update Product when leaving id null.
  + TC34: Validate not being able to update Product when name is null.
  + TC35: Validate not being able to update Product when price is null.
  + TC36: Validate not being able to update Product leaving all Null values.
  + TC37: Validate not being able to update Product because it has 2 duplicate IDs.

**Delete Product**

* + TC38: Validate the ability to delete a product.
  + TC39: Validate that a product cannot be deleted when the ID is invalid.
  + TC40: Validate that a product cannot be deleted when the ID is null.
  + TC41: Validate that a product cannot be deleted when using a non-existent ID.

**Create Order**

* + TC42: Validate successful order creation.
  + TC43: Validate unable to create order using an invalid id.
  + TC44: Validate unable to create order using invalid product Name.
  + TC45: Validate not being able to create order using an invalid quantity.
  + TC46: Validate unable to create order using an invalid status.
  + TC47: Validate unable to create order using all invalid values.
  + TC48: Validate not being able to create order when the id is Null.
  + TC49: Validate the ability to create order by leaving the Product Name field Null.
  + TC50: Validate not being able to create order leaving the quantity field Null.
  + TC51: Validate the ability to create order by leaving the status Null.
  + TC52: Validate not being able to create order, leaving all fields null.
  + TC53: Validate not being able to create order with equal IDs.

**Retrieve Order**

* + TC54: Validate that can successfully obtain all orders.
  + TC55: Validate that you can successfully obtain the requested order.
  + TC56: Validate not being able to obtain order by entering a non-existent ID.
  + TC57: Validate unable to obtain order by leaving the id Null.

**Update Order**

* + TC58: Validate update of an order successfully.
  + TC59: Validate unable to update order using invalid id.
  + TC60: Validate unable to update order using Null id.
  + TC61: Validate not being able to update order using invalid id.
  + TC62: Validate unable to update order using invalid product name.
  + TC63: Validate unable to update order using invalid quantity.
  + TC64: Validate unable to update order using invalid status.
  + TC65: Validate unable to update order using all invalid values.
  + TC66: Validate cannot update order because the id field is null.
  + TC67: Validate unable to update order using null product name field.
  + TC68: Validate cannot update order because the quantity field is null.
  + TC69: Validate cannot update order with null status.
  + TC70: Validate cannot update order by leaving all fields null.
  + TC71: Validate not being able to update an ID when it has already been used previously.

**Delete Order**

* + TC72: Validate successful delete of order.
  + TC73: Validate cannot delete order with non-existent ID.
  + TC74: Validate unable to delete order with invalid id.
  + TC75: Validate cannot delete order with Null id.

**Frontend (ReactJS):**

* + TC1: Validate successful login by entering Username and Password correctly.
  + TC2: Validate that cannot log in by leaving the Username field empty and the Password correct.
  + TC3: Validate Unable to log in using correct Username and empty Password.
  + TC4: Validate Unable to log in leaving all fields empty.
  + TC5: Validate Unable to log in using correct Username and incorrect Password.
  + TC6: Validate Unable to log in using incorrect username and correct password.
  + TC7: Validate Unable to log in leaving Username empty and Password incorrect.
  + TC9: Validate Unable to log in using Incorrect Username and Empty Password.
  + TC10: Validate successful session close.

**Test Report:**

During the evaluation process of the System Under Test (SUT) provided, it was identified that the system development is not fully completed. Therefore, tests will only be performed on the functionalities that are finished, while the incomplete sections cannot be tested due to their current state of development.

An initial review was conducted to determine the scope of the tests, confirming that certain functionalities are operational and ready for validation, while others still require development and/or integration. For this reason, the applied tests will be limited to the functional scope of the completed sections of the SUT.

The following approach was implemented:

1. **Validation of Completed Functionalities:**
   * Exhaustive testing was carried out in the areas of the system that meet the necessary criteria for functional testing.
   * Compliance with requirements in the completed functionalities was verified, following the standards established in the test plan.
2. **Limitations in Pending Functionalities:**
   * Incomplete functionalities were not tested due to the lack of their final implementation.

**Incomplete functionalities:**

**ReactJS (Product Listing & Order Processing):**

* + Verify product listing display and filtering.
  + Test order creation, update, and cancellation workflows.
  + Check UI responsiveness and accessibility.