PARTE 1: TEST PLAN DESIGN

1. Scope:

Backend:

User Authentication:

- ✓ **Login Validation:**
  - Test successful logins with valid credentials to ensure correct generation of access tokens.
  - Test login with invalid credentials to verify that error messages provide clear and specific feedback, such as "User not found", "Invalid credentials", and "Expired token".

Product Management:

- ✓ **CRUD Operations (Create, Get, Update, Delete) for Products:**
  - **Product Creation:** Ensure products are correctly added to the database and stored with accurate details through the CREATE operation.
  - **Product Retrieval:** Validate that products are correctly retrieved by ID and details are returned through the GET operation.
  - **Product Update:** Confirm that changes made to products are reflected properly in the database via the UPDATE operation.
  - **Product Deletion:** Ensure products are deleted correctly from the system when removed via the DELETE operation.

Order Processing:

- **CRUD Operations (Create, Get, Update, Delete) for Orders:**
  - **Order Creation:** Ensure orders are correctly created with all necessary details and stored in the database via the CREATE operation.
  - **Order Retrieval:** Validate that orders are correctly retrieved by their ID and displayed with the correct details via the GET operation.

- **Order Status Update:** Confirm that order statuses or details are updated according to the requested changes via the UPDATE operation.
- **Order Deletion:** Ensure orders are deleted correctly from the system through the DELETE operation.

Frontend:

User Authentication:

- ✓ **Login and Dashboard:** The scope of testing focuses on validating the functionality of the login module on the frontend, covering the following aspects:
  - **Successful Login:**
    - Verify that the system allows access with correct credentials and displays the corresponding message: "Logged in with token: sampletoken".
  - **Failed Login:**
    - Evaluate the system’s response to invalid credentials, including:
      - Incorrect username.
      - Incorrect password.
      - Invalid combinations of username and password.
    - Validate the error message: "Login failed".
  - **Empty Field Validations:**
    - Test the system’s behavior when:
      - The username field is empty.
      - The password field is empty.
      - Both fields are empty.
    - In all cases, the expected error message should be: "Login failed".
  - **Navigation:**
    - Validate the navigation flow between fields and buttons and pages, such as the dashboard, where the authenticated user's data (basic information and personalized details) are displayed.

Product Listing:

- Display all available products in an intuitive and user-friendly format.
- Incorporate a search feature that allows filtering products by ID, name, or price, optimizing the navigation and locating specific items.

Order Processing:

- ✓ **Order Processing:**
  - Allow the creation of new orders directly from the user interface, ensuring an efficient and user-friendly flow.
  - Enable dynamic updates to the statuses of existing orders (in progress, completed, canceled) to reflect real-time progress.
  - Display clear success confirmations and effectively handle errors with detailed messages from the backend to ensure a smooth experience.

2. Objectives:

- ✓ **Functionality Validation:**
  Ensure that all key functionalities of both the backend and frontend operate correctly. This includes the efficient management of data, proper input validation, and the appropriate response to invalid inputs to prevent unexpected behaviors.
- ✓ **Security:**
  Evaluate the effectiveness of the authentication and authorization process, ensuring the protection of sensitive data, secure session management, and the implementation of effective security measures to safeguard the system.
- ✓ **Error Handling:**
  Ensure that the system handles errors properly by providing clear and detailed feedback to the user in cases of failures, system errors, or invalid inputs, maintaining a smooth and understandable user experience.
- ✓ **Accessibility and User Experience:**
  Verify that the user interface is accessible to all users, easy to navigate, and displays correctly on a variety of devices and screen resolutions, ensuring a consistent and satisfying user experience across all platforms.

3. Resources:

Tools:

- **Frontend:** Cypress was used for frontend testing.
- **Backend:** Postman was used for API testing.
- **Global:** GitHub for version control, Visual Studio Code as the code editor, Swagger for API documentation, and Excel and Google Sheets for data and result management and documentation.

Environments:

- **Google Chrome:** The browser used for frontend testing.
- **.NET 8:** Development environment for the backend.
- **Node.js:** Used to run the React application.

Local Environment:

- **Frontend:** qa-app, used to run frontend tests locally.
- **Backend:** qa-api, used to run backend tests locally.

Datasets:

- **Users, Products, and Orders:** These datasets were used for testing, ensuring adequate coverage of various functionalities and scenarios in the system. This includes both valid and invalid data for users, products, and orders. This allowed validation of user authentication, product entries like negative prices or empty fields, and order processing at different stages (creation, update, and cancellation).

4. Risks:

Technical Risks:

- **Incompatibility between tools and development environment** that could cause errors in the execution of tests.
  Mitigation Strategy: Conduct preliminary tests with specific versions of tools

and environment components to ensure they work well together before the full test execution.

Risks in Test Environment Configuration:

- **Errors that could affect test execution** due to incorrect test environment configuration.
  Mitigation Strategy: Use a replica environment to avoid interference with the production environment and ensure reliable test execution.

Data Integrity Risks:

- **Inconsistencies in test data** that could lead to incorrect results.
  Mitigation Strategy: Implement rigorous validation and verification processes for test data sets and maintain regular backups to restore data if necessary.

Security Risks:

- **Vulnerability to common attacks** such as SQL injections and XSS.
  Mitigation Strategy: Implement secure development practices, including input validation and secure data handling, as well as perform penetration testing to detect and fix vulnerabilities before deployment in production.

5. Deliverables:

- **Completed Test Report:** A document summarizing the test cases executed, the results obtained, and the errors identified during the testing process.

- **Test Logs:** A detailed list of errors and failures found, including a description of the problem, steps to reproduce it, assigned priority, and current resolution status.

- **Execution Report:** Screenshots, error logs, and other files that support the results obtained in the tests.

- **Test Scripts:** Delivery of automated test scripts developed to verify the correct execution of critical system flows, using tools like Cypress to ensure efficiency and accuracy in validation.