PART 1: TEST PLAN DESIGN

1. Scope

Backend (C# APIs)

User Authentication:

- Login validation with valid and invalid credentials.
- Verify error message for invalid credentials.
- Authentication token generation, validity and expiration tests.
- Security tests for brute force attacks and protection against SQL injections.

Product Management

- Create, read, update, and delete (CRUD) operations for products.
- Verify error handling for invalid product data.
- Test search and filter functionalities.

Order Processing:

- Create an order and verify order details.
- Test order cancellation and status updates.
- Verify error handling for invalid order operations.

Frontend (ReactJS):

ReactJS (User Authentication & Dashboard):

- Testing of login and logout functionalities.
- Evaluation of the display and updating of data in the Dashboard
- Evaluation of the adaptability and accessibility of the user interface.

ReactJS (Product Listing & Order Processing):

- Testing of the display and filtering of product listings.
- Evaluation of workflows for order creation, update and cancellation.
- Testing of the responsiveness and accessibility of the user interface.

2.Objectives:

- Product quality assurance: Ensure that both the frontend and backend function correctly according to the specified requirements.
- Data integrity validation: Verify that the information provided by users is correctly stored in the database and is properly retrievable.

- Usability: Ensure that users can interact with the application intuitively, without errors or difficulties.
- Performance and scalability: Ensure that the system handles user and data loads under stressful situations without failure.
- Security: Check for critical vulnerabilities, such as code injections, exposure of sensitive data or breaches in access permissions.

3. Resources:

Tools

- Cypress (for automated UI testing).
- Lighthouse (for performance and accessibility auditing).
- Postman: For backend API testing.
- Git: For version control and code storage.
- GPT-4: Technical Consulting: Advanced language modeling assistance to design test strategy, create detailed test cases, and suggest best practices for automation and documentation.

Environments:

- Local Development Environment: Local server for backend testing.
- Browsers: Google Chrome for frontend testing.
- Devices: Tests on desktop, tablet and mobile devices, without specifically evaluating responsiveness.

Test Data:

- Test users: Different credentials (valid and invalid).
- Test Products: Products with various attributes (name, prices, etc).
- Test orders: Orders with different status (pending, in process, completed).

4. Risks

Technical risks:

- Version incompatibilities between test tools and the development environment.
- Errors in the test environment configuration that may affect test execution.

- Lack of test coverage: Some key functionalities may not be tested correctly.
- Performance issues under load: The system may not behave properly with a high volume of users or data.
- Integration errors with external systems: Third-party services or APIs may be buggy or incompatible.

Mitigation strategies:

- Configuration and verification of controlled and well-documented environments prior to the start of testing to ensure that all aspects of the environment are aligned with project requirements.
- Use of an independent and replicable test environment to avoid interference with the production environment, ensuring that testing does not affect the performance of production systems.
- Execution of tests in multiple browsers and devices to ensure that the system is functional on different platforms, avoiding compatibility issues.
- Automation of repetitive and critical tests to ensure consistency of results and reduce the possibility of human error during test executions.
- Implementation of early integration tests to detect incompatibilities between system components or with external services as early as possible.