**Case Interview for Test Automation Engineer**

**Test Plan**

*Purpose: To conduct test for case interview for test automation engineer position.*

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description |
| 24/8 | 1 | Amir |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Overview 3](#_Toc175517160)

[1.1. Purpose 3](#_Toc175517161)

[1.2. Project Overview 3](#_Toc175517162)

[2. Scope 4](#_Toc175517163)

[2.1. Scope of Testing 4](#_Toc175517164)

[3. Test Strategy 5](#_Toc175517165)

[3.1. Test Objectives 5](#_Toc175517166)

[3.2. Test Assumptions 5](#_Toc175517167)

[3.3.  Test Principles 5](#_Toc175517168)

[3.4. Test Type & Approach 5](#_Toc175517169)

[3.5. Unit Testing 6](#_Toc175517170)

[3.6. Integration Testing 6](#_Toc175517171)

[3.7. Regression testing 7](#_Toc175517172)

[4. Test Environment Plan 8](#_Toc175517173)

[4.1. Test Environment Details 8](#_Toc175517174)

[5. QA Process 9](#_Toc175517175)

[5.1. Testing Process 9](#_Toc175517176)

[5.2. Bug Handling Process 9](#_Toc175517177)

[5.3. Recommendations for Improving QA Process 10](#_Toc175517178)

[5.4. Quality Gates 10](#_Toc175517179)

[5.5. Test Summary 10](#_Toc175517180)

[5.6. Conclusion 11](#_Toc175517181)

[6. Entry and Exit Criteria 12](#_Toc175517182)

[6.1. Entry Criteria 12](#_Toc175517183)

[6.2. Exit Criteria 12](#_Toc175517184)

# Overview

## Purpose

This test plan describes the testing approach and overall framework that will drive the testing of the Fruit Product and Order System. The document introduces the test strategy, execution strategy, and test management processes.

## Project Overview

The Fruit Product and Order System is an web app that allows users to view a list of fruit products, create orders, and manage product information. The system includes functionality for listing products, creating new products, updating existing products, deleting products, placing orders and deleting orders.

# Scope

## Scope of Testing

Given that Fruit Product and Order System web app is in initial stage, only the functional and UI testing will be performed. Test cases should include checking all functionalities related to the user and to ads as well as the cooperation with external systems such as the email system and billing system.

### In scope

#### Exploratory Testing

* Purpose: Identify critical defects before structured testing begins.
* Scope: First-level navigation, product listing, and order creation.
* Testers: Tester.
* Timing: At the beginning of each cycle.

#### Functional Testing

* Purpose: Check the functions of the application by feeding input and validating output.
* Scope:
  + Product listing
  + Product creation
  + Product updating
  + Product deletion
  + Order creation
  + API functionality for products and orders
* Testers: Testing Team.
* Method: Performed using Cypress for frontend testing and API testing.
* Timing: After Exploratory testing is completed.

### Out of scope

Performance testing, security testing (to be done separately)

# Test Strategy

## Test Objectives

* Verify that the functionality of the Fruit Product and Order System works according to specifications.
* Execute and verify test scripts, identify and fix all high and medium severity defects.
* Produce a production-ready software and a set of stable, reusable test scripts.

## Test Assumptions

* The application will be accessible at <http://localhost:4200> for frontend testing.
* The API will be accessible at <http://localhost:3000> for backend testing.
* Performance testing is not considered for this estimation.
* The Test Team will have access to the Test environment.
* Test case design activities will be performed by the QA Group.

##  Test Principles

* Testing will focus on meeting business objectives, cost efficiency, and quality.
* Testing processes will be well-defined yet flexible.
* Testing will be a repeatable, quantifiable, and measurable activity.

## Test Type & Approach

|  |  |
| --- | --- |
| Test Type | Objectives |
| Unit Testing | The objectives are to verify that:   * Individual components or functions of the application work correctly in isolation. * Each unit of the software performs as designed. * Edge cases and error conditions are handled properly at the component level. * The smallest testable parts of the application function correctly. |
| Integration Testing | The objectives are to verify that:   * Different components or functions of the application work together correctly. * Data is correctly passed between various units of the application. * The integrated system components meet the specified requirements. * The interactions between integrated units function as expected. * The application's modules or components integrate without error. |
| Regression Testing | The objectives are to verify that:   * Recent program or code changes have not adversely affected existing features * Previously eradicated bugs have not returned * The system remains stable and functional after any changes or additions * New changes have not introduced any new bugs into the existing functionalities * The application as a whole continues to meet the specified requirements after updates. |

## Unit Testing

Food Product and Order System incorporates unit testing methodologies, primarily implemented using Cypress. While Cypress is mainly an end-to-end testing tool, our test suite includes elements of all these testing types.

### Unit Testing Components:

#### Component Rendering Tests:

* Navigation component test: Verifies the existence of "Products" and "Orders" navigation links on the main page.
* Product list component test: Checks for the presence of the "Create" button on the product list page.

#### Individual Function Tests:

* While not explicitly implemented in the current test suite, future unit tests could include isolated checks for individual functions, such as price calculation logic.

## Integration Testing

Food Product and Order System incorporates unit testing methodologies, primarily implemented using Cypress. While Cypress is mainly an end-to-end testing tool, our test suite includes elements of all these testing types.

### Integration Testing Components:

#### Product CRUD Operations:

- Tests that create, read, update, and delete products through the UI, verifying interactions between frontend components and the backend API.

#### Order Creation and Verification:

- Tests that create an order for a specific product (e.g., "Red Apple") and verify its details in the order list, checking interactions across product list, order creation, and order list components.

#### API Tests:

- Comprehensive tests for both product and order APIs, verifying CRUD operations and ensuring proper communication between frontend and backend.

#### Navigation Tests:

- Tests that navigate between different pages (e.g., from product list to order creation), verifying routing and component interactions.

#### Data Flow Tests:

- Tests that ensure data consistency across different views, such as creating a product and verifying its appearance in the product list.

## Regression testing

### Full Application Workflow Tests

Tests that cover the entire user journey from product creation to order placement and verification, ensuring that core functionalities remain intact after changes.

### UI Consistency Tests

Tests that verify the presence and functionality of key UI elements across different pages, helping to catch unintended UI changes.

### API Stability Tests:

- Regular runs of all API tests to ensure that backend changes don't break existing functionalities.

### Cross-component Interaction Tests:

- Tests that verify interactions between different components (e.g., product list affecting order creation) remain functional after updates.

# Test Environment Plan

## Test Environment Details

### Software

* Frontend application accessible at http://localhost:4200
* Backend API accessible at http://localhost:3000
* Testing environment should include browsers supported by Cypress (Chrome, Firefox, Edge)
* Node.js and Cypress installed for test execution

# QA Process

## Testing Process

1. Test Planning: Create and update test plans for each release.
2. Test Case Development: Design and review test cases based on requirements.
3. Test Execution:
   1. Exploratory Testing
   2. Automated Testing using Cypress
   3. API Testing

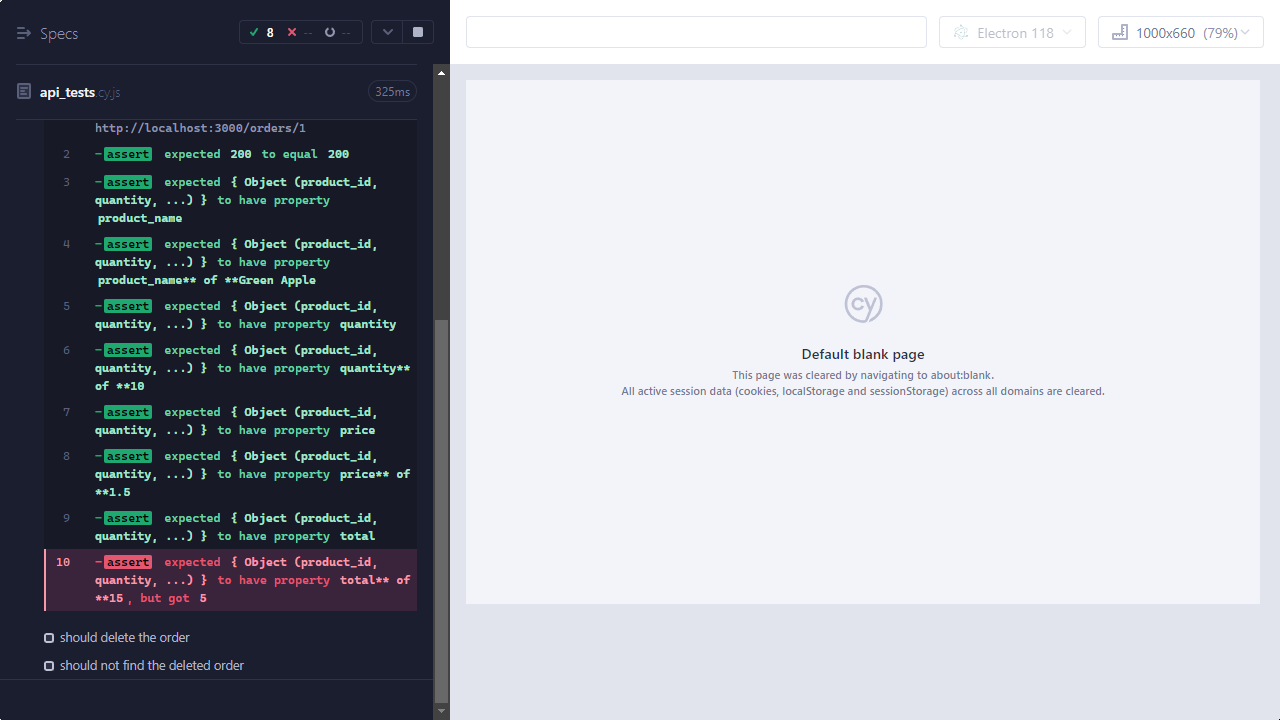
## Bug Handling Process

### Expected vs Actual Results

**Test Case**: Update Order Details and Validate Response via API

**Issue Description**: During the Cypress test execution, the following assertion failed. The test expected the total property in the response body to be 15, but the actual value returned was 5.

**Error Details**:



**Expected Behavior**:

* The total property in the response should be calculated as quantity \* price, which is 10 \* 1.5 = 15.

**Actual Behavior**:

* The total property in the response was 5.

**Possible Causes**:

1. **Incorrect Calculation in Backend**: The backend might not be calculating the total correctly.
2. **Data Update Issue**: The PATCH request may not be updating the order data correctly.
3. **Response Mismatch**: The response may not reflect the updated order details.

**Steps to Reproduce**:

1. Send a PATCH request to update the order with quantity 10 and price 1.5.
2. Verify the response to ensure the total is correctly calculated.

**Suggested Actions**:

* Review the backend logic for calculating the total to ensure it's correct.
* Check the request handling in the backend to confirm that it processes updates accurately.
* Verify that the response data is consistent with the updates made.

## Recommendations for Improving QA Process

1. Implement Continuous Integration/Continuous Deployment (CI/CD) pipeline to automate build and deployment processes.
2. Increase automated test coverage using Cypress, focusing on critical user journeys.
3. Implement performance testing for critical user journeys.
4. Establish a metrics program to track and improve testing efficiency and effectiveness.

## Quality Gates

Establish the following quality gates before proceeding to the next phase:

1. Code Review: All code changes must pass peer review before merging.
2. Unit Test Coverage: Maintain a minimum of 80%-unit test coverage.
3. Automated Test Pass Rate: 100% of automated tests must pass before deployment to QA.

By implementing these QA processes and recommendations, we aim to improve the overall quality of the Fruit Product and Order System, reduce the number of production issues, and increase stakeholder confidence in the releases.

## Test Summary

### Test Scenarios

#### Order List Page:

1. **Objective**: Verify that the order list page displays the correct table headers.
2. **Outcome**: Test passed; all table headers (Order ID, Order Date, Product Name, Price, Quantity, Total, Actions) are visible.

#### Create Order:

1. **Objective**: Create an order for "Red Apple" and verify it appears correctly in the orders list.
2. **Outcome**: Test passed; order was created successfully, and the order details (product name, price, quantity, total) were verified.

#### Delete Order:

1. **Objective**: Delete an order and verify that it is removed from the orders list.
2. **Outcome**: Test passed; the order was deleted successfully and removed from the list.

#### Update Order:

1. **Objective**: Update the order details (quantity and price) and verify the updated details in the response.
   1. **Issue**: Assertion error due to incorrect total calculation in the response.
      1. **Error**: Expected total was 15, but actual total returned was 5.
      2. **Suggested Actions**:
         1. Review backend logic for total calculation.
         2. Verify request handling and response data consistency.

For a detailed breakdown of the test results, including individual test cases, screenshots, and other relevant information, please refer to the detailed HTML report attached in[Test Results HTML Report. This report provides comprehensive information on test execution, including pass/fail status, error details, and other metrics.

### Setup and Configuration

#### Angular Server:

* To avoid issues with localhost recognition in Cypress, configure Angular to use --host 0.0.0.0 to ensure accessibility across different ports.

### Known Issues and Workarounds

#### Localhost vs. 127.0.0.1:

1. **Issue**: Cypress may have issues with 127.0.0.1 but not localhost for some configurations.
2. **Workaround**: Set Angular server to use --host 0.0.0.0 and configure the hosts file for proper domain resolution.

#### Cypress Assertion Errors:

1. **Issue**: Incorrect total calculation in API response.
2. **Workaround**: Verify backend calculations and data handling to ensure consistency with expected results.

## Conclusion

* The testing framework (Cypress) has been used effectively to validate the core functionalities of the fruit product and order system.
* Most tests passed successfully, with the exception of an issue with the total calculation during order updates.
* Workarounds for localhost and port issues have been implemented to ensure reliable testing environments.

# Entry and Exit Criteria

## Entry Criteria

The testing process will commence only when the following conditions are met:

### Test Environment Setup

* + All necessary software, including npm, Cypress and any other testing tools, is installed and configured.

### Test Data Availability

* + Access to JSON data for the Fruit Product and Order System.

### Test Cases Prepared

* + Test scripts for automated testing (using Cypress) have been created and validated.

### Development Completion

* + The web app is available for testing.

### Availability of Resources

* + The tester is available and ready to start the testing process.

## Exit Criteria

The testing process will be considered complete, and the product will be ready for the next phase (release or further development) when the following conditions are met:

### Test Case Execution

* + All planned test cases have been executed, including functional, UI, and integration tests.
  + All critical paths and high-risk areas have been tested and verified.

### Defect Resolution

* + All identified defects have been logged, tracked, and resolved, with no high-severity defects remaining open.
  + All fixed defects have been retested and verified, ensuring they no longer exist.

### Test Coverage

* + The test coverage meets the defined goals, with all critical features and modules tested.
  + Any areas not covered by tests are documented with explanations and risk assessments.

### Test Summary Report

* + A comprehensive test summary report has been prepared, including the results of all test cases, defect logs, and any remaining risks.