Test Plan and Architecture Report

1. Introduction

This document provides a comprehensive test plan for both API and UI testing of the Petstore API and the SauceDemo web application. It also explains the architecture used, along with the benefits of the chosen approach.

2. Test Plan

2.1 API Testing

Objective:

The objective of API testing is to verify the functionality, reliability, performance, and security of the Petstore API endpoints.

Scope:

- Validate API requests and responses for correctness.
- Ensure the API handles errors properly.
- Test the API with valid and invalid data.

Test Scenarios:

| Test Case ID | Test Scenario | Endpoint & Method | Expected Result |
|-----------------|-----------------------------|----------------------|------------------------|
| API-01 | Add a pet with valid data | POST /v2/pet | 200 OK, Pet is added |
| API-02 | Add a pet with invalid data | POST /v2/pet | 500 Internal Server |
| | type | | Error |
| API-03 | Retrieve pet by valid ID | GET /v2/pet/{id} | 200 OK, Pet details |
| | | | returned |
| API-04 | Retrieve pet by invalid ID | GET /v2/pet/{id} | 404 Not Found |

2.2 UI Testing

Objective:

The objective of UI testing is to ensure that the SauceDemo web application functions correctly, provides a seamless user experience, and meets business requirements.

Scope:

- Validate login functionality with different credential scenarios.
- Test product listing and cart management.
- Ensure proper error handling and messages.

Test Scenarios:

| Test Case ID | Test Scenario | Expected Result | |
|--------------|--------------------------------|--------------------------------|--|
| UI-01 | Login with valid credentials | Redirect to product page | |
| UI-02 | Login with invalid credentials | Display error message | |
| UI-03 | Login with empty password | Display "Password is required" | |
| 01-03 | | error | |
| UI-04 | | Display "Username is required" | |
| 01-04 | Login with empty username | error | |
| UI-05 | View product listing page | Display 6 products | |
| UI-06 | Add a product to the cart | Cart badge shows 1 item | |
| 111.07 | Remove a product from the | Cart badge is not visible | |
| UI-07 | cart | | |

3. Architecture Used

3.1 Page Object Model (POM) for UI Testing

The UI tests follow the **Page Object Model (POM)** to improve maintainability and reusability.

Structure:

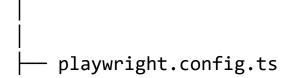
Benefits:

- Encapsulation: Separates UI elements from test logic.
- **Reusability**: Common page interactions are reusable across tests.
- **Maintainability**: Changes in UI require modifications only in the page classes, not all test cases.

3.2 API Testing with Playwright APIRequestContext

The API tests use Playwright's APIRequestContext, which allows testing APIs independently of UI tests.

Structure:



Benefits:

- **Separation of Concerns**: Business logic for API requests is encapsulated in service classes.
- Scalability: New API test cases can be added easily.
- **Maintainability**: Changes in API endpoints require modifications only in service classes, not test scripts.

4. Conclusion

By implementing the **Page Object Model (POM) for UI testing** and **modular service-based API testing**, the test framework ensures maintainability, scalability, and efficiency. These methodologies improve automation test quality, making them ideal for long-term projects.