Test Plan

Project: Fakestore RestAPI

Table of Contents

[1 Introduction 3](#_Toc205798927)

[1.1 Objective 3](#_Toc205798928)

[2 Scope 3](#_Toc205798929)

[2.1 In-Scope 3](#_Toc205798930)

[2.2 Out-of-Scope 3](#_Toc205798931)

[3 Testing Strategy 3](#_Toc205798932)

[3.1 Test Scenarios 3](#_Toc205798933)

[3.2 Test Assumptions 4](#_Toc205798934)

[3.3 Data Approach 4](#_Toc205798935)

[3.4 Level of Testing 4](#_Toc205798936)

[4 Execution Strategy 4](#_Toc205798937)

[4.1 Entry Criteria 4](#_Toc205798938)

[4.2 Exit criteria 5](#_Toc205798939)

[4.3 Defect Management 6](#_Toc205798940)

[5 Environment Requirements 6](#_Toc205798941)

[5.1 Test Environments 6](#_Toc205798942)

[6 Dependencies 6](#_Toc205798943)

[7 Documents Repository 6](#_Toc205798944)

[8 Sign Off 7](#_Toc205798945)

**Purpose of the Document**

This test plan provides a clear roadmap for verifying basic CRUD operations on users in the Fake Store API using Java, Rest Assured, and TestNG in a Maven project

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 08/10/2025 | V1.0 |  | Joseph Kodadhala |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Objective

Validate core CRUD (Create, Read, Update, Delete) operations of the Fake Store RestAPI

# Scope

## In-Scope

These cases collectively provide comprehensive coverage for CRUD operations on the Fake Store API, ensuring reliability, stability, and helpful error messaging for both typical and erroneous usage patterns

Includes Validation of:

* The HTTP status codes for each operation
* Typical success and failure scenarios (e.g. valid/invalid IDs, required/invalid payloads)
* Handling of incorrect HTTP methods and malformed requests

## Out-of-Scope

Excludes the validation of,

* Security, authentication, or authorization testing (since the API publicly exposes endpoints without authentication)
* Data persistence validation, since the Fake Store API is a mock service with simulated, non-persistent responses (data created/updated/deleted is not retained)

# Testing Strategy

## Test Scenarios

| ***HTTP Method*** | ***Test Scenario*** | ***Endpoint*** | ***Expected Result*** |
| --- | --- | --- | --- |
| *GET* | *Retrieve all users* | */users* | *200 OK, JSON array, not empty* |
| *GET* | *Retrieve single user by id* | */users/{id}* | *200 OK, JSON, id matches* |
| *POST* | *Create new user* | */users* | *201, id in response* |
| *PUT* | *Update existing user by id* | */users/{id}* | *200 OK, updated fields match* |
| *DELETE* | *Delete user by id* | */users/{id}* | *200 OK, user removed* |

## Test Assumptions

Key test cases to verify GET, POST, PUT, and DELETE methods on the Fake Store API should cover both standard (positive) operations and error (negative/edge) scenarios to ensure the API behaves as expected

It is the responsibility of the tester to open the defects, retest and close the defect*.*

## Data Approach

Since the Fake Store API is a mock service, created for prototyping and testing purposes, it returns pre-defined product, user, and cart data, and does not persist any new or updated records created through POST or PUT methods testing.

## Level of Testing

Types of testing to be performed

|  |  |  |
| --- | --- | --- |
| **Test Type** | **Description** | **Responsible Parties** |
| Unit Testing |  |  |
| Functional Testing |  |  |
| User Acceptance Testing |  |  |
| Security Testing |  |  |
| Regression Testing |  |  |
| Performance Testing (non-functional) |  |  |

# Execution Strategy

## Entry Criteria

* The entry criteria refer to the desirable conditions in order to start test execution
* Entry criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions and provide a recommendation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Test Team** | **Technical Team** | **Notes** |
| Test environment(s) is available | C:\Users\arxp\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\7F9Z3IW4\MC900441310[1].png |  |  |
| Test data is available |  |  |  |
| Code has been merged successfully |  |  |  |
| Development has completed unit testing |  |  |  |
| Test scripts are completed, reviewed and approved by the Project Team |  |  |  |

## Exit criteria

* The exit criteria are the desirable conditions that need to be met in order proceed with the implementation.
* Exit criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions and provide a recommendation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Exit Criteria** | **Test Team** | **Technical Team** | **Notes** |
| 100% Test Scripts executed | C:\Users\arxp\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\7F9Z3IW4\MC900441310[1].png |  |  |
| 90% pass rate of Test Scripts |  |  |  |
| No open Critical and High severity defects |  |  |  |
| All remaining defects are either cancelled or documented as Change Requests for a future release |  |  |  |
| All test metrics collected based on reports from daily and Weekly Status reports |  |  |  |

## Defect Management

* + It is expected that the testers will execute all the scripts in each of the cycles described in test case/traceability documents.
  + The defects will be tracked through Jira or ALM Quality Center.
  + It is the responsibility of the tester to open the defects, retest and close the defect*.*

Defects found during the Testing should be categorized as below:

|  |  |
| --- | --- |
| **Severity** | **Impact** |
| 1 (Critical) | * Functionality is blocked and no testing can proceed * Application/program/feature is unusable in the current state |
| 2 (High) | * Functionality is not usable and there is no workaround, but testing can proceed |
| 3 (Medium) | * Functionality issues but there is workaround for achieving the desired functionality |
| 4 (Low) | * Unclear error message or cosmetic error which has minimum impact on product use. |

# Environmental Requirements

## Test Environments

* Unit Testing is performed in DEV environment
* Functional, User Acceptance and Regression Testing is performed in UAT/QA environment

# Dependencies

* + Dependencies like Rest Assured, TestNG, and Extent Reports often bring in transitive dependencies which may have version mismatches

# Documents Repository

|  |  |
| --- | --- |
| **Document Name** | **Sharepoint Location** |
| Functional specifications |  |
| QA Test plan |  |
| Test cases/Traceability Matrix |  |

# Sign Off

|  |  |
| --- | --- |
| **Sign-Off For** | **Sign-Off Done By** |
| Development & Unit Test |  |
| QA Validation |  |
| Acceptance & Release Readiness |  |