API DemoQA Test Plan

# 1. Purpose

This document describes the test plan for the Bookstore API on the DemoQA site, aiming to:

* Ensure the Bookstore endpoints function correctly.
* Verify the data integrity of the user and book catalogs.
* Ensure API responses comply with RESTful standards (status codes, JSON structure, error messages).

**Main Business Operations:** The Bookstore API allows users to create an account, authenticate with a token, view the list of books, and add/remove books from their personal collection.

# 2. Scope of Testing

## 2.1 In Scope

* **Module:** Bookstore.
* **Main Endpoints:**

| **Group** | **Endpoint** | **Description** |
| --- | --- | --- |
| User | /Account/v1/User | Create/Get/Delete user information |
| Token | /Account/v1/GenerateToken | Generate an authentication token |
| Book | /BookStore/v1/Book, /BookStore/v1/Books | Get the list of books, and get detailed book information |
| Collection | /BookStore/v1/Book, /BookStore/v1/Books | Add/Delete/Update books in the collection |

* **Functions to be tested:** User CRUD, token authentication, adding/deleting books, and error handling.

## 2.2 Out of Scope

* Performance, load, and advanced security testing (SQL Injection, fuzzing).
* Integration testing with other systems.

# 3. Test Objectives

* Verify valid HTTP response codes (200, 201, 400, 401, 404).
* Ensure response data is in the correct JSON format and matches the schema.
* Test error handling with invalid tokens, invalid ISBNs, and empty input data.
* Ensure user and collection CRUD operations work seamlessly.

# 4. Test Strategy

## 4.1 Methods

* **Black-box Testing:** Based on API input/output.
* **Combined Automation & Manual:**
  + **Automation:** Postman Collection + Newman.
  + **Manual:** Swagger Inspector, Postman GUI.

## 4.2 Test Types

* **Functional testing:** CRUD, authentication, adding/deleting books.
* **Negative testing:** Invalid token, non-existent ISBN, empty required fields.
* **Integration testing:** End-to-End flow (create user → add book → view → delete).

## 4.3 Pass/Fail Criteria

| **Result** | **Description** |
| --- | --- |
| Pass | API returns the correct HTTP code, accurate JSON data, and response time is < 2s. |
| Fail | API returns an incorrect code, JSON structure does not match the schema, 500 error, or timeout. |

# 

# 5. Test Environment

# 

| **Component** | **Description** |
| --- | --- |
| Server | <https://demoqa.com/swagger> |
| Tools | Postman v11, Newman CLI |
| Configuration | HTTPS, Bearer Token authentication |
| Test Data | Valid/Invalid ISBNs |
| OS & Browser | Windows 11, Chrome 141 |
| Test Token | A new token is generated before each test session |

# 

# 

# 6. Deliverables

| **Document** | **Description** |
| --- | --- |
| Test Plan | This test plan document |
| Test Case / Script | Postman Collection file (.json) |
| Execution Report | Test result report from Newman (HTML/JSON) |
| Bug Report | List of defects logged in Jira or Excel |

# 7. Test Scenarios

| **ID** | **Scenario** | **Description** | **Endpoint(s)** | **Objective** |
| --- | --- | --- | --- | --- |
| TS\_01 | User Management | Valid and invalid user CRUD operations | /Account/v1/User | Ensure the account functions correctly |
| TS\_02 | Authentication | Authenticate user, receive token | /Account/v1/GenerateToken | Verify login/token response |
| TS\_03 | Book Management | Get the book list and book details | /BookStore/v1/Books, /BookStore/v1/Book | Ensure data accuracy |
| TS\_04 | Book Collection | Add/Delete/Update books for a user | /BookStore/v1/Book | Ensure Add/Delete/Update logic |
| TS\_05 | End-to-End Flow | Create user → login → add → update → delete → delete user | All | Test the overall flow |
| TS\_06 | Error Handling | Invalid token, ISBN error, empty data | All | Ensure API handles errors according to REST standards |