

Test Plan

A Test Plan describes how testing will be performed, what will be tested, and who will do it.

1. **Test Plan Identifier**

Test Plan ID: TP-BOOKCART-01

2. **Test Plan Objective**

The objective of this test plan is to verify that the Book Cart web application functions correctly according to specified requirements and meets quality standards through manual testing, automation testing, and API testing.

Scope of Testing

In scope	Out of scope
User Registration	Payment gateway integration
User Login & Logout	Third-party service testing
Book Listing & Search	Load and stress testing (unless required)
Add to Cart & Cart Management	
Order Placement& Admin login	
Book Management (Admin)	
API Testing for authentication, books and others	

2. Test Items

- Web UI (Frontend)
- Backend APIs
- Database interactions (indirectly via UI & APIs)

3. Types of Testing

1. Functional Testing
2. Regression Testing
3. Smoke Testing
4. Negative Testing
5. Automation Testing
6. API Testing

4. Test Approach

Manual Testing will be performed using predefined test cases.

Automation Testing will be done using **Selenium** for critical user flows.

API Testing will be conducted using **Postman** with assertions. Bugs will be logged and tracked using **Jira**.

5. Test Environment

Operating System  Windows

Browser  Chrome/Edge

Testing Tools  Selenium, Playwright, Postman, Jira

6. Entry Criteria

Testing will begin when:

- Application is accessible.
- Requirements are understood.
- Test cases are prepared.

7. Exit Criteria

Testing will be completed when:

- All planned test cases are executed.
- Critical defects are fixed or reported.
- Test summary report is prepared.

8. Roles & Responsibilities

Tester: Design and execute test cases, report defects.

Test Lead: Review test cases and reports.

9. Risks & Mitigation

Risk	Mitigation
Application Downtime	Test during availability
Limited test date	Use dummy test data
Time constraints	Prioritize critical modules

10. Roles & Responsibilities

- Test Plan Document
- Manual Test Cases

- Automation Scripts
- Postman Collection
- Bug Reports
- Test Summary Report

Manual Testing

Test Case ID	Test Case Scenario	Test Steps	Expected Result	Actual Result	Status
TC_001	Verify user can open BookCart Homepage	1.Open browser 2.Navigate to https://bookcart.azurewebsites.net/ 3. Press Enter	Homepage loads within 5 seconds.	Homepage loaded successfully	Pass
TC_002	User Registration with Valid Data	1.Enter valid username and password 2. Enter Register	User should be registered successfully	Registered successfully	Pass
TC_003	User Registration with Invalid Email	1.Enter invalid 2. Enter Email	User should be registered successfully	No email required	Fail
TC_004	User Registration with already registered username	Enter already registered username	A message appeared Already registered or not available	Username not available appeared	Pass
TC_005	Register with empty credentials	Keep fields empty	Validation message appear or	Empty fields turned	

			color must change		Pass
TC_006	Login with valid credentials	1.Enter username 2. Enter password	User should be logged in successfully	Successfully logged in.	Pass
TC_007	Login with Invalid Password	1.Enter username 2. Enter invalid password	Logg in should be failed with an error message	Nothing happened	Fail
TC_008	Login with empty fields	Keep login credentials empty	Validation message should appear or color must change	Empty fields turn red	Pass
TC_009	View Book List	Click on Books section	List of books should be displayed	No books appeared.	Fail
TC_010	Search book by name	Search book by entering name “ Sulphite ”	Relevant book results should appear	No books appeared	Fail
TC_011	Search non-existing book	Enter book name that does not exist	No results found	No books found	Pass
TC_012	Check the cart button	Click the cart button	It should display cart module	Cart section appears	Pass
TC_013	When no books added, cart is empty	Check cart when no books added	Validation message appears that cart is empty	Validation message appears	Pass

TC_014	Add book to cart	Add books	Cart should display added books	No addition	Fail
TC_015	Check cart features	Click “Continue Shopping”	Clicking Continue Shopping loads the homepage	Homepage loaded successfully	Pass
TC_016	Checking the Logo Functionality	Click BookCart Logo on the left top	Loads Homepage	Homepage successfully loaded	Pass
TC_017	Access Cart without login	Without login , click cart button	Cart button should not respond	Cart section opens	Fail
TC_018	Check order module	Click My Orders when no orders are placed	Validation message must appear	Validation message appears	Pass
TC_019	Access My Orders Section	Place orders	Orders must be placed and displayed in My Orders Section	No orders placed	Fail
TC_020	Verify that swagger UI page is accessed	Click Swagger UI	Next module should be displayed successfully	Page displayed successfully	Pass
TC_021	Verify GET /api/Book returns list of all books	Expand GET /api/Book , click Try it out and execute	It should return 200 OK status and list of all books	API returns 500 (Error: issue at the server side) with no response body	Fail

TC_02 2	API for validate username returns availability status for existing username	Expand GET /api/User/validateUserName/{userName}	It should display status 200 with response body false: for already existing username	It returned 200 with response body false	Pass
TC_02 3	API for validate username returns availability status for non-existing username	Expand GET /api/User/validateUserName/{userName}	It should display status 200 with response body true: for non-existing username	It returned 200 with response body true:	Pass
TC_02 4	Logout Functionality	Click Logout	User should be logged out and redirected to login page	Redirected to login page after logout	Pass
TC_02 5	UI Responsiveness	Open https://bookcart.azurewebsites.net/ on mobile	Resize to mobile width	Page is incomplete on mobile, only left side is displayed	Fail

Automation Testing

Introduction

BookCart is an online demo bookstore that allows users to browse, search, and purchase books. Automation testing was implemented to verify key functionalities efficiently, reduce manual effort, and ensure reliable testing results.

Purpose of Automation Testing:

- Verify website functionalities automatically
- Reduce manual testing effort and time
- Capture pass/fail results systematically

Tools and Environment

Tools Used:

Tool/Library	Purpose
Python	Scripting language for automation
Selenium WebDriver	Automate browser interactions
Chrome Browser	Web browser used for testing
ChromeDriver	Selenium driver for Chrome
Visual Studio Code	IDE to write and run scripts

Test Environment:

Parameter	Details
Operating System	Windows 10
Browser	Google Chrome (latest stable)
Python Version	3.x
Selenium Version	4.x

TEST CASES:

<i>Test Case #</i>	<i>Test Name</i>	<i>Description</i>	<i>Expected Result</i>
1	Homepage Load	Verify homepage loads	Page title contains "Book"
2	Register Page	Navigate to registration page	Page loads with URL containing 'register'

3	Login Page	Navigate to login page	Page loads with URL containing 'login'
4	Search Books	Verify search functionality	At least one book appears (may fail if no books exist)
5	Add to Cart	Add first book to cart	Book added confirmation visible (may fail if no books)
6	Cart Page	Navigate to cart page	URL contains 'shopping-cart'
7	Book Details Page	Open details of book id 3	Book details displayed (may fail if book missing)
8	Logout	Perform logout	Logout successful

API Testing

TC Id	API endpoint	Method	Description	Expected result	Actual Result	Status
001	/api/Account/register	POST	User Registration	200 OK	405 Method Not Allowed	Fail
002	/api/Account/login	POST	User Login	200 OK	405 Method Not Allowed	Fail
003	/api/Account/login	POST	Invalid Login Credentials	401 Unauthorized	405 Method Not Allowed	Fail
004	/api/Account/profile	GET	Get User Profile	200 OK	200 OK	Pass
005	/api/Account/profile	PUT	Update profile	200 OK	405 Method Not Allowed	Fail

006	/api/Amin/dashboar d	GET	Admin Dashboard Info	200 OK	200 OK	Pass
007	/api/Account/book/1	DELETE	Delete Book	200 OK	405 Method Not Allowe d	Fail