Bookstore API

Client-Server Architectures

Name: Senuri Hansamini Wedamulla

Table of Contents

1. Introduction

This report documents the test cases for the Bookstore API, which is designed to manage book, author, customer, cart and order data for an online bookstore. The API is built using **JAX-RS** (Java API for RESTful Web Services), which facilitates the development of RESTful web services in Java. It uses **JSON** as the format for data exchange between the client and server.

The primary tool used for testing the API is **Postman**, which allows for efficient testing of the various endpoints to ensure that the API behaves as expected under different conditions. The report presents the test cases in a structured format, focusing on various aspects such as functionality, input validation, and error handling.

2. API Endpoint Test Case Tables

Endpoint	Description	HTTP Method	Request Body (JSON)	Expected HTTP Status Code	Expected Response Body (JSON)	Actual Result (Pass/Fail)
Book						
POST /books	Create a book with valid data.	POST	{ "bookId": 106, "title": "Harry Potter and the Prisoner of Azkaban", "authorId": 202, "isbn": "9780439708180", "publicationYear": 2000, "price": 13.99, "stock": 14 }	201 Created	{ "bookId": 106, "title": "Harry Potter and the Prisoner of Azkaban", "authorId": 202, "isbn": "9780439708180", "publicationYear": 2000, "price": 13.99, "stock": 14 }	Pass
POST /books	Create a book with invalid data. (Publication year in the future)	POST	{ "bookId": 106, "title": "Harry Potter and the Prisoner of Azkaban", "authorId": 202, "isbn": "9780439708180", "publicationYear": 2076, "price": 13.99, "stock": 14 }	400 Bad Request	{ "error": "Invalid Input Exception", "message": "Publication year cannot be in the future.", "path": "http://localhost:8080/Bookstore/rest/books", "status": 400, "timestamp": "2025-04-26T10:21:49.835359400+05:30[Asia/Colombo]" }	Pass
POST /books	Create a book with an invalid author ID (not found).	POST	{ "bookId": 106, "title": "Harry Potter and the Prisoner of Azkaban", "authorId": 506, "isbn": "9780439708180", "publicationYear": 2016, "price": 13.99, "stock": 14	404 Not Found	{ "error": "Invalid Input Exception", "message": "Author ID must be between 201 and 299.", "path": "http://localhost:8080/Bookstore/rest/books", "status": 404,	Pass

			}		"timestamp": "2025-04-26T10:20:10.786526500+05:30[Asia/Colombo]" }	
GET /books	Get all books.	GET		200 OK	[{ "bookId": 101, "title": "The Hobbit", "authorId": 201, "isbn": "9780261103344", "publicationYear": 1937, "price": 14.99, "stock": 12 }, { "bookId": 102, "title": "Harry Potter and the Sorcerer's Stone", "authorId": 202, "isbn": "9780439708180", "publicationYear": 1997, "price": 12.99, "stock": 15 }, { "bookId": 103, "title": "The Name of the Wind", "authorId": 203, "isbn": "9780756404741", "publicationYear": 2007, "price": 16.99, "stock": 10 }, { "bookId": 104, "title": "Mistborn: The Final Empire", "authorId": 204, "isbn": "9780765311788", "publicationYear": 2006,	Pass

					"price": 13.99,
					"stock": 8
					},
					{
					"bookId": 105,
					"title": "The Way of Kings",
					"authorId": 204,
					"isbn": "9780765326355",
					"publicationYear": 2010,
					"price": 19.99,
					"stock": 5
					}
					]	

GET /books/{id}	Get book by valid ID	GET		200 OK	{
					"bookId": 102,
					"title": "Harry Potter and the Sorcerer's Stone",
					"authorId": 202,
					"isbn": "9780439708180",
					"publicationYear": 1997,
					"price": 12.99,
					"stock": 15
					}
						Pass

GET /books/{id}	Get book by invalid ID	GET		404 Not Found	{
					"error": "Invalid Input Exception",
					"message": "Invalid input: bookId must be between 101 and 199.",
					"path": "http://localhost:8080/Bookstore/rest/books/300",
					"status": 400,
					"timestamp": "2025-04-25T16:33:41.815876800+05:30[Asia/Colombo]"
					}
						Pass

PUT /books/{id}	Update book with valid data	PUT	{ "bookId": 101, "title": "The Hobbit", "authorId": 201,	200 OK	{ "bookId": 101, "title": "The Hobbit", "authorId": 201, "isbn": "9780261103344",	Pass

			"isbn": "9780261103344", "publicationYear": 2013, "price": 14.99, "stock": 12 }		"publicationYear": 2013, "price": 14.99, "stock": 12 }	
PUT /books/{id}	Update book with invalid data (Negative price)	PUT	{ "bookId": 102, "title": "The Hobbit", "authorId": 201, "isbn": "9780261103344", "publicationYear": 2013, "price": -14.99, "stock": 12 }	400 Bad Request	{ "error": "Invalid Input Exception", "message": "Price cannot be negative.", "path": "http://localhost:8080/Bookstore/rest/books/102", "status": 400, "timestamp": "2025-04-26T10:25:37.379132100+05:30[Asia/Colombo]" }	Pass
PUT /books/{id}	Update book with non-existent ID	PUT	{ "bookId": 109, "title": "The Hobbit", "authorId": 201, "isbn": "9780261103344", "publicationYear": 2013, "price": 14.99, "stock": 12 }	404 Not Found	{ "error": "Book not found", "message": "Book with ID 109 not found.", "path": "books/109", "status": 404, "timestamp": "2025-04-25T16:35:31.282640500+05:30[Asia/Colombo]" }	Pass
DELETE /books/{id}	Delete a book by valid ID	DELETE		204 No Content	204 No Content	Pass
DELETE /books/{id}	Delete a book by invalid ID	DELETE		404 Not Found	{ "error": "Book not found", "message": "Book with ID 109 not found.", "path": "books/109", "status": 404, "timestamp": "2025-04-25T10:33:45.822723700+05:30[Asia/Colombo]" }	Pass
Author						

POST /authors	Create an author with valid data.	POST	{
			"authorId": 205,
			"firstName": "Sulo",
			"lastName": "Swarnamali",
			"biography": "Known for the Swarnamali chronicals."
			}
				201 Created	{
					"authorId": 205,
					"firstName": "Sulo",
					"lastName": "Swarnamali",
					"biography": "Known for the Swarnamali chronicals."
					}
						Pass

POST /authors	Create an author with invalid data. (First name is a number)	POST	{
			"authorId": 205,
			"firstName": 123,
			"lastName": "Swarnamali",
			"biography": "Known for the Swarnamali chronicals."
			}
				400 Bad Request	{
					"error": "Invalid Input Exception",
					"message": "First name cannot be all numbers.",
					"path": "http://localhost:8080/Bookstore/rest/authors",
					"status": 400,
					"timestamp": "2025-04-26T10:31:55.818581500+05:30[Asia/Colombo]"
					}
						Pass

GET /authors	Get all authors.	GET		200 OK	[{ "authorId": 201, "firstName": "J.R.R.", "lastName": "Tolkien", "biography": "Author of The Lord of the Rings." }, { "authorId": 202, "firstName": "J.K.", "lastName": "Rowling", "biography": "Creator of the Harry Potter series." }, { "authorId": 203, "firstName": "Patrick", "lastName": "Rothfuss", "biography": "Author of The Kingkiller Chronicle." },	Pass

					{
					"authorId": 204,
					"firstName": "Brandon",
					"lastName": "Sanderson",
					"biography": "Known for the Mistborn series."
					},
					{
					"authorId": 205,
					"firstName": "Sulo",
					"lastName": "Swarnamali",
					"biography": "Known for the Swarnamali chronicals."
					},
					{
					"authorId": 206,
					"firstName": "123",
					"lastName": "Swarnamali",
					"biography": "Known for the Swarnamali chronicals."
					}
					]	

GET /authors/{id}	Get author by valid ID	GET		200 OK	{
					"authorId": 201,
					"firstName": "J.R.R.",
					"lastName": "Tolkien",
					"biography": "Author of The Lord of the Rings."
					}
						Pass

GET /authors/{id}	Get author by invalid ID	GET		404 Not Found	{
					"error": "Author Not Found",
					"message": "Author with ID 207 not found.",
					"path": "http://localhost:8080/Bookstore/rest/authors/207",
					"status": 404,
					"timestamp": "2025-04-25T16:50:20.717907600+05:30[Asia/Colombo]"
					}
						Pass

PUT /authors/ {id}	Update author with valid data	PUT	{ "authorId": 206, "firstName": "Sulo", "lastName": "Swarnamali", "biography": "xyz" }	200 OK	{ "authorId": 206, "firstName": "Sulo", "lastName": "Swarnamali", "biography": "xyz" }	Pass
PUT /authors/ {id}	Update author with invalid data (first name is a number)	PUT	{ "authorId": 203, "firstName": "124", "lastName": "Swarnamali", "biography": "xyz" }	400 Bad Request	{ "error": "Invalid Input Exception", "message": "First name cannot be all numbers.", "path": "http://localhost:8080/Bookstore/rest/authors/203", "status": 400, "timestamp": "2025-04-26T10:32:57.776878600+05:30[Asia/Colombo]" }	Pass
PUT /authors/ {id}	Update author with non-existent ID	PUT	{ "authorId": 208, "firstName": "Sulo", "lastName": "Swarnamali", "biography": "xyz" }	404 Not Found	{ "error": "Author Not Found", "message": "Author with ID 208 not found.", "path": "http://localhost:8080/Bookstore/rest/authors/208", "status": 404, "timestamp": "2025-04-25T16:51:34.407246+05:30[Asia/Colombo]" }	Pass
DELETE /authors/ {id}	Delete an author by valid ID	DELETE		204 No Content	204 No Content	Pass
DELETE /authors/ {id}	Delete an author by invalid ID	DELETE		404 Not Found	{ "error": "Author Not Found", "message": "Author with ID 209 not found.", "path": "http://localhost:8080/Bookstore/rest/authors/209", "status": 404,	Pass

					"timestamp": "2025-04-25T16:52:18.535177900+05:30[Asia/Colombo]" }	

GET /authors/{id}/books	Get author's books by valid ID	GET		200 OK	[
					{
					"bookId": 104,
					"title": "Mistborn: The Final Empire",
					"authorId": 204,
					"isbn": "9780765311788",
					"publicationYear": 2006,
					"price": 13.99,
					"stock": 8
					},
					{
					"bookId": 105,
					"title": "The Way of Kings",
					"authorId": 204,
					"isbn": "9780765326355",
					"publicationYear": 2010,
					"price": 19.99,
					"stock": 5
					}
					]
						Pass

GET /authors/{id}/books	Get author's books by invalid ID	GET		404 Not Found	{
					"error": "Author Not Found",
					"message": "Author with ID 209 not found.",
					"path": "http://localhost:8080/Bookstore/rest/authors/209/books",
					"status": 404,
					"timestamp": "2025-04-25T16:53:02.430542+05:30[Asia/Colombo]"
					}
						Pass

Customer						
POST /customers	Create a customer with valid data.	POST	{ "customerId": 306, "firstName": "x", "lastName": "y",	201 Created	{ "customerId": 306, "firstName": "x", "lastName": "y", "email": "xy@gmail.com",	Pass

			"email": "xy@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }		"password": "emilyPass99", "cart": {}, "orders": [] }	
POST /customers	Create a customer with invalid data. (Invalid email)	POST	{ "customerId": 306, "firstName": "x", "lastName": "y", "email": "xygmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }	400 Bad Request	{ "error": "Invalid Input Exception", "message": "Invalid email format.", "path": "http://localhost:8080/Bookstore/rest/customers", "status": 404, "timestamp": "2025-04-26T10:40:16.971733900+05:30[Asia/Colombo]" }	Pass
GET /customers	Get all customers.	GET		200 OK	[{ "customerId": 301, "firstName": "John", "lastName": "Doe", "email": "john.doe@gmail.com", "password": "johnPass99", "cart": {}, "orders": [] }, { "customerId": 302, "firstName": "Alice", "lastName": "Smith", "email": "alice.smith@gmail.com", "password": "alicePass99", "cart": {}, "orders": [] }, { "customerId": 303,	Pass

					"firstName": "Bob", "lastName": "Johnson", "email": "bob.johnson@gmail.com", "password": "bobPass99", "cart": {}, "orders": [] }, { "customerId": 304, "firstName": "Emily", "lastName": "Davis", "email": "emily.davis@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }, { "customerId": 305, "firstName": "Michael", "lastName": "Brown", "email": "michael.brown@gmail.com", "password": "michaelPass99", "cart": {}, "orders": [] }, { "customerId": 306, "firstName": "x", "lastName": "y", "email": "xy@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }, { "customerId": 307, "firstName": "x", "lastName": "y",	

					"email": "xy@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }]	
GET /customers/{id}	Get customer by valid ID	GET		200 OK	{ "customerId": 301, "firstName": "John", "lastName": "Doe", "email": "john.doe@gmail.com", "password": "johnPass99", "cart": {}, "orders": [] }	Pass
GET /customers/{id}	Get customer by invalid ID	GET		404 Not Found	{ "error": "Invalid Input Exception", "message": "Invalid input: customerId must be an Integer.", "path": "http://localhost:8080/Bookstore/rest/customers/Q", "status": 404, "timestamp": "2025-04-25T16:56:27.043360200+05:30[Asia/Colombo]" }	Pass
PUT /customers/{id}	Update customer with valid data	PUT	{ "customerId": 301, "firstName": "test", "lastName": "Davis", "email": "emily.davis@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }	200 OK	{ "customerId": 301, "firstName": "test", "lastName": "Davis", "email": "emily.davis@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }	Pass

PUT /customers/{id}	Update customer with invalid data. (first name is a number)	PUT	{ "customerId": 302, "firstName": "123", "lastName": "Davis", "email": "emily.davis@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }	400 Bad Request	{ "error": "Invalid Input Exception", "message": "First name cannot be only numbers.", "path": "http://localhost:8080/Bookstore/rest/customers/302", "status": 400, "timestamp": "2025-04-26T10:41:34.040166400+05:30[Asia/Colombo]" }	Pass
PUT /customers/{id}	Update customer with non-existent ID	PUT	{ "customerId": 309, "firstName": "test", "lastName": "Davis", "email": "emily.davis@gmail.com", "password": "emilyPass99", "cart": {}, "orders": [] }	404 Not Found	{ "error": "Customer Not Found", "message": "Customer with ID 309 not found.", "path": "http://localhost:8080/Bookstore/rest/customers/309", "status": 404, "timestamp": "2025-04-25T16:59:53.788198+05:30[Asia/Colombo]" }	Pass
DELETE /customers/{id}	Delete a customer by valid ID	DELETE		204 No Content	204 No Content	Pass
DELETE /customers/{id}	Delete a customer by invalid ID	DELETE		404 Not Found	{ "error": "Customer Not Found", "message": "Customer with ID 309 not found.", "path": "http://localhost:8080/Bookstore/rest/customers/309", "status": 404, "timestamp": "2025-04-25T17:02:05.353864700+05:30[Asia/Colombo]" }	Pass
Cart						
POST /customers/{custo	Add to a cart with valid data.	POST	{ "bookId": 103, "quantity": 2	201 Created	{ "message": "Item added to cart"	Pass

merId}/cart/items			}		}	

POST /customers/{customerId}/cart/items	Add to a cart with invalid data (quantity is negative)	POST	{
			"bookId": 101,
			"quantity": -15
			}
				400 Bad Request	{
					"error": "Invalid Input Exception",
					"message": "Invalid input: quantity must be greater than zero.",
					"path": "http://localhost:8080/Bookstore/rest/customers/305/cart/items",
					"status": 400,
					"timestamp": "2025-04-26T10:47:40.546877900+05:30[Asia/Colombo]"
					}
						Pass

GET /customers/{customerId}/cart	Get cart by valid customer ID	GET		200 OK	{
					"103: The Name of the Wind (Author ID: 203) - $16.99 | Stock: 10": 0
					}
						Pass

GET /customers/{customerId}/cart	Get cart by invalid customer ID	GET		404 Not Found	{ "error": "Customer Not Found", "message": "Customer with ID 306 not found.", "path": "http://localhost:8080/Bookstore/rest/customers/306/cart", "status": 404, "timestamp": "2025-04-25T17:03:30.513278100+05:30[Asia/Colombo]"	Pass

PUT /customers/{customerId}/cart/items/{bookId}	Update cart by valid customer ID	PUT	{
			"quantity": 3
			}
				200 OK	{
					"message": "Cart item updated"
					}
						Pass

PUT /customers/{customerId}/c	Update cart by invalid customer ID	PUT	{
			"quantity": 3
			}
				404 Not Found	{
					"error": "Customer Not Found",
					"message": "Customer with ID 309 not found.",
						Pass

art/items/{bookId}					"path": "http://localhost:8080/Bookstore/rest/customers/309/cart/items/102", "status": 404, "timestamp": "2025-04-25T17:05:02.533019900+05:30[Asia/Colombo]" }	
PUT /customers/{customerId}/cart/items/{bookId}	Update cart by valid book ID	PUT	{ "quantity": 3 }	200 OK	{ "message": "Cart item updated" }	Pass
PUT /customers/{customerId}/cart/items/{bookId}	Update cart by invalid book ID	PUT	{ "quantity": 3 }	404 Not Found	{ "error": "Invalid Input Exception", "message": "Invalid input: bookId must be between 101 and 199.", "path": "http://localhost:8080/Bookstore/rest/customers/305/cart/items/500", "status": 404, "timestamp": "2025-04-25T17:06:08.243494300+05:30[Asia/Colombo]" }	Pass
PUT /customers/{customerId}/cart/items/{bookId}	Update cart with invalid data (negative quantity)	PUT	{ "quantity": -3 }	400 Bad Request	{ "error": "Invalid Input Exception", "message": "Invalid input: quantity must be greater than zero.", "path": "http://localhost:8080/Bookstore/rest/customers/305/cart/items/103", "status": 400, "timestamp": "2025-04-26T10:51:19.184949200+05:30[Asia/Colombo]" }	Pass

DELETE /customers/{customerId}/cart/items/{bookId}	Delete a book from cart by valid customer ID	DELETE		204 No Content	204 No Content	Pass
DELETE /customers/{customerId}/cart/items/{bookId}	Delete a book from cart by invalid customer ID	DELETE		404 Not Found	{ "error": "Customer Not Found", "message": "Customer with ID 309 not found.", "path": "http://localhost:8080/Bookstore/rest/customers/309/cart/items/103", "status": 404, "timestamp": "2025-04-25T21:36:29.932350+05:30[Asia/Colombo]" }	Pass
DELETE /customers/{customerId}/cart/items/{bookId}	Delete a book from cart by valid book ID	DELETE		204 No Content	204 No Content	Pass
DELETE /customers/{customerId}/cart/items/{bookId}	Delete a book from cart by invalid book ID	DELETE		404 Not Found	{ "error": "Invalid Input Exception", "message": "Invalid input: bookId must be between 101 and 199.", "path": "http://localhost:8080/Bookstore/rest/customers/305/cart/items/500", "status": 404, "timestamp": "2025-04-25T21:36:06.484021300+05:30[Asia/Colombo]" }	Pass
Order						
POST /customers/{custo	Add the cart to orders with valid customer ID	POST		201 Created	Order placed successfully	Pass

merId}/orders						
POST /customers/{customerId}/orders	Add the cart to orders with invalid customer ID	POST		404 Not Found	{ "error": "Customer Not Found", "message": "Customer with ID 309 not found.", "path": "http://localhost:8080/Bookstore/rest/customers/309/orders", "status": 404, "timestamp": "2025-04-25T17:09:16.816187400+05:30[Asia/Colombo]" }	Pass
POST /customers/{customerId}/orders	Add the cart to orders with invalid quantity	POST		400 Bad Request	{ "error": "Out Of Stock Exception", "message": "Book with ID 101 is out of stock.", "path": "http://localhost:8080/Bookstore/rest/customers/305/orders", "status": 400, "timestamp": "2025-04-25T17:10:35.405160600+05:30[Asia/Colombo]" }	Pass
GET /customers/{customerId}/orders	Get order by valid customer ID	GET		200 OK	[{ "103": 3 }]	Pass
GET /customers/{customerId}/orders	Get order by invalid customer ID	GET		404 Not Found	{ "error": "Customer Not Found", "message": "Customer with ID 309 not found.", "path": "http://localhost:8080/Bookstore/rest/customers/309/orders", "status": 404,	Pass

					"timestamp": "2025-04-25T17:11:18.032222500+05:30[Asia/Colombo]" }	

GET /customers/{customerId}/orders/{orderId}	Get order by valid customer ID	GET		200 OK	{
					"103": 3
					}
						Pass

GET /customers/{customerId}/orders/{orderId}	Get order by invalid customer ID	GET		404 Not Found	{
					"error": "Customer Not Found",
					"message": "Customer with ID 309 not found.",
					"path": "http://localhost:8080/Bookstore/rest/customers/309/orders/0",
					"status": 404,
					"timestamp": "2025-04-25T17:11:44.453585200+05:30[Asia/Colombo]"
					}
						Pass

GET /customers/{customerId}/orders/{orderId}	Get order by valid order ID	GET		200 OK	{
					"103": 3
					}
						Pass

GET /customers/{customerId}/orders/{orderId}	Get order by invalid order ID	GET		404 Not Found	Customer or order not found	Pass