Coursework Report

5COS022W - Client Server Architectures

Student Name: Mehmet Gilgil

Student ID: w2046446

Report

Contents

Introduction

This report documents the testing and validation of a RESTful API developed for a fictional Bookstore application. The primary goal of this project was to gain hands-on experience in designing and implementing RESTful web services using the JAX-RS framework in Java. The API supports core functionalities such as managing books, authors, customers, shopping carts, and orders, all structured around REST principles.

The application was tested using Postman, a popular tool for API development, which allowed for detailed and repeatable testing of each endpoint. All data was handled using JSON format, and application data was stored in-memory using Java collections such as ArrayList and HashMap. This report presents the results of those tests in a clear, structured, and tabular format, demonstrating the correctness, error handling, and reliability of the system.

Tests

Each table in this report is designed to clearly document the behavior of a specific API endpoint. The structure is consistent across all tables to make it easy to understand and compare results. For every endpoint, the table includes a description of the test case, the HTTP method used, the request body (if any), the expected status code and response, and the actual result observed during testing.

This format helps demonstrate not only that each endpoint works as intended with valid data, but also how the API handles edge cases and errors—such as invalid inputs or missing resources. By laying everything out in tables, it becomes much easier to verify that the API behaves correctly and reliably under different conditions.

Author Resource

Endpoint	Description	HTTP Method	Request Body (JSON)	Expected HTTP Status Code	Expected Response Body (JSON)	Actual Result (Pass / Fail)
/authors	Create author with valid details	POST	{"name": "John Smith", "biography": "John Smith's life"}	201 Created	{ "id": 3, "name": "John Smith", "biography": "John Smith's life" }	Pass
/authors	Create author with missing name	POST	{ "biography": "No name provided" }	400 Bad Request	Author needs to have a name and biography.	Pass
/authors	Get all authors	GET		200 OK	[{ "id": 1, "name": "John Doe", "biography": " John Doe's Life" }, { "id": 2, "name": "Mehmet Gilgil", "biography": " Mehmet's Story" }, { "id": 3, "name": "John Smith",	Pass

					"biography": "John Smith's life" }]	

/authors/1	Get author by valid ID	GET		200 OK	{
					"id": 1,
					"name": "John Doe",
					"biography": " John Doe's Life"
					}
						Pass

/authors/464	Get author by invalid ID	GET		404 Not Found	Author with ID 464 not found	Pass

/authors/books	Get books with valid author ID	GET		200 OK	[
					{
					"id": 2,
					"title": "Naruto",
					"author": {
					"id": 2,
					"name": "Mehmet Gilgil",
					"biography": " Mehmet's Story"
					},
					"isbn": "8677319356172",
					"publicationYear": 1996,
					"price": 9.5,
					"stockQuantity": 10
					}
					]
						Pass

/authors/books	Get books with invalid author ID	GET		200 OK	Author with ID 464 not found	Pass
/authors/1	Update existing author	PUT	{"name":"John Bob", "biography": "John Bob's Life"}	204 No Content		Pass
/authors/999	Update non-existent author	PUT	{"name":"John Bob", "biography": "John Bob's Life"}	404 Not Found	Author with ID 999 not found for update	Pass
/authors/1	Update existing author without passing name	PUT	{ "biography": "John Bob's Life"}	400 Bad Request	Please enter both name and biography for the update.	Pass

/authors/1	Delete author by valid ID	DELETE		204 No Content		Pass
/authors/464	Delete author with invalid ID	DELETE		404Not Found	Author with ID 464 not found for deletion	Pass

Book Resource

Endpoint	Description	HTTP Method	Request Body (JSON)	Expected HTTP Status Code	Expected Response Body (JSON)	Actual Result (Pass/ Fail)
/books	Create book with valid data	POST	{ "title": "1984", "author": { "id": 1 }, "isbn": "1234567890", "publicationYear": 1949, "price": 15.99, "stockQuantity": 10 }	201 Created	{ "id": 3, "title": "1984", "author": { "id": 1, "name": "John Doe", "biography": " John Doe's Life" }, "isbn": "1234567890", "publicationYear": 1949, "price": 15.99, "stockQuantity": 10 }	Pass
/books	Create book with missing title	POST	{ "author": { "id": 1 }, "isbn": "1234567890", "publicationYear": 1949, "price": 15.99, "stockQuantity": 10 }	400 Bad Request	Author, Title and ISBN are required.	Pass
/books	Create book with invalid future year	POST	{ "title": "Future Book", "author": { "id": 1 }, "isbn": "0987654321", "publicationYear": 2099, "price": 10, "stockQuantity": 5 }	400 Bad Request	Publication year cannot be in the future.	Pass
/books		POST	{ "title": "Negative", "author": { "id": 1 }, "isbn": "000111222",	400 Bad Request	Price cannot be negative	Pass

	Create book with negative price		"publicationYear": 2020, "price": -5, "stockQuantity": 5 }			
/books	Creating book with invalid author ID	POST	{ "title": "1984", "author": { "id": 66667 }, "isbn": "1234567890", "publicationYear": 1949, "price": 15.99, "stockQuantity": 10 }	404 Not Found	Author with ID 66667 not found.	Pass

/books	Get all books	GET		200 OK	[
					{
					"id": 1,
					"title": "One Piece",
					"author": {
					"id": 1,
					"name": "John Doe",
					"biography": "John Doe's Life"
					},
					"isbn": "9788420467283",
					"publicationYear": 1996,
					"price": 9.5,
					"stockQuantity": 100
					},
					{
					"id": 2,
					"title": "Naruto",
					"author": {
					"id": 2,
					"name": "Mehmet Gilgil",
					"biography": "Mehmet's Story"
					},
					"isbn": "8677319356172",
					"publicationYear": 1996,
					"price": 9.5,
					"stockQuantity": 10
					},
						Pass

					{
					"id": 3,
					"title": "1984",
					"author": {
					"id": 1,
					"name": "John Doe",
					"biography": "John Doe's Life"
					},
					"isbn": "1234567890",
					"publicationYear": 1949,
					"price": 15.99,
					"stockQuantity": 10
					}	

/books/1	Get book by valid ID	GET		200 OK	{
					"id": 1,
					"title": "One Piece",
					"author": {
					"id": 1,
					"name": "John Doe",
					"biography": "John Doe's Life"
					},
					"isbn": "9788420467283",
					"publicationYear": 1996,
					"price": 9.5,
					"stockQuantity": 100
					}
						Pass

/books/999	Get book by invalid ID	GET		404 Not Found	Book with ID 999 not found	Pass
/books/1	Update book with valid data	PUT	{ "title": "Updated Title", "author": { "id": 2 }, "isbn": "1111111111", "publicationYear": 2000, "price": 12.99, "stockQuantity": 5 }	204 No Content		Pass
/books/999	Update non-existent book	PUT	{ "title": "Updated Title", "author": { "id": 2 }, "isbn":	404 Not Found	Book with ID 999 not found for update	Pass

			"1111111111", "publicationYear": 2000, "price": 12.99, "stockQuantity": 5 }			
/books/1	Update book with invalid details (this case future publication year)	PUT	{ "title": "Updated Title", "author": { "id": 2 }, "isbn": "1111111111", "publicationYear": 2067, "price": 12.99, "stockQuantity": 5 }	400Bad Request	Publication year cannot be in the future.	Pass
/books/1	Delete book by valid ID	DELETE		204 No Content		Pass
/books/999	Delete book by invalid ID	DELETE		404 Not Found	Book with ID 999 not found for deletion	Pass

Customer Resource

Endpoint	Description	HTTP Method	Request Body (JSON)	Expected HTTP Status Code	Expected Response Body (JSON)	Actual Result (Pass / Fail)
/customers	Create customer with valid data	POST	{ "name": "Alice Smith", "email": "alice@example.com", "password": "pass1234" }	201 Created	{ "id": 3, "name": "Alice Smith", "email": "alice@example.com", "password": "pass1234" }	Pass
/customers	Create customer with missing fields	POST	{ "email": "incomplete@example.com", "password": "nopass" }	400 Bad Request	Customer must have a name, email and password.	Pass
/customers	Get all customers	GET		200 OK	[{ "id": 1, "name": "Lionel Messi", "email": "lm10@gmail.com", "password": "Messi123"	Pass

					},
					{
					"id": 2,
					"name": "Cristiano Ronaldo",
					"email": "cr7@gmail.com",
					"password": "Ronaldo123"
					},
					{
					"id": 3,
					"name": "Alice Smith",
					"email": "alice@example.com",
					"password": "pass1234"
					}
					]	

/customers/1	Get customer by valid ID	GET		200 OK	{
					"id": 1,
					"name": "Lionel Messi",
					"email": "lm10@gmail.com",
					"password": "Messi123"
					}
						Pass

/customers/999	Get customer by invalid ID	GET		404 Not Found	Customer with ID 999 not found	Pass
/customers/1	Update customer with valid data	PUT	{ "name": "Leo Messi", "email": "leo@example.com", "password": "newpass" }	204 No Content		Pass
/customers/999	Update non-existent customer	PUT	{ "name": "Ghost", "email": "ghost@example.com", "password": "ghostpass" }	404 Not Found	Customer with ID 999 not found for update	Pass
/customers/1	Update customer with invalid data	PUT	{ "name": "Leo Messi", "email": "leo@example.com" }	400 Bad Request	Customer must have a name, email and password.	Pass

/customers/1	Delete customer by valid ID	DELETE		204 No Content		Pass
/customers/999	Delete customer by invalid ID	DELETE		404 Not Found	Delete customer by valid ID	Pass

Cart Resource

Endpoint	Description	HTTP Method	Request Body (JSON)	Expected HTTP Status Code	Expected Response Body (JSON)	Actual Result (Pass/ Fail)
/customers/1/ cart/items	Add valid item to cart	POST	{ "bookId": 2, "quantity": 2 }	201 Created	{ "bookId": 2, "quantity": 2 }	Pass
/customers/1/ cart/items	Add item with invalid book ID	POST	{ "bookId": 999, "quantity": 1 }	400 Bad Request	Book with ID 999 not found or quantity is invalid.	Pass
/customers/1/ cart/items	Add item with out-of-stock quantity	POST	{ "bookId": 2, "quantity": 9999 }	400 Bad Request	Not enough stock for book with ID: 2	Pass
/customers/999/cart/items	Add item for non-existent customer	POST	{ "bookId": 2, "quantity": 1 }	404 Not Found	Customer with ID 999 not found.	Pass
/customers/1/ cart	Get cart contents for valid customer	GET		200 OK	[{ "bookId": 1, "quantity": 10 }, { "bookId": 2, "quantity": 2 }]	Pass
/customers/999/cart/	Get cart for invalid customer	GET		404 Not Found	Customer with ID 999 not found.	Pass
/customers/1/ cart/1	Update quantity of existing book in cart	PUT	{ "quantity": 3 }	204 No Content		Pass

/customers/1/cart/999	Update book that's not in the cart	PUT	{ "quantity": 3 }	404 Not Found	Book with ID 999 not found in the cart for update.	Pass
/customers/999/cart/1	Update cart for non-existent customer	PUT	{ "quantity": 3 }	404 Not Found	Customer with ID 999 not found.	Pass
/customers/1/cart/1	Update item with out-of-stock quantity	PUT	{ "quantity": 1000}		Not enough stock for book with ID: 1	Pass
/customers/1/cart/1	Delete item from cart	DELETE		204 No Content		Pass
/customers/1/cart/999	Delete non-existent book from cart	DELETE		404 Not Found	Book with ID 999 not found in the cart for deletion.	Pass
/customers/999/cart/1	Delete item for invalid customer	DELETE		404 Not Found	Customer with ID 999 not found.	Pass

Order Resource

Endpoint	Description	HTTP Method	Request Body (JSON)	Expected HTTP Status Code	Expected Response Body (JSON)	Actual Result (Pass/Fail)
/customers/1/orders	Place order with valid cart	POST		201 Created	{ "id": 2, "customerId": 1, "cart": [{ "bookId": 1, "quantity": 10 }], "total": 95.0 }	Pass

/customers/999/orders	Place order for invalid customer	POST		404 Not Found	Customer with ID 999 not found.	Pass
/customers/2/orders	Place order with empty cart	POST		400 Bad Request	Cart is empty or missing for customer ID 2	Pass

/customers/1/orders	Get all orders for a valid customer	GET		200 OK	[
					{
					"id": 1,
					"customerId": 1,
					"cart": [
					{
					"bookId": 1,
					"quantity": 10
					}
					],
					"total": 95.0
					},
					{
					"id": 2,
					"customerId": 1,
					"cart": [
					{
					"bookId": 1,
					"quantity": 10
					}
					],
					"total": 95.0
					}
					]
						Pass

/customers/999/orders	Get orders for non-existent customer	GET		404 Not Found	Customer with ID 999 not found.	Pass

/customers/1/orders/1	Get order by valid ID	GET		200 OK	{
					"id": 1,
					"customerId": 1,
					"cart": [
					{
					"bookId": 1,
					"quantity": 10
					}
					],
					"total": 95.0
					}
						Pass

/customers/1/orders/999	Get order by invalid ID	GET		404 Not Found	Order with ID 999 not found for customer ID 1	Pass

customers/999/orders/1	Customer with ID 999 not found.	GET		404 Not Found	Customer with ID 999 not found.	Pass