Student Number: W20457325
Student Name: Sikandar Ali

5COSC022W Client Server Architecture Coursework

Introduction

This report outlines the testing process for the RESTful Bookstore API I developed as part of my Client-Server Architectures coursework. The purpose of this report is to clearly document the test cases I designed and executed in order to validate the functionality and robustness of the API. The API was built using Java, utilising the JAX-RS framework (Jersey implementation) to create RESTful web services. JSON was used throughout for data exchange between client and server. To thoroughly test the API endpoints, I used Postman for API development and testing, which allowed me to simulate real-world API interactions easily and efficiently.
This report focuses on presenting the test cases in a clear and structured table format, organised by each resource class. Both positive and negative test scenarios are included, ensuring that the API behaves correctly under normal conditions and handles errors normally when something goes wrong.
Building and testing this API has been an enjoyable learning experience, as it allowed me to strengthen my understanding of RESTful design principles, exception handling, and the importance of thorough testing in software development. This coursework has given an insight into real world scenarios, and I have learned new skills which have been really helpful.

POSTMAN TEST CASE TEMPLATES
BookResource TestCases

Description	**HTTP Method**	**Endpoint**	**Request Body**	**Expected HTTP Status**	**Expected Response Body(JSON)**	**Actual Status**	**Actual Response**	**TEST: PASS/ FAIL**
Create a valid book	POST	/api/books	{ "title": "Why We Sleep", "authorId": 1, "isbn": "9780141983769", "publicationYear": 2017, "price": 9.95, "stock": 100 }	201 Created	JSON of created book with generated id	201 Created New book has been successfully created and stored	{ "authorId": 1, "id": 1, "isbn": "9780141983769", "price": 9.95, "publicationYear": 2017, "stock": 100,	PASS

Student Number: W20457325
Student Name: Sikandar Ali

							"title": "Why We Sleep" }	
Create a book with invalid input – price is negative	POST	/api/books	{ "title": "Some Book", "price": -50.0, "stock": 10, "authorId": 1 }	400 error	{ "error": "Author Not Found", "message": "..." }	400 Bad Request	{ "error": "Invalid Input", "message": "Book price must be a positive value." }	PASS
Get All books	GET	/api/books		200 OK	Array of books	200 Request successful retrieving all books	[{ "authorId": 1, "id": 1, "isbn": "9780141983769", "price": 9.95, "publicationYear": 2017, "stock": 100, "title": "Why We Sleep" }]	PASS

Student Number: W20457325
Student Name: Sikandar Ali

Get book by valid id	GET	/api/books/1		200 OK	JSON of book with id =1	200 OK Request successful retrieving book with id 1	{
							"authorId": 1,
							"id": 1,
							"isbn": "9780141983769",
							"price": 9.95,
							"publicationYear": 2017,
							"stock": 100,
							"title": "Why We Sleep"
							}
								PASS

Get book by invalid id	GET	/api/books/999		404 Not found	{“error” : Book Not Found, “message”: “”}	404 Not Found	{
							"error": "Book Not Found",
							"message": "Book with ID 999 not found."
							}
								PASS

Update book with valid id	PUT	/api/books1	{ "title": "The Hobbit", "authorId": 1, "isbn": "978-0-00-752549-2", "publicationYear": 1937,	200 OK	Updated book JSON	200 OK Book ID 1 is now updated to The Hobbit Book	{ "authorId": 1, "id": 1, "isbn": "978-0-00-752549-2",	PASS

Student Number: W20457325
Student Name: Sikandar Ali

			"price": 15.99, "stock": 90 }			instead now	"price": 150.99, "publicationYear": 1937, "stock": 90, "title": "The Hobbit Updated" }	
Delete book by valid id	DELETE	/api/books/1		204 No content – book becomes deleted	Empty Response	204 No Content – This means the deletion was successful and now book id 1 is now empty	No Content returned – this is now passed	PASS
Delete book by invalid id	DELETE	/api/books/999		404 Not Found	{“error” : “Book Not Found”, “message”: “…”}	404 Not Found because the book does not exist to begin with so we can not delete it .	{ "error": "Book Not Found", "message": "Book with ID 999 not found." }	PASS

Student Number: W20457325
Student Name: Sikandar Ali

AuthorResource Tests

Description	HTTP Method	Endpoint	Request Body	Expected HTTP Status	Expected Response Body(JSON)	Actual Status	Actual Response	TEST: PASS/FAIL
Create a valid author	POST	/api/authors	{ "name": "J.R.R. Tolkien", "biography": "Author of The Hobbit." }	201 Created	JSON of created author	201 Created	{ "biography": "Author of The Hobbit.", "id": 1, "name": "J.R.R. Tolkien" }	PASS
Get all authors	GET	/api/authors	-	200 OK	Array of authors	200 OK – gives back array of authors	[{ "biography": "Author of The Hobbit.", "id": 1, "name": "J.R.R. Tolkien" }]	PASS
Get author by valid id	GET	/api/authors/1	-	200 OK	JSON of author	200 OK	{ "biography": "Author	PASS

Student Number: W20457325
Student Name: Sikandar Ali

							<pre>of The Hobbit.", "id": 1, "name": "J.R.R. Tolkien" }</pre>	
Get author by invalid id	GET	/api/authors/999	-	404 Not Found	{ "error": "Author Not Found", "message": "..." }	404 Not Found	<pre>{ "error": "Author Not Found", "message": "Author with id:999not found" }</pre>	PASS
Update author by valid id	PUT	/api/authors/1	{ "name": "John Tolkien", "biography": "Updated Bio." }	200 OK	Updated author JSON	200 ok	<pre>{ "biography": "Updated Bio.", "id": 1, "name": "John Tolkien" }</pre>	PASS
Delete author by valid id	DELETE	/api/authors/1	-	204 No Content	Empty	204 No Content	Empty Body – (Item has been successfully deleted)	PASS

Student Number: W20457325
Student Name: Sikandar Ali

CustomerResource Tests

Description	HTTP Method	Endpoint	Request Body	Expected HTTP Status	Expected Response Body(JSON)	Actual Status	Actual Response	TEST: PASS/ FAIL
Create valid customer	POST	/api/customers	{ "name": "Sikandar", "surname": "Ali", "email": "randomemail@email.com", "address": "address 123", "password": "Random5678" }	201 Created	Customer JSON	201 Created	{ "address": "address 123", "email": "randomemail@email.com", "id": 1, "name": "Sikandar", "password": "Random5678", "surname": "Ali" }	PASS
Get all customers	GET	/api/customers	-	200 OK	Array of customers	200 OK	[{ "address": "address 123", "email": "randomemail@email.com", "id": 1,	PASS

Student Number: W20457325
Student Name: Sikandar Ali

							"name": "Sikandar", "password": "Random5678", "surname": "Ali" }]	
Get customer by valid id	GET	/api/customers/1	-	200 OK	JSON of customer	200 OK	{ "address": "address 123", "email": "randomemail@email.com", "id": 1, "name": "Sikandar", "password": "Random5678", "surname": "Ali" }	PASS
Get customer by invalid id	GET	/api/customers/999	-	404 Not Found	{ "error": "Customer Not Found", "message": "..." }	404 Not Found	{ "error": "Customer Not Found", "message": "Customer with ID999 not found" }	PASS
Update custo	PUT	/api/customers/1	{ "name": "Sikandar Updated", "surname":	200 OK	Updated custom	200 OK	{ "address":	PASS

Student Number: W20457325
Student Name: Sikandar Ali

mer by id			"Ali", "email": "newemail@random.com", "address": "Tottenham Court Road", "password": "newpass" }		er JSON		"Tottenham Court Road", "email": "newemail@random.com", "id": 1, "name": "Sikandar Updated", "password": "newpass", "surname": "Ali" }	
Delete customer	DELETE	/api/customers/1	-	204 No Content	Empty	204 No Content	Empty Body- successful deletion of customer with id 1	PASS

CartResource Tests

Description	**HTTP Method**	**Endpoint**	**Request Body**	**Expected HTTP Status**	**Expected Response Body(JSON)**	**Actual Status**	**Actual Response**	**TEST: PASS/ FAIL**
Add item to cart	POST	/api/customers/1/cart/items	{ "bookId": 3, "quantity": 2 }	201 Created	CartItem JSON	201 Created	{ "bookId": 1, "quantity": 2 }	PASS
View cart	GET	/api/customers/1/cart	-	200 OK	List of CartItems	200 OK	{	PASS

Student Number: W20457325
Student Name: Sikandar Ali

							<pre>"bookId": 1,
							"quantity": 2
							}</pre>	

Update quantity of cart item	PUT	/api/customers/1/cart/items/1	{ "bookId": 1, "quantity": 5 }	200 OK	Updated CartItem JSON	200 OK	<pre>{
							"bookId": 1,
							"quantity": 5
							}</pre>
								PASS

Remove item from cart	DELETE	/api/customers/1/cart/items/1	-	204 No Content	Empty	204 No Content	Empty Body	PASS

OrderResource Tests

Description	**HTTP Method**	**Endpoint**	**Request Body**	**Expected HTTP Status**	**Expected Response Body(JSON)**	**Actual Status**	**Actual Response**	**TEST: PASS/ FAIL**

Place order from cart	POST	/api/customers/1/orders	-	201 Created	Order JSON	201 Created	<pre>{
							"cartItems": [
							{
							"bookId": 1,
							"quantity": 2
							}
							],
							"customerId": 1,</pre>
								PASS

Student Number: W20457325
Student Name: Sikandar Ali

							<pre>"id": 1, "orderDate": "2025-04-28T04:25:31.809702", "totalPrice": 301.98 }</pre>	
View all orders	GET	/api/customers/1/orders	-	200 OK	Array of orders	200 OK	<pre>[{ "cartItems": [{ "bookId": 1, "quantity": 2 }], "customerId": 1, "id": 1, "orderDate": "2025-04-28T04:25:31.809702", "totalPrice": 301.98 }]</pre>	PASS
Get specific order	GET	/api/customers/1/orders/1	-	200 OK	Order JSON	200 OK	<pre>{</pre>	PASS

Student Number: W20457325
Student Name: Sikandar Ali

							"cartItems": [{ "bookId": 1, "quantity": 2 }], "customerId": 1, "id": 1, "orderDate": "2025-04-28T04:25:31.809702", "totalPrice": 301.98 }	
Place order from empty cart	POST	/api/customers/1/orders	-	404 Not Found	{ "error": "Cart Not Found", "message": "" }	404 Not Found	{ "error": "Cart Not Found", "message": "Cart is empty or does not exist for customer ID: 1" }	PASS
Get non-existent order	GET	/api/customers/1/orders/999	-	404 Not Found	{ "error": "Order Not Found"	404 Not Found	{ "error": "Item is	PASS

Student Number: W20457325
Student Name: Sikandar Ali

					, "messa ge": "..." }	Order Not found message proving that the order does not exist.	out of Stock", "message": "Order not found." }	