 

Informatics Institute of Technology

University of Westminster

Client Server Architecture Coursework

5COSC022C

Student Name- Gani Achchige Lakindu Chandira Perera

Module Code- 5COSC022C

IIT Student ID- 20231153

UOW ID- W2051869

# **Declaration**

confirm that I understand what plagiarism, collusion, and contract cheating are, and I have read and understood the section on Assessment Offences in the *Essential Information for Students*. I declare that the work I have submitted is entirely my own, and any work taken from other sources has been properly referenced and acknowledged.

Signed: \_\_\_\_\_****\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_22/04/2025\_\_\_\_

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **Table Of Content**

[**Declaration**](#_o61kz1n43t3z) **1**

[**Table Of Content**](#_br1karwyt5g9) **1**

[**1.Introduction**](#_8g0q19khcd27) **2**

[**2. API Endpoint Test Case Tables**](#_v4z5xln1ye7v) **2**

[2.1 AuthorResource (/authors)](#_2w6s967sv3ba) 2

[2.1.1 POST /authors Endpoint](#_cxrwox3sib84) 3

[2.1.2 GET /authors Endpoint](#_7bv6hz74q8w1) 3

[2.1.3 GET /authors/{id} Endpoint](#_3tp0hgg1ke5k) 5

[2.1.4 PUT /authors/{id}](#_do0vrgnal59y) 6

[2.1.4 DELETE /authors/{id}](#_n9fhpqvngnpp) 7

[2.1.5 GET /authors/{id}/books](#_9kgh9aqb8t3y) 7

[2.2 BookResourse (/books)](#_hsf7m7j1veaq) 8

[2.2.1 POST /books Endpoint](#_296car25o1i3) 8

[2.2.2 GET /books Endpoint](#_a1su7cayl3jm) 9

[2.2.3 GET /books/{id} Endpoint](#_1j7f5lxzs04a) 10

[2.2.4 PUT /books/{id} Endpoint](#_s3cdbeypx0i9) 11

[2.2.5 DELETE /books/{id} Endpoint](#_pq9h8iazqk3n) 12

[2.3 CustomerResource (/customers)](#_uhtvdum1tlo0) 12

[2.3.1 POST /customers Endpoint](#_1d726846ucdz) 12

[2.3.2 GET /customers](#_bhqmhms6l6sw) 13

[2.3.3 GET /customers/{id}](#_au31t86ujboq) 14

[2.3.4 PUT /customers/{id}](#_8t32duqd6v45) 15

[2.3.5 DELETE /customers/{id}](#_2teal4285xbc) 16

[2.4 CartResource (/customers/{customerId}/cart)](#_mx7bqmigw90j) 16

[2.4.1 POST /customers/{customerId}/cart/items](#_4jlgy81fju6s) 16

[2.4.2 GET /customers/{customerId}/cart](#_pu7y1x78m96x) 17

[2..PUT /customers/{customerId}/cart/items/{bookId}](#_oaijznety62o) 18

[2.4.4 DELETE /customers/{customerId}/cart/items/{bookId}](#_ix3e5vz9jo5o) 19

[2.5 OrderResource (/customers/{customerId}/orders)](#_9l35e8yq4ld7) 19

[2.5.1 POST /customers/{customerId}/orders](#_1d8ii5ehjzu8) 19

[2.5.2 GET /customers/{customerId}/orders](#_wi6vr3x3s1tp) 20

[2.5.3 GET /customers/{customerId}/orders/{orderId}](#_599l5dpxntbf) 20

# 

# 

# 

# 

# **1.Introduction**

This report documents the comprehensive test cases designed for the RESTful Bookstore API. The API has been implemented using JAX-RS with JSON as the data interchange format for requests and responses. All testing was conducted using Postman to validate the functionality of each endpoint.

The Bookstore API manages books, authors, customers, shopping carts, and orders through a series of interconnected resources. Backend data is maintained using in-memory data structures (ArrayList, HashMap) for simplicity and educational purposes. The test cases presented in this report are organized in a tabular format, systematically covering all API endpoints with positive, negative, and edge cases to ensure complete functionality validation.

# 2. API Endpoint Test Case Tables

#### 

## **2.1 AuthorResource (/authors)**

### **2.1.1 POST /authors Endpoint**

### 

| **Endpoint** | **Description** | **HTTP Method** | **Request Body(JSON)** | **Expected HTTP status Code** | **Expected Response Body (JSON)** | **Actual Result** |
| --- | --- | --- | --- | --- | --- | --- |
| /authors | Create author | POST | {  "firstName":"Stephen",  "lastName": "King",  "biography": "American author of horror, supernatural fiction, suspense, and fantasy novels"  } | 201 Created | {  "id": 4,  "firstName" : "Stephen",  "lastName" : "King",  "Biography" : "American author of horror, supernatural fiction, suspense, and fantasy novels",  "fullName" : "Stephen King"  } | pass |
| /authors | Create author with Missing name | POST | {  "biography": "Updated biography for J.K. Rowling"  } | 400 Bad Request | {  "status": 400,  "message": "Author first name and last name are required",  "timestamp": 1745215673527  } | pass |

### 

### **2.1.2 GET /authors Endpoint**

### 

| **Endpoint** | **Description** | **HTTP Method** | **Request Body(JSON)** | **Expected HTTP status Code** | **Expected Response Body (JSON)** | **Actual Result** |
| --- | --- | --- | --- | --- | --- | --- |
| /authors | Retrieve all Authors | GET | - | 200 OK | [  {  "id": 1,  "firstName": "J.K.",  "lastName": "Rowling",  "biography": "British author best known for the Harry Potter series",  "fullName": "J.K. Rowling"  },  {  "id": 2,  "firstName": "George",  "lastName": "Orwell",  "biography": "English novelist and essayist, journalist and critic",  "fullName": "George Orwell"  },  {  "id": 3,  "firstName": "Harper",  "lastName": "Lee",  "biography": "American novelist widely known for her novel To Kill a Mockingbird",  "fullName": "Harper Lee"  },  {  "id": 4,  "firstName": "Stephen",  "lastName": "King",  "biography": "American author of horror, supernatural fiction, suspense, and fantasy novels",  "fullName": "Stephen King"  }  ] | pass |

### **2.1.3 GET /authors/{id} Endpoint**

### 

| **Endpoint** | **Description** | **HTTP Method** | **Request Body(JSON)** | **Expected HTTP status Code** | **Expected Response Body (JSON)** | **Actual Result** |
| --- | --- | --- | --- | --- | --- | --- |
| /authors/4 | Retrieve the author by valid id | GET | - | 200 OK | {"id": 4,  "firstName": "Stephen",  "lastName": "King",  "biography": "American author of horror, supernatural fiction, suspense, and fantasy novels",  "fullName": "Stephen King"  } | pass |
| /authors/ 100 | Retrieve the author by invalid id | GET |  | 404 Not Found | {  "status": 404,  "message": "Author with ID 100 not found",  "timestamp": 1745216350414  } | pass |

#### 

### 

### 

### **2.1.4 PUT /authors/{id}**

### 

| **Endpoint** | **Description** | **HTTP Method** | **Request Body(JSON)** | **Expected HTTP status Code** | **Expected Response Body (JSON)** | **Actual Result** |
| --- | --- | --- | --- | --- | --- | --- |
| /authors/4 | Update the author info | PUT | {  "id": 4,  "firstName": "Elena",  "lastName": "Starbrook",  "biography": "A reclusive sci-fi author from Greenland known for her quantum-inspired plots and fictional languages.",  "fullName": "Elena Starbrook"  } | 200 OK | {  "id": 4,  "firstName": "Elena",  "lastName": "Starbrook",  "biography": "A reclusive sci-fi author from Greenland known for her quantum-inspired plots and fictional languages.",  "fullName": "Elena Starbrook"  } | pass |
| /author/ 100 | Update the non exist author | PUT | {"id": 4,  "firstName": "Elena",  "lastName": "Starbrook",  "biography": "A reclusive sci-fi author from Greenland known for her quantum inspired plots and fictional languages.",  "fullName": "Elena Starbrook"  } | 404 Not Found | { "status": 404,  "message": "Author with ID 100 not found",  "timestamp": 1745216783952  } | pass |

### 2.1.4 **DELETE /authors/{id}**

### 

| **Endpoint** | **Description** | **HTTP Method** | **Request Body(JSON)** | **Expected HTTP status Code** | **Expected Response Body (JSON)** | **Actual Result** |
| --- | --- | --- | --- | --- | --- | --- |
| /authors/4 | Delete the valid author | DELETE |  | 204 No Contet |  | pass |
| /authors/ 100 | Delete the invalid author | DELETE | - | 404 Not Found | {  "status": 404,  "message": "Author with ID 100 not found",  "timestamp": 1745217233409  } | pass |

### **2.1.5 GET /authors/{id}/books**

| **Endpoint** | **Description** | **HTTP Method** | **Request Body(JSON)** | **Expected HTTP status Code** | **Expected Response Body (JSON)** | **Actual Result** |
| --- | --- | --- | --- | --- | --- | --- |
| /authors/ 1/ books | retrieve the valid authors books | GET |  | 200 OK | [  {  "id": 1,  "title": "Harry Potter and the Philosopher's Stone",  "authorId": 1,  "isbn": "9780747532743",  "publicationYear": 1997,  "price": 19.99,  "stock": 50  },  {  "id": 2,  "title": "Harry Potter and the Chamber of Secrets",  "authorId": 1,  "isbn": "9780747538486",  "publicationYear": 1998,  "price": 19.99,  "stock": 40  }  ] | pass |
| /author | Retrieve the invalid author books | GET | - | 404 Not Found | {  "status": 404,  "message": "Author with ID 100 not found",  "timestamp": 1745217719418  } | pass |

## 2.2 BookResourse (/books)

### 2.2.1 POST /books Endpoint

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /books | Create a book with valid data | POST | {  "title":"The Shining",  "authorId":5,  "isbn":"9780307743657",  "publicationYear": 1977,  "price":14.99,  "stock": 30  } | 201 Created | {  "id": 6,  "title": "The Shining",  "authorId": 5,  "isbn": "9780307743657",  "publicationYear": 1977,  "price": 14.99,  "stock": 30  } | 201 Created |
| /books | Create book with invalid author id | POST | {  "title": "The Shining",  "authorId": 120,  "isbn": "9780307743657",  "publicationYear": 1977,  "price": 14.99,  "stock": 30  } | 404 Not Found | {  "status": 404,  "message": "Cannot add book. Author with ID 120 does not exist",  "timestamp": 1745219431084  } | pass |
| /books | Create with future publication year | POST | {  "title": "The Shining",  "authorId":5,  "isbn": "9780307743657",  "publicationYear": 2050,  "price": 14.99,  "stock": 30  } | 400 Bad Request | {  "status": 400,  "message": "Publication year cannot be in the future",  "timestamp": 1745219579749  } | pass |

### 2.2.2 GET /books Endpoint

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /books | Get all books | GET | - | 200 OK | [  {  "id": 1,  "title": "Harry Potter and the Philosopher's Stone",  "authorId": 1,  "isbn": "9780747532743",  "publicationYear": 1997,  "price": 19.99,  "stock": 50  },  {  "id": 2,  "title": "Harry Potter and the Chamber of Secrets",  "authorId": 1,  "isbn": "9780747538486",  "publicationYear": 1998,  "price": 19.99,  "stock": 40  }, ….  ] | pass |

### **2.2.3 GET /books/{id} Endpoint**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /books/6 | Get valid book | GET | - | 200 OK | {  "id": 6,  "title": "The Shining",  "authorId": 5,  "isbn": "9780307743657",  "publicationYear": 1977,  "price": 14.99,  "stock": 30  } | pass |
| /books /120 | Get invalid book | GET | - | 404 Bad Request | {  "status": 404,  "message": "Book with ID 120 not found",  "timestamp": 1745219967821  } | pass |

### 2.2.4 PUT /books/{id} Endpoint

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /books/ 1 | Update book with valid data | PUT | {  "title": "Harry Potter and the Philosopher's Stone",  "authorId": 1,  "isbn": "9780747532743",  "publicationYear": 1997,  "price": 24.99,  "stock": 100  } | 200 OK | {  "id": 1,  "title": "Harry Potter and the Philosopher's Stone",  "authorId": 1,  "isbn": "9780747532743",  "publicationYear": 1997,  "price": 24.99,  "stock": 100  } | pass |
| /books/ 120 | Update non exist book | PUT | {  "title": "Harry Potter and the Philosopher's Stone",  "authorId": 1,  "isbn": "9780747532743",  "publicationYear": 1997,  "price": 24.99,  "stock": 100  } | 404 Not Found | {  "status": 404,  "message": "Book with ID 120 not found",  "timestamp": 1745220202736  } | pass |

### **2.2.5 DELETE /books/{id} Endpoint**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /books/5 | Delete book by valid id | DELETE | - | 200 No Content | - | pass |
| /books /120 | Delete book by invalid id | DELETE | - | 404 Not Found | {  "status": 404,  "message": "Book with ID 120 not found",  "timestamp": 1745220384019  } | pass |

## 

## 2.3 CustomerResource (/customers)

### **2.3.1 POST /customers Endpoint**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body(JSON)** | **Expected HTTP Status code** | **Expected Response Body** | **Actual Result** |
| --- | --- | --- | --- | --- | --- | --- |
| /customers | Create a customer | POST | {  "firstName": "Pulitha",  "lastName": "Amarasignha",  "email": "pulitha45@example.com",  "password": "securepass123"  } | 201 Created | {  "id": 3,  "firstName": "Pulitha",  "lastName": "Amarasignha",  "email": "pulitha45@example.com",  "password": "securepass123",  "fullName": "Pulitha Amarasignha"  } | pass |
| /customers | Create customer missing password | POST | {  "firstName": "Pulitha",  "lastName": "Amarasignha",  "email": "pulitha45@example.com"  } | 400 Bad Request | {  "status": 400,  "message": "Customer password is required",  "timestamp": 1745221433895  } | pass |

#### 

### **2.3.2 GET /customers**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /customers | Get all customer | GET |  | 200 OK | [  {  "id": 1,  "firstName": "Lakindu",  "lastName": "Perera",  "email": "lakinduperera123@gmail.com",  "password": "laki123",  "fullName": "Lakindu Perera"  },  {  "id": 2,  "firstName": "Janidu",  "lastName": "Costa",  "email": "janiducosta321@gmail.com",  "password": "jani123",  "fullName": "Janidu Costa"  },  {  "id": 3,  "firstName": "Pulitha",  "lastName": "Amarasignha",  "email": "pulitha45@example.com",  "password": "securepass123",  "fullName": "Pulitha Amarasignha"  }  ] | pass |

#### 

### **2.3.3 GET /customers/{id}**

| **EndPoint** | **Description** | **HTTP Method** | * **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /customers/3 | Get customer by valid id | GET |  | 200 OK | {  "id": 3,  "firstName": "Pulitha",  "lastName": "Amarasignha",  "email": "pulitha45@example.com",  "password": "securepass123",  "fullName": "Pulitha Amarasignha"  } | pass |

#### 

#### 

#### 

#### 

### **2.3.4 PUT /customers/{id}**

#### 

| **EndPoint** | **Description** | **HTTP Method** | * **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /customers/2 | Update the customer info | PUT | {  "firstName": "Johnny",  "lastName": "Depp",  "email": "johnny@example.com",  "password": "newpassword456"  } | 200 OK | {  "id": 2,  "firstName": "Johnny",  "lastName": "Depp",  "email": "johnny@example.com",  "password": "newpassword456",  "fullName": "Johnny Depp"  } | pass |
| /customer/120 | Update the customer by invalid id | PUT | {  "firstName": "Johnny",  "lastName": "Depp",  "email": "johnny@example.com",  "password": "newpassword456"  } | 404 NOT Found | {  "status": 404,  "message": "Customer with ID 120 not found",  "timestamp": 1745221903622  } | pass |

#### 

### **2.3.5 DELETE /customers/{id}**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| customer/2 | Delete customer by valid id | DELETE | - | 204 No Content | - | pass |
| Customer /120 | Delete cusotmer by invalid id | DELETE | - | 404 Not Found |  | pass |

#### 

#### 

#### 

#### 

#### 

#### 

## 2.4 CartResource (/customers/{customerId}/cart)

### **2.4.1 POST /customers/{customerId}/cart/items**

| EndPoint | Description | HTTP Method | Request body(JSON) | Expected HTTP Status Code | Expected Response Body (JSON) | Actual Result |
| --- | --- | --- | --- | --- | --- | --- |
| /customers/1/cart/items | Add valid book to cart | POST | {  "bookId": 6,  "quantity": 2  } | 201 Created | {  "bookId": 6,  "quantity": 2  } | pass |
| /customers/1/cart/items | Add out of stock book | POST | {  "bookId": 6,  "quantity": 1000  } | 404 Not Found | {  "status": 400,  "message": "Book with ID 6 has insufficient stock. Requested: 1000, Available: 30",  "timestamp": 1745232787059  } | pass |
| /customers /120/cart/items | Invalid customer id | POST | {  "bookId": 6,  "quantity": 2  } | 404 Not Found | {  "status": 404,  "message": "Customer with ID 120 not found",  "timestamp": 1745232884734  } | pass |

### **2.4.2 GET /customers/{customerId}/cart**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| /customers/1/cart/items | View customer cart with items | GET |  | 200 Ok | [  {  "bookId": 4,  "quantity": 5  },  {  "bookId": 6,  "quantity": 2  }  ] | pass |
| /customers /100/cart/items | Invalid customer id | GET |  | 404 Not Found | {  "status": 404,  "message": "Customer with ID 100 not found",  "timestamp": 1745233352891  } | pass |

### 2..PUT /customers/{customerId}/cart/items/{bookId}

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| customers/1/cart/items/6 | Update the quantity | PUT | {  "quantity": 6  } | 200 OK | {  "bookId": 6,  "quantity": 6  } | pass |
| customers/1/cart/items/6 | Invalid quantity (negative) | PUT | {  "quantity": -5  } | 400 Bad Request | {  "status": 400,  "message": "Valid quantity is required",  "timestamp": 1745233776888  } | pass |
| customers /120/cart/items/6 | Invalid customer | PUT | {  "quantity": 6  } | 404 Not Found | {  "status": 404,  "message": "Customer with ID 120 not found",  "timestamp": 1745233957228  } | pass |

### 2.4.4 DELETE /customers/{customerId}/cart/items/{bookId}

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| customers/1/cart/items/4 | Delete the cart | DELETE |  | 204 No Content |  | pass |

## 2.5 OrderResource (/customers/{customerId}/orders)

### 2.5.1 POST /customers/{customerId}/orders

| Endpoint | Description | HTTP Method | Request Body (JSON) | Expected HTTP status code | Expected responses body | Actual Result |
| --- | --- | --- | --- | --- | --- | --- |
| Customers /1/orders | Place a order from cart | POST |  | 201 Created | {  "id": 1,  "customerId": 1,  "items": [  {  "bookId": 6,  "quantity": 6  }  ],  "totalPrice": 89.94  } | pass |
| Customers /2/orders | Place order with empty cart | POST |  | 400 Not Found | {  "status": 404,  "message": "Customer with ID 2 not found",  "timestamp": 1745239242635  } | pass |
| Customers /100/orders | Invalid customer id | POST |  | 404 Not Found | {  "status": 404,  "message": "Customer with ID 100 not found",  "timestamp": 1745239329258  } | pass |

### **2.5.2 GET /customers/{customerId}/orders**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| Customers /1/orders | View all orders of specific customer | GET |  | 200 OK | [  {  "id": 1,  "customerId": 1,  "items": [  {  "bookId": 6,  "quantity": 6  }  ],  "totalPrice": 89.94  }  ] | pass |
| Customers /1/orders/1 | Valid order retrievel | GET |  | 200 OK | {  "id": 1,  "customerId": 1,  "items": [  {  "bookId": 6,  "quantity": 6  }  ],  "totalPrice": 89.94  } | pass |

### 

### **2.5.3 GET /customers/{customerId}/orders/{orderId}**

| **EndPoint** | **Description** | **HTTP Method** | **Request Body (JSON)** | **Expected HTTP Status Code** | **Expected Response Body (JSON)** | **Actual Result (Pass/Fail)** |
| --- | --- | --- | --- | --- | --- | --- |
| Customers /1/orders /120 | Invalid order id | GET |  | 404 Not Found | {  "status": 400,  "message": "Order not found or doesn't belong to the customer",  "timestamp": 1745239717415  } | pass |
| Customers /1/orders /120 | Invalid customer id | GET |  | 404 Not Found | {  "status": 400,  "message": "Order not found or doesn't belong to the customer",  "timestamp": 1745239868892  } | pass |