

## **Bazar Bookstore – Program Output**

**D. Samer Arandi**

<b>Rand Johari</b>	<b>12027653</b>
<b>Abeer Kharouf</b>	<b>12028125</b>

### **External via Postman:**

- ✓ Search Book by Topic (Frontend → Catalog).
- ✓ Get Book Info by ID (Frontend → Catalog)..
- ✓ Purchase Book (Frontend → Order → Catalog ).

### **Internal via curl:**

- ✓ Internal Request: Update Book Quantity (Order → Catalog).

## 1. 🔍 Search Book by Topic.

**URL:** <http://localhost:3002/search/distributed systems>

**Method:** GET

### Expected Behavior:

Returns a list of books under the topic "distributed systems".

Response output: JSON – 200

The screenshot shows a REST client interface. At the top, a GET request is configured for the URL `http://localhost:3002/search/distributed systems`. Below the URL bar, tabs for Params, Authorization, Headers (6), Body, Scripts, and Settings are visible. The 'Params' tab is active, showing a table for Query Params with columns for Key, Value, and Description. Below this, the 'Body' tab is selected, displaying the JSON response. The response is a 200 OK status with a response time of 43 ms. The JSON data is a list of two book objects. The first object has an id of 1 and a title 'How to get a good grade in DOS in 40 minutes a day'. The second object has an id of 2 and a title 'RPCs for Noobs'.

Key	Value	Description
Key	Value	Description

```
1  [
2    {
3      "id": 1,
4      "title": "How to get a good grade in DOS in 40 minutes a day"
5    },
6    {
7      "id": 2,
8      "title": "RPCs for Noobs"
9    }
10 ]
```

```
2025-04-04 02:27:05 Front-end server running on port 3100
2025-04-04 02:27:10 ✓ Search results for topic 'distributed systems':
2025-04-04 02:27:10 - [1] How to get a good grade in DOS in 40 minutes a day
2025-04-04 02:27:10 - [2] RPCs for Noobs
```

```
2025-04-04 02:27:04 Catalog Service running on http://localhost:3000
2025-04-04 02:27:10 🔍 Search request for topic: "distributed systems"
2025-04-04 02:27:10 ➡ ID: 1, Title: How to get a good grade in DOS in 40 minutes a day
2025-04-04 02:27:10 ➡ ID: 2, Title: RPCs for Noobs
```

## 2. 📖 Get Book Info by ID.

**URL:** <http://localhost:3002/info/2>.

**Method:** GET

**Expected Behavior:**

Returns detailed information about book with ID 2.

Response output: JSON - 200

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method: GET, URL: `http://localhost:3002/info/2`
- Tabs:** Params, Authorization, Headers (6), Body, Scripts, Settings. The 'Params' tab is selected.
- Query Params Table:**

	Key	Value	Description
	Key	Value	Description
- Response Bar:** Status: 200 OK, Time: 10 ms, Size: 300 B. The 'Body' tab is selected.
- Response Body:** JSON format showing book details:

```
1 {
2   "id": 2,
3   "title": "RPCs for Noobs",
4   "topic": "distributed systems",
5   "quantity": 5,
6   "price": 50
7 }
```

Front:

```
2025-04-04 02:41:31 Book details:
2025-04-04 02:41:31 ID: 2
2025-04-04 02:41:31 Topic: undefined
2025-04-04 02:41:31 Title: RPCs for Noobs
2025-04-04 02:41:31 Quantity: 5
2025-04-04 02:41:31 Price: 50
```

Catalog:

```
2025-04-04 02:41:31 Info for book ID 2
2025-04-04 02:41:31 Title: RPCs for Noobs
2025-04-04 02:41:31 Topic: distributed systems
2025-04-04 02:41:31 Price: 50
2025-04-04 02:41:31 Quantity: 5
```

### 3. Purchase Book.

**URL:** <http://localhost:3002/purchase/2>

**Method:** **POST.**

**Expected Behavior:**

When a purchase request is made:

1. The **frontend** sends a POST request to the **order** service.
2. The **order** service:
  - Queries the **catalog** service using GET /info/:id to retrieve the book's current quantity.
  - If quantity > 0, it proceeds to PUT /update/:id to decrease the quantity by 1.
  - If quantity == 0, the purchase fails.

Response output: JSON – 200

POST

▼

http://localhost:3002/purchase/1

Params

Authorization

Headers (7)

Body

Scripts

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body

Cookies

Headers (7)

Test Results

🕒

200 OK

• 144 ms •

{ } JSON ▼


▶ Preview

🔗 Visualize ▼

```
1 {
2   "success": true,
3   "message": "Book purchased successfully"
4 }
```

## Order:


2025-04-04 02:59:42 Trying to contact catalog with: http://catalog:3000/info/1


2025-04-04 02:59:42 bought book: How to get a good grade in DOS in 40 minutes a day


## Fronted:


2025-04-04 02:59:42 bought book with ID: 1


## Catalog:


2025-04-04 02:59:42 Info for book ID 1


2025-04-04 02:59:42 Title: How to get a good grade in DOS in 40 minutes a day

2025-04-04 02:59:42 Topic: distributed systems

2025-04-04 02:59:42 Price: 60

2025-04-04 02:59:42 Quantity: 3

2025-04-04 02:59:42 Received update request for ID: 1

2025-04-04 02:59:42 New data: { quantity: 2 }

After 2 purchases:

POST

http://localhost:3002/purchase/1

Params

Authorization

Headers (7)

Body

Scripts

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body

Cookies

Headers (7)

Test Results

200 OK · 8 ms · 28

{}

JSON

Preview

Visualize

```
1 {
2   "success": false,
3   "message": "Book is out of stock"
4 }
```

Order:

```
2025-04-04 03:23:06 Trying to contact catalog with: http://catalog:3000/info/1
2025-04-04 03:23:06 ❌ Book "How to get a good grade in DOS in 40 minutes a day" is out of stock
```

Fronted:

```
2025-04-04 03:23:01 Front-end server running on port 3100
2025-04-04 03:23:06 ❌ purchase failed - Book with ID 1 is out of stock
```

Catalog:

```
2025-04-04 03:23:06 ✅ info for book ID 1
2025-04-04 03:23:06 📖 Title: How to get a good grade in DOS in 40 minutes a day
2025-04-04 03:23:06 📖 Topic: distributed systems
2025-04-04 03:23:06 📖 Price: 60
2025-04-04 03:23:06 📖 Quantity: 0
```

#### 4. 📡 Internal Request: Get Book Info.

**URL:** curl http://catalog:3000/info/2

**Method:** PUT.

**Executed inside:** order container

To test updating a book manually (simulate internal communication):

1. Open a shell into the order container:

```
docker exec -it <order_container_id> sh
```

2. Run the following command:

```
curl -X PUT http://catalog:3000/update/2 \  
-H "Content-Type: application/json" \  
-d '{"quantity": 4, "price": 40 }'
```

```
C:\Users\pc\bazar-bookstore\backend\order>docker exec -it e64cfbec0f3b89f2834149feea26d08a7c2a932686d54a80aa8114d82121d1  
58 sh  
#  
# curl -X PUT http://catalog:3000/update/2 \  
-H "Content-Type: application/json" \  
-d '{"quantity": 4, "price": 40 }'  
> > {"status":"success","updated":{"id":2,"price":40,"quantity":4}}#
```

Nothing in Front and Order:

```
2025-04-04 12:28:52 Front-end server running on port 3100
```

```
2025-04-04 12:28:52 Order Service running on http://localhost:3001
```

Catalog:

```
2025-04-04 12:28:52 Catalog Service running on http://localhost:3000  
2025-04-04 12:34:49 🟢 received update request for ID: 2  
2025-04-04 12:34:49 📄 New data: { quantity: 4, price: 40 }
```