An-Najah National University

Faculty of Engineering & Information Technology

Dos

Project - Part 1

Students:

Fadi Hamad 12112857

Ayman Abu Hijleh 12112985

Supervisor:

Dr.Samer Arandi

arandi@najah.edu

April 3, 2025

1. Introduction

This project focuses on developing a system for managing a book catalog, purchasing books, and tracking orders. It is a full-stack application that utilizes a server built with Node.js and Express, and the data is stored and manipulated using CSV files. The project includes multiple components:

- Book catalog management (adding, updating, and searching books).
- Order management (purchasing books, logging orders).
- Stock management (decreasing stock based on purchases).

The project simulates an e-commerce-like system for books, where users can search for books by topic, purchase them, and the stock is updated accordingly.

2. Technology Stack

- **Backend**: Node.js, Express
- **CSV Handling**: csv-parser, csv-writer
- **Data Storage**: CSV files (proj.csv for catalog data, orders.csv for logged orders)
- **Testing Tool**: Postman for API testing
- **Docker**

3. Features

3.1. Book Catalog Management

The system allows users to:

- Search books by topic.
- View detailed information about a specific book by its ID.
- Update book prices and stock based on purchasing.

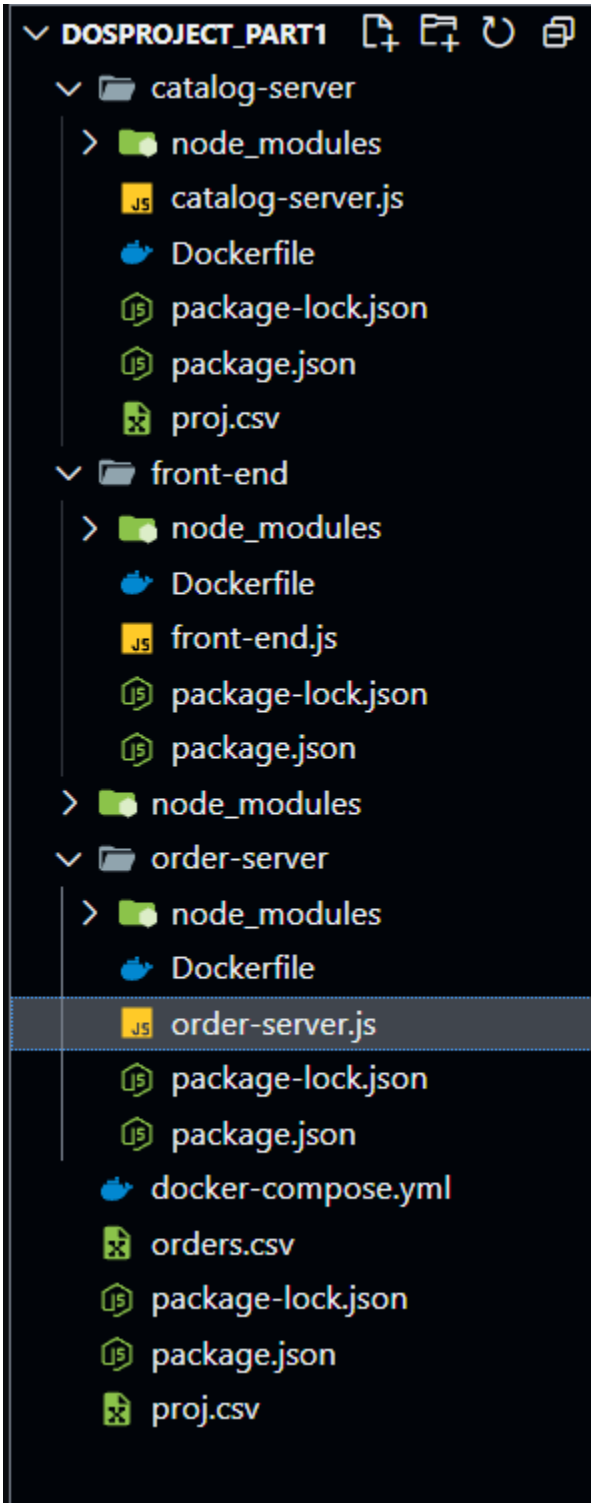
3.2. Order Management

The system also allows users to:

- Purchase books by specifying the quantity.

- Log each purchase into the orders.csv file.

4. Project structure



5. API Endpoints

Below is a list of the core API endpoints exposed by the system:

Method	Endpoint	Description
GET	**/search/:topic**	**Searches for books based on the topic specified in the request.**
GET	**/info/:item_number**	**Retrieves detailed information about a specific book using its item number (ID).**
PUT	**/update**	**Updates the price and stock of a book specified in the request body.**
POST	**/purchase/:item_number**	**Allows a user to purchase a book and logs the order in orders.csv.**

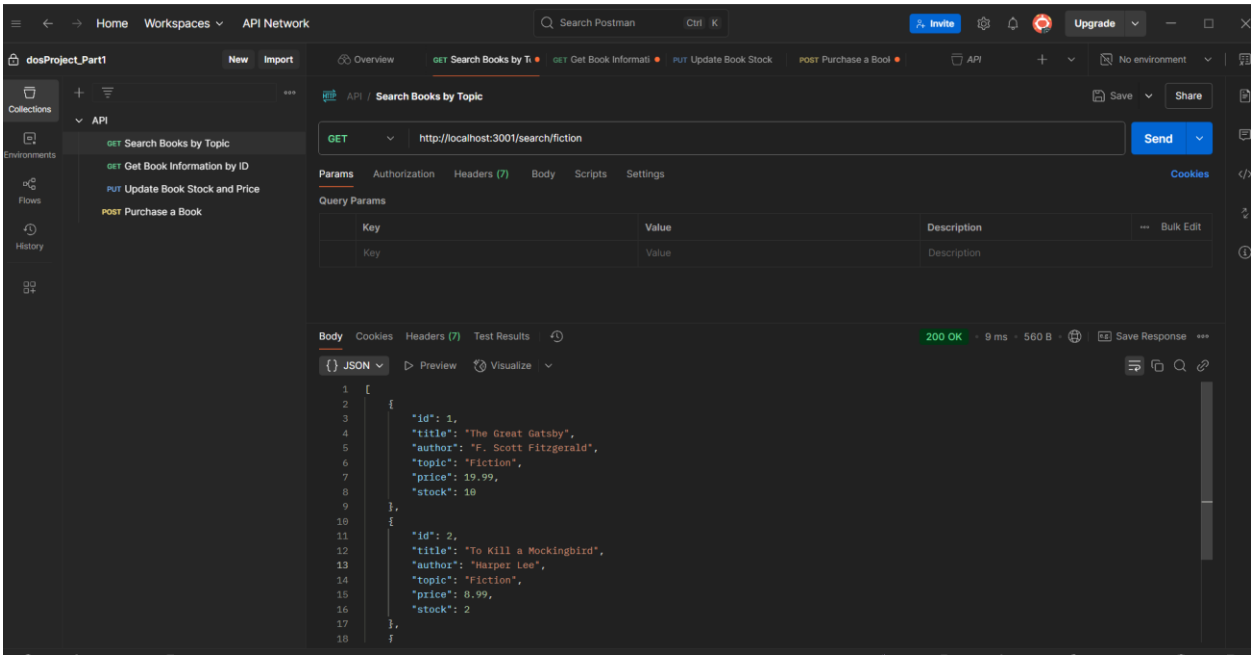
6. API Testing

The following section provides step-by-step instructions on how to test the API using Postman.

6.1. Testing the /search/:topic Endpoint

- **Method:** GET
- **URL:** http://localhost:3002/search/Fiction
- **Description:** This endpoint allows searching books based on the specified topic.

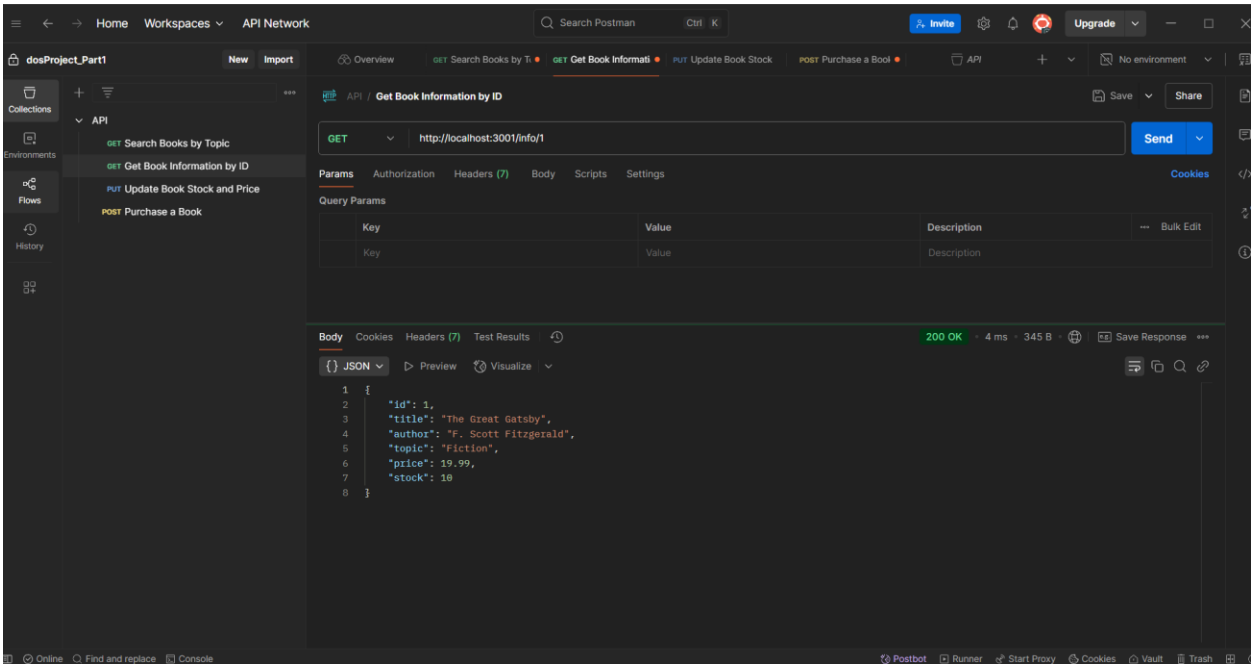
Expected Response: A list of books with the topic "Fiction" will be returned in the response.



6.2. Testing the /info/:item_number Endpoint

- **Method:** GET
- **URL:** http://localhost:3002/info/1
- **Description:** This endpoint retrieves detailed information about a specific book by its ID.

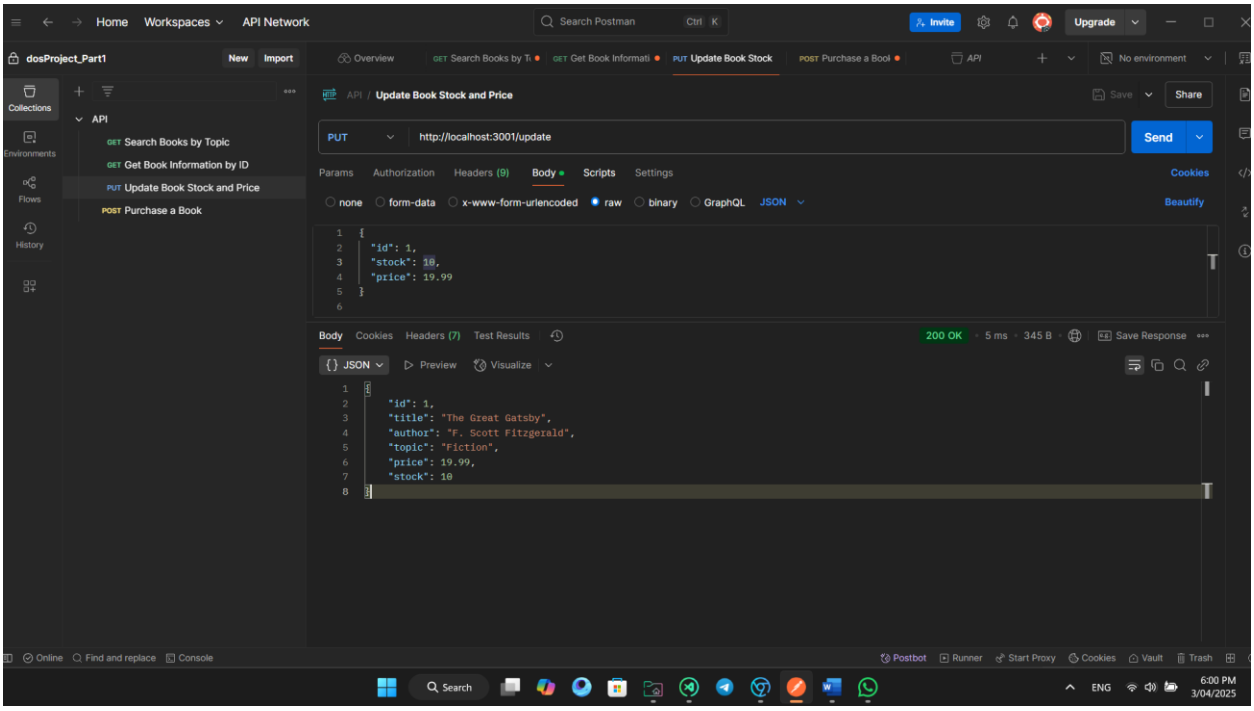
Expected Response: The detailed information about the book with the specified ID will be returned in the response.

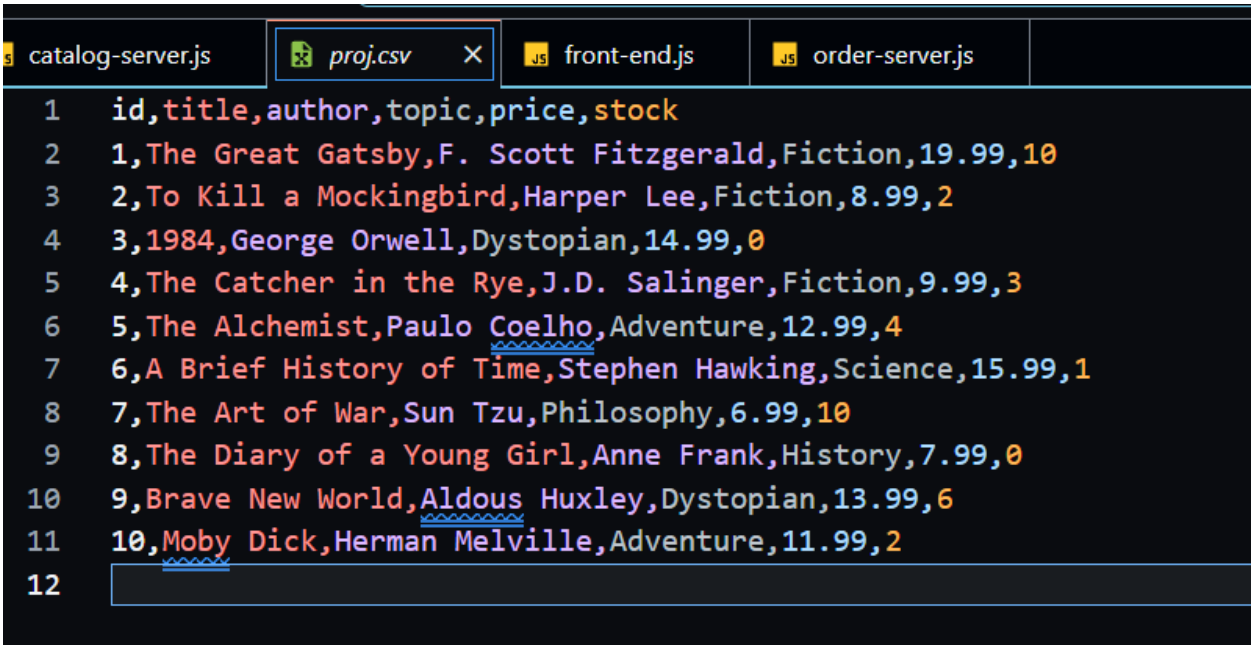


6.3. Testing the /update Endpoint

- **Method:** PUT
- **URL:** http://localhost:3002/update
- **Description:** This endpoint updates the price and stock of a specified book.

Expected Response: The updated book object, including the new price and stock values, will be returned.

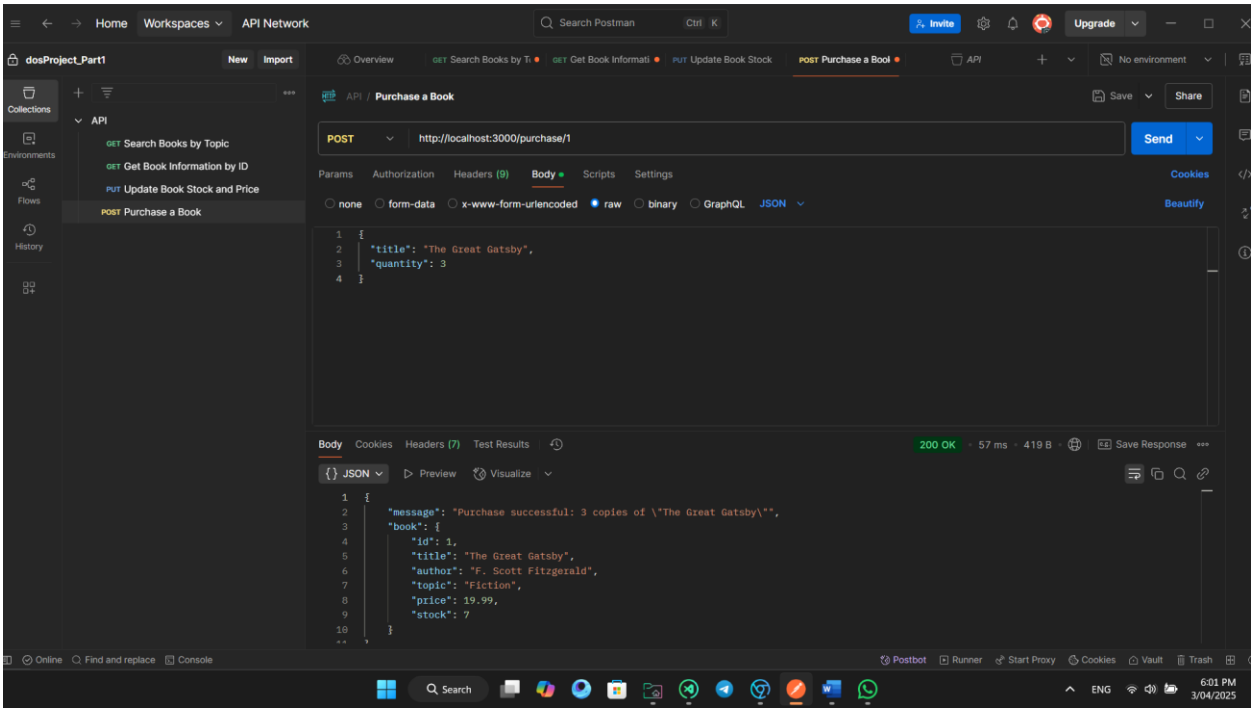

Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
dosProject_Part1
New
Import
Collections
Environments
Flows
History
API
GET Search Books by Topic
GET Get Book Information by ID
PUT Update Book Stock and Price
POST Purchase a Book
Overview
GET Search Books by Tr
GET Get Book Informati
PUT Update Book Stock
POST Purchase a Bool
API
No environment
API / Update Book Stock and Price
Save
Share
PUT
http://localhost:3001/update
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 1,
"stock": 10,
"price": 19.99
}
Body
Cookies
Headers (7)
Test Results
200 OK
5 ms
345 B
Save Response
JSON
Preview
Visualize
{
"id": 1,
"title": "The Great Gatsby",
"author": "F. Scott Fitzgerald",
"topic": "Fiction",
"price": 19.99,
"stock": 10
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
Search
ENG
6:00 PM
3/04/2025

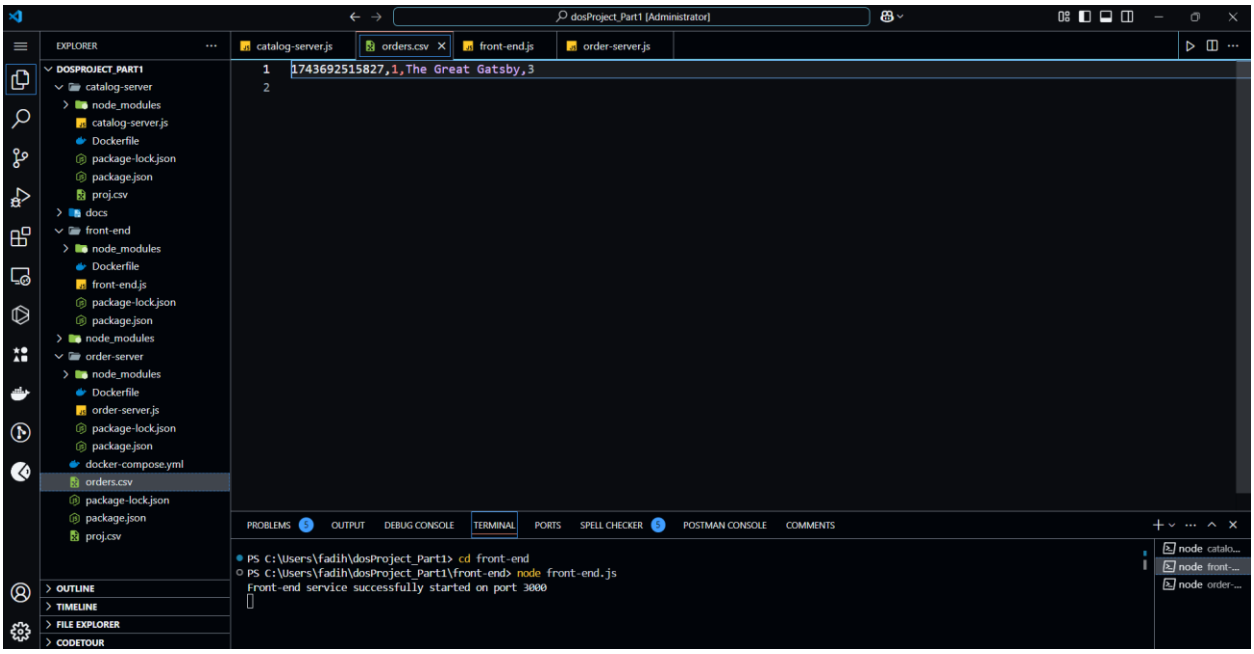

catalog-server.js
proj.csv
front-end.js
order-server.js
1 id,title,author,topic,price,stock
2 1,The Great Gatsby,F. Scott Fitzgerald,Fiction,19.99,10
3 2,To Kill a Mockingbird,Harper Lee,Fiction,8.99,2
4 3,1984,George Orwell,Dystopian,14.99,0
5 4,The Catcher in the Rye,J.D. Salinger,Fiction,9.99,3
6 5,The Alchemist,Paulo Coelho,Adventure,12.99,4
7 6,A Brief History of Time,Stephen Hawking,Science,15.99,1
8 7,The Art of War,Sun Tzu,Philosophy,6.99,10
9 8,The Diary of a Young Girl,Anne Frank,History,7.99,0
10 9,Brave New World,Aldous Huxley,Dystopian,13.99,6
11 10,Moby Dick,Herman Melville,Adventure,11.99,2
12

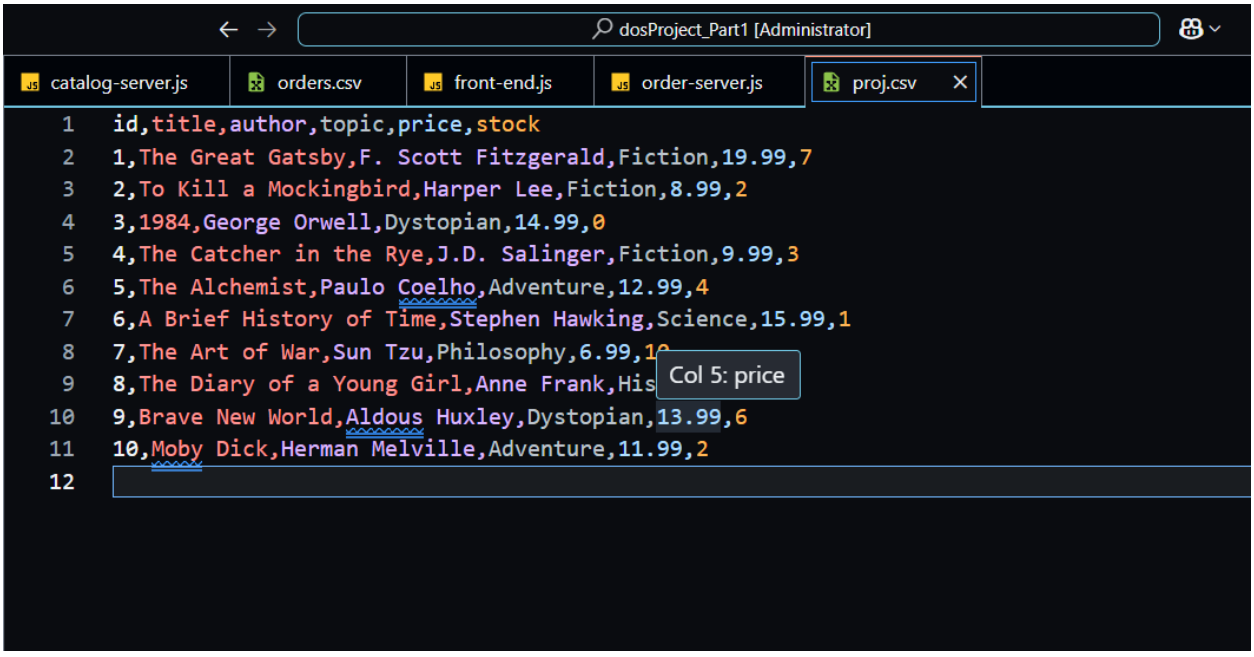
6.4. Testing the /purchase/:item_number Endpoint

- **Method:** POST
- **URL:** http://localhost:3002/purchase/1
- **Description:** This endpoint processes a book purchase and decreases the stock based on the quantity specified.

  Expected Response: A message indicating that the purchase was successful, along with the updated book details.




dosProject_Part1 [Administrator]
EXPLORER
catalog-server.js
orders.csv
front-end.js
order-server.js
1743692515827,1,The Great Gatsby,3
DOSPROJECT_PART1
catalog-server
node_modules
catalog-server.js
Dockerfile
package-lock.json
package.json
proj.csv
docs
front-end
node_modules
Dockerfile
front-end.js
package-lock.json
package.json
node_modules
order-server
node_modules
Dockerfile
order-server.js
package-lock.json
package.json
docker-compose.yml
orders.csv
package-lock.json
package.json
proj.csv
OUTLINE
TIMELINE
FILE EXPLORER
CODETOUR
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SPELL CHECKER
POSTMAN CONSOLE
COMMENTS
PS C:\Users\fadih\dosProject_Part1> cd front-end
PS C:\Users\fadih\dosProject_Part1\front-end> node front-end.js
Front-end service successfully started on port 3000
node catalo...
node front-...
node order-...

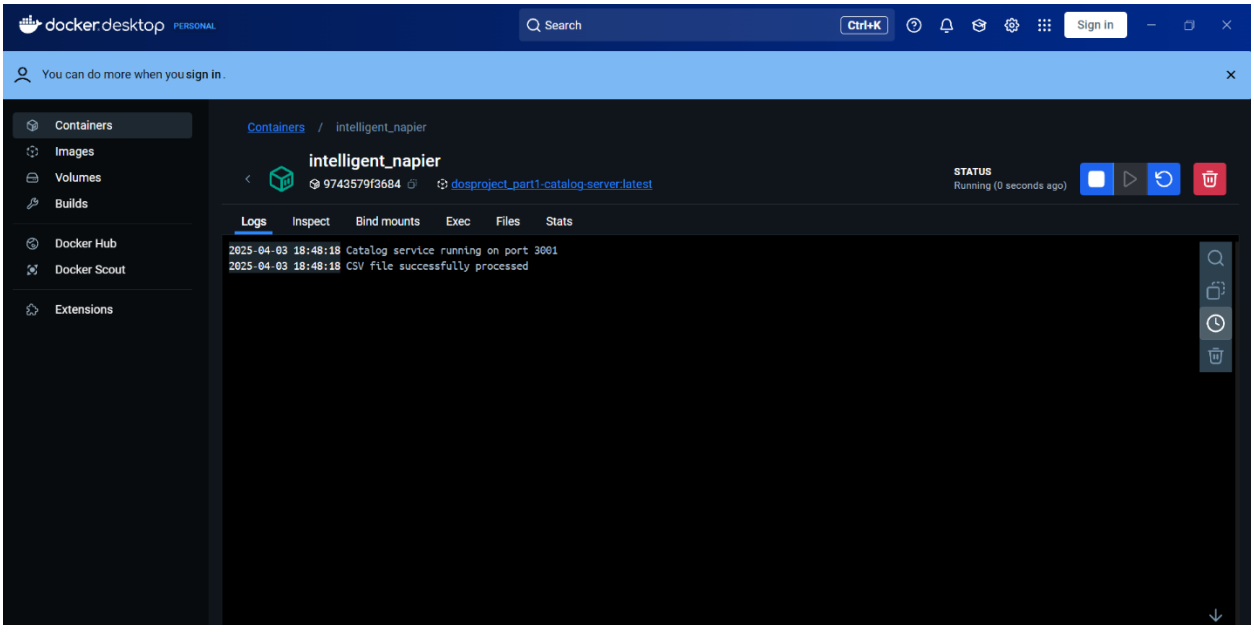

dosProject_Part1 [Administrator]
catalog-server.js
orders.csv
front-end.js
order-server.js
proj.csv
id,title,author,topic,price,stock
1,The Great Gatsby,F. Scott Fitzgerald,Fiction,19.99,7
2,To Kill a Mockingbird,Harper Lee,Fiction,8.99,2
3,1984,George Orwell,Dystopian,14.99,0
4,The Catcher in the Rye,J.D. Salinger,Fiction,9.99,3
5,The Alchemist,Paulo Coelho,Adventure,12.99,4
6,A Brief History of Time,Stephen Hawking,Science,15.99,1
7,The Art of War,Sun Tzu,Philosophy,6.99,1
8,The Diary of a Young Girl,Anne Frank,His
Col 5: price
9,Brave New World,Aldous Huxley,Dystopian,13.99,6
10,Moby Dick,Herman Melville,Adventure,11.99,2

7.Docker Container Test

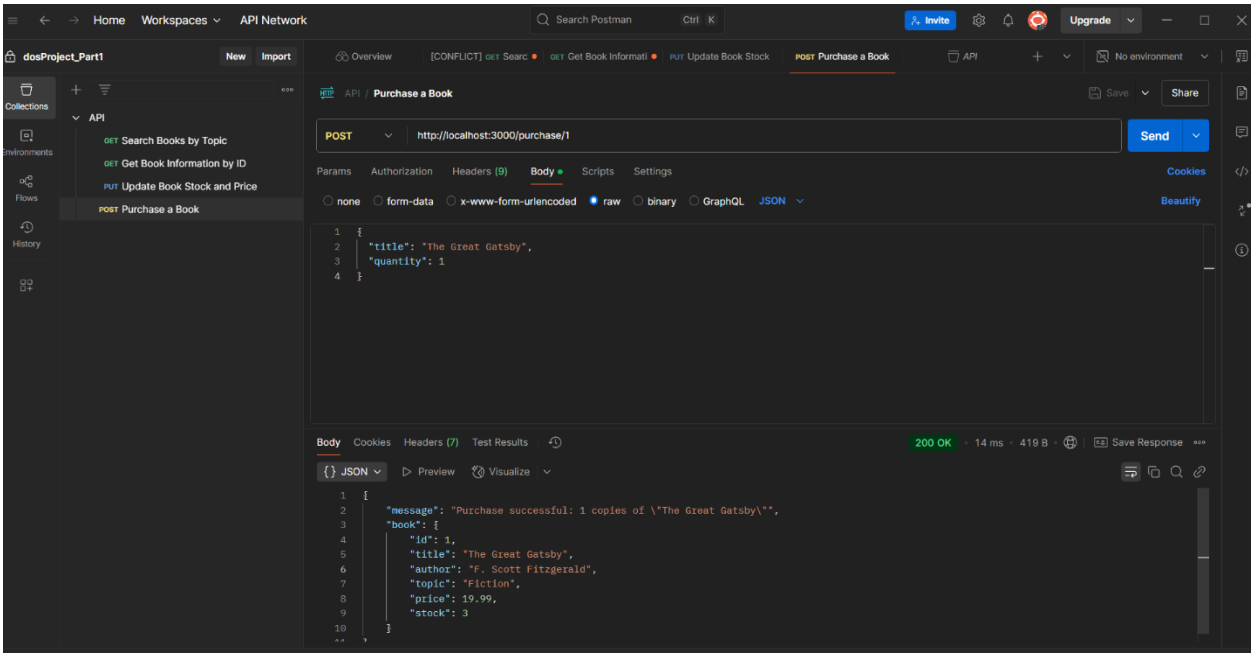
```
[+] Running 7/7
 ✓ catalog-server                              Built
 ✓ front-end                                   Built
 ✓ order-server                                Built
 ✓ Network dosproject_part1_default            Created
 ✓ Container dosproject_part1-order-server-1   Created
 ✓ Container dosproject_part1-front-end-1      Created
 ✓ Container dosproject_part1-catalog-server-1 Created
Attaching to catalog-server-1, front-end-1, order-server-1
front-end-1       | Front-end service successfully started on port 3000
order-server-1    | Order service running on port 3002
catalog-server-1  | Catalog service running on port 3001
catalog-server-1  | CSV file successfully processed
order-server-1    | CSV file successfully processed
intelligent_napier  | Catalog service running on port 3001
intelligent_napier  | CSV file successfully processed
catalog-server-1    | CSV file updated successfully
```

```
C:\Users\fadih>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED         STATUS         PORTS
          NAMES
d91f63b32587   dosproject_part1-front-end        "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   0.0.0.0:3000->3
000/tcp   dosproject_part1-front-end-1
7d4bf77cc2c7   dosproject_part1-catalog-server   "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   0.0.0.0:3001->3
001/tcp   dosproject_part1-catalog-server-1
90976ba6f777   dosproject_part1-order-server     "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   0.0.0.0:3002->3
002/tcp   dosproject_part1-order-server-1

C:\Users\fadih>
```




```
C:\Users\fadih\dosProject_Part1>curl http://localhost:3001/search/fiction
[{"id":1,"title":"The Great Gatsby","author":"F. Scott Fitzgerald","topic":"Fiction","price":19.99,"stock":10},{"id":2,"
title":"To Kill a Mockingbird","author":"Harper Lee","topic":"Fiction","price":8.99,"stock":2},{"id":4,"title":"The Catc
her in the Rye","author":"J.D. Salinger","topic":"Fiction","price":9.99,"stock":3}]
C:\Users\fadih\dosProject_Part1>
```



8. File Management

The data for the catalog and orders are stored in CSV files:

- **proj.csv:** This file contains the catalog of books, with details such as title, author, topic, price, and stock.
- **orders.csv:** This file logs all orders made by users, including the order ID, book ID, title, and quantity.

9. Error Handling

The application has basic error handling mechanisms:

- **404 Not Found: Returned when a book is not found in the catalog.**
- **400 Bad Request: Returned when there is not enough stock for a purchase.**
- **500 Internal Server Error: Returned if there are issues logging the order or updating the CSV files.**

10. Conclusion

This Book Catalog & Order Management System is a simple yet efficient way to manage books and process orders. The system allows for easy catalog searching, updating book details, and logging orders. Using Postman, you can thoroughly test all API endpoints to ensure the system works as expected.

The use of CSV files for storing book and order data simplifies the process, though in a production environment, a more robust database system would be recommended.

11. Future Improvements

- **Database Integration:** Move from CSV files to a proper database like MySQL or MongoDB for better performance and scalability.
- **User Authentication:** Implement user authentication so that only registered users can make purchases.
- **Order History:** Add functionality to view past orders for each user.