Case Study: "Book Bazaar" Inventory Management System

Background:

"Book Bazaar" is a small, independent bookstore located in a bustling city center. The store offers a wide variety of books, from bestsellers to rare editions. To keep track of their inventory, the bookstore currently uses a manual system involving spreadsheets, which is time-consuming and prone to errors. To streamline their operations and improve inventory management, the bookstore decides to develop a simple digital inventory management system.

Objective:

Develop a backend system for "Book Bazaar" using Node.js and Express.js. The system should allow bookstore staff to manage their inventory efficiently by supporting CRUD operations. The data will be stored in an in-memory array to facilitate quick development and testing. The system should provide a RESTful API to interact with the book inventory.

Requirements:

1. **Create a New Book:**
    - **Endpoint:** `POST /books`
    - **Description:** Adds a new book to the inventory. Each book should have a unique `id`, a `title`, an `author`, and a `price`.

Request Body:

```
{
  "title": "string",
  "author": "string",
  "price": "number"
}
```

- **Response:**

Success (201 Created):

```
{
  "id": "integer",
  "title": "string",
  "author": "string",
  "price": "number"
```

```
}
```

-

Error (400 Bad Request): If any required field is missing or invalid.

```
{
  "error": "All fields are required and price must be a positive
number"
}
```

-

2. **Read All Books:**
    - **Endpoint:** `GET /books`
    - **Description:** Retrieves a list of all books currently in the inventory.
    - **Response:**

Success (200 OK):

```
[
  {
    "id": "integer",
    "title": "string",
    "author": "string",
    "price": "number"
  },
  ...
]
```

-

3. **Read a Specific Book:**
    - **Endpoint:** `GET /books/:id`
    - **Description:** Retrieves details of a single book by its unique `id`.
    - **Parameters:**
        - `id` (path parameter): The unique identifier of the book.
    - **Response:**

Success (200 OK):

```
{
  "id": "integer",
  "title": "string",
  "author": "string",
  "price": "number"
}
```

-

Error (404 Not Found): If the book with the specified `id` does not exist.

```
{
  "error": "Book not found"
}
```

-

4. **Update Book Information:**
    - **Endpoint:** `PUT /books/:id`
    - **Description:** Updates the details of an existing book identified by its `id`.
    - **Parameters:**
        - `id` (path parameter): The unique identifier of the book to be updated.

Request Body:

```
{
  "title": "string",
  "author": "string",
  "price": "number"
}
```

-
- **Response:**

Success (200 OK):

```
{
  "id": "integer",
  "title": "string",
```

```
  "author": "string",
  "price": "number"
}
```

-

Error (400 Bad Request): If any required field is missing or invalid.

```
{
  "error": "All fields are required and price must be a positive
number"
}
```

-

Error (404 Not Found): If the book with the specified `id` does not exist.

```
{
  "error": "Book not found"
}
```

-

5. **Delete a Book:**
    - **Endpoint:** `DELETE /books/:id`
    - **Description:** Removes a book from the inventory by its `id`.
    - **Parameters:**
        - `id` (path parameter): The unique identifier of the book to be deleted.
    - **Response:**
        - **Success (204 No Content):** Indicates that the book was successfully deleted.

Error (404 Not Found): If the book with the specified `id` does not exist.

```
{
  "error": "Book not found"
}
```

-

Constraints:

- The system will use an in-memory array to store book data. This means that data will not persist across server restarts.
- Input validation is required to ensure that all fields are provided and that the `price` is a positive number.
- Error handling should provide clear and informative messages for invalid operations (e.g., missing fields, non-existent books).

Assumptions:

- The backend system will be used internally by bookstore staff for managing inventory.
- The system is intended for development and testing, and the in-memory storage is suitable for this purpose.
- The server will run on localhost and listen on port 3001.