**Exercise 8: Online Bookstore - Implementing CRUD Operations**

**Business Scenario:**

Implement Create, Read, Update, and Delete (CRUD) operations for the Book and Customer entities in the Online Bookstore application.

**Instructions:**

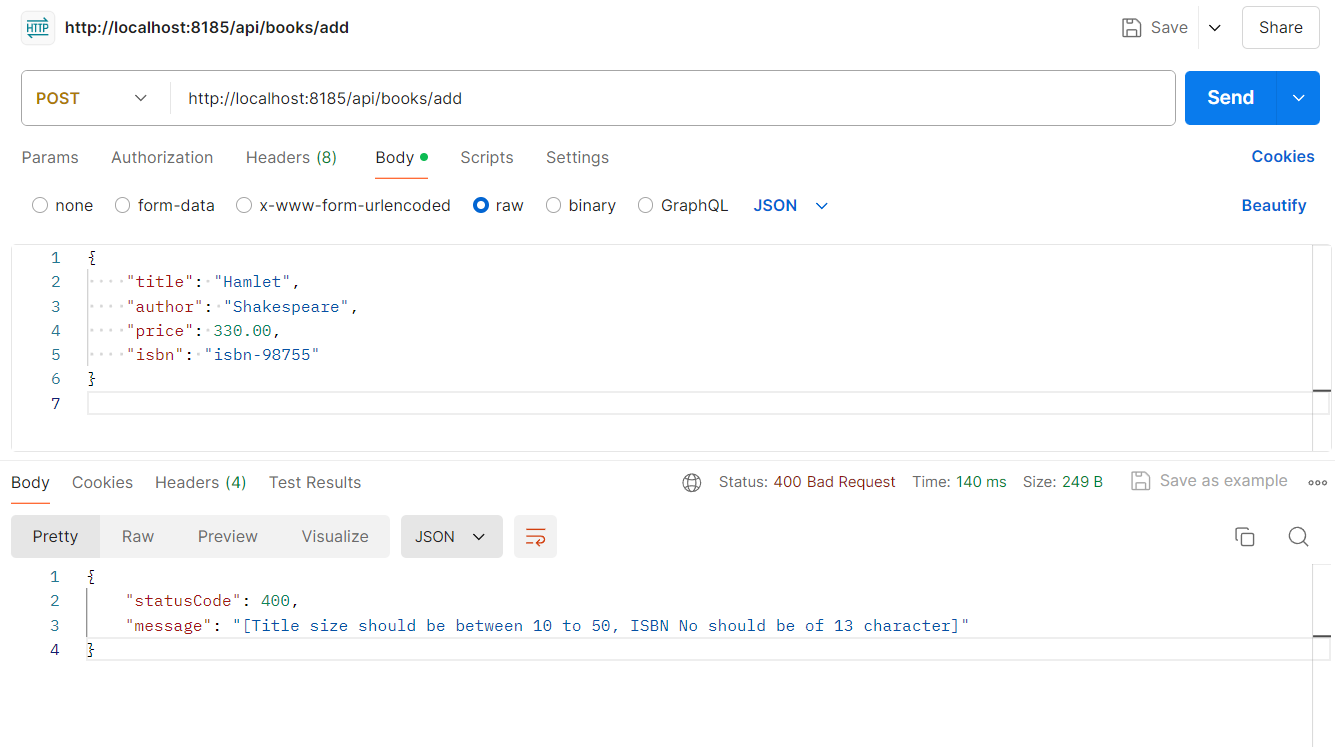
1. **CRUD Endpoints:**
   * Implement endpoints for creating, reading, updating, and deleting books and customers.
   * **Books Endpoints:**
     + POST /api/books/add: Adds a new book to the bookstore.
     + GET /api/books/: Retrieves a list of all books.
     + GET /api/books/{id}: Retrieves a specific book by its ID.
     + PUT /api/books/edit/{id}: Updates the details of a book.
     + DELETE /api/books/delete/{id}: Deletes a book by its ID.
   * **Customers Endpoints:**
     + POST /api/customers/register: Registers a new customer.
     + GET /api/customers/{id}: Retrieves a specific customer by their ID.
     + PUT /api/customers/edit/{id}: Updates customer details.
     + DELETE /api/customers/delete/{id}: Deletes a customer by their ID.
2. **Validating Input Data:**
   * Validation is implemented using annotations in the Book and Customer entity classes to ensure that the input data adheres to the required constraints.
   * **Book Entity:**
     + @NotNull for mandatory fields like title and author.
     + @Size for the title to ensure it is within a certain length.
     + @Min for price to ensure it is greater than a certain value.
     + @Size for isbn to ensure it is exactly 13 characters.
   * **Customer Entity:**
     + @NotNull for mandatory fields like name and emailid.
     + @Size for the emailid to ensure it adheres to the format and length.
3. **Optimistic Locking:**
   * Implemented using JPA versioning to manage concurrent updates. The @Version annotation is used in the Book entity to handle optimistic locking. This ensures that when multiple transactions are trying to update the same entity, they do not overwrite each other's changes.

**Annotations usage:**

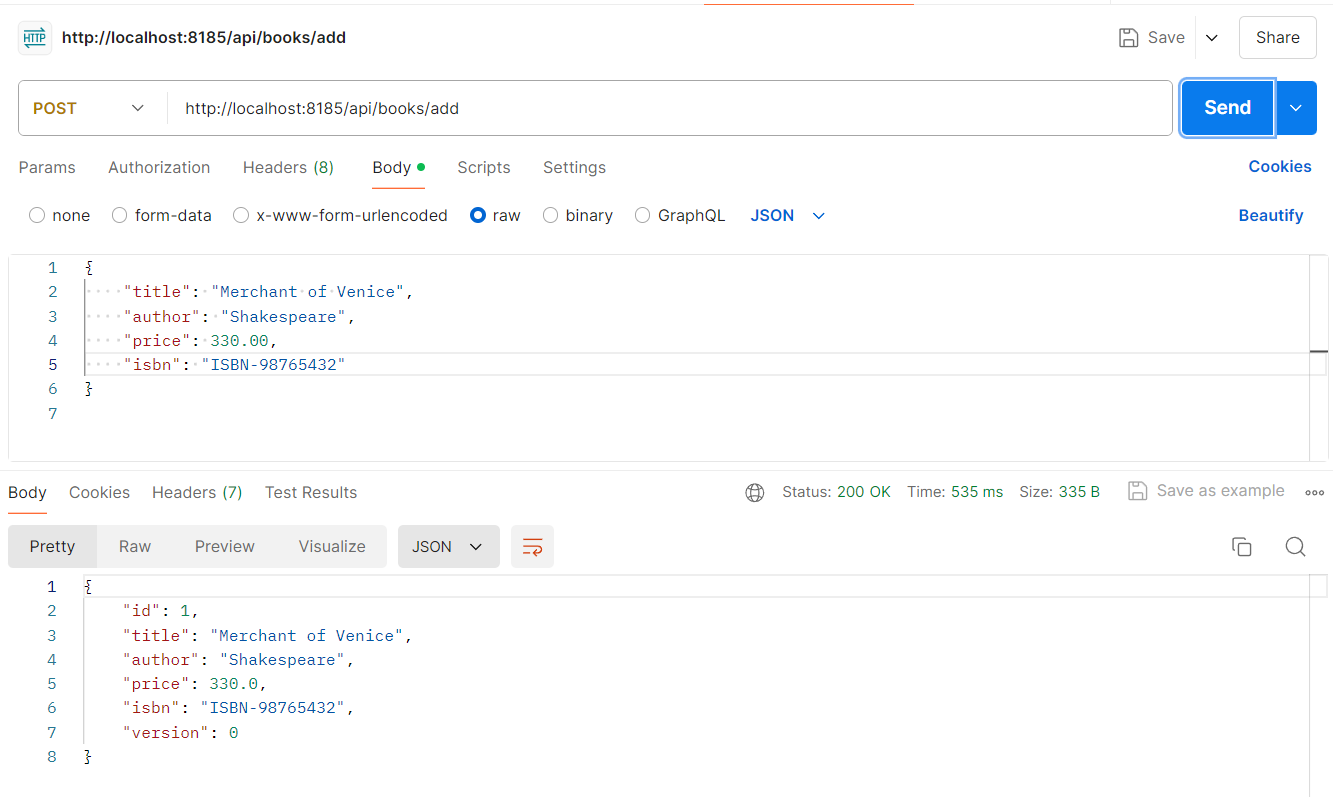
* **@NotNull**: Ensures that the annotated field is not null.
* **@Size**: Specifies the minimum and maximum length of a field.
* **@Min**: Ensures that the annotated field is greater than or equal to the specified value.
* **@Version**: Used for optimistic locking to prevent concurrent modifications.

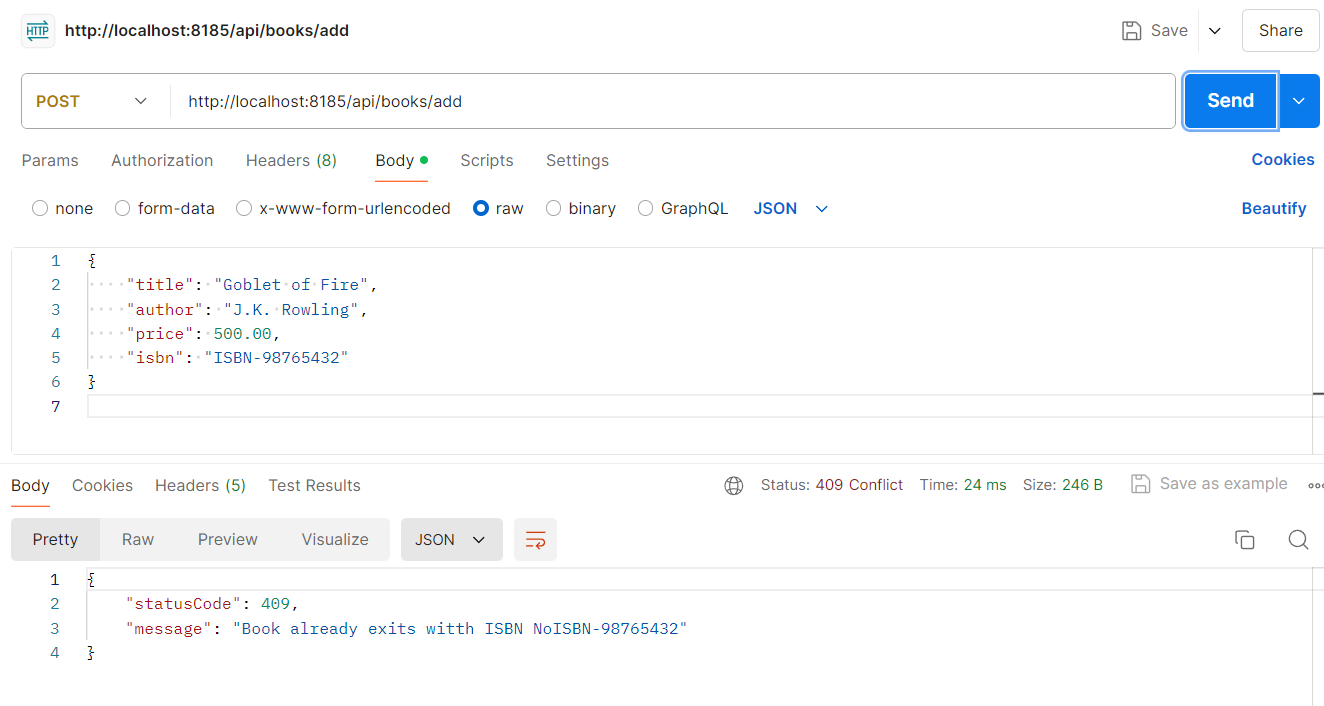
**Example Response Handling:**

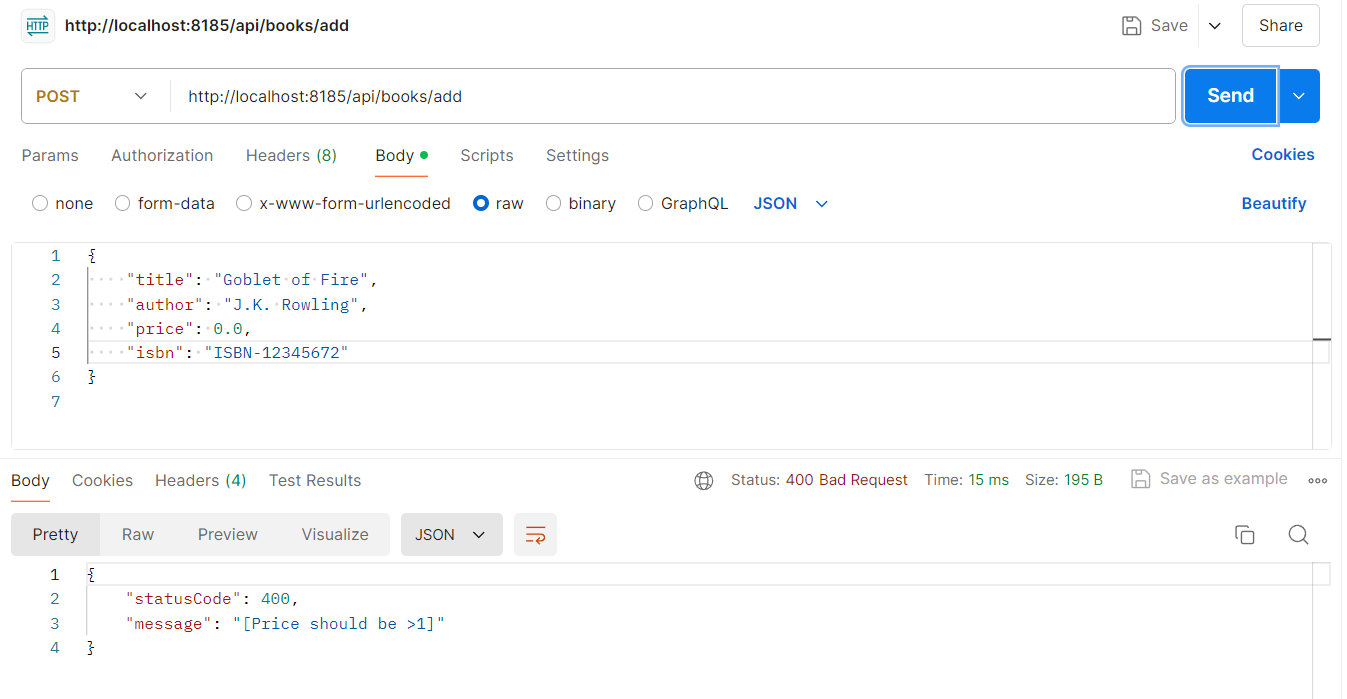
Here showing 2 cases together, when book **ISBN is below 13 character** and **title is below 10 character**:

****

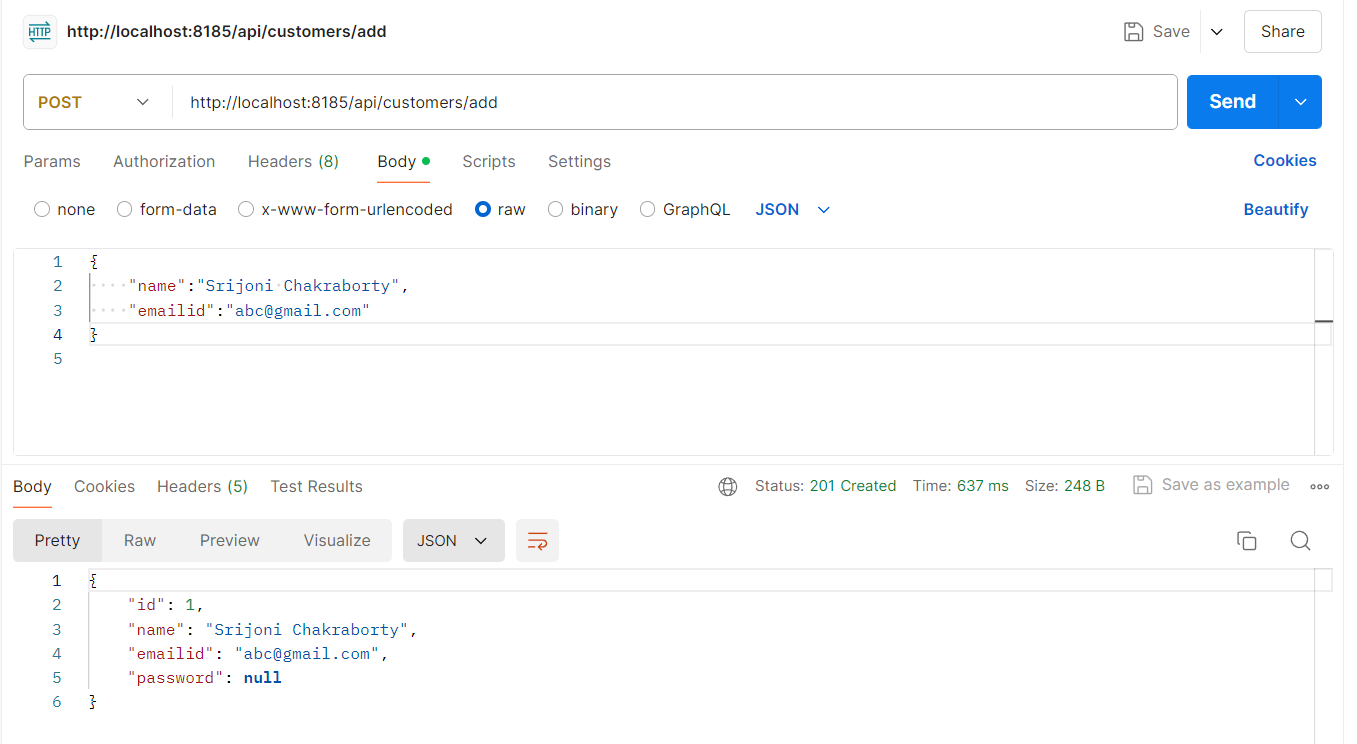
**Add Book:**

****

**When Book to be Added has same ISBN as of another book:**

**When Adding a Book with Price 0.0:  
**

**Add Customer:**

****

* **GET /api/books/**:

Returns a list of all books with HTTP status 200 OK. If no books are found, it returns HTTP status 204 No Content.

* **POST /api/books/add**:

Adds a new book and returns the book details along with HTTP status 201 Created.

* **PUT /api/books/edit/{id}**:

Updates an existing book and returns the updated book details with HTTP status 200 OK.

* **DELETE /api/books/delete/{id}**:

Deletes a book and returns HTTP status 204 No Content upon successful deletion.

**Exception Handling:**

Specific exceptions like BookIsbnAlreadyExitsException are handled to return appropriate HTTP status codes and error messages.