### ****Capstone Project: API Testing for FreeTestAPI (Books API)****

**Detailed Test Scenarios with Specific Endpoints**

**1. GET /api/v1/books**

* **Scenario 1**: Retrieve all books.
  + **Endpoint**: GET /api/v1/books
  + **Expected Result**: Status 200; response contains a list of books.
* **Scenario 2**: Validate the structure of each book in the list.
  + **Endpoint**: GET /api/v1/books
  + **Expected Result**: Status 200; each book object contains the fields: id, title, author, and isbn.

**2. GET /api/v1/books/{id}**

* **Scenario 3**: Retrieve a book by a valid ID.
  + **Endpoint**: GET /api/v1/books/{id}
  + **Example**: GET /api/v1/books/1
  + **Expected Result**: Status 200; response contains the details of the book with ID 1.
* **Scenario 4**: Attempt to retrieve a book by an invalid ID.
  + **Endpoint**: GET /api/v1/books/{id}
  + **Example**: GET /api/v1/books/9999
  + **Expected Result**: Status 404; response contains an error message indicating the book was not found.

**3. POST /api/v1/books**

* **Scenario 5**: Add a new book with valid data.
  + **Endpoint**: POST /api/v1/books
  + **Body**:

json

Copy code

{

"title": "Effective Java",

"author": "Joshua Bloch",

"isbn": "9780134685991"

}

* + **Expected Result**: Status 201; response contains the details of the newly created book.
* **Scenario 6**: Attempt to add a book with missing mandatory fields.
  + **Endpoint**: POST /api/v1/books
  + **Body**:

json

Copy code

{

"title": "Incomplete Book"

}

* + **Expected Result**: Status 400; response contains an error message indicating the missing fields.
* **Scenario 7**: Add a book with a duplicate ISBN.
  + **Endpoint**: POST /api/v1/books
  + **Body**:

json

Copy code

{

"title": "Duplicate ISBN",

"author": "Author Name",

"isbn": "9780134685991"

}

* + **Expected Result**: Status 409; response contains an error message indicating that the ISBN already exists.

**4. PUT /api/v1/books/{id}**

* **Scenario 8**: Update an existing book’s title.
  + **Endpoint**: PUT /api/v1/books/{id}
  + **Example**: PUT /api/v1/books/1
  + **Body**:

json

Copy code

{

"title": "Effective Java (Updated Edition)"

}

* + **Expected Result**: Status 200; response contains the updated book details.
* **Scenario 9**: Attempt to update a book that doesn’t exist.
  + **Endpoint**: PUT /api/v1/books/{id}
  + **Example**: PUT /api/v1/books/9999
  + **Body**:

json

Copy code

{

"title": "Non-Existent Book"

}

* + **Expected Result**: Status 404; response contains an error message indicating the book was not found.

**5. DELETE /api/v1/books/{id}**

* **Scenario 10**: Delete a book by a valid ID.
  + **Endpoint**: DELETE /api/v1/books/{id}
  + **Example**: DELETE /api/v1/books/1
  + **Expected Result**: Status 204; no content in the response body.
* **Scenario 11**: Attempt to delete a book by an invalid ID.
  + **Endpoint**: DELETE /api/v1/books/{id}
  + **Example**: DELETE /api/v1/books/9999
  + **Expected Result**: Status 404; response contains an error message indicating the book was not found.

**6. GET /api/v1/books?title={title}**

* **Scenario 12**: Search for books by a partial title.
  + **Endpoint**: GET /api/v1/books?title=Java
  + **Expected Result**: Status 200; response contains a list of books that match the title search.
* **Scenario 13**: Search for books by a non-existent title.
  + **Endpoint**: GET /api/v1/books?title=NonExistentTitle
  + **Expected Result**: Status 200; response contains an empty list.

**7. GET /api/v1/books?author={author}**

* **Scenario 14**: Filter books by an author's name.
  + **Endpoint**: GET /api/v1/books?author=Joshua Bloch
  + **Expected Result**: Status 200; response contains a list of books by Joshua Bloch.
* **Scenario 15**: Filter books by a non-existent author's name.
  + **Endpoint**: GET /api/v1/books?author=Unknown Author
  + **Expected Result**: Status 200; response contains an empty list.

**8. GET /api/v1/books?page={page}&limit={limit}**

* **Scenario 16**: Retrieve paginated books.
  + **Endpoint**: GET /api/v1/books?page=1&limit=5
  + **Expected Result**: Status 200; response contains a list of 5 books on page 1.
* **Scenario 17**: Request a page that exceeds the total number of pages.
  + **Endpoint**: GET /api/v1/books?page=100&limit=5
  + **Expected Result**: Status 200; response contains an empty list.

**9. POST /api/v1/books/batch**

* **Scenario 18**: Add multiple books in a single request.
  + **Endpoint**: POST /api/v1/books/batch
  + **Body**:

json

Copy code

[

{

"title": "Book One",

"author": "Author One",

"isbn": "1111111111"

},

{

"title": "Book Two",

"author": "Author Two",

"isbn": "2222222222"

}

]

* + **Expected Result**: Status 201; response contains the details of the newly created books.
* **Scenario 19**: Attempt to add multiple books with one having an invalid field.
  + **Endpoint**: POST /api/v1/books/batch
  + **Body**:

json

Copy code

[

{

"title": "Valid Book",

"author": "Valid Author",

"isbn": "3333333333"

},

{

"title": "",

"author": "Invalid Author",

"isbn": "4444444444"

}

]

* + **Expected Result**: Status 400; response indicates which book entry caused the validation error.

**10. GET /api/v1/books/{id}/authors**

* **Scenario 20**: Retrieve the author of a specific book by book ID.
  + **Endpoint**: GET /api/v1/books/{id}/authors
  + **Example**: GET /api/v1/books/1/authors
  + **Expected Result**: Status 200; response contains the author details for the book.