API Endpoints:

1. Get Token

- Method: GET
- Endpoint: /getToken
- Response:
    - Status: 200 OK
    - Body: An array of JSON objects representing user details with accesstoken

  Example Response:

```
{
  "name": "Syed Usman",
  "id": 1002,
  "type": "author",
  "isSuperUser": false,

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiU3llZCBV
c21hbiIsImlkIjoxMDAyLCJ0eXBlIjoiYXV0aG9yIiwiaXNTdXBlclVzZXIiOmZhbHN
lLCJpYXQiOjE3MTM5NzU0NjV9.Zpqwp5EN9pf1nlw_j7r13vPjFy-
PdXozw3JpyBEkXtU"
}
```

2. Get books

- Method: Get
- End Point: `books/`
- Description: Retrieves books from the collection based on optional query parameters.
- Query Parameters:
    a. `author`: Filter books by author name (exact match).
    b. `year`: Filter books published in a specific year (exact match).
    c. `id`: Retrieve a single book by its unique identifier.
    d. `title`: Filter books by title (exact match).
- Response:

```
[
    {
        "title": "JavaScript",
        "author": "Syed Usman",
        "authorId": "1002",
        "id": "1001",
        "year": 2022
    },
    {
        "title": "Python",
        "author": "Syed Lukman",
        "authorId": "1001",
        "id": "1002",
        "year": 2022
    }
]
```

- Status: 200 OK, Body: An array of JSON objects representing the matching books. Each object contains book information (e.g., `id`, `title`, `author`, etc.).
- Status: 404 Not Found, Body: An error message indicating requested books weren't found based on the provided filters.

3. **Add Book**

- Method**:** POST
- End Point: `/addBook`
- Description**:** Adds a new book to the collection.
- Authorization**:** Requires a valid JWT token.
- Request Body**:**
  `authorId` (required): The unique identifier of the book's author.
- Example Request**:**

```
{
"authorId": 123,
"title": "The Hitchhiker's Guide to the Galaxy",
"year": 1979,
“authorName”: “John”
}
```

- Validation**:** The request body is validated using the `addBookValidation` middleware using joi validator
- Response**:**

1. Status: 200 OK, Body: A success message: "Book added successfully!!"
2. Status: 400 Bad Request, Body: An error message: "Only registered author's book can be added" (returned if the provided author ID doesn't match a registered user).

4. **Add User**

- Method**:** POST
- End Point: /addUser
- Description**:** Adds a new user.
- Authorization**:** Requires a valid JWT token.
- Validation**:** The request body is validated using the `addUserValidation` middleware using joi validator.
- Response**:** Status: 200 OK, Body: Sends all users with new user.

5. Update Book by id:

- Method: Put
- EndPoint: /updateBook/:id
- Description: Updates a book.
- Authorization: Requires a valid JWT token
- Validation: The request body is validated using the `updateBookValidation` middleware using joi validator.
- Checks if the user is a super user or author of the book.
- Response: Status: 200 OK. Body: Send the updated book.

6. Delete Book by id:

Method: delete
End Point: /deleteBook
Description: Deletes a book either by id or multiple books of an author by authorId.
Authorization: Requires a valid JWT token
Validation: If to delete a book by id, checks if book exists or not,
If book with given id doesn't exists, then 404 “Book not found error is thrown”.
If to delete a book of an author, checks if books exists or not, If books with given authorId doesn't exists, then 404 “Book not found error thrown”