**API Design Specification**

**(*Library management* )**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 08-Nov-2022 | Sanket Aware (Flexur Systems) | Initial draft |
| 2.0 | 10-Nov-2023 | Sanket Aware (Flexur Systems) | First draft |

Github : https://github.com/Sanketaware/LibraryAPI.git

This project is an implementation of a very simple library management system.

We are implementing a book API which will have five endpoints in the server to perform the

specified operations.

## 

## 

Endpoints

[1.](#_bit7idv2ancx) Retrieve all Books

[Request](#_kd6tjl5w5wjo)

[Respons](#_8b7ijrpuamb8)e

[2](#_bit7idv2ancx). Retrieve all Recycled Books

[Request](#_kd6tjl5w5wjo)

[Respons](#_8b7ijrpuamb8)e

3.Get specific Book

[Request](#_iaps2dh7l2m2)

[Response](#_fvf6lulvf6wv)

4. Add Book

[Request](#_kd6tjl5w5wjo)

[Respons](#_8b7ijrpuamb8)e

[5. Edit/Update](#_9t094e45sg90) Book

[Request](#_38pypefkb82)

[Response](#_47awg09ehquu)

[6. Delet](#_9mc75gzfq3m3)e Book

[Request](#_xtszbus9yomc)

[Response](#_xuhh3y4ufz7)

[Usage Conventions](#_uk42zio712fb)

[Status Codes](#_of6ke88xwee4)

## 

Methods

## 1. Retrieve Books

Retrieve all the books from the server

## Request

| **Method** | **URL** |
| --- | --- |
| **GET** | {{BaseUrl}}/Books/ |

| **Type** | **Params** | **Values** |
| --- | --- | --- |
| Header | - | - |

## Response

| **Status** | **Response** |
| --- | --- |
| 200 | **Response will be an object containing the list of books (array) . Each item in the Books array has the following structure.**  [  {  "title": "Ficciones",  "author": "Jorge Luis Borges",  "id": "ZUST9JFx-Sd9X0k",  "genre": "fiction",  "yearPublished": 1944,  },  ] |
| 400 | {"error":"Bad Request."} |
| 500 | {"error":"Something went wrong. Please try again later."} |
| 503 | {"error":"Empty Shelf."} |

## 2. Retrieve Recycled Books

Retrieve all the books from the Recycled section of server

## Request

| **Method** | **URL** |
| --- | --- |
| **GET** | {{BaseUrl}}/RecycledBooks/ |

| **Type** | **Params** | **Values** |
| --- | --- | --- |
| Header | - | - |

## Response

| **Status** | **Response** |
| --- | --- |
| 200 | **Response will be an object containing the list of books from the Recycled Array .**  [  {  "title": "Ficciones",  "author": "Jorge Luis Borges",  "id": "ZUST9JFx-Sd9X0k",  "genre": "fiction",  "yearPublished": 1944,  },  ] |
| 400 | {"error":"Bad Request."} |
| 500 | {"error":"Something went wrong. Please try again later."} |
| 503 | {"error":"Empty Shelf."} |

## 3. Get specific book

Fetch the single required book

## Request

| **Method** | **URL** |
| --- | --- |
| **GET** | {{BaseUrl}}/books/:id |

| **Type** | **Params** | **Values** |
| --- | --- | --- |
| HEADER | id | number |

## Response

| **Status** | **Response** |
| --- | --- |
| 200 | **Response will be an object containing the specific book**  {  "title": "Ficciones",  "author": "Jorge Luis Borges",  "id": "ZUST9JFx-Sd9X0k",  "genre": "fiction",  "yearPublished": 1944  } |
| 400 | {"error":"Bad Request."} |
| 404 | {"error":"Book Not Found."} |
| 500 | {"error":"Something went wrong. Please try again later."} |

## 4. Add a book

Add a new book to the server database.

## Request

| **Method** | **URL** |
| --- | --- |
| **POST** | {{BaseUrl}}/books/ |

| **Type** | **Params** | **Values** |
| --- | --- | --- |
| REQUEST BODY | JSON | {  "title": "One Hundred Years,  "author": "Gabriel García Márquez",  "genre": "fiction",  "yearPublished": 1967  } |

## Response

| **Status** | **Response** |
| --- | --- |
| 200 | **{**  **"message": "Successfully Added , But also already exists in recycle bin"**  **}** |
| 201 | **{**  **"message": "Successfully Added"**  **}** |
| 400 | {"error":"Bad Request."} |
| 500 | {"error":"Something went wrong. Please try again later."} |
| 503 | {"error":"Book already exists."} |

## 

## 5. Edit/Update a book

Get more information on a particular recipe

## Request

| **Method** | **URL** |
| --- | --- |
| **PUT** | {{BaseUrl}}/books/:{id} |

| **Type** | **Params** | **Values** |
| --- | --- | --- |
| HEADER  REQUEST BODY | Id  JSON | Number  {  "title": "One Hundred Years,  "genre": "fiction"  }  Only title and Genre are editable fields |

## Response

| **Status** | **Response** |
| --- | --- |
| 200 | **{**  **"message": "OK"**  **}** |
| 400 | {"error":"Bad Request."} |
| 404 | {"error":"Book Not Found."} |
| 500 | {"error":"Something went wrong. Please try again later."} |

## 6. Delete a specific book

Delete the book specified from the Main Books Array and Add it to the Junk Array Which will be Cleared After 30 Days.

## Request

| **Method** | **URL** |
| --- | --- |
| **DELETE** | {{BaseUrl}}/books/:{id} |

| **Type** | **Params** | **Values** |
| --- | --- | --- |
| HEADER | id | number |

## 

## Response

| **Status** | **Response** |
| --- | --- |
| 200 | **{**  **"message": "DELETED"**  **}** |
| 400 | {"error":"Bad Request"} |
| 404 | {"error": "Book Not Found”} |
| 500 | {"error":"Something went wrong. Please try again later."} |

Glossary

## Conventions

* **Client** - Client application. [ POSTMAN for Mock Server testing ]
* **Status** - HTTP status code of response.
* All the possible responses are listed under ‘Responses’ for each method. Only one of them is issued per request server.
* All responses are in JSON format.
* All request parameters are mandatory unless explicitly marked as [optional]

## Status Codes

The Status codes used in the API are custom to cater the applications needs in various methods. Below is the reference to Standard codes and error messages.

Standard HTTP status codes.

2XX - Success of some kind

4XX - Error occurred in client’s part

5XX - Error occurred in server’s part

| **Status Code** | **Description** |
| --- | --- |
| 200 | OK |
| 201 | Created |
| 202 | Accepted (Request accepted, and queued for execution) |
| 400 | Bad request |
| 404 | Resource not found |
| 500 | Internal Server Error |
| 503 | Service Unavailable |