**BOOK LIBRARY – API**

****

**Overview**

A Book Library needs a REST API to be created. The library has a few books and some subscribers. Every time a subscriber walks in to the Library, the Librarian will check the list of Books for availability of copies. Once a Subscriber picks a book a subscription entry will be made by the Librarian to record it.

**Database Tables -**  
The data will be stored in the following two Database tables -

**USER**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **SUBSCRIBER\_NAME** | **PASSWORD** | **EMAIL** |
| John | John Millar | ASDJAS1@21SDDEEEE | abc@xyz.com |
| Mark | Mark Waugh | ASDJAS12#231SDDEEE | [abc2@pqr.com](mailto:abc2@pqr.com) |
| Peter | Peter Parker | ASDJ1231A@S121SDD | abc3@lmn.com |

**BOOK**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BOOK\_ID** | **BOOK\_NAME** | **AUTHOR** | **AVAILABLE\_COPIES** | **TOTAL\_COPIES** |
| B1212 | History of Amazon Valley | Ross Suarez | 0 | 2 |
| B4232 | Language Fundamentals | H S Parkmay | 5 | 5 |

**SUBSCRIPTION**

|  |  |  |  |
| --- | --- | --- | --- |
| **SUBSCRIBER\_NAME** | **DATE\_SUBSCRIBED** | **DATE\_RETURNED** | **BOOK\_ID** |
| John | 12-JUN-2020 |  | B1212 |
| Mark | 26-APR-2020 | 14-May-2020 | B4232 |
| Peter | 22-JUN-2020 |  | B1212 |



Please create the following **three** REST APIs (using a SpringBoot application) to help the Librarian with his daily functions. The APIs will be used to fetch a list of Books, fetch a list of Subscriptions and create a new Subscription record.

**Project # 1: Books Services**

|  |  |
| --- | --- |
| URL | GET /books |
| Response Body | Array of Book objects. Each object would contain details of the Book. |
| Response JSON | [  {  bookId : “string”,  name : “string”,  author : “string”,  copiesAvailable : number,  totalCopies : number  }  ] |
| Response Http Status code | 200 |

|  |  |
| --- | --- |
| URL | GET /books/{bookId} |
| Response Body | Return Object would contain details of the given Book Id. |
| Response JSON | [  {  bookId : “string”,  name : “string”,  author : “string”,  copiesAvailable : number,  totalCopies : number  }  ] |
| Response Http Status code | 200 |

|  |  |
| --- | --- |
| URL | POST / books/UpdateAvailability/{bookId}/{incremental\_count} |
| Response Body | Return Object would contain details of the given Book Id. |
| Response JSON | [  {  bookId : “string”,  name : “string”,  author : “string”,  copiesAvailable : number,  totalCopies : number  }  ] |
| Response Http Status code | 200 – Successful creation of subscription record  500 – if bookid is not found in BOOK table |
| Notes | * Add AVAILABLE\_COPIES with “incremental\_count” for given bookid * If someone has opted for “notify” for this bookid, send an email to him if AVAILABLE\_COPIES is greater than zero now. Look “POST /subscriptions” for more detail on this. |

**Project # 2: Subscription Services**

|  |  |
| --- | --- |
| URL | GET /subscriptions |
| Response Body | Array of Subscription objects. Each object would contain details of the books subscribed. |
| Optional Query parameter | subscriberName.  If provided, show the record for that particular subscriber name. |
| Response JSON | [  {  subscriberName : “string”,  bookId : “string”,  dateSubscribed : date,  dateReturned : date  }  ] |
| Response Http Status code | 200 |

|  |  |
| --- | --- |
| URL | POST /subscriptions |
| Request Body | A single Subscription object. |
| Request body JSON | {  subscriberName : “string”,  bookId : “string”,  dateSubscribed : date,  notify : “no”  } |
| Response Http Status code | 201 – Successful creation of subscription record  422 – If book copies not available for subscription. |
| Notes | * Create a record in the SUBSCRIPTION table. * For the given bookId check if one or more copies are available using “books” services (Don’t Hit DB). * Update AVAILABLE\_COPIES with **“-1”** using “/book/UpdateAvailability” service * If not available, then fail the operation. * If AVAILABLE\_COPIES =0 and “notify” is “yes” then put this in (Message queue/topic), so that user will be notified when anybody will return the same book. |

|  |  |
| --- | --- |
| URL | POST /returns |
| Request Body | A single Subscription object. |
| Request body JSON | {  subscriberName : “string”,  bookId : “string”,  dateSubscribed : date,  dateReturned : date  } |
| Response Http Status code | 201 – Successful creation of subscription record |
| Notes | * Update a record in the SUBSCRIPTION table for return book. * Update AVAILABLE\_COPIES with +**1** using “/book/UpdateAvailability” service |

**NOTES -**

* Implement both services using SpringBoot.
* Follow the standard pattern of - REST Controller -> Service -> Repository
* Use any Database or in-memory database of your choice.
* “books” and “subscriptions” needs to be created and deployed as different projects and services
* Use API Gateway and Service Registry
* Use External Configuration to read service registry SERVER URI via spring cloud bus.
* Use RestTemplate to call an external API
* Use JWT token to secure services endpoints.
* User Kafka for “notify” functionality
* Implement Centralize logging and monitoring for each microservice via ELK stack.
* Create/run 3 instances of “books” services
* Create/run 2 instances of “subscriptions” services
* Use Docker to deploy multiple instances

**IF INTERESTED, PLEASE ALSO TRY -**

If you are interested in trying out a few more things, please implement the below -

* Basic Authentication – add a security layer to authenticate the REST Services
* Use Circuit Breaker in case of “returns” service is unable to get response from “UpdateAvailability”. Or use Asynchronous call (using Kafka) for “UpdateAvailability” service

**COMLETION DURATION: 4 WEEKS**