I barely have 5 hours on this problem since I have 2 midterms, several assignment dues and regular class to attend at the same day. I apology for my delay.

Here are some functions that I believe I could (and am able to) implement, if the time was not this short.

* More error catching
* Rate limit using Redis
* Database migration automation
* Relationships table foreign key relationship
* Input validation check (ISBN, Email address, etc.)
* Flask-Blueprint for API version management
* GAE deployment and SSL(https)

Thank you for your time.

[Intentionally Blank]

Reading List REST API Documentation

The reading list API allows users to manage their private and public reading lists, and to share and explore them.

Base URL

All URLs referenced in the documentation have the following base:

|  |
| --- |
| http://127.0.0.1:5000/ |

To ensure data privacy, the API is also served over HTTPS.

Errors

You will see a HTML 4XX status code and an error message if an exception happens.

|  |
| --- |
| 403 Forbidden  {  “error”: “Wrong user, permission denied.”  } |

User Authenication

Create User

Make a HTTP POST to:

|  |
| --- |
| /register |

with the following parameters (in JSON):

|  |
| --- |
| {  "email": "ok@ok.com",  "password": "python"  } |

API would respond:

|  |
| --- |
| {  "user\_id": 10  } |

indicating that a registration is successful, and a user\_id is created.

Login and Get a Token

Make a HTTP request to:

|  |
| --- |
| /user |

with your login credentials, API would respond with your home address and a token:

|  |
| --- |
| 200 OK  {  "home": "/user/6",  "token": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ4ODM1MjkzNCwiaWF0IjoxNDg4MzUyMzM0fQ.eyJpZCI6Nn0.pBES4gYM3xCytdq4sEjYkAbomReiIX1HXbP49Y0kxGc"  } |

Note that by default all functions does not accept credentials but a token, and a token is valid within 10 minutes. Make another request once a token is expired.

List Management (Token Needed)

Read All Lists (User Home)

Make a HTTP GET to:

|  |
| --- |
| /user/<user\_id> |

with your token, and get all lists infomation:

|  |
| --- |
| 200 OK  {  "lists": [  {  "address": "/list/1",   "isprivate": true,   "list\_id": 1  },   {  "address": "/list/2",   "isprivate": false,   "list\_id": 2  }  ],   "owner": {  "user\_id": 6  } } |

Create a New List

Make a HTTP POST to:

|  |
| --- |
| /user/<user\_id> |

with your token and properties of the new list:

|  |
| --- |
| {  "isprivate": true  } |

API will return the following message if succeeded.

|  |
| --- |
| 201 CREATED  {  "address": "/list/3",  "isprivate": true,  "list\_id": 3  } |

Delete a List and All of its Contents

Make a HTTP DELETE to:

|  |
| --- |
| /user/<user\_id> |

with your token and properties of the list to be deleted:

|  |
| --- |
| {  "list\_id": 2  } |

API will return the following message if succeeded.

|  |
| --- |
| 201 CREATED  {  “status”: “List successfully deleted.”  } |

Show All Books in a List

Make a HTTP GET to:

|  |
| --- |
| /list/<list\_id> |

with your token, and get all books in a list with brief information.

|  |
| --- |
| 200 OK  {  "books": [  {  "author": "sample\_author\_1",  "book\_id": 1,  "title": "sample\_title\_1"  },  {  "author": "sample\_author\_2",  "book\_id": 3,  "title": "sample\_title\_2"  }  ],  "list\_info": {  "isprivate": false,  "list\_id": 2,  "user\_id": 6  } } |

Add a Book to the List

Make a HTTP POST to:

|  |
| --- |
| /list/<list\_id> |

with your token and properties of the new book:

|  |
| --- |
| {  "isbn": "0000000000000",  "title": "sample\_title\_1",  "author": "sample\_author\_1",  "category": "sample\_category\_1",  "coverurl": "sample\_url\_1",  "summary": "sample\_summary\_1"  } |

API will return the following message if succeeded.

|  |
| --- |
| 201 CREATED  {  "book\_id": 1  } |

Note: If you add a book with the same ISBN with an existing book, a new book will not be created and the existing one will be linked to the reading list.

Delete a Book in the List

Make a HTTP DELETE to:

|  |
| --- |
| /list/<list\_id> |

with your token and properties of the book to be deleted:

|  |
| --- |
| {  “book\_id”: 2  } |

API will return the following message if succeeded.

|  |
| --- |
| 201 CREATED  {  "status": "Book successfully deleted." } |

Note that if a book is shared within multiple lists, only the relationship will be deleted.

Return Details of a Book

Make a HTTP GET to:

|  |
| --- |
| /book/<book\_id> |

and get back the details:

|  |
| --- |
| 200 OK  {  "author": "sample\_author\_2",  "book\_id": 3,  "category": "sample\_category\_2",  "coverurl": "sample\_url\_2",  "isbn": "0000000000001",  "summary": "sample\_summary\_2",  "title": "sample\_title\_2"  } |

Discovery for Anonymus Users

See Public Reading Lists

Make a HTTP GET to:

|  |
| --- |
| /discovery |

and get back the public reading lists:

|  |
| --- |
| 200 OK |

See Details of a Public Reading List

Make a HTTP GET to:

|  |
| --- |
| /discovery/<list\_id> |

and get back the details of a public reading list:

|  |
| --- |
| 200 OK  {  "books": [  {  "author": "sample\_author\_1",  "book\_id": 1,  "title": "sample\_title\_1"  }  ],  "list\_info": {  "list\_id": "2"  } } |

See Book Details

The behavior is similar to registered users.