

Library Management System – Postman API Documentation

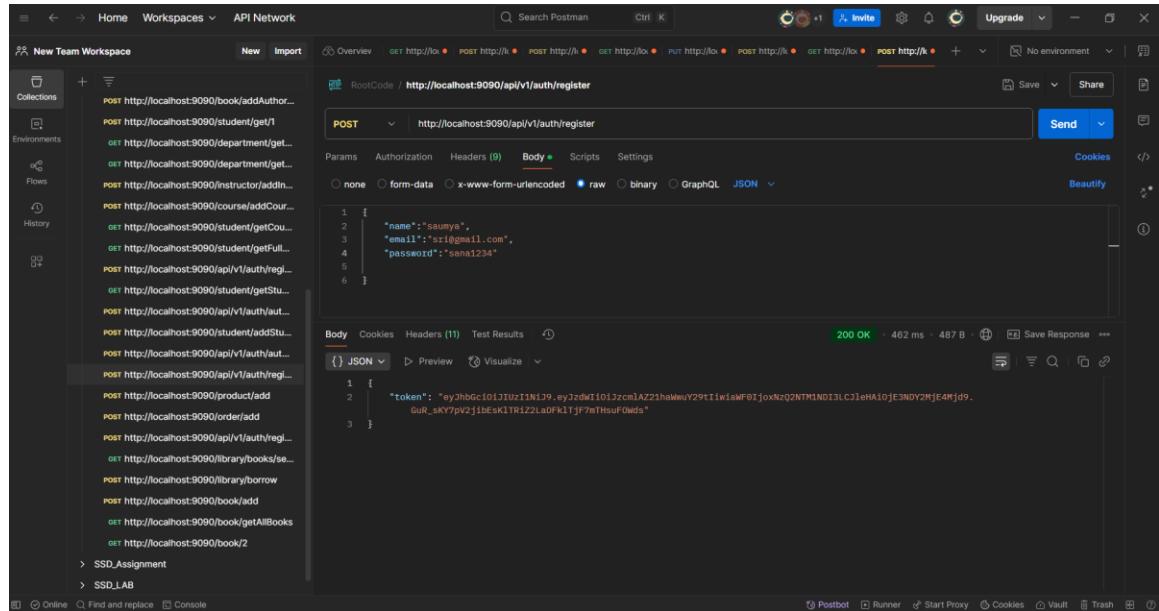
User APIs

User Register API

Endpoint: POST /api/v1/auth/register

Request Body:

```
{  
  "name": "John Doe",  
  "email": "john@example.com",  
  "password": "securePassword"  
}
```



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'New Team Workspace' selected, showing various collections like 'Collections', 'Environments', 'Flows', and 'History'. The main workspace area has a list of API endpoints for a local host. A specific POST request to 'http://localhost:9090/api/v1/auth/register' is selected. The 'Body' tab is active, showing a JSON payload with three fields: 'name' (set to 'saumya'), 'email' (set to 'saumya@gmail.com'), and 'password' (set to 'sana1234'). Below the body, the response status is '200 OK' with a response time of 462 ms and a size of 487 B. The response body is a JSON object containing a single key 'token' with a long string value.

```
{}  
{"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ2cmFkaWwuY29tLmluaWF0IjoxNzQ2NTM1NDI3LCJleHAiOjE3NDY2MjE4Mjg9.GuR_SKV7pv2JibEsK1TRIZ2te0fK1TjP7wHsuFDwds"}
```

User Login API

Endpoint: POST /api/v1/auth/authenticate

Request Body:

```
{  
  "email": "john@example.com",  
  "password": "securePassword"  
}
```

Response: JWT Token

The screenshot shows the Postman interface with a collection named 'New Team Workspace'. A specific POST request to `http://localhost:9090/api/v1/auth/authenticate` is selected. The request body is set to JSON with the following content:

```
1 {  
2   "email": "sana2@gmail.com",  
3   "password": "sana1234"  
4 }
```

The response status is **200 OK**, and the response body contains a single key-value pair:

```
1 {  
2   "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzYmshMkdnbeFpbC5jb280LClC3pYXQiOjE3NDY1Mjg2MjgsImV4cCI6MTc0MjYxNTAyOH0.  
Przg5t6vUM72BaR96o4u-JG8d3fKhZae-E8Bnevx8"  
3 }
```

Get Logged-in User Info

Endpoint: GET /user/{userId}

Header: Authorization: Bearer <JWT>

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:9090/user/1`. The response status is `200 OK` with a response time of 76 ms and a size of 384 B. The response body is a JSON object:

```

1 | {
2 |   "id": 1,
3 |   "name": "sanaya",
4 |   "email": "sana2@gmail.com"
5 |

```


The screenshot shows the Postman interface with an unsuccessful API call. The URL is `http://localhost:9090/user/7`. The response status is `404 Not Found` with a response time of 512 ms and a size of 469 B. The response body is a JSON object:

```

1 | {
2 |   "error": "User Not Found",
3 |   "message": "user with id: 7 could not be found",
4 |   "timestamp": "2025-05-06T17:42:24.685388Z",
5 |   "status": 404
6 |

```

Get All Users

Endpoint: GET /user/getAllUsers

Header: Admin access required

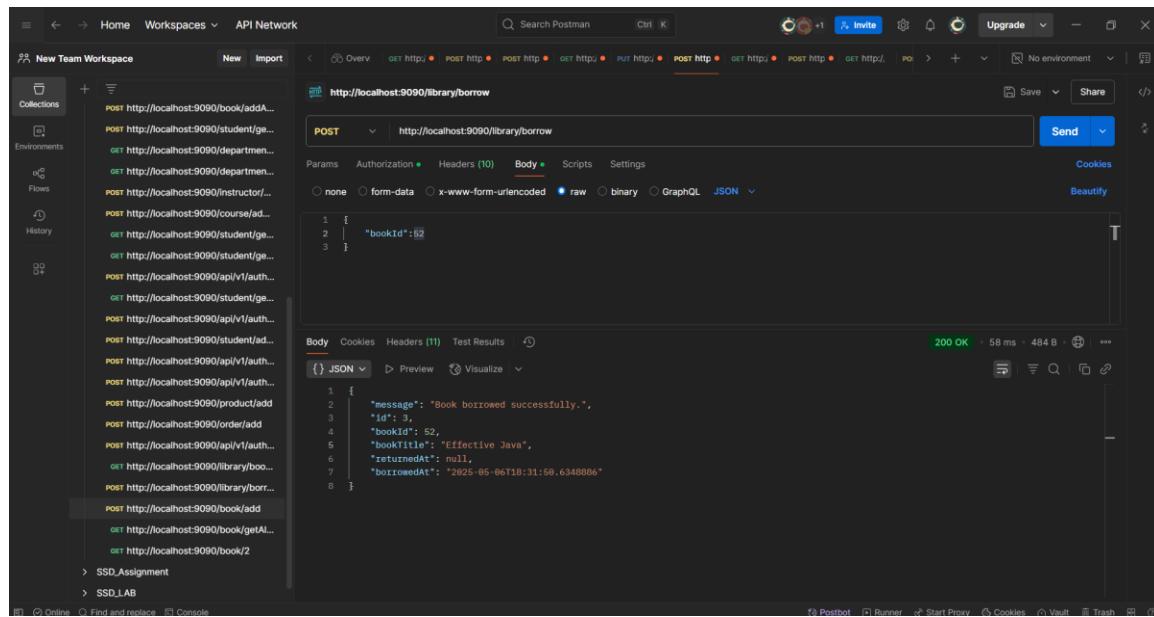
Book APIs

Borrow a Book

Endpoint: POST /library/borrow

Request Body:

```
{  
  "bookId": 1  
}
```



The screenshot shows the Postman interface with a POST request to `http://localhost:9090/library/borrow`. The request body is set to `JSON` and contains the following data:

```
{  
  "bookId": 1  
}
```

The response status is `200 OK` with a response time of `58 ms` and a size of `484 B`.

Return a Book

Endpoint: POST /library/return

Request Body:

```
{  
  "bookId": 1  
}
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'New Team Workspace' selected, showing various API endpoints like 'post http://localhost:9090/book/addA...', 'post http://localhost:9090/student/get...', etc. The main area has a tab for 'http://localhost:9090/library/return'. A POST request is selected with the URL 'http://localhost:9090/library/return'. The 'Body' tab is active, showing a JSON payload: { "bookId": 1 }. Below the body, the response status is '200 OK' with a duration of '772 ms' and a size of '505 B'. The response body is displayed as JSON: { "message": "Book returned successfully.", "id": 1, "bookId": 1, "bookTitle": "Harry Potter", "returnedAt": "2025-05-06T16:25:06.6858975", "borrowedAt": "2025-05-06T16:21:09.296623" }.

Borrowing History

Endpoint: GET [/library/history](#)

```
{  
  "email": "sana@gmail.com,  
  
  "password":1234  
}
```

The screenshot shows the Postman interface with a collection named "New Team Workspace". A GET request is made to `http://localhost:9090/library/history`. The request body is a JSON object:

```
1: {  
2:   "email": "sana2@gmail.com",  
3:   "password": "sana1234"  
4: }
```

The response status is 200 OK, with a response time of 133 ms and a size of 380 B. The response body is a JSON array containing one element:

```
1: [  
2:   "Harry Potter on 2025-05-06T18:21:09.296623"  
3: ]
```

Get All Available Books

Endpoint: GET /library/books

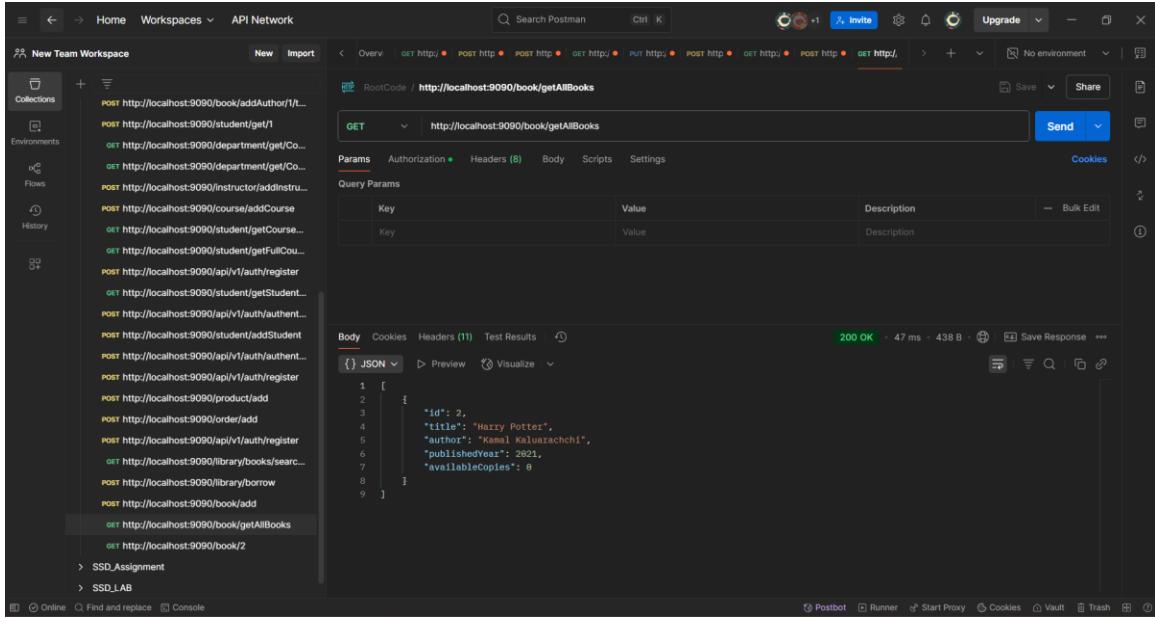
The screenshot shows the Postman interface with a collection named "New Team Workspace". A GET request is made to `http://localhost:9090/library/books`. The request body is empty, indicated by the message "This request does not have a body".

The response status is 200 OK, with a response time of 738 ms and a size of 435 B. The response body is a JSON array containing one element:

```
1: [  
2:   {  
3:     "id": 52,  
4:     "title": "Effective Java",  
5:     "author": "Joshua Bloch",  
6:     "publishedYear": 2018,  
7:     "availableCopies": 3  
8:   }  
9: ]
```

Get All Books

Endpoint: GET /book/getAllBooks

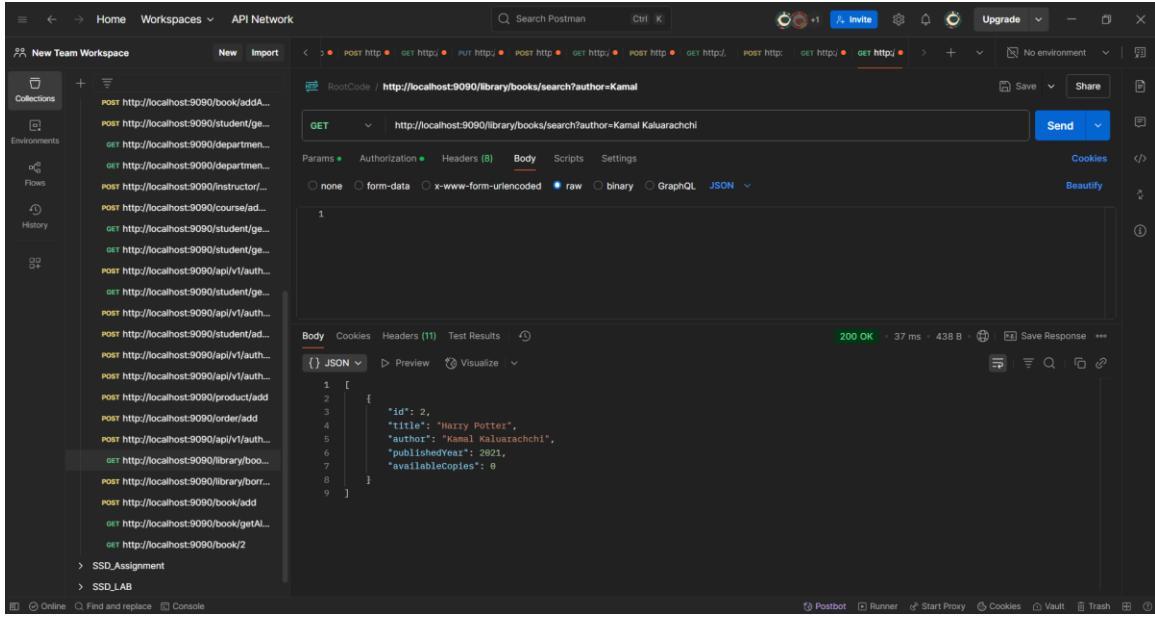


The screenshot shows the Postman interface with the 'RootCode' collection selected. The 'getAllBooks' endpoint is highlighted in the list. The request details show a GET method to 'http://localhost:9090/book/getAllBooks'. The response body is displayed as JSON, showing a single book entry:

```
[{"id": 2, "title": "Harry Potter", "author": "Kamal Kaluarachchi", "publishedYear": 2021, "availableCopies": 0}]
```

Search Books by Author

Endpoint: GET library/books/search?author=Kamal



The screenshot shows the Postman interface with the 'RootCode' collection selected. The 'search?author=Kamal' endpoint is highlighted in the list. The request details show a GET method to 'http://localhost:9090/library/books/search?author=Kamal'. The response body is displayed as JSON, showing the same book entry as the previous screenshot:

```
[{"id": 2, "title": "Harry Potter", "author": "Kamal Kaluarachchi", "publishedYear": 2021, "availableCopies": 0}]
```

Search Books by Published Year

Endpoint: GET /library/books/search?year=2021

The screenshot shows the Postman interface with a collection named "New Team Workspace". A specific POST request for adding a book is visible in the list. The main focus is a GET request to `http://localhost:9090/library/books/search?year=2021`. The "Body" tab is selected, showing an empty JSON object. The "Headers" tab shows the URL with the query parameter. The "Test Results" tab indicates a successful 200 OK response with a response time of 281 ms and a size of 438 B. The response body is displayed as JSON:

```
[{"id": 2, "title": "Harry Potter", "author": "Kamal Kaluzachchi", "publishedYear": 2021, "availableCopies": 0}]
```

Add New Book

Endpoint: POST /books/add

Request Body:

```
{  
  "title": "Effective Java",  
  "author": "Joshua Bloch",  
  "publishedYear": 2018,  
  "availableCopies": 5  
}
```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'New Team Workspace' selected, showing various collections like 'RootCode', 'SSD_Assignment', and 'SSD_LAB'. The main area displays a POST request to 'http://localhost:9090/book/add'. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   "title": "Effective Java",
3   "author": "Joshua Bloch",
4   "publishedYear": 2018,
5   "availableCopies": 5
6 }
```

Below the request, the response is shown as a 200 OK status with a 101 ms duration and 433 B size. The response body is identical to the request body.

Edit Existing Book

Endpoint: PUT /books/edit/{bookId}

Request Body:

```
{
  "title": "Clean Code",
  "author": "Robert C. Martin",
  "publishedYear": 2008,
  "availableCopies": 3
}
```

The screenshot shows the Postman interface with a PUT request to `http://localhost:9090/book/edit/2`. The request body is a JSON object:

```

1 {
2   "id": 2,
3   "title": "Harry Potter",
4   "author": "Kamel Kaluarachchi",
5   "publishedYear": 2021,
6   "availableCopies": 6
7 }

```

The response is a 200 OK status with a response time of 39 ms and a size of 436 B.

Delete Existing Book

Endpoint: PUT /books/edit/{bookId} }

The screenshot shows the Postman interface with a DELETE request to `http://localhost:9090/book/delete/2`. The request body is empty, indicated by the message "This request does not have a body".

The response is a 200 OK status with a response time of 540 ms and a size of 436 B.

POSTMAN COLLECTION

The screenshot shows the Postman application interface with a dark theme. At the top, there are navigation icons (three horizontal lines, back, forward) and menu items: Home, Workspaces (with a dropdown arrow), and API Network. Below the header, the title "New Team Workspace" is displayed, along with "New" and "Import" buttons. On the left side, there is a sidebar with icons for Collections (selected), Environments, Flows, History, and a plus sign for creating new items. The main area lists several API requests:

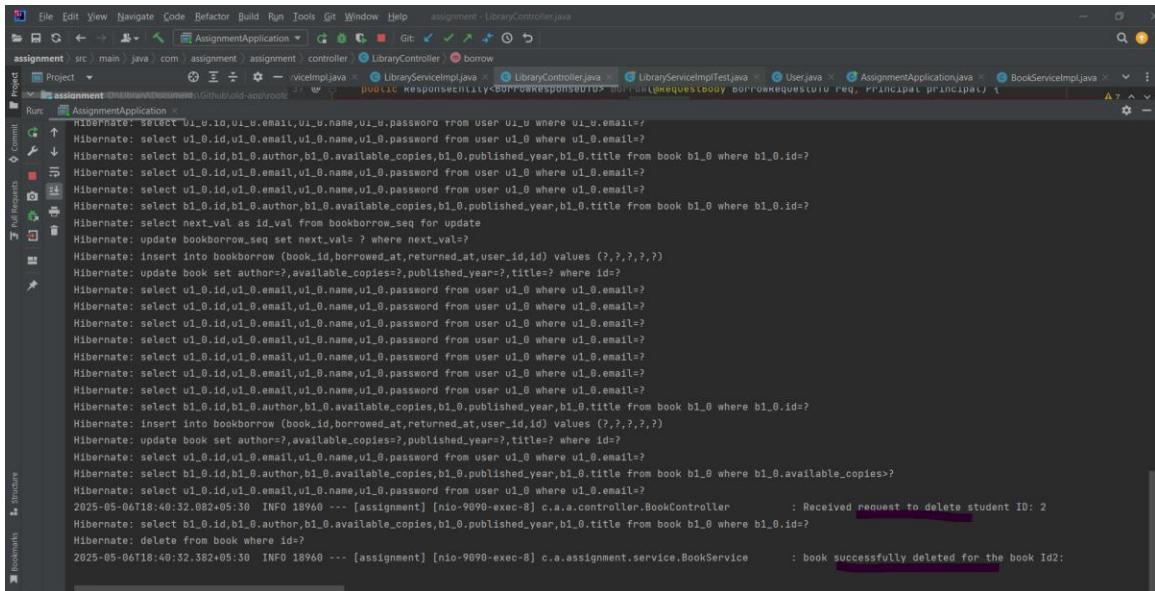
- POST `http://localhost:9090/student/get/1`
- GET `http://localhost:9090/department/get/Computing`
- GET `http://localhost:9090/department/get/Computing`
- POST `http://localhost:9090/instructor/addInstructor`
- POST `http://localhost:9090/course/addCourse`
- GET `http://localhost:9090/student/getCourses/for/2`
- GET `http://localhost:9090/student/getFullCourseHours/for/2`
- POST `http://localhost:9090/api/v1/auth/register`
- GET `http://localhost:9090/student/getStudents/for/52`
- POST `http://localhost:9090/api/v1/auth/authenticate`
- POST `http://localhost:9090/student/addStudent`
- POST `http://localhost:9090/api/v1/auth/authenticate`
- POST `http://localhost:9090/api/v1/auth/register`
- POST `http://localhost:9090/product/add`
- POST `http://localhost:9090/order/add`
- POST `http://localhost:9090/api/v1/auth/register`
- GET `http://localhost:9090/library/books/search?author=Kamal`
- POST `http://localhost:9090/library/borrow`
- POST `http://localhost:9090/book/add`
- GET `http://localhost:9090/book/getAllBooks`
- GET `http://localhost:9090/book/2`

Below the list, there are three collapsed sections:

- > SSD_Assignment
- > SSD_LAB
- > UserManagement

At the bottom of the interface, there are status indicators: "Online" (with a checked checkbox), "Find and replace" (with a magnifying glass icon), and "Console" (with a terminal icon).

Loggers



The screenshot shows a Java IDE interface with multiple tabs open. The active tab is 'AssignmentApplication' under the 'Run' section. The log output is displayed in the bottom half of the window, showing various Hibernate SQL queries and application logs. Key logs include:

```
Hibernate: select v1_0.id,v1_0.email,v1_0.name,v1_0.password from user v1_0 where v1_0.email=?
Hibernate: select b1_0.id,b1_0.author,b1_0.available_copies,b1_0.published_year,b1_0.title from book b1_0 where b1_0.id=?
Hibernate: select v1_0.id,v1_0.email,v1_0.name,v1_0.password from user v1_0 where v1_0.email=?
Hibernate: insert into bookborrow (book_id,borrowed_at,returned_at,user_id,id) values (?, ?, ?, ?, ?)
Hibernate: update book set author=?,available_copies=?,published_year=?,title=? where id=?
Hibernate: select v1_0.id,v1_0.email,v1_0.name,v1_0.password from user v1_0 where v1_0.email=?
Hibernate: insert into bookborrow (book_id,borrowed_at,returned_at,user_id,id) values (?, ?, ?, ?, ?)
Hibernate: update book set author=?,available_copies=?,published_year=?,title=? where id=?
Hibernate: select v1_0.id,v1_0.email,v1_0.name,v1_0.password from user v1_0 where v1_0.email=?
Hibernate: select b1_0.id,b1_0.author,b1_0.available_copies,b1_0.published_year,b1_0.title from book b1_0 where b1_0.id=?
2025-05-06T18:40:32.082+05:30 INFO 18960 --- [assignment] [nio-9090-exec-8] c.a.a.controller.BookController : Received request to delete student ID: 2
Hibernate: select b1_0.id,b1_0.author,b1_0.available_copies,b1_0.published_year,b1_0.title from book b1_0 where b1_0.id=?
Hibernate: delete from book where id=?
2025-05-06T18:40:32.382+05:30 INFO 18960 --- [assignment] [nio-9090-exec-8] c.a.assignment.service.BookService : book successfully deleted for the book Id2:
```