Library Management System

I. Setup and Run Instructions

Prerequisites

- Node.js (v18+)
- npm
- Docker
- Git

Environment Configuration

Create a `.env` file in the project root with the following content:

```
DB_HOST=mysql
DB_USER=root
DB_PASSWORD=root
DB_NAME=library_db
DB_PORT=3306
BASIC_AUTH_USER=admin
BASIC_AUTH_PASS=admin
```

Running with Docker Compose

```
docker-compose up --build
```

- Containers started:
    - `library_app` (Node.js application)
    - `library_mysql` (MySQL database with tables from `mysql-init/init.sql`)
- Access API: http://localhost:3000

II. API Documentation

1. Books

Add a Book

- **Method:** POST
- **URL:** `/books`
- **Description:** Adds a new book to the library.

Request Body:

```
{
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald",
  "isbn": "1234567890",
  "available_quantity": 5,
  "shelf_location": "A1"
}
```

Response (Success):

```
{
  "success": true,
  "book": {
    "id": 1,
    "title": "Book Title",
    "author": "Author",
    "isbn": "12345",
    "available_quantity": 5,
    "shelf_location": "A1",
    "created_at": "...",
    "updated_at": "..."
  }
}
```

Response (Error):

```
{
  "success": false,
  "error": "Book addition failed: <error message>"
}
```

Update a Book

- **Method:** PUT
- **URL:** `/books/:id`
- **Description:** Updates book details by ID.

Request Body: Same as Add Book (all fields optional except id in URL)

Response (Success):

```
{
  "success": true,
  "book": { ...updated book object... }
}
```

Response (Not Found):

```
{
  "success": false,
  "error": "Book not found"
}
```

Delete a Book

- **Method:** DELETE
- **URL:** `/books/:id`
- **Description:** Deletes a book by ID.

Response (Success):

```
{ "success": true }
```

Response (Not Found):

```
{ "success": false, "error": "Book not found" }
```

List All Books

- **Method:** GET
- **URL:** `/books`
- **Description:** Returns all books (cached).

Response (Success):

```
{
  "success": true,
  "books": [ {...}, {...} ]
}
```

Search Books

- **Method:** GET
- **URL:** `/books/search?query=<search_term>`
- **Description:** Search books by title, author, or ISBN (cached per query).

Response (Success):

```
{
  "success": true,
  "books": [ {...}, {...} ]
}
```

---

2. Borrowers

Add/Register a Borrower

- **Method:** POST
- **URL:** `/borrowers`
- **Description:** Registers a new borrower.

Request Body:

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "registered_date": "2026-01-14"
}
```

Response (Success/Error): Similar structure to Books.

Update Borrower

- **Method:** PUT
- **URL:** `/borrowers/:id`

Request Body:

```
{
  "name": "New Name",
  "email": "newemail@example.com"
}
```

Responses: Success / Not found same as Books.

Delete Borrower

- **Method:** DELETE
- **URL:** `/borrowers/:id`
- **Response:** Same as Books.

List Borrowers

- **Method:** GET
- **URL:** `/borrowers`
- **Response:** Returns all borrowers (cached).

---

3. Borrowings

Checkout Books

- **Method:** POST
- **URL:** `/borrowings/checkout`
- **Description:** Allows a borrower to check out multiple books.

Request Body:

```
{
  "borrowerId": 1,
  "books": [
    { "bookId": 1, "dueDate": "2026-01-30" },
    { "bookId": 2, "dueDate": "2026-01-30" }
  ]
}
```

Response (Success/Error):

```
{
  "success": true,
  "results": [
    { "bookId": 1, "success": true },
    { "bookId": 2, "success": false, "error": "Book not available" }
  ]
}
```

Return Books

- **Method:** POST
- **URL:** `/borrowings/return`

- **Request Body:**

```
{ "borrowingIds": [1,2,3] }
```

Response: Similar structure to checkout.

List Borrowed Books

- **Method:** GET
- **URL:** `/borrowings/borrowed/:borrowerId`
- **Description:** Returns currently borrowed books for a borrower (cached).

List Overdue Books

- **Method:** GET
- **URL:** `/borrowings/overdue`
- **Description:** Returns overdue books (cached).

---

4. Reports

Borrowing Report

- **Method:** GET
- **URL:** `/reports/borrowings?startDate=<>&endDate=<>&exportFormat=csv|xlsx`
- **Description:** Export borrowings in a date range.

Overdue Last Month

- **Method:** GET
- **URL:** `/reports/overdue-last-month?exportFormat=csv|xlsx`

Borrowings Last Month

- **Method:** GET
- **URL:** `/reports/borrowings-last-month?exportFormat=csv|xlsx`

Notes for All Reports:

- `exportFormat` defaults to `csv`. Can also use `xlsx`.
- If no data, response:

```
{ "success": true, "message": "No data available", "data": [] }
```

What’s next?

1. Adding bulk queries for adding borrowing entries instead of hitting the database per pair.

2. Adding pagination for responses to avoid sending an overwhelming response and for in-memory cache to be reasonable.