Library Management System Documentation

Project Overview:

The **Library Management System** is a RESTful API built with Node.js and MySQL for managing books, borrowers, and the borrowing process in a library. It allows users to add books, register borrowers, and track borrowing activities. The system ensures basic authentication for certain endpoints and includes additional features like overdue book tracking and CSV export functionality. The project is fully containerized using Docker for easy deployment.

Key Features:

1. **Books Management**:
   - Add a book with details like title, author, ISBN, available quantity, and shelf location.
   - Update or delete a book.
   - List all books or search for a book by title, author, or ISBN.
2. **Borrowers Management**:
   - Register a borrower with details like name, email, and registered date.
   - Update or delete a borrower.
   - List all registered borrowers.
3. **Borrowing Process**:
   - A borrower can borrow a book, and the system keeps track of the borrowed book and due dates.
   - A borrower can return a book, updating the book's availability in the system.
   - Track the books currently borrowed by a borrower.
   - List overdue books (books not returned by their due date).
4. **CSV Export**:
   - Export borrowing processes in CSV format.
   - Export overdue books data from the last month.
5. **Authentication**:
   - Basic authentication using username `admin` and password `let.me.in` to protect specific API routes.

6. **Rate Limiting**:
    - Prevent API abuse by limiting certain endpoints (e.g., POST requests for adding books and borrowers) to 3 requests per 5 seconds.
7. **Dockerized**:
    - The entire application is containerized using Docker with a MySQL database running in a separate container.
    - Running the project with `docker-compose` automatically sets up the containers for the Node.js app and MySQL database.

Technologies Used:

- **Node.js**: Server-side JavaScript runtime for building the application.
- **Express.js**: Web framework for building the RESTful API.
- **Sequelize**: ORM for connecting and interacting with MySQL.
- **MySQL**: Relational database for storing books, borrowers, and borrowing records.
- **Docker**: Containerization tool for easy deployment of the Node.js application and MySQL database.
- **Express-Rate-Limit**: Middleware for implementing rate limiting on certain API endpoints.
- **Basic-Auth**: Middleware for securing specific API routes with username/password authentication.
- **Moment.js**: Library for handling dates, particularly for calculating due dates and exporting records from a specific time range.
- **json2csv**: Library for exporting data to CSV format.
- **Jest**: Testing framework for writing and running unit tests.

Installation & Setup:

1. **Clone the Repository**:

    ```
    git clone <https://github.com/your-username/library-management-system.
    git>
    cd library-management-system
    ```

2. **Environment Variables**:
    Create a
    `.env` file in the project's root directory with the following content:

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=yourpassword
DB_NAME=library_management
DB_DIALECT=mysql
```

3. **Run the Application Locally** (Without Docker):
   - Install the necessary dependencies:

     ```
     npm install
     ```

   - Make sure MySQL is installed and running on your machine.
   - Run the migrations to set up the database tables:

     ```
     npx sequelize-cli db:migrate
     ```

   - Start the application:

     ```
     nodemon app.js
     ```

4. **Running the Application with Docker**:
   - If Docker is installed, use the following command to build and run the app and database containers:

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

   - This will set up the app and MySQL database in separate containers.
   - **Note**: Migrations will automatically run when Docker starts up.
5. **Testing the Application**:
   - **With Postman**:
     - Import the provided Postman collection (if available) or manually test the following API endpoints:
     - **Books API**:
       - `POST /books`: Add a new book.
       - `PUT /books/:id`: Update book details.
       - `DELETE /books/:id`: Delete a book.

        - `GET /books`: List all books.
        - `GET /books/search`: Search books by title, author, or ISBN.
    - **Borrowers API**:
        - `POST /borrowers`: Register a new borrower.
        - `PUT /borrowers/:id`: Update borrower details.
        - `DELETE /borrowers/:id`: Delete a borrower.
        - `GET /borrowers`: List all borrowers.
    - **Borrowing Process API**:
        - `POST /borrows`: Borrow a book.
        - `PUT /returns/:borrow_id`: Return a book.
        - `GET /borrows/:borrower_id`: List books borrowed by a specific borrower.
        - `GET /overdue`: List overdue books.
        - **Bonus**:
            - `GET /export-csv`: Export borrowing processes to CSV.
            - `GET /export-overdue-csv`: Export overdue borrows of the last month to CSV.

API Documentation:

Books API:

1. **Add a New Book**:
    - **Method**: `POST /books`
    - **Input** (JSON):

    ```
    {
      "title": "The Great Gatsby",
      "author": "F. Scott Fitzgerald",
      "ISBN": "9780743273565",
      "available_quantity": 5,
      "shelf_location": "A1"
    }
    ```

    - **Output** (Success 201):

    ```
    {
      "id": 1,
    ```

```
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald",
  "ISBN": "9780743273565",
  "available_quantity": 5,
  "shelf_location": "A1",
  "createdAt": "...",
  "updatedAt": "..."
}
```

2. **List All Books**:
   - **Method**: `GET /books`
   - **Output** (Success 200):

   ```
   [
     {
       "id": 1,
       "title": "The Great Gatsby",
       "author": "F. Scott Fitzgerald",
       "ISBN": "9780743273565",
       "available_quantity": 5,
       "shelf_location": "A1",
       "createdAt": "...",
       "updatedAt": "..."
     },
     ...
   ]
   ```

3. **Search for a Book by Title, Author, or ISBN**:
   - **Method**: `GET /books/search?title=The Great Gatsby&author=F. Scott Fitzgerald&isbn=9780743273565`
   - **Output** (Success 200):

   ```
   [
     {
       "id": 1,
       "title": "The Great Gatsby",
       "author": "F. Scott Fitzgerald",
       "ISBN": "9780743273565",
       "available_quantity": 5,
       "shelf_location": "A1"
   ```

```
  }
]
```

4. **Update Book Details**:
   - **Method**: `PUT /books/:id`
   - **Input** (JSON):

```
{
  "title": "The Great Gatsby (Updated)"
}
```

   - **Output** (Success 200):

```
{
  "id": 1,
  "title": "The Great Gatsby (Updated)",
  "author": "F. Scott Fitzgerald",
  "ISBN": "9780743273565",
  "available_quantity": 5,
  "shelf_location": "A1",
  "createdAt": "...",
  "updatedAt": "..."
}
```

5. **Delete a Book**:
   - **Method**: `DELETE /books/:id`
   - **Output** (Success 204): No content.

Borrowers API:

1. **Register a New Borrower**:
   - **Method**: `POST /borrowers`
   - **Input** (JSON):

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "registered_date": "2023-10-01"
}
```

- **Output** (Success 201):

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "registered_date": "2023-10-01",
  "createdAt": "...",
  "updatedAt": "..."
}
```

2. **List All Borrowers**:

- **Method**: `GET /borrowers`
- **Output** (Success 200):

```
[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john.doe@example.com",
    "registered_date": "2023-10-01"
  },
  ...
]
```

3. **Update Borrower Details**:

- **Method**: `PUT /borrowers/:id`
- **Input** (JSON):

```
{
  "email": "john.doe@newmail.com"
}
```

- **Output** (Success 200):

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@newmail.com",
  "registered_date": "2023-10-01",
  "createdAt": "...",
  "updatedAt": "..."
}
```

4. **Delete a Borrower**:
   - **Method**: `DELETE /borrowers/:id`
   - **Output** (Success 204): No content.

Borrowing Process API:

1. **Borrow a Book**:
   - **Method**: `POST /borrows`
   - **Input** (JSON):

```
{
  "borrower_id": 1,
  "book_id": 1,
  "due_date": "2023-11-01"
}
```

   - **Output** (Success 201):

```
{
  "id": 1,
  "borrower_id": 1,
  "book_id": 1,
  "borrow_date": "2023-10-01",
  "due_date": "2023-11-01",
  "return_date": null,
  "createdAt": "...",
  "updatedAt": "..."
}
```

2. **Return a Book**:
   - **Method**: `PUT /returns/:borrow_id`
   - **Output** (Success 204): No content.
3. **List Overdue Books**:
   - **Method**: `GET /overdue`

- **Output** (Success 200):

```
[
  {
    "id": 1,
    "borrower_id": 1,
    "book_id": 1,
    "borrow_date": "2023-10-01",
    "due_date": "2023-10-15",
    "return_date": null
  },
  ...
]
```

Models:

1. **Book Model** (`Book`):
   - **Attributes**:
     - `title`: Book title (String).
     - `author`: Author's name (String).
     - `ISBN`: ISBN code (String).
     - `available_quantity`: Number of available copies (Integer).
     - `shelf_location`: Location in the library (String).
2. **Borrower Model** (`Borrower`):
   - **Attributes**:
     - `name`: Borrower's full name (String).
     - `email`: Borrower's email (String).
     - `registered_date`: The date the borrower registered (Date).
3. **Borrow Model** (`Borrow`):
   - **Attributes**:
     - `borrower_id`: ID of the borrower (Foreign Key).
     - `book_id`: ID of the borrowed book (Foreign Key).
     - `borrow_date`: The date when the book was borrowed (Date).
     - `due_date`: The due date for returning the book (Date).

    - `return_date`: The date when the book was returned (Date, null if not returned).

Security & Authentication:

Basic authentication has been implemented on specific routes to secure sensitive operations like deleting books, updating records, and exporting data. The protected routes include:

- `GET /overdue`: To list overdue books.
- `PUT /books/:id`: To update book details.
- `DELETE /books/:id`: To delete a book.
- `GET /export-csv`: To export borrowing processes to CSV.
- `GET /export-overdue-csv`: To export overdue books of the last month to CSV.

To access these routes, users must authenticate with a username and password (`admin` / `let.me.in`).

Dockerization:

1. **Docker Setup**: The project is fully Dockerized for easier deployment. The Docker setup includes two containers:
    - **Node.js Application**: Handles API requests and runs the business logic.
    - **MySQL Database**: Stores data for books, borrowers, and borrowing processes.
2. **Docker Compose**: The `docker-compose.yml` file is used to define the services. To run the system with Docker:

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

Rate Limiting:

To prevent abuse of certain routes, rate limiting has been implemented on the following routes:

- `POST /books`: Add new books.
- `GET /books`: List all books.

The rate limit is set to allow **3 requests per 6 seconds**. This is configurable in the `rateLimit` middleware.

CSV Export:

Two endpoints are available to export borrowing process data to CSV:

1. **Export All Borrowing Processes**:
    - **Method**: `GET /export-csv`
    - **Description**: Export all borrowing processes (active and returned) within the past month to CSV.
2. **Export Overdue Books**:
    - **Method**: `GET /export-overdue-csv`
    - **Description**: Export overdue books from the past month to CSV.

Both exports will generate a CSV file that can be downloaded.

Final Note:

This Library Management System (LMS) is designed for flexibility and scalability. Additional features like book reservations, user roles, and reviews can be added in the future with minimal changes to the underlying architecture.