GUID TO RUN THE LIBRARY MANAGEMENT SYSTEM

abdullah

[COMPANY NAME] [Company address]

Step 1: Clone the Project from Git

1. Clone the Project from GitHub:

First, the user needs to clone the project from GitHub by running the following command:

```
git clone https://github.com/your-username/library-project.git
```

Step 2: Move to the Docker Folder

1. Navigate to the Docker Folder:

After cloning the project, the user needs to navigate to the `docker` directory:

```
cd docker
```

Step 3: Set Up and Run the System with Docker Compose

1. Ensure Docker is Running:

Make sure Docker and Docker Compose are installed and running:

```
docker --version

docker-compose –version
```

2. Start the System with Docker Compose:

Run the following command to build and start all the services (library system management, MySQL, Redis):

```
docker-compose up –build
```

This will start the system with:

- Spring Boot application on port 8000.

- MySQL database on port 3306.

- Redis cache on port 6379.

Step 4: Test Endpoints Using Postman

Once the system is running, the user can test the APIs using Postman, following the steps below.

1. **Sign Up**

- **HTTP Method**: POST
- **URL**: http://localhost:8000/api/auth/signup
- **Body**:
    - Choose **raw** and select **JSON** from the dropdown.
    - Provide the required data for sign-up (e.g., username, password, email):

```
{
  "username": "abdullah",
  "password": "aaa123",
  "email": "abd@example.com"
}
```

- **Response**:

```
{
  "status": true,
  "message": "User signed up successfully."
}
```

2. **Login** After signing up, you need to log in to obtain a JWT token.

- **HTTP Method**: POST
- **URL**: http://localhost:8000/api/auth/login
- **Body**:
    - Choose **raw** and select **JSON** from the dropdown.
    - Provide the login credentials (e.g., email, password):

```
{
  "email": "abd@example.com",
  "password": "aaa123"
}
```

- **Response**:
    - You should receive a response with the JWT token:

      ```
      {
        "token": "your-jwt-token-here",
        "expiresin": "3600000"  // in seconds
      }
      ```
- requests.

3. Test Book Endpoints

1. **Get All Books**
    - **HTTP Method**: GET
    - **URL**: http://localhost:8000/api/books
    - **Authorization**:
        - In the Authorization tab, select **Bearer Token** and paste your JWT token.
    - **Response**:
        - You should receive a response with a list of all books:

          ```
          [
            {
              "id": 1,
              "title": "Sample Book",
              "author": "John",
              "publicationYear": 2024,
              "isbn": "9781234567890",
              "genre": "Fiction"
            },
            {
              "id": 2,
              "title": "Another Book",
              "author": "Jane",
              "publicationYear": 2023,
              "isbn": "9789876543210",
          ```

```
  "genre": "Non-Fiction"
 }
]
```

2. **Get Book by ID**
   - **HTTP Method**: GET
   - **URL**: http://localhost:8000/api/books/{id} (Replace {id} with the book's ID)
   - **Authorization**:
     - In the Authorization tab, select **Bearer Token** and paste your JWT token.
   - **Response**:
     - You should receive a response with the details of the book with the given ID:

       ```
       {
        "id": 1,
        "title": "Sample Book",
        "author": "John",
        "publicationYear": 2024,
        "isbn": "9781234567890",
        "genre": "Fiction"
       }
       ```

3. **Create a Book**
   - **HTTP Method**: POST
   - **URL**: http://localhost:8000/api/books
   - **Authorization**:
     - In the Authorization tab, select **Bearer Token** and paste your JWT token.
   - **Body**:
     - Choose **raw** and select **JSON** from the dropdown.
     - Include the data for the new book (e.g., title, author, publicationYear, isbn, genre):

       ```
       {
        "title": "New Book",
        "author": "Alice",
        "publicationYear": 2025,
        "isbn": "9781122334455",
       ```

```
  "genre": "Thriller"
}
```

- **Response**:
  - You should receive a response confirming the book creation:

    ```
    {
      "status": true,
      "message": "Book created successfully."
    }
    ```

4. **Update Book**
   - **HTTP Method**: PUT
   - **URL**: http://localhost:8000/api/books/{id} (Replace {id} with the book's ID)
   - **Authorization**:
     - In the Authorization tab, select **Bearer Token** and paste your JWT token.
   - **Body**:
     - Choose **raw** and select **JSON** from the dropdown.
     - Include the data you want to update for the book (e.g., title, author, publicationYear, isbn, genre):

       ```
       {
         "title": "Updated Book Title",
         "author": "Updated Author",
         "publicationYear": 2026,
         "isbn": "9782233445566",
         "genre": "Science Fiction"
       }
       ```

   - **Response**:
     - You should receive a response confirming that the book's information was updated:

       ```
       {
         "status": true,
         "message": "Book updated successfully."
       }
       ```

5. **Delete Book**

- **HTTP Method**: DELETE
- **URL**: http://localhost:8000/api/books/{id} (Replace {id} with the book's ID)
- **Authorization**:
  - In the Authorization tab, select **Bearer Token** and paste your JWT token.
- **Response**:
  - You should receive a response confirming the deletion of the book:

    ```
    {
      "status": true,
      "message": "Book deleted successfully."
    }
    ```

---

1. **Return Book**
   - **HTTP Method**: POST
   - **URL**: http://localhost:8000/api/return/{bookId}/patron/{patronId} (Example: http://localhost:8000/api/return/1/patron/1)
   - **Authorization**:
     - In the Authorization tab, select **Bearer Token** and paste your JWT token.
   - **Response**:
     - You should receive a 200 Returned response confirming that the book has been returned:

       ```
       {
         "status": true,
         "message": "Book returned successfully."
       }
       ```

4. Test Patron Endpoints

1. Get All Patrons

- **HTTP Method:** GET
- **URL:** http://localhost:8000/api/patrons
- **Authorization:**

    - In the **Authorization** tab, select **Bearer Token** and paste your JWT token.
- **Response:**
    - You should receive a response with a list of all patrons

```
[
{
   "id": 1,
   "name": "abdullah",
   "email": "abd@example.com",
   "phone": "phonenumber"
 },
 {
   "id": 2,
   "name": "dd",
   "email": "e@example.com",
   "phone": " phonenumber "
 }
]
```

---

2. Get Patron by ID

- **HTTP Method:** GET
- **URL:** http://localhost:8000/api/patrons/{id} (Replace {id} with the patron's ID)
- **Authorization:**
    - In the **Authorization** tab, select **Bearer Token** and paste your JWT token.

- **Response:**
    - You should receive a response with the details of the patron with the given ID:

```
{
  "id": 1,
  "name": "abdullah",
  "email": "abd@example.com",
  "phone": " phonenumber "
}
```

---

3. Create a Patron

- **HTTP Method:** POST
- **URL:** http://localhost:8000/api/patrons
- **Authorization:**
  - In the **Authorization** tab, select **Bearer Token** and paste your JWT token.
- **Body:**
  - Choose **raw** and select **JSON** from the dropdown.
  - Include the data for the new patron (e.g., name, email, phone).

  ```
  {
        "name": "Alice Smith",
        "email": "alice.smith@example.com",
        "phone": "555-123-4567"
      }
  ```

- **Response:**
  - You should receive a response confirming the patron creation:

```
{
  "status": true,
  "message": "Patron created successfully."
}
```

---

4. Update Patron

- **HTTP Method:** PUT
- **URL:** http://localhost:8000/api/patrons/{id} (Replace {id} with the patron's ID)
- **Authorization:**
  - In the **Authorization** tab, select **Bearer Token** and paste your JWT token.
- **Body:**
  - Choose **raw** and select **JSON** from the dropdown.
  - Include the data you want to update for the patron (e.g., name, email, phone).

```
{
  "name": "abdo",
  "email": "abd0@example.com",
  "phone": "9876543210"
}
```

- **Response:**
  - You should receive a response confirming that the patron's information was updated:

```
{
  "status": true,
  "message": "Patron updated successfully."
}
```

---

5. Delete Patron

- **HTTP Method:** DELETE
- **URL:** http://localhost:8000/api/patrons/{id} (Replace {id} with the patron's ID)
- **Authorization:**
  - In the **Authorization** tab, select **Bearer Token** and paste your JWT token.
- **Response:**
  - You should receive a response confirming the deletion of the patron:

```
{
  "status": true,
  "message": "Patron deleted successfully."
}
```

Test Borrow Endpoints

1. **Borrow Book**
   - **HTTP Method**: POST
   - **URL**: http://localhost:8000/api/borrow/{bookId}/patron/{patronId} (Example: http://localhost:8000/api/borrow/1/patron/1)
   - **Authorization**:

        - In the Authorization tab, select **Bearer Token** and paste your JWT token.
    - **Response**:
        - You should receive a 200 Created response with the borrow record details:

```
{
  "status": true,
  "message": "Book borrowed successfully."
}
```

Step 6: Additional Notes

- Logs

If the user needs to view the logs, they can check the ./logs directory or view the logs of the Docker container by running:

```
docker logs library-service
```

- Stop the System

To stop the Docker containers, the user can run:

```
docker-compose down
```