**Documentation of APIs**

**1. Auth**

This controller is responsible for handling authentication-related actions, such as user registration and login.

* **POST /register**:
  + **Description**: Registers a new user in the system.
  + **Request**:
    - username (String): The username of the user (must be unique).
    - password (String): The password of the user.
  + **Response**:
    - **200 OK**: Returns the created user information in a UserDTO object.
    - **400 Bad Request**: If the username already exists, returns a validation error message.
* **POST /login**:
  + **Description**: Authenticates the user and generates a JWT token for the session.
  + **Request**:
    - username (String): The username of the user.
    - password (String): The password of the user.
  + **Response**:
    - **200 OK**: Returns a JWT token if authentication is successful.
    - **401 Unauthorized**: If authentication fails, returns an error message.

**2. Book**

This controller handles the operations related to managing books in the library system.

* **GET /books**:
  + **Description**: Retrieves a list of all available books in the library.
  + **Response**:
    - **200 OK**: Returns a list of books in JSON format.
* **POST /books**:
  + **Description**: Adds a new book to the library.
  + **Request**:
    - title (String): The title of the book.
    - author (String): The author of the book.
    - isbn (String): The ISBN number of the book.
  + **Response**:
    - **201 Created**: Returns the created book information.
    - **400 Bad Request**: If validation fails (e.g., missing fields).
* **GET /books/{id}**:
  + **Description**: Retrieves a specific book by its ID.
  + **Response**:
    - **200 OK**: Returns the book details.
    - **404 Not Found**: If the book is not found.
* **PUT /books/{id}**:
  + **Description**: Updates an existing book's details.
  + **Request**:
    - title (String): The title of the book.
    - author (String): The author of the book.
    - isbn (String): The ISBN number of the book.
  + **Response**:
    - **200 OK**: Returns the updated book information.
    - **404 Not Found**: If the book does not exist.

**3. Patron**

This controller manages patron-related actions, including viewing and updating patron details.

* **GET /patrons**:
  + **Description**: Retrieves a list of all patrons in the system.
  + **Response**:
    - **200 OK**: Returns a list of patrons in JSON format.
* **POST /patrons**:
  + **Description**: Adds a new patron to the system.
  + **Request**:
    - name (String): The name of the patron.
    - email (String): The email of the patron.
  + **Response**:
    - **201 Created**: Returns the created patron information.
* **GET /patrons/{id}**:
  + **Description**: Retrieves the details of a specific patron.
  + **Response**:
    - **200 OK**: Returns the patron's details.
    - **404 Not Found**: If the patron is not found.

**4. Borrowing**

This controller manages book borrowing actions, including borrowing and returning books.

* **POST /borrow**:
  + **Description**: Allows a patron to borrow a book.
  + **Request**:
    - patronId (Long): The ID of the patron borrowing the book.
    - bookId (Long): The ID of the book to be borrowed.
  + **Response**:
    - **200 OK**: Returns a success message and details of the borrowing.
    - **400 Bad Request**: If the book is already borrowed.
* **POST /return**:
  + **Description**: Allows a patron to return a borrowed book.
  + **Request**:
    - patronId (Long): The ID of the patron returning the book.
    - bookId (Long): The ID of the book being returned.
  + **Response**:
    - **200 OK**: Returns a success message indicating the book was returned.
    - **400 Bad Request**: If the book is not found or not borrowed by the patron.