Library Management System API

Capstone Project: Library Management System API

Project Overview:

As a backend developer, your goal is to design and implement a Library Management System API using Django and Django REST Framework. The API will serve as the backend for managing library resources, allowing users to interact with the system by borrowing, returning, and viewing books. You will be tasked with creating and deploying a fully functional API, simulating a real-world development environment, where backend logic, database management, and API design play crucial roles.

Functional Requirements:

Books Management (CRUD):

Implement the ability to Create, Read, Update, and Delete (CRUD) books.

Each book should have the following attributes: Title, Author, ISBN, Published Date, Number of Copies Available.

Ensure validations such as a unique ISBN number for each book.

Users Management (CRUD):

Implement CRUD operations for library users.

A user should have a unique Username, Email, Date of Membership, and Active Status.

Check-Out and Return Books:

Create an endpoint to allow users to check out available books.

Only one copy of a book can be checked out per user at a time.

When a book is checked out, reduce the number of available copies.

Ensure that users can only check out books if there are available copies.

Create an endpoint to allow users to return checked-out books.

Once a book is returned, increase the number of available copies.

Log the date when the user checked out and returned each book.

View Available Books:

Create an endpoint to view all books and filter by availability (i.e., only show books with available copies).

Implement optional query filters to search by Title, Author, or ISBN.

Technical Requirements:

Database:

Use Django ORM to interact with the database.

Define models for Books, Users, and Transactions (to track check-outs and returns).

Authentication:

Implement basic user authentication using Django’s built-in authentication system.

Users should be able to log in and access their own borrowing history.

Optionally, you can explore token-based authentication (JWT) for a more secure API experience.

API Design:

Use Django Rest Framework (DRF) to create and expose the necessary API endpoints.

Follow RESTful principles: endpoints should be well-named and make use of appropriate HTTP methods (GET, POST, PUT, DELETE).

Ensure proper error handling and return appropriate HTTP status codes.

Deployment:

Deploy your API on Heroku or PythonAnywhere.

Ensure that your deployed API is accessible, functional, and secure.

Stretch Goals (Optional):

User Roles: Add different user roles such as Admin (who can manage books and users) and Member (who can check out and return books).

Overdue Tracking: Add functionality to track overdue books and implement a penalty system for late returns.

Pagination and Filters: Add pagination to the book list and implement more advanced filtering options.

Notifications: Implement an email notification system that alerts users when a book is overdue or when a previously unavailable book becomes available.