Classwork: Building and Interacting with a Bookstore API

Theme: **"Empowering a Digital Library with Dynamic Frontend-Backend Integration"**

Overview

In this classwork, students will create and interact with a **Bookstore API** that serves as the foundation of a digital library. They will explore backend API development using Django REST Framework (DRF) and use JavaScript's fetch API to build a dynamic, interactive frontend. The goal is to simulate real-world development workflows where backend and frontend teams collaborate to deliver seamless user experiences.

The Digital Library Theme

Scenario:
Your team has been tasked with creating a **Digital Bookstore** for managing book collections and their authors. This system is intended to help readers browse available books, learn about their favorite authors, and interact with the system (e.g., adding books, deleting records).

Phases of the Classwork

Phase ١: Backend Development - Building the API

Objective:
Set up the backend infrastructure that will serve the book and author data for the application.

١. **Design the Models**
   - Create models for Book and Author.
   - A Book will have fields such as:
     - Title (string)
     - Description (text, optional)
     - Published Date (date)
     - Author (foreign key to the Author model)
   - An Author will have fields such as:
     - Name (string)
     - Biography (text, optional).

٢. **Implement Serializers**
   - Use DRF serializers to control how the models are serialized into JSON format.
   - Implement nested serializers so that books can display their associated author.

٣. **Create API Views**
   - Develop views to support the following operations:
     - Retrieve a list of all books.
     - Retrieve details of a single book.
     - Add a new book.

        - Update or delete a book.
    - Add similar views for authors.

٤. **Configure URLs**
    - Define API endpoints for books and authors in the urls.py file, e.g., /api/books/ and /api/authors/.

٥. **Optional Enhancements**
    - Add JWT authentication to secure certain endpoints.
    - Implement pagination if the book collection is large.

Example API Endpoints:

- /api/books/ (GET: list books, POST: add a book)
- /api/books/{id}/ (GET: book details, PUT: update book, DELETE: delete book)

Phase ٢: Frontend Development - Building the User Interface

Objective:
Create a simple web page to interact with the API and display book and author data dynamically.

١. **HTML Structure**
    - Design an HTML page with:
        - A heading for the page title, e.g., "Digital Library".
        - An empty section to display a list of books dynamically.
        - A form to add new books.

٢. **Fetch Data and Display Books**
    - Use JavaScript's fetch API to send a GET request to the /api/books/ endpoint.
    - Parse the JSON response and display the book titles in a list.

٣. **Interactivity**
    - Add a form to allow users to submit new books to the /api/books/ endpoint.
    - On form submission, send a POST request using fetch.
    - Update the displayed book list to include the newly added book without refreshing the page.

٤. **Error Handling**
    - Display an error message if the API call fails, e.g., "Unable to fetch data. Please try again."

٥. **Refresh Button**
    - Add a button that allows users to reload the book list by fetching the data again.

Phase ٣: Enhancements (Optional for Advanced Students)

١. **Search Functionality**
    - Add a search bar to filter books by title or author name.

- Use query parameters with fetch to filter results from the /api/books/ endpoint.

٢. **Book Details View**
   - Allow users to click on a book title to view more details, such as the description and published date.

٣. **Pagination**
   - If the API supports pagination, add buttons or links to navigate through pages of books.

٤. **Edit and Delete Options**
   - Add buttons for each book to allow editing its details or deleting it.
   - Use PUT and DELETE requests to update or remove books via the API.

٥. **Authentication**
   - Use JWT authentication for secure interactions. Add a login form to obtain a token, and include the token in subsequent API requests.

Expected Deliverables

١. **Backend API**:
   - Fully functional API with endpoints for managing books and authors.
   - API documentation (if possible) for the frontend team to use.

٢. **Frontend Web Page**:
   - A dynamic and interactive web page where users can view, add, and refresh books.
   - Error messages for failed API requests.

٣. **Optional Features** (if implemented):
   - Search bar, book details view, pagination, and authentication.

Learning Outcomes

- Understand the fundamentals of building a REST API using Django REST Framework.
- Learn how to use JavaScript's fetch API to interact with a backend.
- Gain hands-on experience with dynamic DOM manipulation and error handling.
- Build an end-to-end application combining backend and frontend skills.