Step-by-Step Guide for Building a Library Management System

1. Define Your Goal

Understand the core functionalities of the application:

- Add new books.
- Borrow books and mark them as unavailable.
- Return borrowed books and mark them as available.
- Filter available books.
- Sort books alphabetically by title.
- Dynamically render the book list.

---

2. Plan the Structure

Break the application into smaller components:

1. **Data Storage**: Use an array to store book details.
2. **Methods**: Define functions for adding, borrowing, returning, filtering, sorting, and rendering books.
3. **Event Handling**: Handle interactions like form submissions and button clicks.

---

3. Start Writing Code

Begin with the basics and build incrementally:

A. Set Up a Library Management Object

Create a JavaScript object (`libraryManagement`) to manage books and their associated actions:

- Properties:
  - `books`: Array to store book objects.
- Methods:
  - `addBook()`
  - `borrowBook()`
  - `returnBook()`
  - `filterAvailableBooks()`
  - `sortBooks()`
  - `renderBooks()`

    - `resetForm()`

B. Implement Core Methods

Write the methods in a modular way:

- `addBook`: Add a new book with details like title, author, and availability status.
- `borrowBook`: Mark a book as unavailable if it is currently available.
- `returnBook`: Mark a book as available if it is currently borrowed.
- `filterAvailableBooks`: Return a list of books that are available.
- `sortBooks`: Sort the book list alphabetically by title.
- `renderBooks`: Dynamically update the DOM to display the book list.
- `resetForm`: Clear the input fields in the form.

C. Attach Event Listeners

Handle the interactions:

- **Form Submission**: Use `addEventListener` on the form to trigger `addBook` for adding new books.
- **Borrow and Return Buttons**: Dynamically generate buttons and attach handlers for borrowing and returning books.

D. Test and Debug

- Test each method individually in the browser console.
- Validate inputs for edge cases (e.g., empty fields, borrowing an unavailable book).

---

4. Order of Implementation

Follow this sequence:

Initialize the Library Management Object:

const libraryManagement = { books: [] };

1.
2. **Add Core Methods to the Object**:
    - `addBook()`
    - `borrowBook()`
    - `returnBook()`
    - `filterAvailableBooks()`

    - `sortBooks()`
    - `renderBooks()`
    - `resetForm()`
3. **Create Event Handlers**:
    - Write the logic for form submission and attach it to the "Submit" button.
    - Write the logic for borrow and return actions and attach them to dynamically generated buttons.
4. **DOM Manipulation**:
    - Use `document.createElement` and `appendChild` to render books.
    - Dynamically update the DOM for book list and actions.
5. **Test the Flow**:
    - Add books.
    - Borrow and return books.
    - Sort books alphabetically.
    - Filter available books.
    - Ensure the DOM updates correctly.

---

5. Add Features Incrementally

Once the basics work, enhance the application:

- **Advanced Filters**: Allow filtering by author or specific genres.
- **Borrowing Limits**: Restrict borrowing to a certain number of books per user.
- **Styling**: Apply CSS classes for better UI and user experience.
- **Validation**: Ensure fields are filled correctly before submission.

---

6. Checklist for Completion

- Books are added correctly with unique IDs.
- Books can be borrowed and marked as unavailable.
- Books can be returned and marked as available.
- The book list can be filtered to show only available books.
- The book list can be sorted alphabetically by title.
- The book list renders dynamically and updates after actions.

By following this roadmap, you will systematically build the Library Management System with clarity and focus.