1. Work on the Angular Service that interacts with json-server

Update the service that uses HttpClient to perform CRUD operations against our json-server API.

a. Update the Service File

Create a file called `book-inventory.service.ts` (if you haven't already):

```
// src/app/book-inventory/book-inventory.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Book } from '../book';

@Injectable({
  providedIn: 'root'
})
export class BookInventoryService {
  // The json-server endpoint for books
  private apiUrl = 'http://localhost:3000/books';

  constructor(private http: HttpClient) { }

  // GET: Retrieve all books
  getInventory(): Observable<Book[]> {
    return this.http.get<Book[]>(this.apiUrl);
  }

  // GET: Retrieve a specific book (optional for editing)
  getBook(isbn: string): Observable<Book> {
    return this.http.get<Book>(`${this.apiUrl}/${isbn}`);
  }

  // POST: Create a new book
  createBook(book: Book): Observable<Book> {
    return this.http.post<Book>(this.apiUrl, book);
  }
```

```
  // PUT: Update an existing book
  updateBook(book: Book): Observable<Book> {
    return this.http.put<Book>(`${this.apiUrl}/${book.ISBN}
`, book);
  }

  // DELETE: Delete a book by ISBN
  deleteBook(book: Book): Observable<any> {
    return this.http.delete(`${this.apiUrl}/${book.ISBN}`);
  }
}
```

3. Build an Admin Component for CRUD Operations

We'll create an admin interface that lets you view, add, update, and delete books.

a. Create the Admin Component

Create a new file called `admin.component.ts` (or generate one using Angular CLI):

```
// src/app/admin/admin.component.ts
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { Book } from '../book';
import { BookInventoryService } from '../book-inventory/
book-inventory.service';
import { BookFilterPipe } from '../book-filter.pipe';
import { HoverHighlightDirective } from '../book-inventory/
hover-highlight.directive';

@Component({
  selector: 'app-admin',
  standalone: true,
  imports: [CommonModule, FormsModule, HttpClientModule,
BookFilterPipe, HoverHighlightDirective],
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.css']
})
export class AdminComponent implements OnInit {
```

```
  inventory: Book[] = [];
  searchTerm: string = '';
  // For form editing/adding
  currentBook: Book = this.initializeBook();
  isEditMode: boolean = false;

  constructor(private bookService: BookInventoryService) {}

  ngOnInit(): void {
    this.getBooks();
  }

  // Initialize an empty Book object for the form
  initializeBook(): Book {
    return {
      ISBN: '',
      title: '',
      author: '',
      year: new Date().getFullYear(),
      price: 0,
      featured: false,
      coverImages: [],
      description: ''
    } as Book;
  }

  // Retrieve books from the API
  getBooks(): void {
    this.bookService.getInventory().subscribe({
      next: (books: Book[]) => this.inventory = books,
      error: (err) => console.error('Error fetching
inventory', err)
    });
  }

  // Save a book (create new or update existing)
  saveBook(): void {
    if (this.isEditMode) {

this.bookService.updateBook(this.currentBook).subscribe({
```

```
        next: (updatedBook: Book) => {
          this.getBooks();
          this.resetForm();
        },
        error: (err) => console.error('Error updating
book', err)
      });
    } else {

this.bookService.createBook(this.currentBook).subscribe({
        next: (newBook: Book) => {
          this.getBooks();
          this.resetForm();
        },
        error: (err) => console.error('Error creating
book', err)
      });
    }
  }

  // Load a book into the form for editing
  editBook(book: Book): void {
    this.currentBook = { ...book }; // Clone the book
    this.isEditMode = true;
  }

  // Delete a book
  deleteBook(book: Book): void {
    this.bookService.deleteBook(book).subscribe({
      next: () => this.getBooks(),
      error: (err) => console.error('Error deleting book',
err)
    });
  }

  // Reset the form fields
  resetForm(): void {
    this.currentBook = this.initializeBook();
    this.isEditMode = false;
  }
```

```
  // TrackBy function for ngFor
  trackByISBN(index: number, book: Book): string {
    return book.ISBN;
  }
}
```

b. Create the Admin Component Template

File: src/app/admin/admin.component.html

```
<div class="container my-4">
  <h1 class="mb-4 text-center">Admin Panel: Manage Book
Inventory</h1>

  <!-- Search Input -->
  <div class="mb-3 text-center">
    <input type="text" [(ngModel)]="searchTerm"
placeholder="Search books" class="form-control w-50 mx-
auto" />
  </div>

  <!-- Book Form for Add/Edit -->
  <div class="card mb-4">
    <div class="card-body">
      <h3>{{ isEditMode ? 'Edit Book' : 'Add New Book' }}</
h3>
      <form (ngSubmit)="saveBook()">
        <div class="form-group mb-2">
          <label for="isbn">ISBN:</label>
          <input id="isbn"
type="text" [(ngModel)]="currentBook.ISBN" name="isbn"
class="form-control" [readonly]="isEditMode" required />
        </div>
        <div class="form-group mb-2">
          <label for="title">Title:</label>
          <input id="title"
type="text" [(ngModel)]="currentBook.title" name="title"
class="form-control" required />
        </div>
```

```
        <div class="form-group mb-2">
          <label for="author">Author:</label>
          <input id="author"
type="text" [(ngModel)]="currentBook.author" name="author"
class="form-control" required />
        </div>
        <div class="form-group mb-2">
          <label for="year">Year:</label>
          <input id="year"
type="number" [(ngModel)]="currentBook.year" name="year"
class="form-control" required />
        </div>
        <div class="form-group mb-2">
          <label for="price">Price:</label>
          <input id="price"
type="number" [(ngModel)]="currentBook.price" name="price"
class="form-control" required />
        </div>
        <div class="form-group mb-2">
          <label for="description">Description:</label>
          <textarea
id="description" [(ngModel)]="currentBook.description"
name="description" class="form-control"></textarea>
        </div>
        <div class="form-group mb-2">
          <label>
            <input
type="checkbox" [(ngModel)]="currentBook.featured"
name="featured" /> Featured
          </label>
        </div>
        <div class="form-group mb-2">
          <label for="cover">Cover Image URL:</label>
          <input id="cover"
type="text" [(ngModel)]="currentBook.coverImages[0]"
name="cover" class="form-control" />
        </div>
        <button type="submit" class="btn btn-
success">{{ isEditMode ? 'Update Book' : 'Add Book' }}</
button>
```

```
        <button type="button" class="btn btn-secondary
ms-2" (click)="resetForm()">Cancel</button>
      </form>
    </div>
  </div>

  <!-- Book List -->
  <div class="row">
    <div class="col-12 col-sm-6 col-md-4 col-lg-3 mb-4"
         *ngFor="let book of (inventory |
bookFilter:searchTerm); trackBy: trackByISBN">
      <div class="card h-100 shadow-sm"
appHoverHighlight="#e1e1e1">
        <div class="ratio ratio-4x3">
          <img *ngIf="book.coverImages &&
book.coverImages.length > 0"
               [src]="book.coverImages[0]"
               class="card-img-top"
               alt="{{ book.title }} cover"
               style="object-fit: cover;">
        </div>
        <div class="card-body d-flex flex-column">
          <h5 class="card-title">{{ book.title }}</h5>
          <p class="card-text flex-grow-1">
            <strong>Author:</strong> {{ book.author }}<br>
            <strong>Year:</strong> {{ book.year }}<br>
            <strong>ISBN:</strong> {{ book.ISBN }}<br>
            <strong>Price:</strong> {{ book.price |
currency:'USD':'symbol' }}<br>
            <strong>Description:</strong>
{{ book.description }}
          </p>
          <p *ngIf="book.featured" class="badge bg-primary
mb-2">Featured Book</p>
          <div [ngSwitch]="book.price > 30 ? 'expensive' :
'affordable'" class="mb-2">
            <p *ngSwitchCase="'expensive'" class="text-
danger mb-0">Premium selection!</p>
            <p *ngSwitchCase="'affordable'" class="text-
success mb-0">Budget-friendly!</p>
```

```
            <p *ngSwitchDefault class="text-muted
mb-0">Great value!</p>
          </div>
          <div class="mt-auto">
            <button (click)="editBook(book)" class="btn
btn-warning me-2">Edit</button>
            <button (click)="deleteBook(book)" class="btn
btn-danger">Delete</button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

4. Testing Your Admin Page

1. **Start json-server:**
   Make sure your json-server is running:

   ```
   json-server --watch db.json
   ```

   Your API is available at http://localhost:3000/books.

2. **Run Your Angular App:**
   Your admin page should now fetch book data via HttpClient and display it. You can use the form to add or update a book, and the Edit/Delete buttons should work with the API.