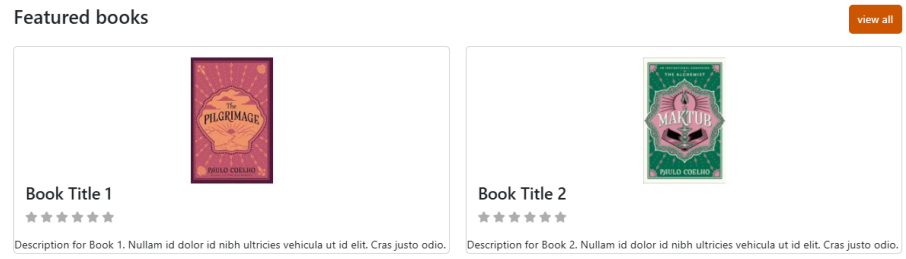
IT Elec 1: WEB DEVELOPMENT

Populating Pages

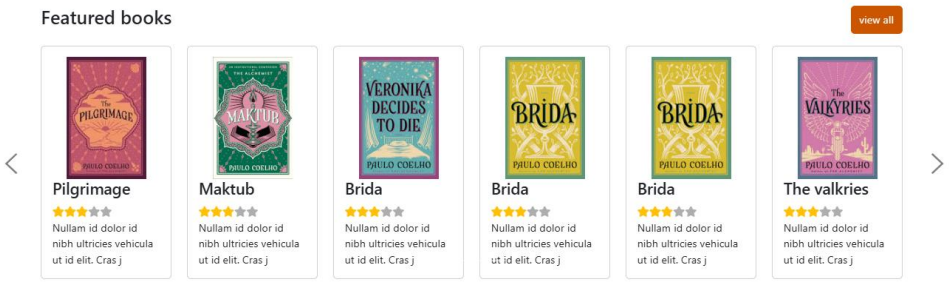
Manuel Andrei L. Lleva
BSIT-3C

WEB PAGES

Displaying 2 Featured Books:



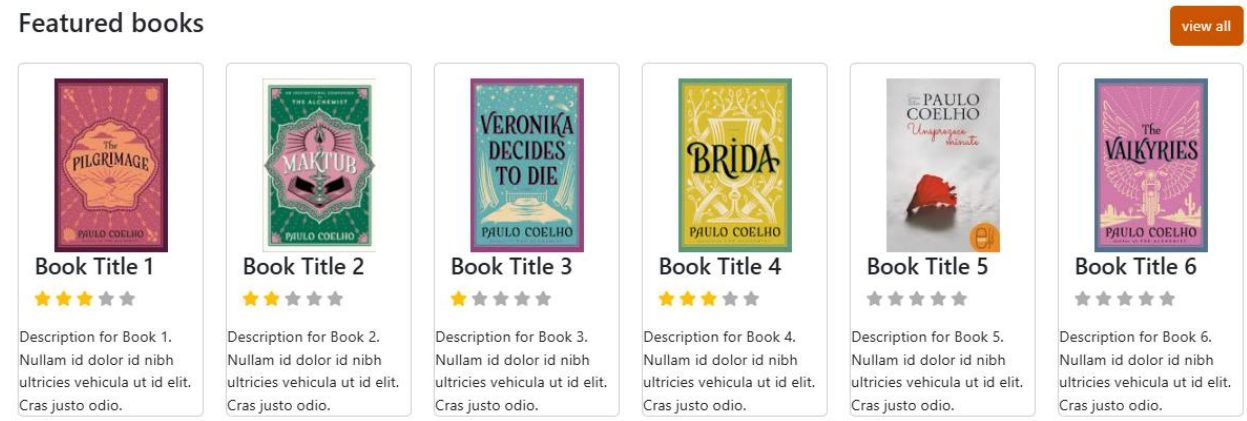
Displaying 6 Featured Books:

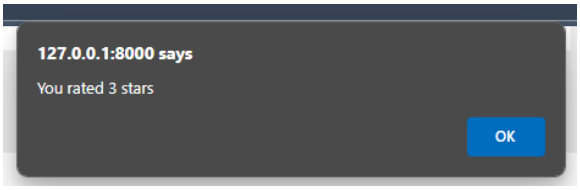


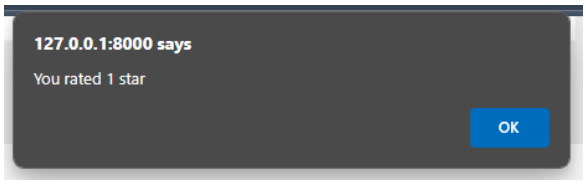

```php
// Generate 12 books dynamically
for ($i = 1; $i <= 6; $i++) {
    $featuredBooks[] = [
        'id' => $i, // Unique ID for the book
        'title' => "Book Title $i",
        'image' => "assets/img/cover$i.jpg", // Assuming different i
        'rating' => 0, // Default to 0 for unrated books
        'description' => "Description for Book $i. Nullam id dolor i
    ];
}
```

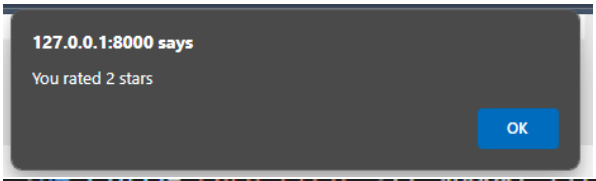
This PHP code snippet dynamically generates 12 books. This code snippet creates arrays of books with unique IDs, titles, images, default ratings, and descriptions, simulating a larger set of book entries. The first section showcases 2 featured books, while the second section displays 6 featured books.

Displaying Star ratings:










```
<div class="rating" data-book-id="{{ $book['id'] }}">
    @for ($i = 1; $i <= 5; $i++)
    <i class="fas fa-star" style="color: rgb(174, 174, 174)"></i>
    @endfor
</div>
```

Users can rate these books, and their ratings are saved in localStorage, ensuring persistence even after page refreshes. Pop-up messages provide immediate feedback on the ratings, enhancing interactivity. Additionally, a snippet of HTML code is shown, responsible for rendering the star rating interface using a loop to generate star icons. Overall, the page allows dynamic user interaction and efficient data display.

Controller

app > Http > Controllers > LandingPageController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class LandingPageController extends Controller
{
    public function index()
    {
        // Simulated featured books data
        $featuredBooks = [];

        // Generate 12 books dynamically
        for ($i = 1; $i <= 6; $i++) {
            $featuredBooks[] = [
                'id' => $i, // Unique ID for the book
                'title' => "Book Title $i",
                'image' => "assets/img/cover$i.jpg", // Assuming different images for each book
                'rating' => 0, // Default to 0 for unrated books
                'description' => "Description for Book $i. Nullam id dolor id nibh ultricies vehicula ut id elit.
            ];
        }

        return view('LandingPage', compact('featuredBooks'));
    }
}
```

This code defines a LandingPageController class that extends the base Controller class. Within this class, there's a method named index, which simulates the creation of an array called $featuredBooks containing 12 dynamically generated book entries. Each entry in the array includes a unique ID, a title, an image path, a default rating of 0, and a brief description. Once the array is populated, the method returns a view called LandingPage, passing the $featuredBooks array to it for display. This approach is often used to dynamically generate content for a web page, such as a landing page showcasing featured books.

Route

```php
Route::get('/', [LandingPageController::class, 'index'])->name('landing');
```

This defines a route that responds to a GET request made to the root URL (/). When a user navigates to this URL, the application will call the index method of the LandingPageController class. This setup is beneficial for organizing the application's entry points, as it allows you to direct specific URLs to particular controller methods efficiently. Additionally, by naming the route landing with ->name('landing'), it can easily refer to this route in other parts of the application, such as views or redirects, making the code more readable and maintainable.

What I learned :

This Project helped me understand how to display information clearly and concisely without cluttering the section. The project improved both my technical skills and my ability to design user-friendly interfaces. I learned how to create a dynamic and responsive landing page using Laravel's Blade templates and controllers. I also used a carousel to display books and employed loops to manage the number of cards displayed. My Group mate implemented features such as user ratings for books, with ratings saved in localStorage for persistence. Additionally created controllers to handle content and registered them in routes, enhancing data management and navigation.