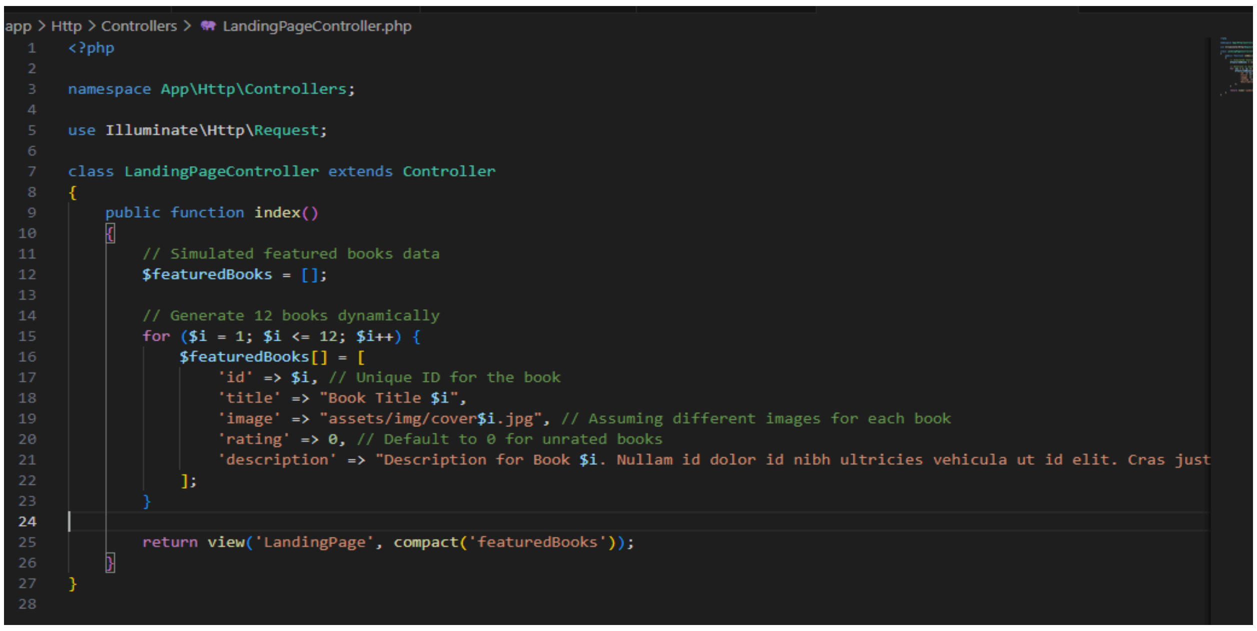
IT Elec 1: WEB DEVELOPMENT
POPULATING PAGES

WENDEE DIANE FLORES LLONA
BSIT 3C

CONTROLLER: app/Http/Controllers/LandingPageController.php

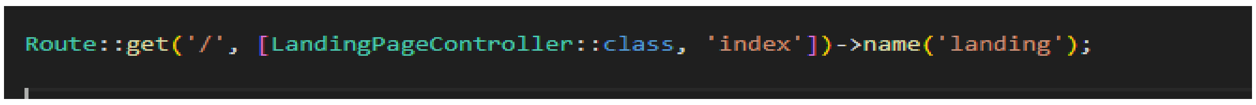


app > Http > Controllers > LandingPageController.php

```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 <= 12; $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. Cras just
            ];
        }

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

This code defines the LandingPageController, which handles requests to the landing page. The index method generates an array of 12 featured books with unique IDs, titles, images, default ratings, and descriptions. The featuredBooks array can be easily updated, and the data is passed to the view using the compact function for rendering.

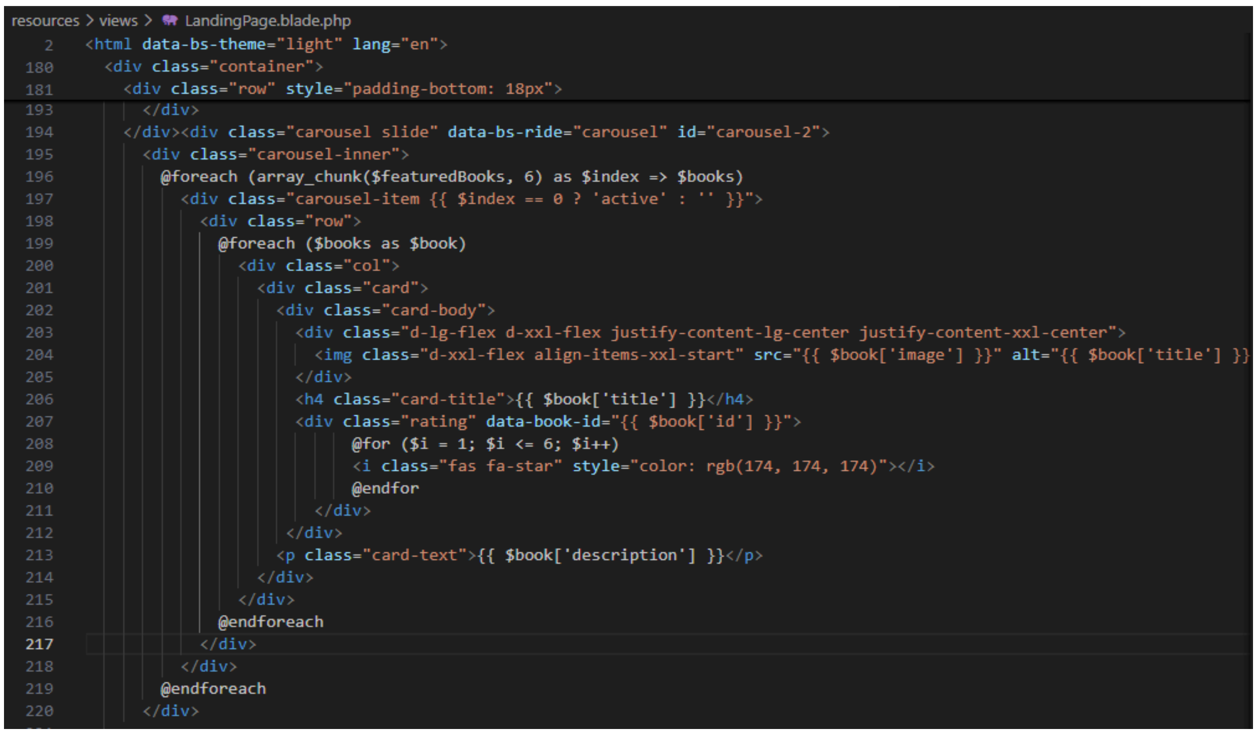
ROUTES: routes/web.php



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

This route maps the root URL (/) to the index method of LandingPageController, which generates an array of featured books and passes them to the LandingPage view. The route is named landing, making it easy to reference elsewhere in the application.

VIEWS: resources/views/LandingPage.blade.php

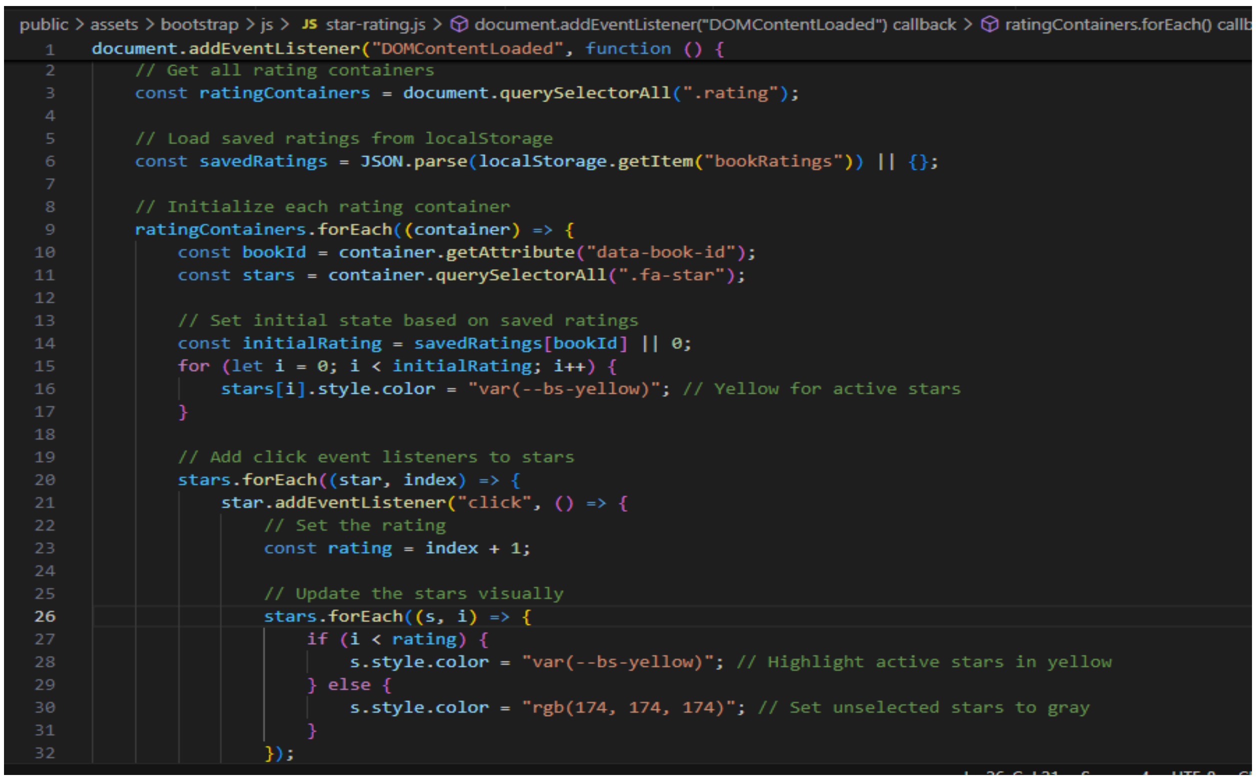


resources > views > LandingPage.blade.php

```html
<html data-bs-theme="light" lang="en">
  <div class="container">
    <div class="row" style="padding-bottom: 18px">
      </div>
    </div><div class="carousel slide" data-bs-ride="carousel" id="carousel-2">
      <div class="carousel-inner">
        @foreach (array_chunk($featuredBooks, 6) as $index => $books)
          <div class="carousel-item {{ $index == 0 ? 'active' : '' }}">
            <div class="row">
              @foreach ($books as $book)
                <div class="col">
                  <div class="card">
                    <div class="card-body">
                      <div class="d-lg-flex d-xxl-flex justify-content-lg-center justify-content-xxl-center">
                        <img class="d-xxl-flex align-items-xxl-start" src="{{ $book['image'] }}" alt="{{ $book['title'] }}
                      </div>
                      <h4 class="card-title">{{ $book['title'] }}</h4>
                      <div class="rating" data-book-id="{{ $book['id'] }}">
                          @for ($i = 1; $i <= 6; $i++)
                          <i class="fas fa-star" style="color: rgb(174, 174, 174)"></i>
                          @endfor
                        </div>
                      </div>
                    <p class="card-text">{{ $book['description'] }}</p>
                  </div>
                </div>
              @endforeach
            </div>
          </div>
        @endforeach
      </div>
```

This view displays the featured books passed from the controller by iterating through the $featuredBooks array. It uses a carousel to show the books in chunks of 6 per slide. Each book displays the image, title, placeholder rating with stars, and description. The data from the controller populates the carousel, creating an interactive display of the featured books.

STAR RATINGS: public/assets/bootstrap/js/star-rating.js



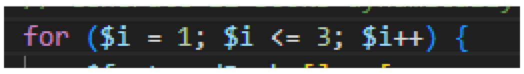
public > assets > bootstrap > js > JS star-rating.js > document.addEventListener("DOMContentLoaded") callback > ratingContainers.forEach() callb

```javascript
document.addEventListener("DOMContentLoaded", function () {
    // Get all rating containers
    const ratingContainers = document.querySelectorAll(".rating");

    // Load saved ratings from localStorage
    const savedRatings = JSON.parse(localStorage.getItem("bookRatings")) || {};

    // Initialize each rating container
    ratingContainers.forEach((container) => {
        const bookId = container.getAttribute("data-book-id");
        const stars = container.querySelectorAll(".fa-star");

        // Set initial state based on saved ratings
        const initialRating = savedRatings[bookId] || 0;
        for (let i = 0; i < initialRating; i++) {
            stars[i].style.color = "var(--bs-yellow)"; // Yellow for active stars
        }

        // Add click event listeners to stars
        stars.forEach((star, index) => {
            star.addEventListener("click", () => {
                // Set the rating
                const rating = index + 1;

                // Update the stars visually
                stars.forEach((s, i) => {
                    if (i < rating) {
                        s.style.color = "var(--bs-yellow)"; // Highlight active stars in yellow
                    } else {
                        s.style.color = "rgb(174, 174, 174)"; // Set unselected stars to gray
                    }
                });
```

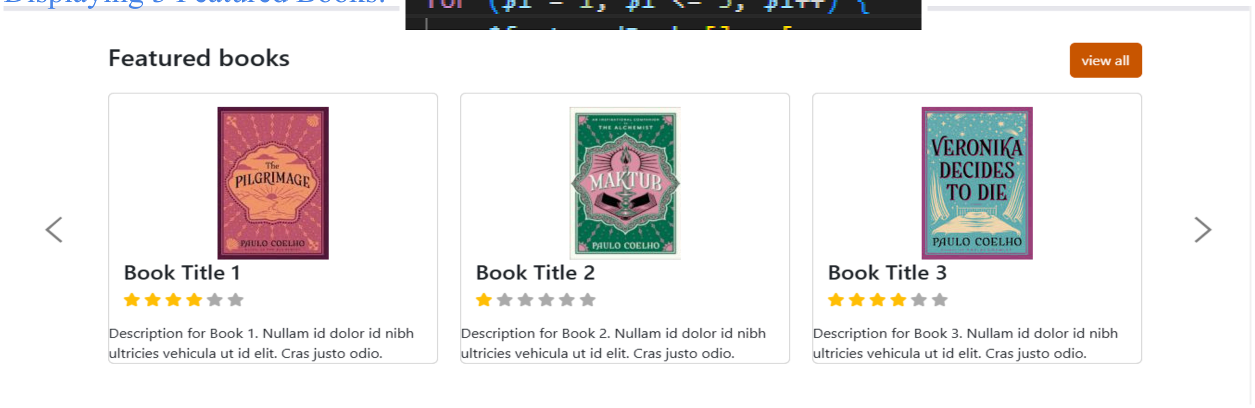
This script enables users to rate the featured books displayed in the view. Each book's rating is stored in localStorage, allowing the ratings to persist across page reloads. When a user clicks on the stars in the rating section, the rating for the respective book is visually updated and saved, ensuring the ratings reflect on the landing page every time the user visits.

RENDERED PAGES: Featured Books

Displaying 3 Featured Books:



```php
for ($i = 1; $i <= 3; $i++) {
```



Featured books

view all

Book Title 1

★★★★★★

Description for Book 1. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

Book Title 2

★★★★★★

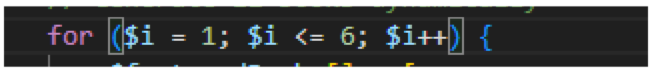
Description for Book 2. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

Book Title 3

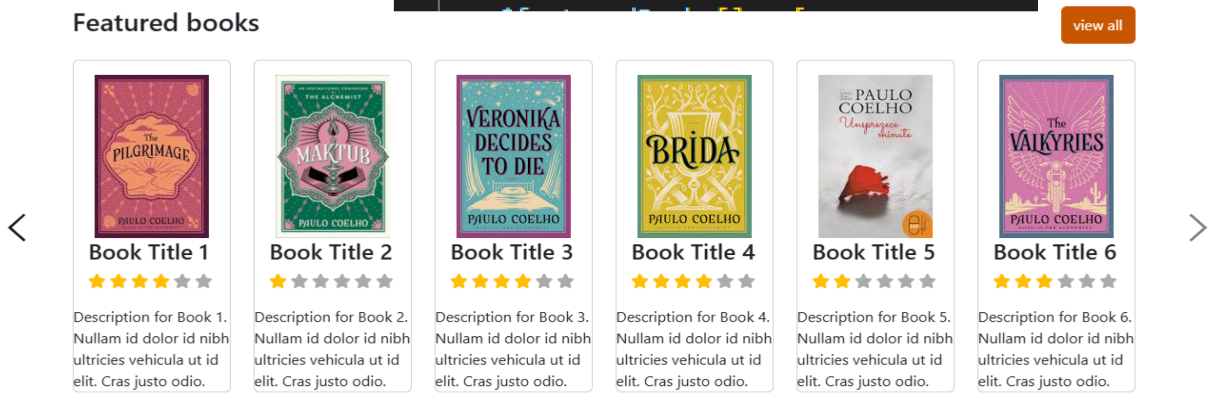
★★★★★★

Description for Book 3. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

Displaying 6 Featured Books:



```php
for ($i = 1; $i <= 6; $i++) {
```



Featured books

view all

Book Title 1

★★★★★★

Description for Book 1. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

Book Title 2

★★★★★★

Description for Book 2. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

Book Title 3

★★★★★★

Description for Book 3. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

Book Title 4

★★★★★★

Description for Book 4. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

Book Title 5

★★★★★★

Description for Book 5. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

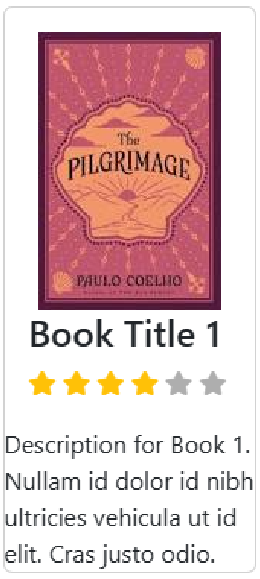
Book Title 6

★★★★★★

Description for Book 6. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.

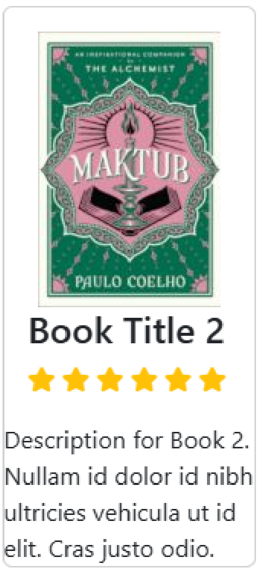
Note: We can display as many featured books as we want.

RENDERED PAGES: Star Rating



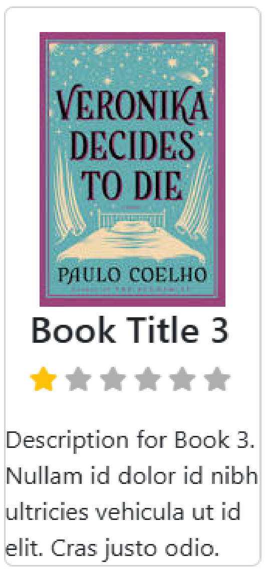
Book Title 1

Description for Book 1. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.



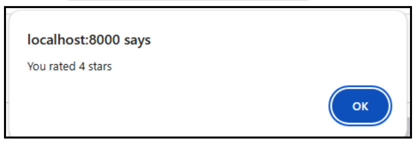
Book Title 2

Description for Book 2. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.



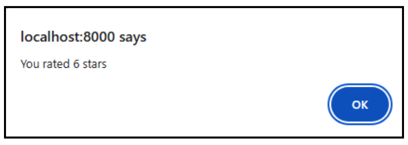
Book Title 3

Description for Book 3. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio.



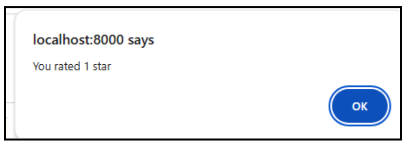
localhost:8000 says

You rated 4 stars



localhost:8000 says

You rated 6 stars



localhost:8000 says

You rated 1 star

SUMMARY OF WHAT I LEARNED:

In this project, I learned how to create a dynamic landing page by using Blade templates for the views and controllers to manage the data. We passed featured book details (title, image, rating, description) from the controller to the view. Users can rate books, with ratings saved in localStorage for persistence. A carousel displays the books in chunks, and we can easily add more books. To accomplish this, we created controllers to load content, registered them in routes, and used arrays to simulate database access for displaying the data.