

## LABORATORY 6

### POPULATING PAGES

Bongat, Mae Anne B.

BSIT-3C

Web Development

1. In our controller, we've designed a method called `loadBooks` that acts as a central hub for our book collection. It's where we gather and organize all the key details about various books to make them accessible for our application. This is like creating a virtual library where every book is stored neatly with its information ready to be shared. Each book in our collection has a rich set of details: its title, author, a brief description, an image of the cover, genres, average ratings, the number of reviews, and even some reader comments. By putting all of this together, we ensure that our users or anyone accessing this data gets a comprehensive view of what each book has to offer.

We believe this setup not only keeps things well-organized but also sets a solid foundation for presenting information effectively. It feels like we're building something meaningful—a system that could grow and evolve as we add more features or integrate real-world data.

```
app > Http > Controllers > BooksController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class BooksController extends Controller
8  {
9      public function loadBooks()
10     {
11         $posts = [
12             [
13                 'book_id' => 1,
14                 'title' => "Omniscient Reader's Viewpoint",
15                 'author' => 'Sing Shong',
16                 'rating' => 4.5,
17                 'cover' => 'asset/images/orv.jpg',
18                 'description' => "Only I know the end of this world. One day our MC finds himself stuck in the world of his favorite webnovel. Wh",
19                 'num_ratings' => 121212,
20                 'num_reviews' => 20212,
21                 'genres' => ['Fantasy', 'Manga', 'Light Novel', 'Science Fiction', 'Manhwa', 'Fiction'],
22                 'comments' => [
23                     ['reader' => 'Renato Jr.',
24                      'text' => 'An incredible read! So many twists and unforgettable moments!',
25                      'time' => '1 day ago']
26                 ],
27             ],
28         ];
```

2. This method demonstrates parameter handling in our application. When the show(\$id) function is called, it takes a dynamic parameter, \$id, typically passed from a route like /books/{id}. Using this \$id, we search for a specific book in our collection (\$this->loadBooks()) by matching the book\_id. If no book is found, we handle it gracefully by showing a "404 Not Found" error. Otherwise, we pass the found book to the book-details view so that it can be displayed to users. This ensures that our application handles dynamic inputs efficiently while providing meaningful responses.

```
210     public function show($id)
211     {
212
213         $posts = $this->loadBooks();
214
215         $book = collect($posts)->firstWhere('book_id', $id);
216
217         if (!$book) {
218             abort(404);
219         }
220
221         return view('book-details', compact('book'));
222     }
223
224
225 }
```

## LABORATORY 6 POPULATING PAGES

Bongat, Mae Anne B.

BSIT-3C

Web Development

```
web.php x
routes > web.php
1 use Illuminate\Support\Facades\Route;
2 use App\Http\Controllers\HomeController;
3 use App\Http\Controllers\DashboardController;
4 use App\Http\Controllers\AuthController;
5 use App\Http\Controllers\LandingPageController;
6
7
8 Route::get('/', [LandingPageController::class, 'index'])->name('landing-page');
9
10
11 // Authentication Routes (only for guests)
12 Route::middleware('guest')->group(function () {
13     Route::get('/login', [AuthController::class, 'login'])->name('login');
14     Route::post('/login', [AuthController::class, 'loginPost'])->name('login.post');
15     Route::get('/register', [AuthController::class, 'register'])->name('register');
16     Route::post('/register', [AuthController::class, 'registerPost'])->name('register.post');
17 });
18
19 // Authenticated Routes (for logged-in users)
20 Route::middleware('auth')->group(function () {
21     Route::get('/contact', function () {
22         return view('contact');
23     });
24
25     Route::get('/home', [HomeController::class, 'index'])->name('home.index');
26     Route::get('/dashboard', [DashboardController::class, 'index'])->name('dashboard.show');
27
28     Route::post('/logout', function () {
29         auth()->logout(); // Logs out the user
30         return redirect(route('landing-page'));
31     }->name('logout');
32
33 // Content loading routes
34 Route::get('/home/content', [BooksController::class, 'loadBooks'])->name('home.books');
35 Route::get('/dashboard/content', [BooksController::class, 'loadBooks'])->name('dashboard.books');
36
37 Route::get('/books/{id}', [BooksController::class, 'show'])->name('books.show');
```

3. In this section of the project, the route assignments are defined to manage navigation across different pages based on the user's authentication status. The **public routes** allow access to the landing page, which is the first page that users see when they visit the site. The **authentication routes** are reserved for guests, providing them with the ability to log in or register for an account. Once a user is logged in, they are directed to the **authenticated routes**, which include pages such as the home page, dashboard, and a contact page. Additionally, the **content routes** handle the loading of dynamic content, such as books, and allow users to view detailed information about individual books. These routes are managed by controllers like `AuthController`, `HomeController`, and `BooksController`, which contain the logic for returning the appropriate views. This structure helps ensure that users are directed to the correct pages depending on their authentication status, maintaining a smooth and organized flow throughout the application.

The screenshot shows a VS Code editor with a PHP Blade template for a book review dashboard. The file explorer on the left shows a project structure with files like login.blade.php, register.blade.php, and dashboard.blade.php. The main editor displays the dashboard.blade.php code, which includes a navigation bar, a 'Recent Reviews' section, and a 'Review List' section. The code uses Laravel Blade syntax for loops and conditionals. The status bar at the bottom shows the file is at line 123, column 1, and the editor is using UTF-8 encoding and LF line endings.

```

11 <main>
12 <div class="container">
13     <div class="row">
14         <div class="col-md-3">
15             <div class="card">
16                 <div class="card-body">
17                     <div class="text-center">
18                         <h3>Recent Reviews</h3>
19                         <button id="view-all-reviews" class="view-all-link">View All</button>
20                     </div>
21                 </div>
22             </div>
23         </div>
24         <div class="col-md-9">
25             <div class="card">
26                 <div class="card-body">
27                     <div class="text-center">
28                         <h3>Review List</h3>
29                         <button id="view-all-reviews" class="view-all-link">View All</button>
30                     </div>
31                     <div class="row" id="review-list">
32                         @foreach (array_slice($posts, 0, 4) as $post)
33                             <div class="col-md-3 mb-4 review-item">
34                                 <div class="card">
35                                     <div class="card-body">
36                                         <div class="reviews">
37                                             <h5 class="book_title">{{ $post['title'] }}</h5>
38                                             @foreach ($post['comments'] as $comment)
39                                                 <span class="text">{{ $comment['text'] }}</span>
40                                                 <span class="time">{{ $comment['time'] }}</span>
41                                             @endforeach
42                                         </div>
43                                     </div>
44                                 </div>
45                             </div>
46                         @endforeach
47                     </div>
48                 </div>
49             </div>
50         </div>
51     </div>
52 </div>
53 <!-- Hidden reviews that will be shown when the "View All" button is clicked -->
54 @foreach (array_slice($posts, 4) as $post)
55     <div class="col-md-3 mb-4 review-item" style="display: none;">
56         <div class="card">
57             <div class="card-body">
58                 <div class="reviews">
59                     <h5 class="book_title">{{ $post['title'] }}</h5>
60                     @foreach ($post['comments'] as $comment)
61                         <span class="text">{{ $comment['text'] }}</span>
62                         <span class="time">{{ $comment['time'] }}</span>
63                     @endforeach
64                 </div>
65             </div>
66         </div>
67     </div>
68 @endforeach
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
101 </div>
102 </div>
103 </div>
104 </div>
105 </div>
106 </div>
107 </div>
108 </div>
109 </div>
110 </div>
111 </div>
112 </div>
113 </div>
114 </div>
115 </div>
116 </div>
117 </div>
118 </div>
119 </div>
120 </div>
121 </div>
122 </div>
123 </div>

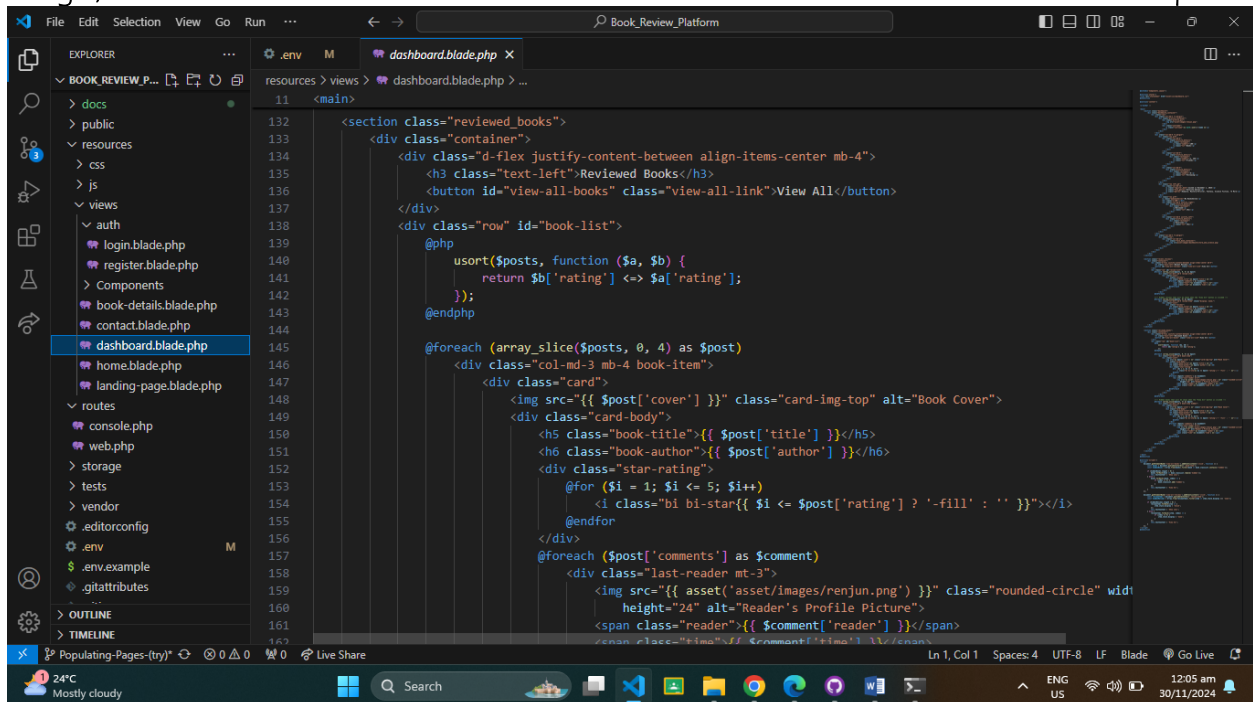
```

## LABORATORY 6 POPULATING PAGES

Bongat, Mae Anne B.

BSIT-3C

Web Development



```
11 <main>
12 <section class="reviewed_books">
13 <div class="container">
14 <div class="d-flex justify-content-between align-items-center mb-4">
15 <h3 class="text-left">Reviewed Books</h3>
16 <button id="view-all-books" class="view-all-link">View All</button>
17 </div>
18 <div class="row" id="book-list">
19 @php
20 usort($posts, function ($a, $b) {
21     return $b['rating'] <=> $a['rating'];
22 });
23 @endphp
24 @foreach (array_slice($posts, 0, 4) as $post)
25 <div class="col-md-3 mb-4 book-item">
26 <div class="card">
27 
28 <div class="card-body">
29 <h5 class="book-title">{{ $post['title'] }}</h5>
30 <h6 class="book-author">{{ $post['author'] }}</h6>
31 <div class="star-rating">
32 @for ($i = 1; $i <= 5; $i++)
33 <i class="bi bi-star" {{ $i <= $post['rating'] ? '-fill' : '' }}></i>
34 @endfor
35 </div>
36 @foreach ($post['comments'] as $comment)
37 <div class="last-reader mt-3">
38 
39 <span class="reader">{{ $comment['reader'] }}</span>
40 </div>
41 </div>
42 </div>
43 @endforeach
44 </div>
45 </div>
46 </main>
```

### Recent Reviews

[Show Less](#)

<b>Omniscient Reader's Viewpoint</b> "An incredible read! So many twists and unforgettable moments!" 1 day ago	<b>Lord of the Mysteries</b> "A thrilling mystery that keeps you on the edge of your seat. Highly recommend!" 2 days ago	<b>A Gentle Reminder</b> "It's a beautiful book with gentle life lessons. Very thought-provoking." 2 days ago	<b>Pride and Prejudice</b> "A classic with timeless romance and witty characters. A must-read!" 3 days ago
<b>First Lie Wins</b> "A fast-paced, exciting thriller. Great book!" 5 days ago	<b>Funny Story</b> "Great book! It's light and fun." 5 days ago	<b>The Women</b> "A heartfelt drama. The story really stays with you after finishing it" 7 days ago	<b>The Teacher</b> "I loved this mystery! It's suspenseful and keeps you guessing." 7 days ago
<b>The Cruel Prince</b> "A captivating fantasy! The world-building is amazing."	<b>Storm and Silence</b> "A charming historical fiction with a strong heroine. Loved	<b>Waves of Memories</b> "A beautiful romance! It'll pull at your heartstrings."	<b>The Invisible Girl</b> "A sweet romance with an intriguing plot. Loved it!"

## LABORATORY 6 POPULATING PAGES

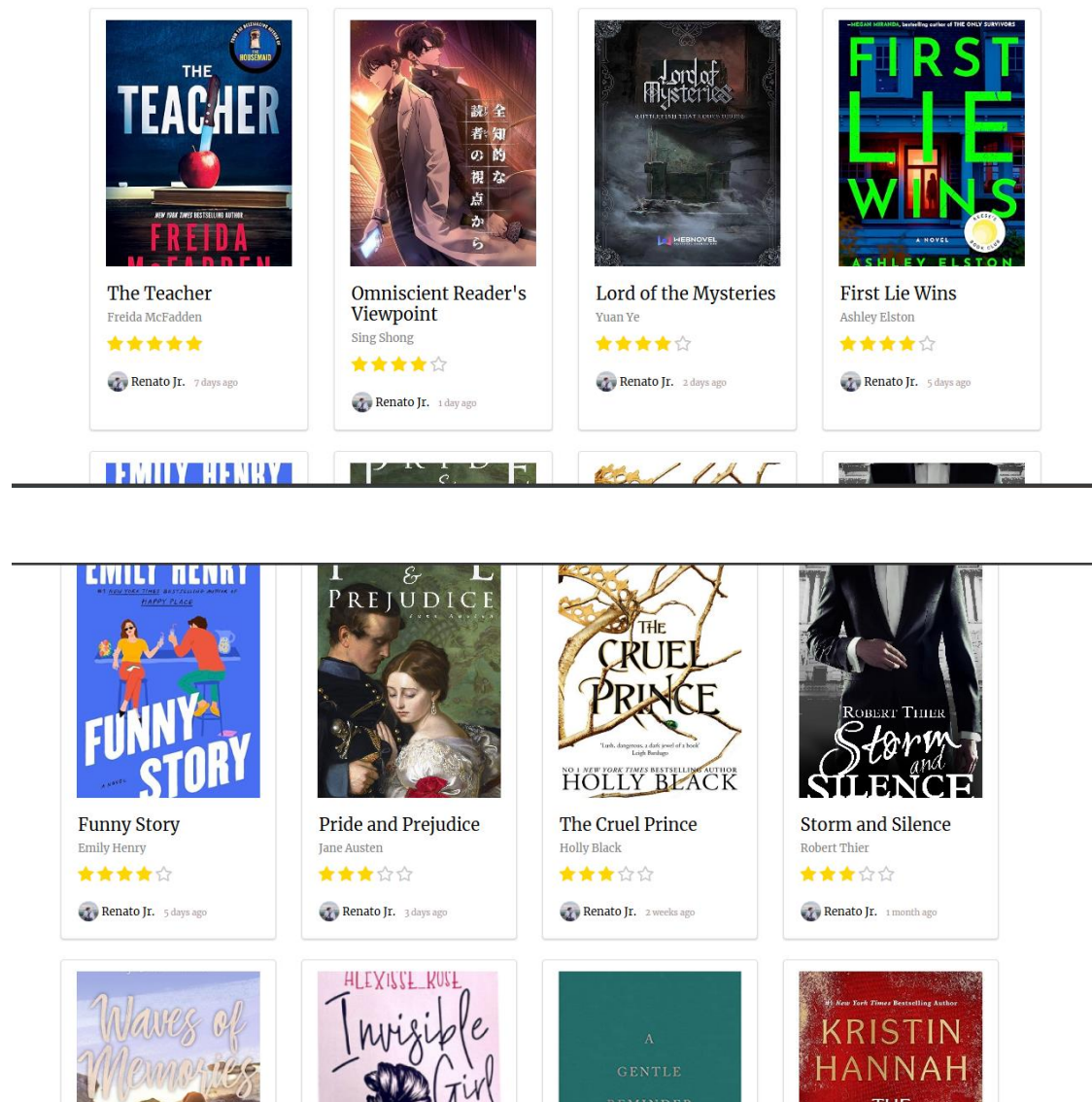
Bongat, Mae Anne B.

BSIT-3C

Web Development

### Reviewed Books

Show Less



In working on this project, I learned how to create views using Blade templates, such as 'landing-page', 'home', 'dashboard', and 'contact', and display them based on routes. I created controllers like 'BooksController', 'HomeController', 'DashboardController', and 'AuthController' to handle the logic for loading views and content. I also defined routes to link URLs to controller methods, which allowed me to navigate between pages for both authenticated users and guests. By applying middleware ('guest' and 'auth'), I ensured that certain routes were accessible only to the appropriate users, such as login and registration for guests, and protected routes like '/home' and '/dashboard' for authenticated users. Additionally, I simulated database access by loading content, like books or posts, from arrays, which helped me understand how data can be fetched and displayed on the front end. This process gave me a deeper understanding of Laravel's routing, controllers, views, and middleware, which are essential for building dynamic web applications.