



**BICOL UNIVERSITY COLLEGE OF SCIENCE**  
**LEGAZPI CITY, ALBAY**



# **IT Elect 1 – WEB DEVELOPMENT**

# **DOCUMENTATION**

**(Laboratory 5, CONTROLLER)**

**SUBMITTED BY:**

**JERALD JAY G. BUBAN**

**BSIT – 3C**

## Part 1: Create and Register Controllers

- Create HomeController (loads home page) and DashboardController (loads dashboard, feed, or equivalent pages).

HomeController:

EXPLORER

BOOK\_REVIEW...

app

Http\ Controllers

AuthController.php

Controller.php

DashboardController.php

HomeController.php

Models

Providers

bootstrap

config

database

public

asset

htaccess

HomeController.php

app > Http > Controllers > HomeController.php > ...

1 <?php

2

3 namespace App\Http\Controllers;

4

5 use Illuminate\Http\Request;

6 -----

7 class HomeController extends Controller

8 {

9 | 1 reference | 0 overrides

10 | public function index (): Factory|View

11 | | return view (view: 'landing-page');

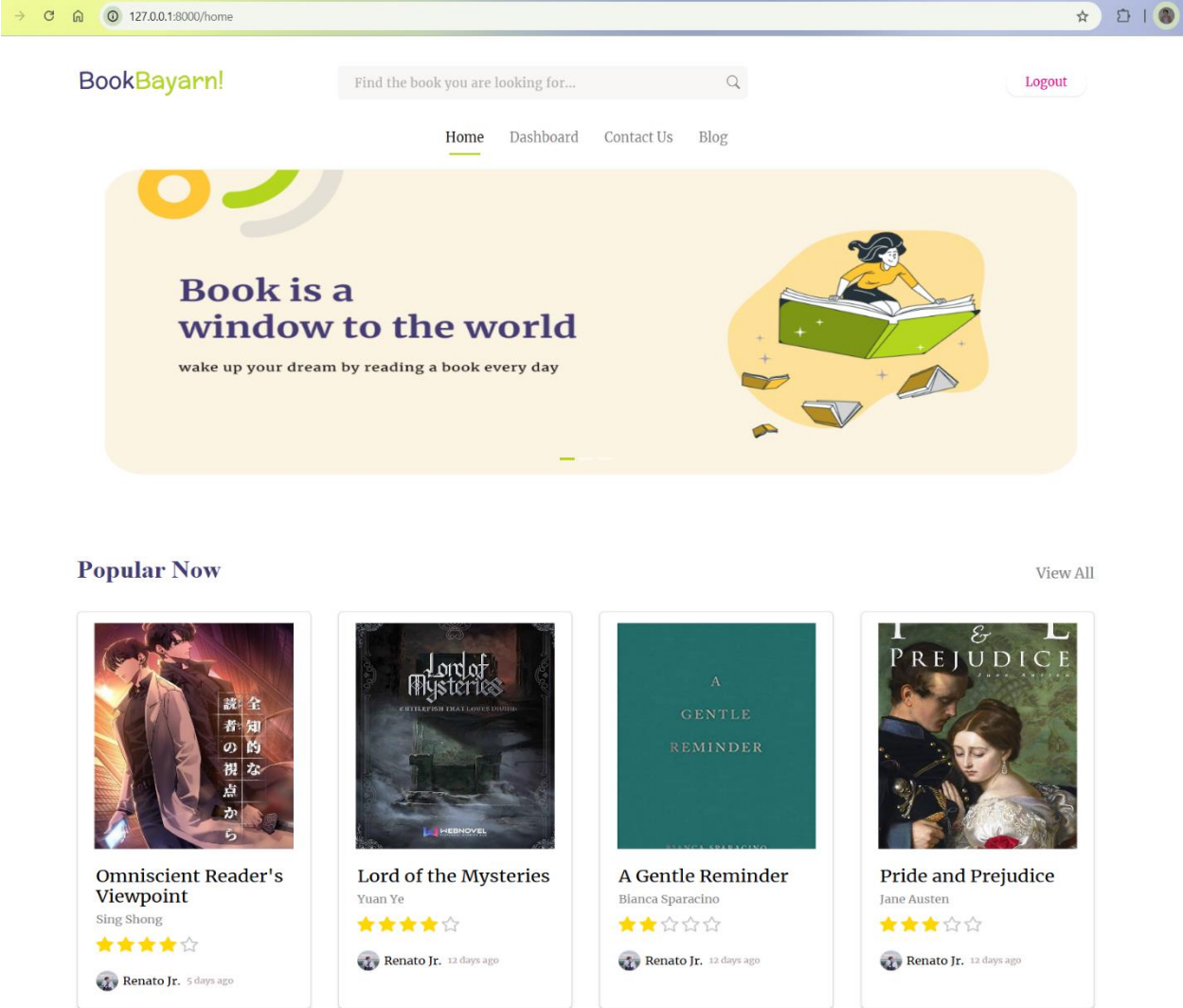
12 | }

13 }

14

- The `HomeController` serves as the controller for handling the home or the landing page of our website. The `index` method within this controller is responsible for responding to a request made to the home route. When the `index` method is triggered, it returns the `landing-page` view, which displays the main landing content of our website. This setup allows our website to render the landing page when users visit the homepage, making it an effective and organized way to manage route handling and page rendering.

Loads Home Page:



Latest Books

[View All](#)



First Lie Wins

Ashley Elston

★★★★☆

 Renato Jr. 5 days ago

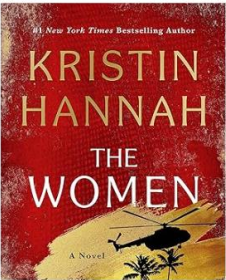


Funny Story

Emily Henry

★★★★☆


 Renato Jr. 12 days ago



The Women

Kristin Hannah

★★★☆☆

 Renato Jr. 12 days ago



The Teacher

Freida McFadden

★★★★★

 Renato Jr. 12 days ago

BookBayarn!

Find your next favorite read!

Quick Links

- [Home](#)
- [Categories](#)
- [Community](#)
- [Blog](#)

About

Developed by 3rd Year IT students of Bicol University  
[bookbayarn!@gmail.com](mailto:bookbayarn!@gmail.com)

© 2024 BookBayarn! All rights reserved.

DashboardController:

DashboardController.php

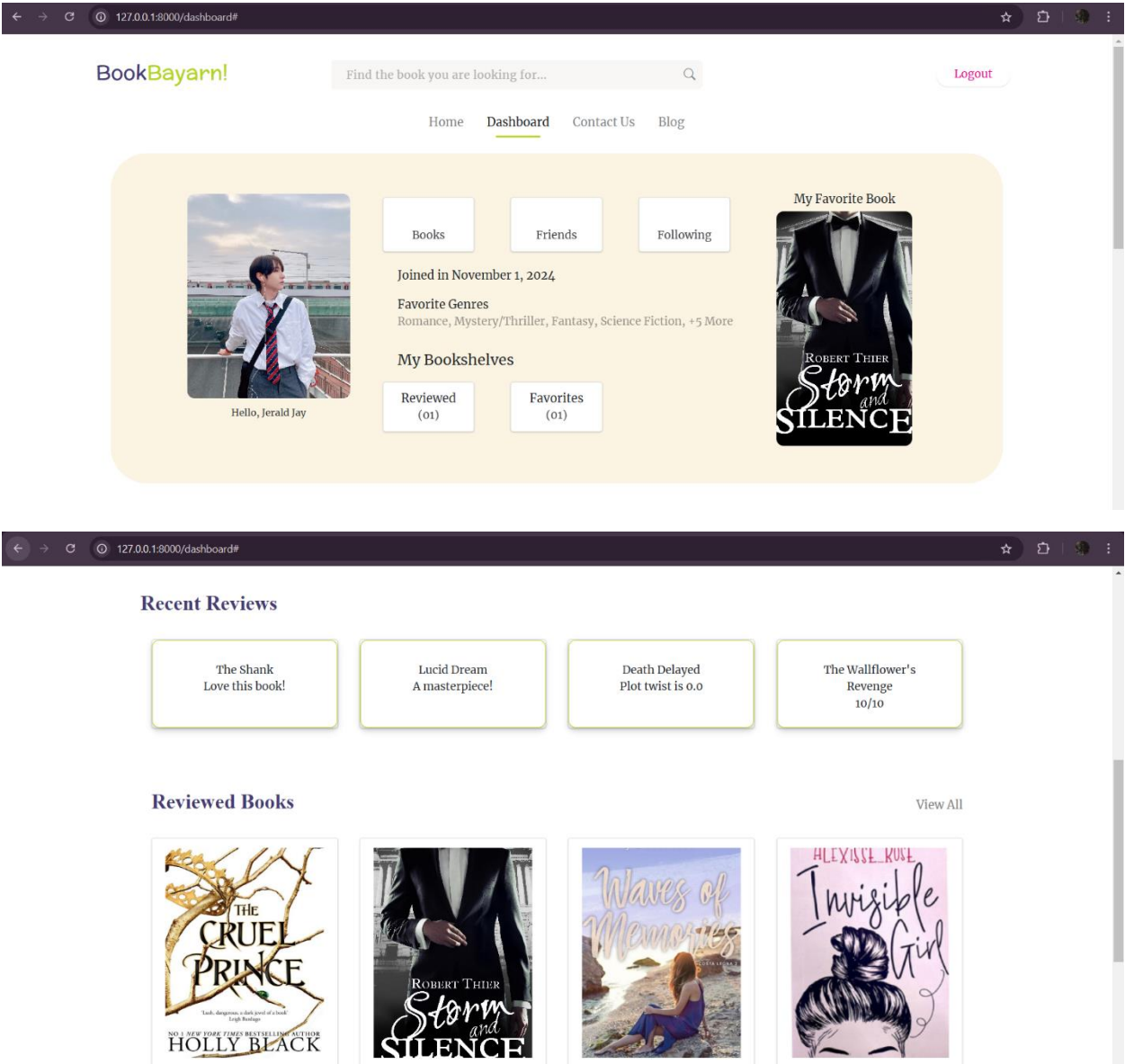
app > Http > Controllers > DashboardController.php > ...

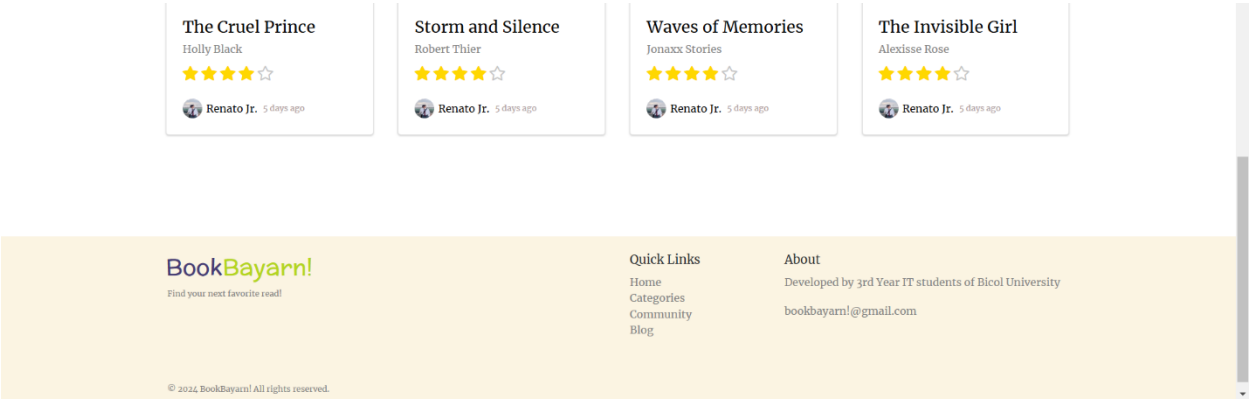
```
1 <?php
2
3 // app/Http/Controllers/DashboardController.php
4
5 namespace App\Http\Controllers;
6
7 use Illuminate\Http\Request;
8 use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
9 use Illuminate\Support\Facades\Auth;
10 use App\Models\User;
11
12
13 3 references | 0 implementations
14 class DashboardController extends Controller
15 {
16     1 reference | 0 overrides
17     public function index(): Factory|View
18     {
19         return view(view: 'dashboard');
20     }
21
22     1 reference | 0 overrides
23     public function showBook($book_id): Factory|View
24     {
25         // Dummy book data
26         $books = [
27             1 => [
28                 'title' => 'The Great Gatsby',
29                 'author' => 'F. Scott Fitzgerald',
30                 'review' => 'A fascinating story of the Jazz Age.',
31                 'rating' => 5
32             ],
33             2 => [
34                 'title' => '1984',
35                 'author' => 'George Orwell',
36                 'review' => 'A dystopian novel about totalitarianism.',
37                 'rating' => 4
38             ]
39         ];
40     }
41 }
```

```
35         ]
36     };
37
38     // Check if the book exists in the dummy data
39     if (!isset($books[$book_id])) {
40         abort(code: 404, message: 'Book not found');
41     }
42
43     // Return the view with the book details
44     return view(view: 'book', data: ['book' => $books[$book_id]]);
45 }
46
47 }
48 }
49
50
51
```

- The `DashboardController` controls access to the user dashboard and specific book details. The `index` method loads and returns the main `dashboard` view when accessed, allowing users to interact with the dashboard interface. The `showBook` method takes a `\$book\_id` parameter and uses it to retrieve specific book data from a dummy list of books. Each book entry contains details like title, author, review, and rating. If the requested book ID does not exist in the predefined data, the method triggers a 404 error, signaling that the book was not found. If the book exists, it returns a `book` view that displays the selected book's details, enhancing the user experience by allowing users to view specific book information directly from the dashboard.

Loads dashboard, feed, or equivalent pages:





- The dashboard also shows the users profile with its name on it, and the books he reviewed and the reviews he/she wrote.

• **Register controllers in routes to link methods to URLs.**

```
web.php M X
routes > web.php
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\HomeController;
5 use App\Http\Controllers\DashboardController;
6 use App\Http\Controllers\AuthController;
7
8 // Landing Page
9 Route::get(uri: '/', action: [HomeController::class, 'index'])->name(name: 'landing-page');
10
11
12 // Authentication Routes (only for guests)
13 Route::middleware(middleware: 'guest')->group(callback: function(): void {
14     Route::get(uri: "/login", action: [AuthController::class, 'login'])->name(name: 'login');
15     Route::post(uri: "/login", action: [AuthController::class, "loginPost"])->name(name: "login.post");
16     Route::get(uri: "/register", action: [AuthController::class, "register"])->name(name: "register");
17     Route::post(uri: "/register", action: [AuthController::class, "registerPost"])->name(name: "register.post");
18 });
19
20 // Authenticated Routes (for logged-in users)
21 Route::middleware(middleware: 'auth')->group(callback: function(): void {
22     Route::get(uri: '/contact', action: function (): Factory|View {
23         return view(view: 'contact');
24     });
25
26     Route::get(uri: '/home', action: function (): Factory|View {
27         return view(view: 'home');
28     });
29     Route::get(uri: '/dashboard', action: [DashboardController::class, 'index'])->name(name: 'dashboard.show');
30
31     Route::get(uri: '/dashboard/book/{book_id}', action: [DashboardController::class, 'showBook']);
32
33     Route::post(uri: '/logout', action: function (): Redirector|RedirectResponse { // Logout Route
34         Auth::logout(); // Logs out the user
35         return redirect(to: route(name: "landing-page"));
36     })->name(name: 'logout');
37 });
```

- The landing page is handled by the `HomeController`'s `index` method and is accessible via the `/` route. Authentication-related routes are grouped under the `guest` middleware, meaning they are only accessible to unauthenticated users. Within this group, the `/login` and /register` routes load login and registration pages via `AuthController` and handle login and registration submissions.
- For logged-in users, routes are grouped under the `auth` middleware, restricting them to authenticated users only.
- This includes routes for `/contact` and `/home`, which display contact and home views, respectively.
- The `/dashboard` route triggers the `DashboardController`'s `index` method to display the main dashboard page.
- Users can view specific books by visiting `/dashboard/book/{book\_id}`, where the `showBook` method loads detailed book information based on the ID.
- The logout route logs out the user, redirecting them back to the landing page.

Authentication Controller:

AuthController.php X

app > Http > Controllers > AuthController.php > ...

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\User;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\Auth;
8 use Illuminate\Support\Facades\Hash;
9
10 class AuthController extends Controller
11 {
12     1 reference | 0 overrides
13     public function login(): Factory|View
14     {
15         return view(view: 'auth.login');
16     }
17
18     1 reference | 0 overrides
19     function loginPost(Request $request): mixed|RedirectResponse
20     {
21         $request->validate(rules: [
22             "email" => 'required',
23             "password" => 'required',
24         ]);
25         $credentials = $request->only(keys: "email","password");
26         if (Auth::attempt(credentials: $credentials)){
27             return redirect()->route(route: 'dashboard.show', parameters: ['email' => Auth::user()->email]);
28         }
29         return redirect(to: route(name: "login"))->with(key: "error",value: "Login failed");
30     }
31
32     1 reference | 0 overrides
33     function register(): Factory|View{
34         return view(view: 'auth.register');
35     }
36
37     1 reference | 0 overrides
38     function registerPost(Request $request): mixed|RedirectResponse
39     {
40         $request->validate(rules: [
41             "fullname" => 'required',
42             "email" => 'required',
43             "password" => 'required',
44         ]);
45         $user = new User();
46         $user->name = $request->fullname;
47         $user->email = $request->email;
48         $user->password = Hash::make(value: $request->password);
49
50         if ($user->save()){
51             return redirect(to: route(name: "login"))->with(key: "success", value: "User created successfully");
52         }
53         return redirect(to: route(name: "register"))->with(key: "error",value: "Failed to create account");
54     }
55
56
57 }
58
```

- The `AuthController` manages user authentication processes, including login and registration.
- The `login` method loads the login view, displaying a form for users to enter their credentials. When a user submits the form, the `loginPost` method is triggered, validating the input to ensure both "email" and "password" fields are filled. It then attempts to authenticate the user with these credentials. If successful, the user is redirected to the dashboard with their email in the route. If the login fails, the user is redirected back to the login page with an error message.
- The `register` method loads the registration view, allowing new users to input their details. When the form is submitted, the `registerPost` method validates that "fullname," "email," and "password" fields are completed. It then creates a new `User` instance, assigning the name, email, and a hashed version of the password.



## Part 2: Assign Controllers to Routes

- Use middleware (e.g., authentication) to protect specific routes and controllers.

```
// Authentication Routes (only for guests)
Route::middleware(middleware: 'guest')->group(callback: function(): void {
    Route::get(uri: "/login", action: [AuthController::class, 'login'])->name(name: 'login');
    Route::post(uri: "/login", action: [AuthController::class, "loginPost"])->name(name: "login.post");
    Route::get(uri: "/register", action: [AuthController::class, "register"])->name(name: "register");
    Route::post(uri: "/register", action: [AuthController::class, "registerPost"])->name(name: "register.post");
});
```

- The `guest` middleware ensures that the routes are only accessible to users who have not authenticated.
- The `/login` route loads the login view through the `login` method in the `AuthController`, while `/login` with a POST request triggers the `loginPost` method to process login submissions.
- The `/register` loads the registration view via the `register` method, and a POST request to `/register` activates the `registerPost` method to handle registration form submissions.
- By grouping these routes under the `guest` middleware, the code restricts these pages to only unauthenticated users.

```
// Authenticated Routes (for logged-in users)
Route::middleware(middleware: 'auth')->group(callback: function(): void {
    Route::get(uri: '/contact', action: function (): Factory|View {
        | return view(view: 'contact');
    });

    Route::get(uri: '/home', action: function (): Factory|View {
        | return view(view: 'home');
    });
    Route::get(uri: '/dashboard', action: [DashboardController::class, 'index'])->name(name: 'dashboard.show');

    Route::get(uri: '/dashboard/book/{book_id}', action: [DashboardController::class, 'showBook']);

    Route::post(uri: '/logout', action: function (): Redirector|RedirectResponse { // Logout Route
        | Auth::logout(); // Logs out the user
        | return redirect(to: route(name: "landing-page"));
    })->name(name: 'logout');
});
```

- The `/contact` and `/home` routes load their respective views directly, displaying contact and home page content for authenticated users.
- The `/dashboard` route uses the `DashboardController`'s `index` method to load the main dashboard view, while `/dashboard/book/{book\_id}` uses the `showBook` method in the same controller to display details for a specific book based on its ID.
- The `logout` route logs the user out by calling `Auth::logout()` and redirects them back to the landing page.
- By grouping these routes with the `auth` middleware, the code restricts access to only authenticated users, keeping certain sections of the application secure and only available to logged-in users.

- Test routes (e.g. /, /dashboard) to ensure proper page loading.

```
// Test routes
Route::get(uri: '/dashboard', action: [DashboardController::class, 'index'])->name(name: 'dashboard.show');

Route::get(uri: '/dashboard/book/{book_id}', action: [DashboardController::class, 'showBook']);
```

- The `/dashboard` route uses the `DashboardController`'s `index` method to load the main dashboard view, while `/dashboard/book/{book\_id}` uses the `showBook` method in the same controller to display details for a specific book based on its ID.

## Part 3: Controllers with Parameters

- Modify DashboardController to accept dynamic parameters (e.g., user ID).

book.blade.php

resources > views > book.blade.php

1

<h1>Book Details</h1>

2

3

<h2>{{ \$book['title'] }}</h2>

4

<p><strong>Author:</strong> {{ \$book['author'] }}</p>

5

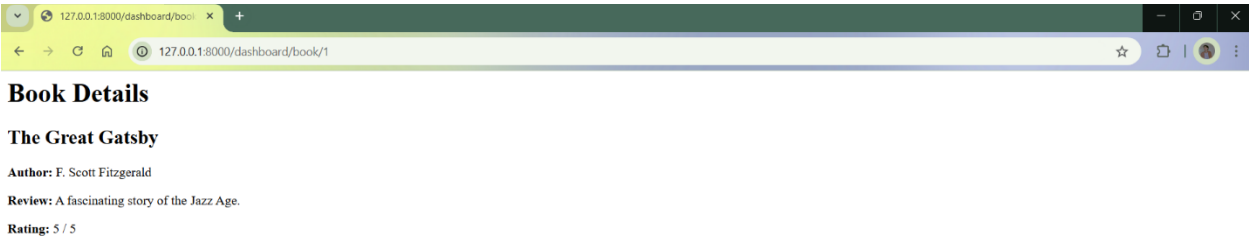
<p><strong>Review:</strong> {{ \$book['review'] }}</p>

6

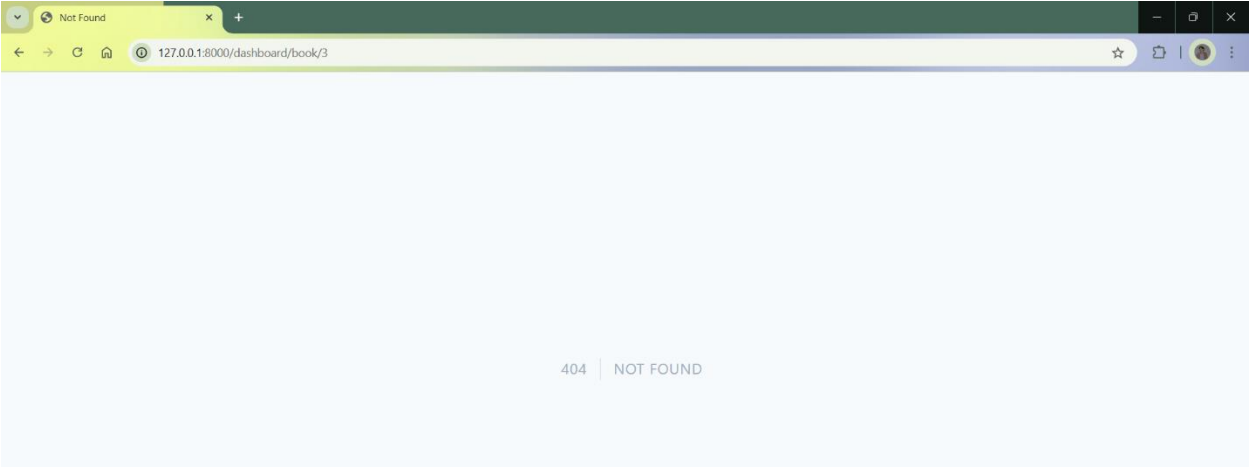
<p><strong>Rating:</strong> {{ \$book['rating'] }} / 5</p>

7

- Book sample for controller with dynamic parameter purpose.



- The book details will show, since we put a default information to the book 1 and 2.



- Since we didn't put a default information to book 3, it will redirect to 404 error.



Other Rendered Page:

Contact Us

## CONTACT US

### We'd Love To Hear From You! Let's Get in Touch

Full Name

Enter name

Company

Company name

Email

example@gmail.com

Phone number


+1 (555) 000 - 0000

Address

Your Message

Type Your Message here

Send Message



Contact Us Code

```
36 |         <div class="col-md-6">
37 |             <label>Phone Number</label>
38 |             <textarea name="phone" placeholder="+1 (555) 000 - 0000" rows="1"></textarea>
39 |         </div>
40 |     </div>
41 |     <div class="mb-3">
42 |         <label>Address</label>
43 |         <textarea name="address" placeholder="Brgy, City, Province" rows="1"></textarea>
44 |     </div>
45 |     <div class="message-form mb-3">
46 |         <label>Your Message</label>
47 |         <textarea name="message" placeholder="Type your message here" rows="5"></textarea>
48 |     </div>
49 |     <button type="submit" class="btn btn-primary">Send Message</button>
50 | </div>
51 </div>
52 
53 </div>
54 </main>
55 </body>
56 </html>
57
```

```
contact.blade.php
resources > views > contact.blade.php
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Contact Us</title>
7 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
8 | integrity="sha384-QMtkZyPPEjiv5WmaRU90fErpokenvwmDr5pNlyT2BRjXh03MhJv6hW+ALEwIH" crossorigin="anonymous">
9 <link rel="stylesheet" href="asset/css/contact.css">
10 <link href="https://fonts.googleapis.com/css2?family=Merriweather:ital,wght@0,300;0,400;0,700;0,900;1,300;1,400;1,700;1,900&family=raleway:1
11 </head>
12
13 <body>
14 <main>
15 <h1>CONTACT US</h1>
16 <div class="container">
17 <div class="contact-box">
18 <p>We'd Love To Hear From You! <br> Let's Get in Touch</p>
19
20 <div class="form-fields">
21 <div class="row mb-3">
22 <div class="col-md-6">
23 <label>Full Name</label>
24 <textarea name="fullname" placeholder="Enter name" rows="1"></textarea>
25 </div>
26 <div class="col-md-6">
27 <label>Company</label>
28 <textarea name="company" placeholder="Company name" rows="1"></textarea>
29 </div>
30 </div>
31 <div class="row mb-3">
32 <div class="col-md-6">
33 <label>Email</label>
34 <textarea name="email" placeholder="example@gmail.com" rows="1"></textarea>
35 </div>
```