BICOL UNIVERSITY

COLLEGE OF SCIENCE

Legaspi City

Controllers

Laboratory Acitvity

RICHARD D. BILAN JR.

BSIT 3-C

It Elec 1 - Web Development

- Part 1: Create and Register Controllers
    - Create HomeController (loads home page) and DashboardController (loads dashboard, feed, or equivalent pages).
    - Register controllers in routes to link methods to URLs.
    - You may create and register more controllers if needed.
- Part 2: Assign Controllers to Routes
    - Use middleware (e.g., authentication) to protect specific routes and controllers.
    - Test routes (e.g. /, /dashboard) to ensure proper page loading.
- Part 3: Controllers with Parameters
    - Modify DashboardController to accept dynamic parameters (e.g., user ID).
    - Test parameterized routes to load user-specific data.
- Part 4: Documentation (Individual)
    - Take screenshots of controllers, routes, and rendered pages.
    - Write brief explanations of controller logic, parameter handling, and route assignments.
-
Controller Creation and Registration (30%):
    - Correctly define and implement HomeController and DashboardController.
    - Ensure controllers are properly registered in the routes file to link methods to URLs.
    - If applicable, create and register additional controllers as needed for handling different routes.
- Assigning Controllers to Routes (30%):
    - Properly assign controllers to routes, ensuring the correct methods are linked to specific URLs (e.g., home page, dashboard, feed).
    - Use middleware (e.g., authentication) where necessary to protect specific routes.
    - Ensure that routes are tested and load the correct pages as expected.
- Handling Controllers with Parameters (20%):
    - Modify DashboardController to accept dynamic parameters (e.g., user ID).
    - Ensure that parameterized routes are properly configured to load user-specific data.
    - Test the parameterized routes to confirm they are functioning correctly.
- Documentation and Explanation (20%):
    - Take clear screenshots of Blade templates, route configurations, and rendered pages.
    - Provide concise and detailed explanations of controller logic, how parameters are handled, and how routes are assigned.
    - Ensure that the documentation is well-organized and demonstrates a strong understanding of the concepts.

- Part 1: Create and Register Controllers

```
php artisan make:controller HomeController
php artisan make:controller DashboardController
```

I used Laravel's Artisan command to create the HomeController, which will handle loading the home page

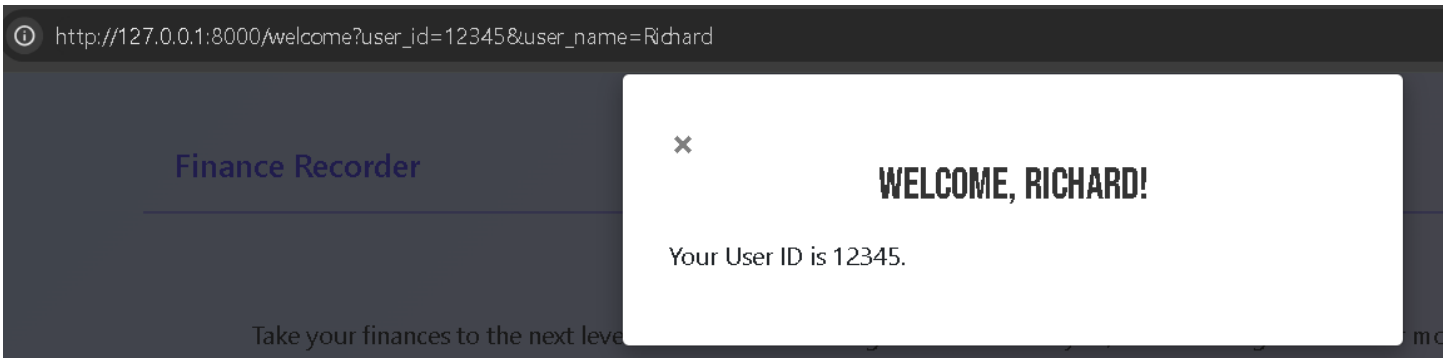
```php
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

4 references | 0 implementations
class HomeController extends Controller
{
    2 references | 0 overrides
    public function welcome(Request $request): Factory|View
    {
        $userId = $request->query('user_id', '');  // Get userId from URL
        $userName = $request->query('user_name', '');  // Get userName from URL
        return view(view: 'welcome', data: compact(var_name: 'userId', var_names: 'userName'));
    }
}

```

The purpose of the welcome method in the HomeController is to retrieve dynamic data from the URL query parameters and pass them to a view. In this case, the method extracts the user_id and user_name from the query string of the URL, allowing for a personalized response based on the user's input.



The method works by Request object to capture the query parameters. If the parameters are present in the URL, they are assigned to the $userId and $userName variables. If not, default values (empty strings) are used. These values are then passed to the welcome view using the compact() function, which makes them accessible in the view. This allows the view to dynamically display the user ID and name, creating a more personalized experience for the user.

Part 2: Assign Controllers to Routes

Middleware is applied to protect specific routes from unauthorized access. Here's how it's implemented:

Using the auth Middleware

```
// Simulate login for demonstration (replace with actual login logic)
Auth::loginUsingId($userId);

// Debugging output to verify role
\Log::info('User ID: ' . Auth::id());  // Logs the user ID
\Log::info('User Role: ' . Auth::user()->role);  // Logs the user's role

// Redirect based on role
```

- The middleware auth is applied to the /dashboard/{userId} route, which restricts access to this route for only authenticated users.
- This is done by wrapping the route inside a Route::middleware('auth')->group(function() {...}) block. The auth middleware checks if the user is logged in.
- If the user is authenticated, the route proceeds to the DashboardController method, where further checks for the user's role (admin) are done.

- Part 3: Controllers with Parameters

To make the DashboardController more dynamic, pass the user ID parameter to the dashboard route. This way, I can load user-specific data based on the user ID provided in the URL.

```
// Access-denied route for unauthorized access
Route::get(uri: '/access-denied', action: function (): Factory|View {
    return view(view: 'access-denied'); // Ensure access-denied.blade.php exists
})->name('access-denied');

// Dashboard route restricted to admins only
Route::get(uri: '/dashboard/{userId}', action: function ($userId): never {
    dd(vars: 'Dashboard route hit', $userId);

    if (Auth::check() && Auth::user()->role == 'admin') {
        return app(abstract: DashboardController::class)->show($userId);
    } else {
        return redirect()->route('access-denied');
    }
})->name('dashboard.show');

// Catch-all route for unauthorized access to any undefined paths within the dashboard
Route::get(uri: '/dashboard/{any}', action: function (): mixed|RedirectResponse {
    return redirect()->route('access-denied');
})->where('any', '.*');
```

Access Denied Route

The `/access-denied` route redirects users to an access-denied page when they try to access restricted areas without proper permissions.

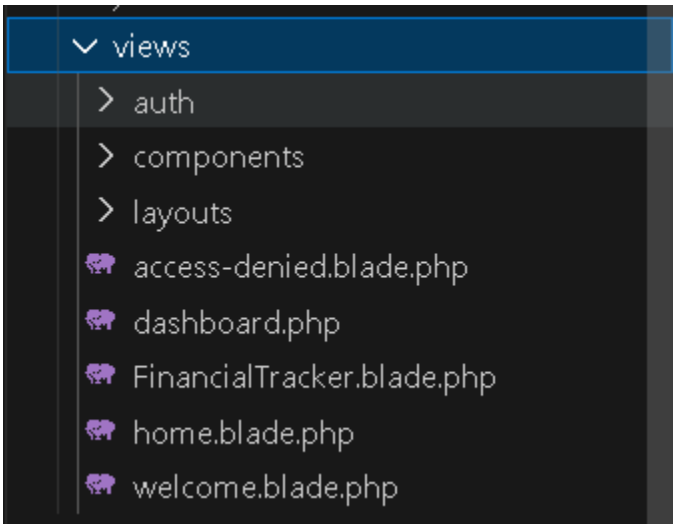
Dashboard Route Restricted to Admins

The `/dashboard/{userId}` route is protected, allowing only authenticated users with an `admin` role to access it. If the user is not logged in or lacks the correct role, they are redirected to the access-denied page.

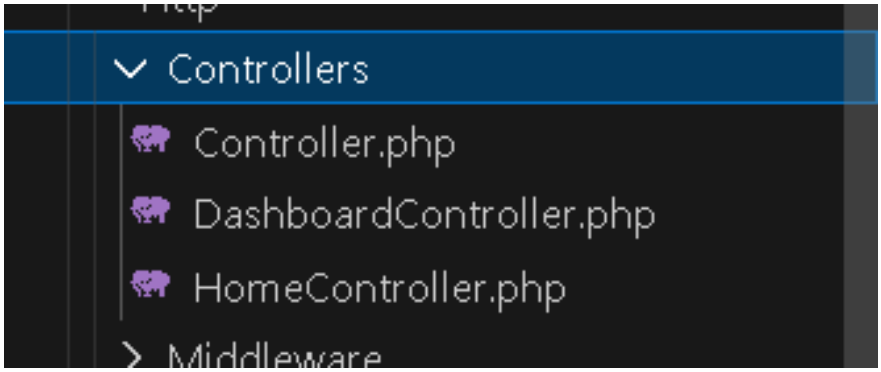
Catch-All Route for Unauthorized Dashboard Paths

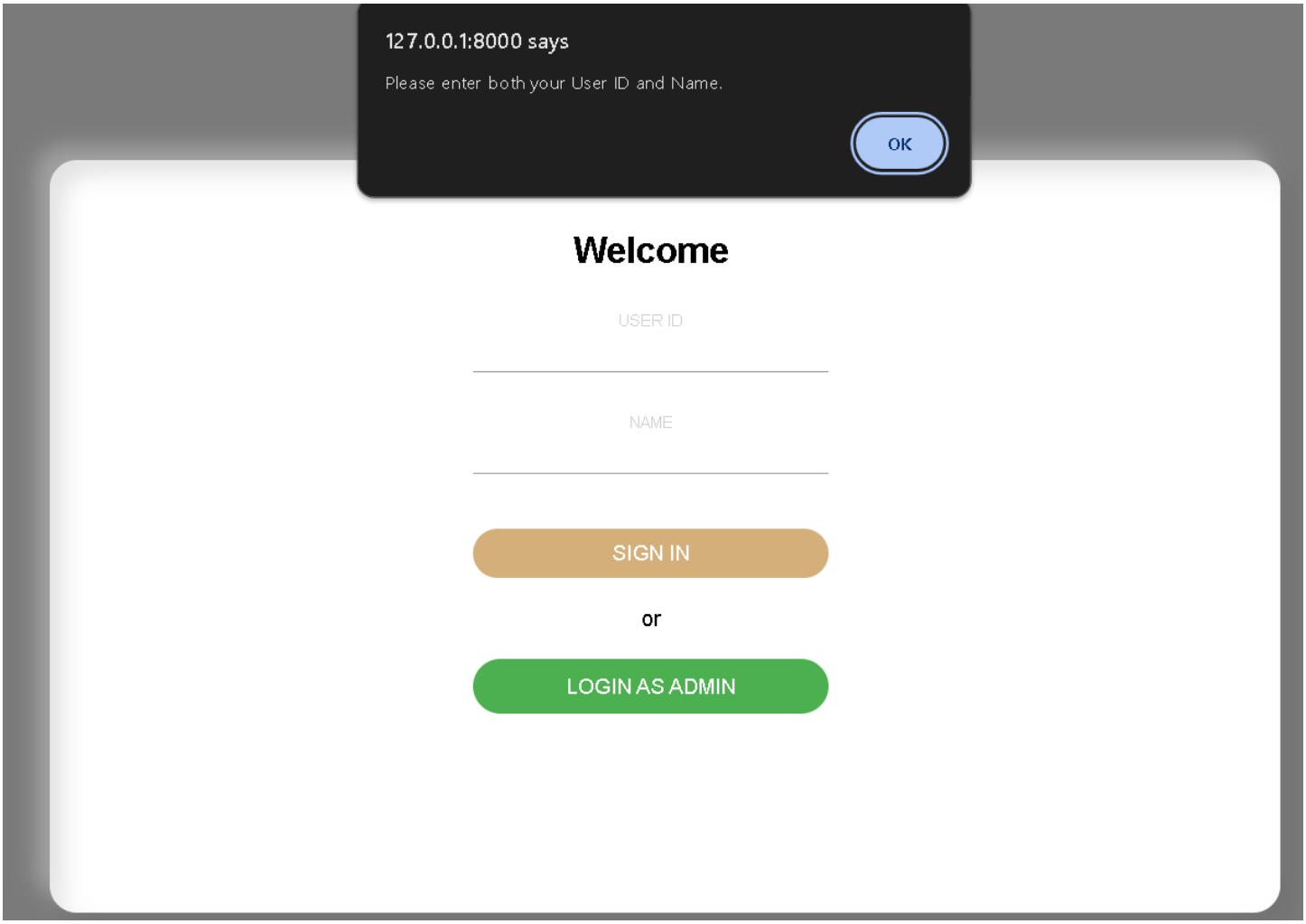
The `/dashboard/{any}` route captures any undefined dashboard URL paths and redirects unauthorized users to the access-denied page.

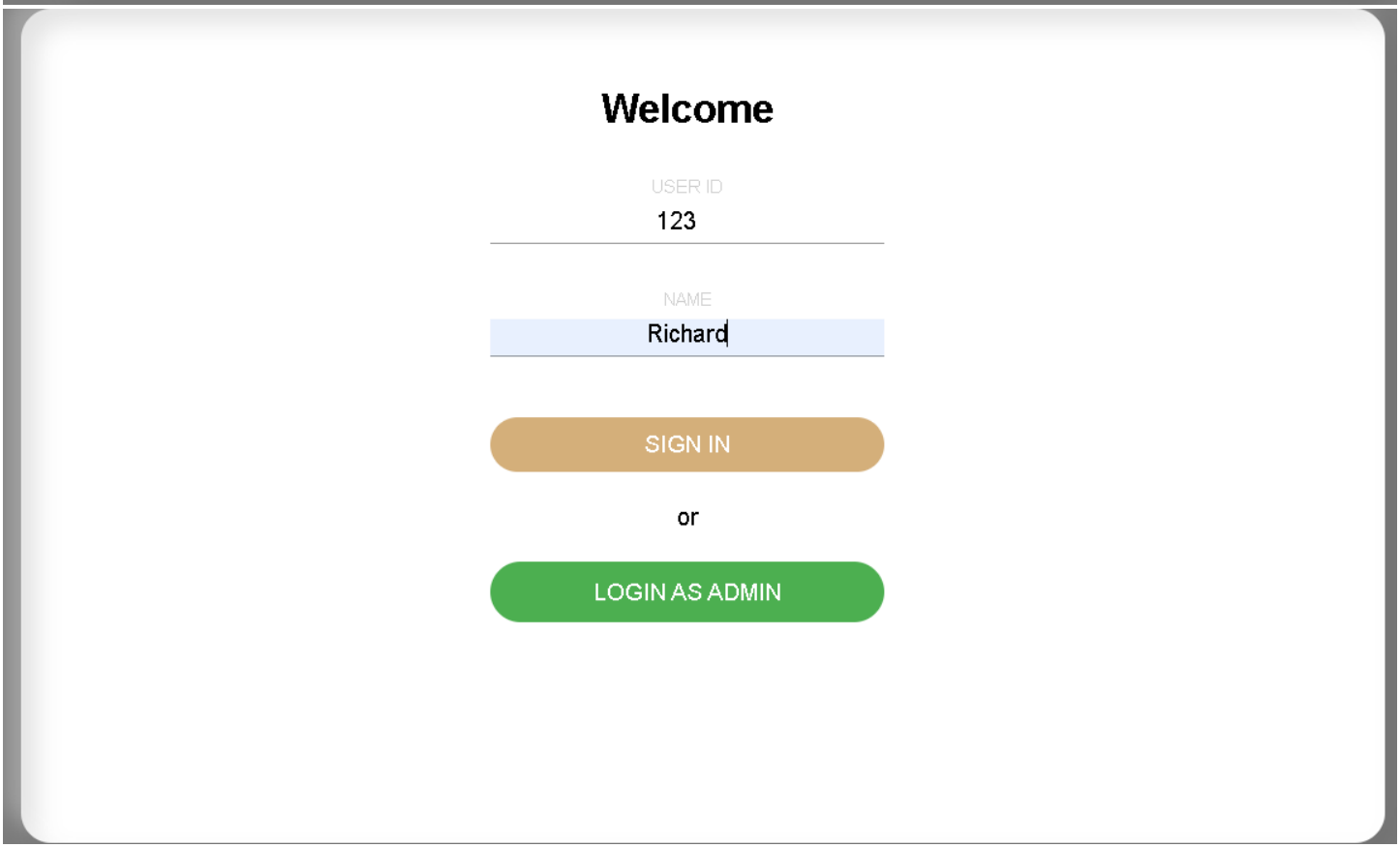
VIEWS

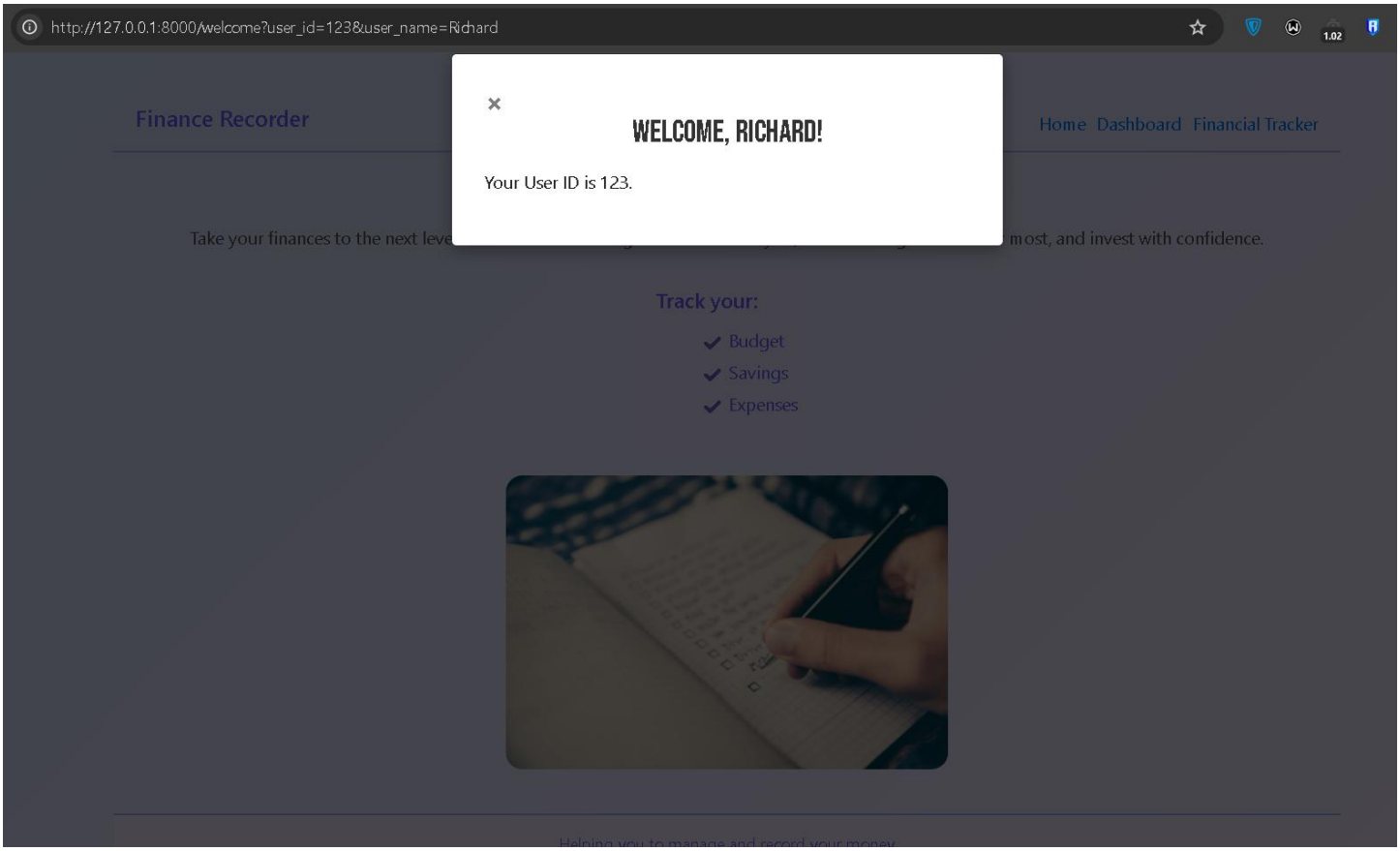


CONTROLLERS




127.0.0.1:8000 says
Please enter both your User ID and Name.
OK
Welcome
USER ID
NAME
SIGN IN
or
LOGIN AS ADMIN


Welcome
USER ID
123
NAME
Richard
SIGN IN
or
LOGIN AS ADMIN


http://127.0.0.1:8000/welcome?user_id=123&user_name=Richard
Finance Recorder
Home Dashboard Financial Tracker
WELCOME, RICHARD!
Your User ID is 123.
Take your finances to the next leve
most, and invest with confidence.
Track your:
Budget
Savings
Expenses