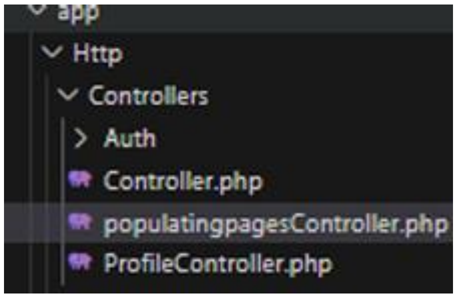
Casimiro, Althea M.
BSIT - 3C

Web Development

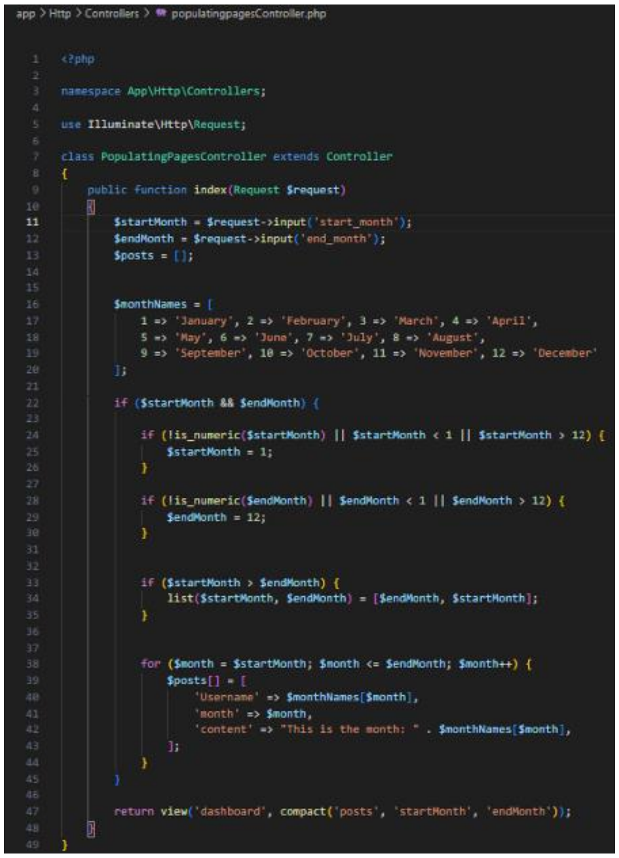
Lab Activity

Populating Pages

- These are the controllers we created or added.

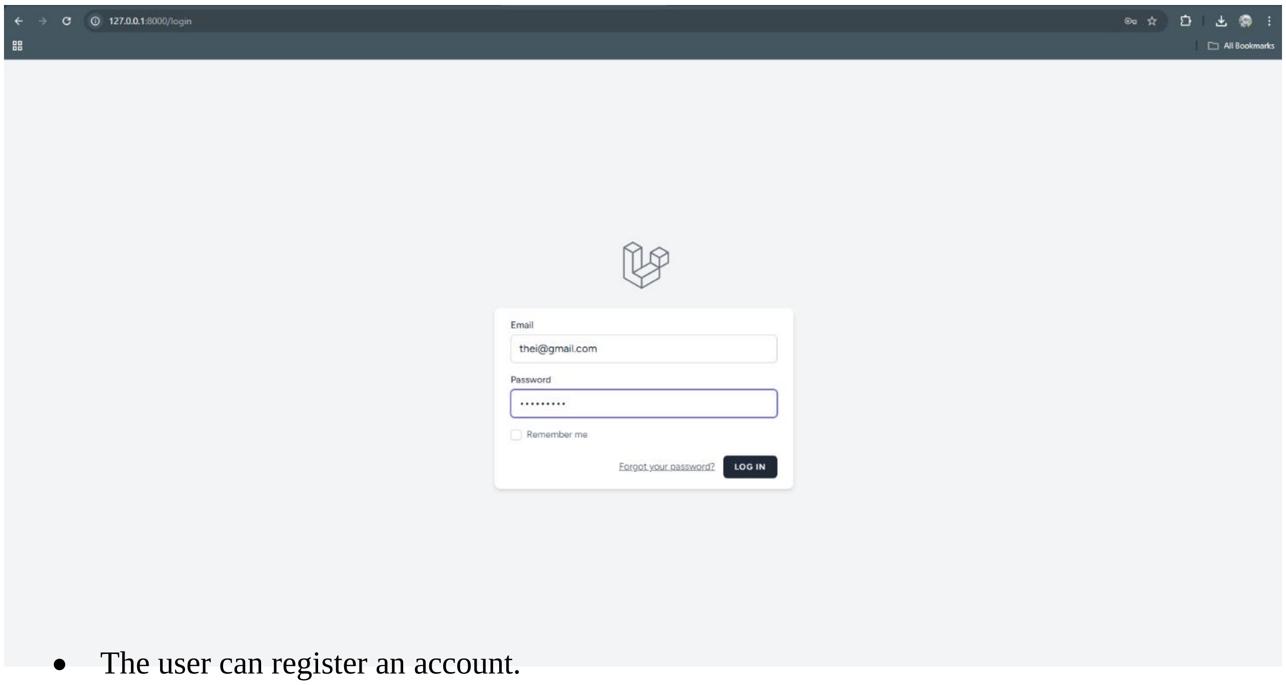


- This controller is used to populate the dashboard by allowing users to specify a range of months to track their expenses. The user provides a starting month (start_month) and an ending month (end_month), with months represented as numbers (e.g., January = 1, December = 12). Based on the selected range, the dashboard generates and displays the corresponding months, such as showing January to March if the user selects 1 as the start and 3 as the end.


app > Http > Controllers > populatingpagesController.php

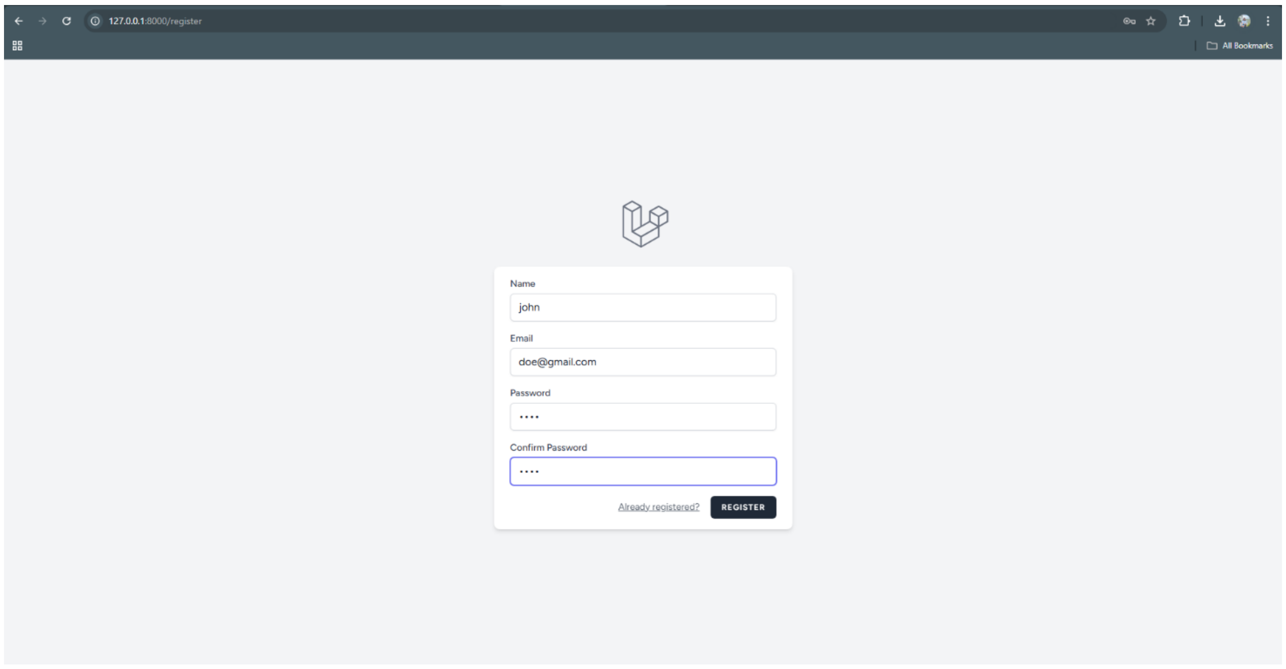
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PopulatingPagesController extends Controller
{
    public function index(Request $request)
    {
        $startMonth = $request->input('start_month');
        $endMonth = $request->input('end_month');
        $posts = [];


        $monthNames = [
            1 => 'January', 2 => 'February', 3 => 'March', 4 => 'April',
            5 => 'May', 6 => 'June', 7 => 'July', 8 => 'August',
            9 => 'September', 10 => 'October', 11 => 'November', 12 => 'December'
        ];

        if ($startMonth && $endMonth) {

            if (!is_numeric($startMonth) || $startMonth < 1 || $startMonth > 12) {
                $startMonth = 1;
            }

            if (!is_numeric($endMonth) || $endMonth < 1 || $endMonth > 12) {
                $endMonth = 12;
            }


            if ($startMonth > $endMonth) {
                list($startMonth, $endMonth) = [$endMonth, $startMonth];
            }


            for ($month = $startMonth; $month <= $endMonth; $month++) {
                $posts[] = [
                    'Username' => $monthNames[$month],
                    'month' => $month,
                    'content' => "This is the month: " . $monthNames[$month],
                ];
            }
        }

        return view('dashboard', compact('posts', 'startMonth', 'endMonth'));
    }
}
```

- Sample login page.


127.0.0.1:8000/login

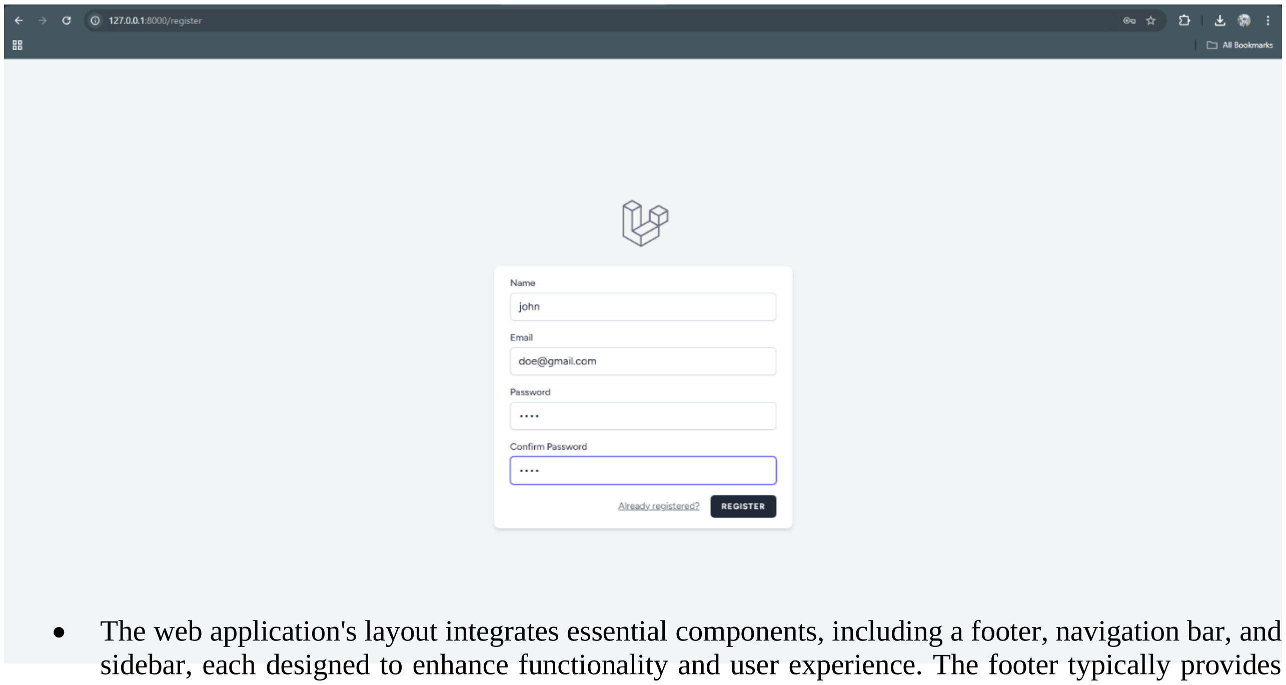
Email
thei@gmail.com
Password
•••••••••
Remember me

- The user can register an account.


127.0.0.1:8000/register

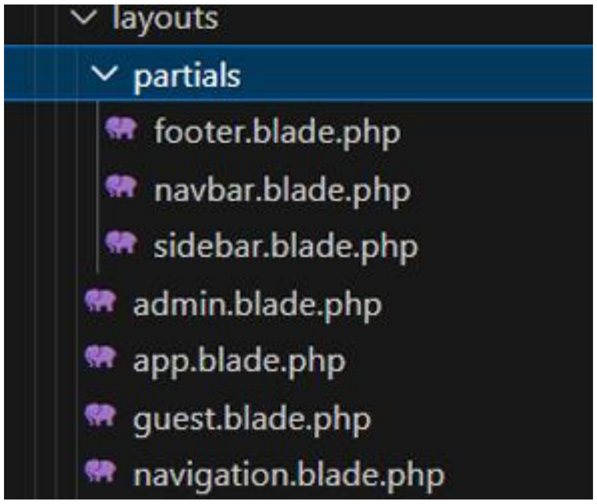
Name
john
Email
doe@gmail.com
Password
••••
Confirm Password
••••

- This is what it looks like after logging in.



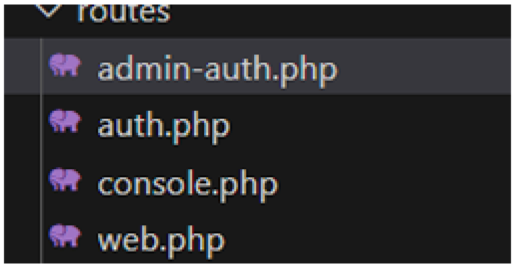
Name
john
Email
doe@gmail.com
Password
····
Confirm Password
····

- The web application's layout integrates essential components, including a footer, navigation bar, and sidebar, each designed to enhance functionality and user experience. The footer typically provides supplementary information, while the navigation bar ensures seamless access to different sections of the app. Additionally, the sidebar and other design elements contribute to the overall usability and aesthetic appeal of the interface.



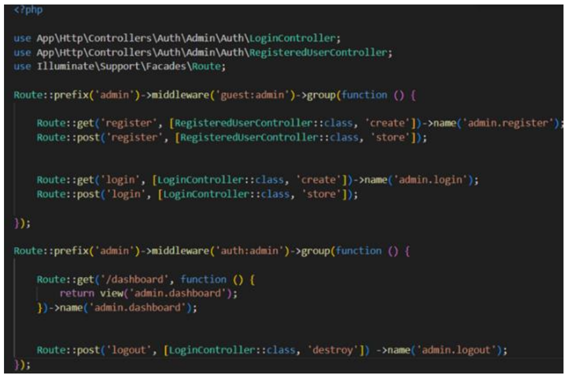
```
layouts
  partials
    footer.blade.php
    navbar.blade.php
    sidebar.blade.php
  admin.blade.php
  app.blade.php
  guest.blade.php
  navigation.blade.php
```

- A dedicated setup was created for admin routes to keep administrative functions separate from user routes. This separation improves security and simplifies application management. The admin routes include features like managing user data, monitoring system activity, and handling other key tasks.
- An admin/login page was implemented to allow users to choose between logging in as a regular user or an admin. This interface clearly separates the two roles, granting admins access to advanced features while restricting regular users to appropriate functionalities.

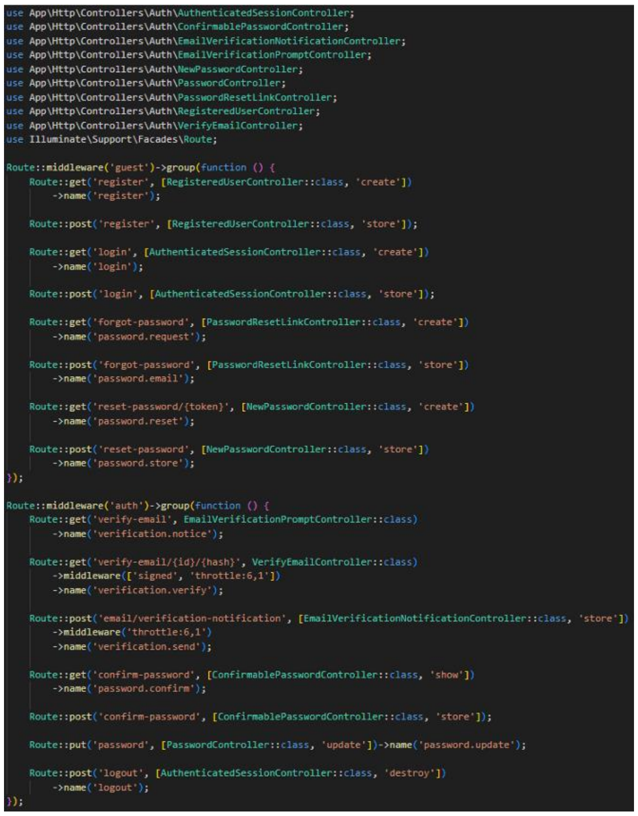


```
routes
  admin-auth.php
  auth.php
  console.php
  web.php
```

- The system provides CRUD functionality for user registration, allowing users to create, view, update, and delete their accounts. It also includes a log-out feature for secure and convenient account management.



```php
<?php

use App\Http\Controllers\Auth\Admin\Auth\LoginController;
use App\Http\Controllers\Auth\Admin\Auth\RegisteredUserController;
use Illuminate\Support\Facades\Route;

Route::prefix('admin')->middleware('guest:admin')->group(function () {

    Route::get('register', [RegisteredUserController::class, 'create'])->name('admin.register');
    Route::post('register', [RegisteredUserController::class, 'store']);


    Route::get('login', [LoginController::class, 'create'])->name('admin.login');
    Route::post('login', [LoginController::class, 'store']);

});

Route::prefix('admin')->middleware('auth:admin')->group(function () {

    Route::get('/dashboard', function () {
        return view('admin.dashboard');
    })->name('admin.dashboard');


    Route::post('logout', [LoginController::class, 'destroy']) ->name('admin.logout');
});
```



```php
use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;

Route::middleware('guest')->group(function () {
    Route::get('register', [RegisteredUserController::class, 'create'])
        ->name('register');

    Route::post('register', [RegisteredUserController::class, 'store']);

    Route::get('login', [AuthenticatedSessionController::class, 'create'])
        ->name('login');

    Route::post('login', [AuthenticatedSessionController::class, 'store']);

    Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])
        ->name('password.request');

    Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])
        ->name('password.email');

    Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])
        ->name('password.reset');

    Route::post('reset-password', [NewPasswordController::class, 'store'])
        ->name('password.store');
});

Route::middleware('auth')->group(function () {
    Route::get('verify-email', EmailVerificationPromptController::class)
        ->name('verification.notice');

    Route::get('verify-email/{id}/{hash}', VerifyEmailController::class)
        ->middleware(['signed', 'throttle:6,1'])
        ->name('verification.verify');

    Route::post('email/verification-notification', [EmailVerificationNotificationController::class, 'store'])
        ->middleware('throttle:6,1')
        ->name('verification.send');

    Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])
        ->name('password.confirm');

    Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);

    Route::put('password', [PasswordController::class, 'update'])->name('password.update');

    Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
        ->name('logout');
});
```

- We also explored connecting the database to understand its functionality. Both registered users and admins will be stored in the database, along with key data such as budget and expense trackers. This setup will allow us to manage and track user-related information efficiently.

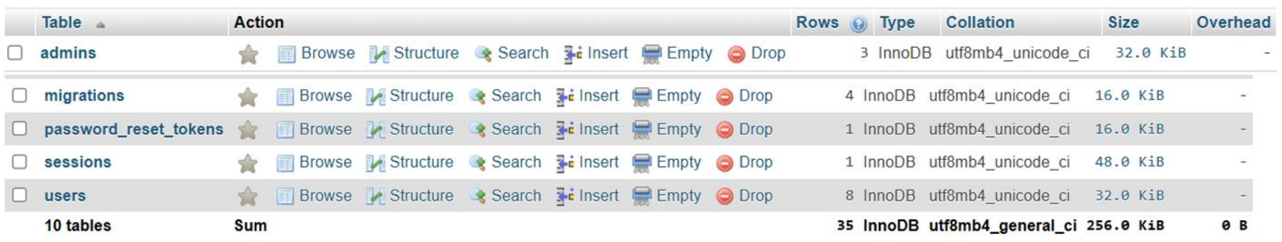


Table	Action	Rows	Type	Collation	Size	Overhead
admins	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
migrations	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
password_reset_tokens	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
sessions	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
users	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
10 tables	Sum	35	InnoDB	utf8mb4_general_ci	256.0 KiB	0 B