

Quintanilla, Lord Zaro Fiber A.
BS IT – 3 C
Web Dev

This is the controller responsible for populating the dashboard side to track on how many months does a user want to track his/her expenses

```
blade.php M | dashboard.blade.php | populatingpagesController X | admin-auth.php M
ttp > Controllers > populatingpagesController > ...
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

0 references | 0 implementations
class populatingpagesController extends Controller
{
    0 references | 0 overrides
    public function index(Request $request): View
    {
        $startMonth = $request->input(key: 'start_month');
        $endMonth = $request->input(key: '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(value: $startMonth) || $startMonth < 1 || $startMonth > 12) {
                $startMonth = 1;
            }

            if (!is_numeric(value: $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(view: 'dashboard', data: compact(var_name: 'posts', var_names: 'startMonth', 'endMonth'));
    }
}
```

This is the generated output

The screenshot shows a web dashboard with a 'MONTH TRACKER' section. At the top, there's a 'Start Month' dropdown menu set to '3', an 'End Month' dropdown menu set to '5', and a 'Submit' button. Below these are three boxes representing the months March, April, and May. Each box contains the month name, the current month (e.g., 'Month: March'), and a 'Continue' button.

A controller that is registered in the routes and this is responsible for the editing of profile for the users.

```
4 references | 0 implementations
class ProfileController extends Controller
{
    /**
     * Display the user's profile form.
     */
    1 reference | 0 overrides
    public function edit(Request $request): View
    {
        return view(view: 'profile.edit', data: [
            'user' => $request->user(),
        ]);
    }

    /**
     * Update the user's profile information.
     */
    1 reference | 0 overrides
    public function update(ProfileUpdateRequest $request): RedirectResponse
    {
        $request->user()->fill($request->validated());

        if ($request->user()->isDirty('email')) {
            $request->user()->email_verified_at = null;
        }

        $request->user()->save();

        return Redirect::route(route: 'profile.edit')->with(key: 'status', value: 'profile-updated');
    }

    /**
     * Delete the user's account.
     */
    1 reference | 0 overrides
    public function destroy(Request $request): RedirectResponse
    {
        $request->validateWithBag(errorBag: 'userDeletion', rules: [
            'password' => ['required', 'current_password'],
        ]);

        $user = $request->user();

        Auth::logout();

        $user->delete();

        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return Redirect::to(path: '/');
    }
}
```


This is routes and this is the profile controller registered in it

```
login.blade.php M dashboard.blade.php web.php X admin-auth.php M
routes > web.php
1 <?php
2
3 use App\Http\Controllers\ProfileController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get(uri: '/', action: function (): View {
7     return view(view: 'welcome');
8 });
9
10 Route::get(uri: '/dashboard', action: function (): View {
11     return view(view: 'dashboard');
12 }->middleware(middleware: ['auth', 'verified'])->name(name: 'dashboard');
13
14 Route::middleware(middleware: 'auth')->group(callback: function (): void {
15     Route::get(uri: '/profile', action: [ProfileController::class, 'edit'])->name(name: 'profile.edit');
16     Route::patch(uri: '/profile', action: [ProfileController::class, 'update'])->name(name: 'profile.update');
17     Route::delete(uri: '/profile', action: [ProfileController::class, 'destroy'])->name(name: 'profile.destroy');
18 });
19
20 require __DIR__.'/auth.php';
21 require __DIR__.'/admin-auth.php';
```

this controller is responsible for the creation of the account

```
login.blade.php M dashboard.blade.php RegisteredUserController.php X admin-auth.php M
app > Http > Controllers > Auth > RegisteredUserController.php > RegisteredUserController > store()
3 namespace App\Http\Controllers\Auth;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\User;
7 use Illuminate\Auth\Events\Registered;
8 use Illuminate\Http\RedirectResponse;
9 use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\Support\Facades\Hash;
12 use Illuminate\Validation\Rules;
13 use Illuminate\View\View;
14
15 class RegisteredUserController extends Controller
16 {
17     /**
18      * Display the registration view.
19      */
20     public function create(): View
21     {
22         return view(view: 'auth.register');
23     }
24
25     /**
26      * Handle an incoming registration request.
27      *
28      * @throws \Illuminate\Validation\ValidationException
29      */
30     public function store(Request $request): RedirectResponse
31     {
32         $request->validate(rules: [
33             'name' => ['required', 'string', 'max:255'],
34             'email' => ['required', 'string', 'lowercase', 'email', 'max:255', 'unique:'.User::class],
35             'password' => ['required', 'confirmed', Rules\Password::defaults()],
36         ]);
37
38         $user = User::create(attributes: [
39             'name' => $request->name,
40             'email' => $request->email,
41             'password' => Hash::make(value: $request->password),
42         ]);
43
44         event(args: new Registered(user: $user));
45
46         Auth::login(user: $user);
47
48         return redirect(to: route(name: 'dashboard', absolute: false));
49     }
50 }
51
```

And here it is seen in the log in page.



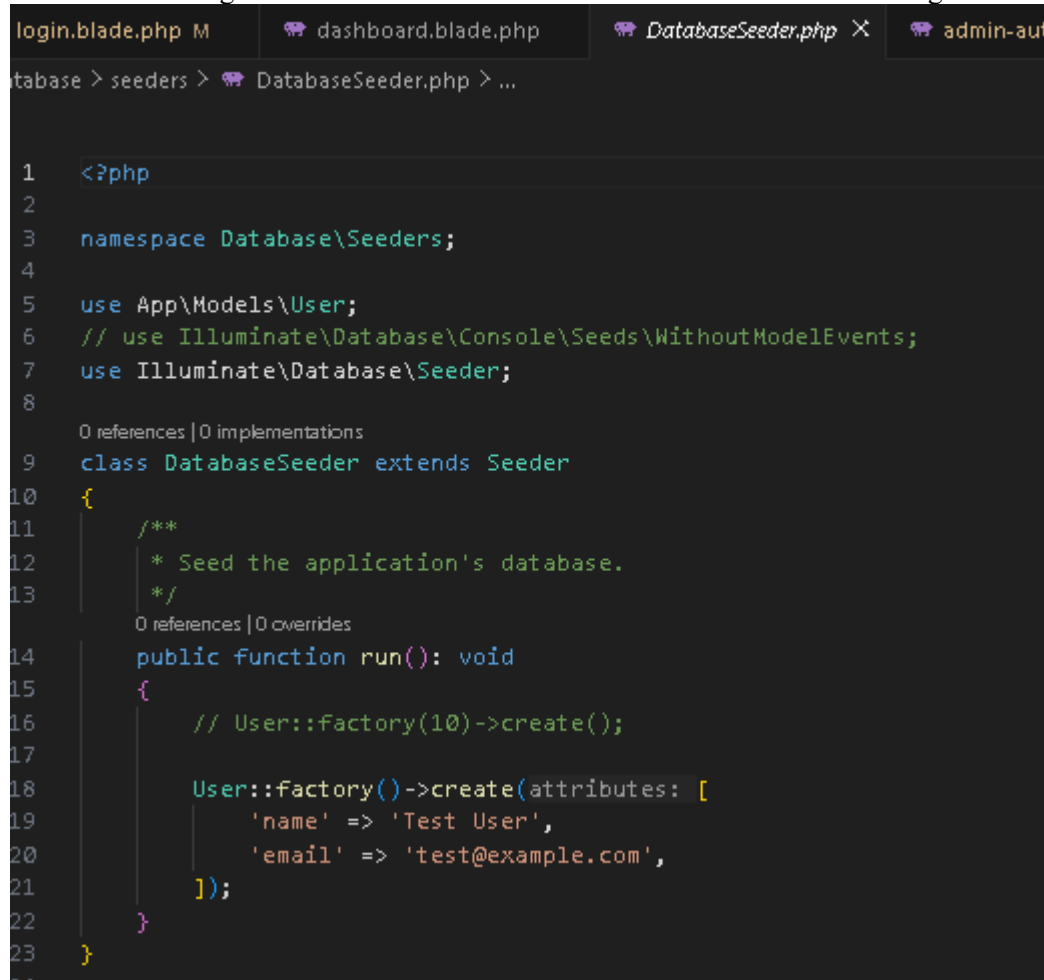
Email

Password

☐ Remember me

[Forgot your password?](#)

Here is the seeding and its runs and makes a tester email that can be used to log in as a user.



```
login.blade.php M dashboard.blade.php DatabaseSeeder.php X admin-aut
atabase > seeders > DatabaseSeeder.php > ...

1  <?php
2
3  namespace Database\Seeders;
4
5  use App\Models\User;
6  // use Illuminate\Database\Console\Seeds\WithoutModelEvents;
7  use Illuminate\Database\Seeder;
8
9  class DatabaseSeeder extends Seeder
10 {
11     /**
12      * Seed the application's database.
13      */
14     public function run(): void
15     {
16         // User::factory(10)->create();
17
18         User::factory()->create(attributes: [
19             'name' => 'Test User',
20             'email' => 'test@example.com',
21         ]);
22     }
23 }
```

```
n.blade.php M dashboard.blade.php Admin.php X admin-auth
Models > Admin.php > ...
<?php
The use directive is unnecessary. PHP(PHP7101)
Quick Fix... (Ctrl+)

use Illuminate\Database\Eloquent\Model;
use Illuminate\Notifications\Notifiable;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
4 references | 0 implementations
class Admin extends Authenticatable
{
    use HasFactory, Notifiable;

    0 references
    protected $guard = 'admin';

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    0 references
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    0 references
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    0 references | 0 overrides
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
```

Here is a model for the admin and this can perform the simple functions like updating and deletion of users.