Lab Activity

Creating a simple application with Laravel

Step 0: Prerequisites

You already have Node.js, PHP, and Composer installed. You may install the Laravel installer via Composer:

// Laravel installer
Run Command:

```
> composer global require laravel/installer
```

Step 1: Laravel Installation

// Environment

```
> cd htdocs
```

// Create new project
Run Command:

```
> laravel new your-project-name
```

// Follow through

```
 Which starter kit would you like to install? [None]:
  [none    ] None
  [react   ] React
  [vue     ] Vue
  [livewire] Livewire
 > None

 Which testing framework do you prefer? [Pest]:
  [0] Pest
  [1] PHPUnit
 > Pest

 Which database will your application use? [MySQL]:
  [mysql  ] MySQL
  [mariadb] MariaDB
  [sqlite ] SQLite (Missing PDO extension)
  [pgsql  ] PostgreSQL (Missing PDO extension)
  [sqlsrv ] SQL Server (Missing PDO extension)
 > MySQL

 Default database updated. Would you like to run the default database migrations? (yes/no) [yes]:
 > no

 Would you like to run npm install and npm run build? (yes/no) [yes]:
 > Yes
```

Step 2: .env Configurations

Open your project using VSCode

Edit .env (set your database & Email configurations)

```
APP_NAME=Your_project

[
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dbname
DB_USERNAME=root
DB_PASSWORD=dbpass
]

[
MAIL_MAILER=smtp
MAIL_SCHEME=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=your_auth_email@gmail.com
MAIL_PASSWORD="your_auth_pass"
MAIL_FROM_ADDRESS="noreply@domain.com"
MAIL_FROM_NAME="${APP_NAME}"
]
```

Step 3: Authentication with Breeze

Open the Terminal

// Install breeze
Run Command:

```
> composer require laravel/breeze --dev
> php artisan breeze:install
```

Which Breeze stack would you like to install?

```
> blade
```

Would you like dark mode support

```
> yes
```

Which testing framework do you prefer? [Pest]:
[0] Pest
[1] PHPUnit

```
> 0
```

Run Command:

```
> php artisan migrate
> php artisan serve
```

Step 4: Migrations & Seeders

Create the roles table

- roleId is the primary key
- roleName is a unique key

// Create roles migrations
Run Command:

```
> php artisan make:migration create_roles_table
```

Edit new roles migration (in >> database >> migrations)

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
   public function up(): void
    {
        Schema::create('roles', function (Blueprint $table) {
             $table->unsignedTinyInteger('roleId')->autoIncrement();
             $table->string('roleName', 100)->unique();
             $table->timestamp('created_at')->useCurrent();
             $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('roles');
    }
};
```

Run Command:

```
> php artisan migrate
```

Roles seeders may include admin, editor, viewer, student, instructor, guest, moderator

Run Command

```
> php artisan make:seeder RolesTableSeeder
```

Edit new Seeder (in >> database >> seeders)

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class RolesTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //
        $roles = [
             ['roleName' => 'Admin'],
             ['roleName' => 'Editor'],
             ['roleName' => 'Viewer'],
             ['roleName' => 'Student'],
             ['roleName' => 'Instructor'],
             ['roleName' => 'Guest'],
             ['roleName' => 'Moderator'],
        ];
        foreach ($roles as $role) {
             \DB::table('roles')->insert($role);
        }
    }
}
```

Run Command

```
> php artisan db:seed --class=RolesTableSeeder
```

In the users table,

- Make the password field nullable
- Add the roleId foreign key

Run Command

```
> php artisan make:migration add_roleId_constraint_to_users_table
```

Edit new migration (in >> database >> migrations)

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {
            // Add the roleId column as nullable first
            $table->unsignedTinyInteger('roleId')->default(4)->after('remember_token')->nullable(false);
        });

        // Update existing users to have roleId = 1
        \DB::table('users')->update(['roleId' => 1]);

        Schema::table('users', function (Blueprint $table) {
            // make password not nullable
            $table->string('password')->nullable(true)->change();
            // Then, add the foreign key constraint
            $table->foreign('roleId')->references('roleId')->on('roles')->onDelete('cascade')->onUpdate('cascade');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {
            // Drop the foreign key constraint first
            $table->dropForeign(['roleId']);
            // Then drop the roleId column
            $table->dropColumn('roleId');
        });
    }
};
```

Run Command:

```
> php artisan migrate
```

Step 5: Create Models

Run Command

```
> php artisan make:model SpotRoles
```

Edit new Model (in >> app >> Models)

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class SpotRoles extends Model
{
    // Model to select roles from roles table
    protected $table = 'roles';
    protected $primaryKey = 'roleId';
    public $timestamps = true;
    protected $fillable = [
        'roleName',
    ];
}
```

Run Command

> php artisan make:model AddUser

Edit new Model (in >> app >> Models)

```
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class AddUser extends Model
{
    // Model for adding a user
    protected $table = 'users';
    protected $primaryKey = 'id';
    public $timestamps = true;
    protected $fillable = [
        'name',
        'email',
        'roleId',
    ];
}
```

Create a Blade File Form to Add a user

Form fields may include full name, email address, and role (dropdown)

Step 6: Create Views & Routes

Go to resources >> views >> components
Create file *select-input.blade.php*

```
@props(['disabled' => false])

@php
$selectClasses = 'border-gray-300 dark:border-gray-700 dark:bg-gray-900 dark:text-gray-300 ' .
    'focus:border-indigo-500 dark:focus:border-indigo-600 focus:ring-indigo-500 ' .
    'dark:focus:ring-indigo-600 rounded-md shadow-sm';
@endphp

<select :disabled="$disabled" {{ $attributes->merge(['class' => $selectClasses]) }}>
    <option value="" disabled selected>Select a Role</option>
    {{ $slot }}
</select>
```

Go to resources >> views >> profile >> partials
Create file *add-user-form.blade.php*

```
<section>
    <header>
        <h2 class="text-lg font-medium text-gray-900 dark:text-gray-100">
            {{ __('User Information') }}
        </h2>

        <p class="mt-1 text-sm text-gray-600 dark:text-gray-400">
            {{ __("Add new user's information and email address.") }}
        </p>
    </header>

    <form id="send-verification" method="post" action="{{ route('verification.send') }}">
        @csrf
    </form>

    <form method="post" action="{{ route('addUser.create') }}" class="mt-6 space-y-6">
        @csrf
          <div>
            <x-input-label for="name" :value="__('Name')" />
            <x-text-input id="name" name="name" type="text" class="mt-1 block w-full"  required autofocus autocomplete="name" />
            <x-input-error class="mt-2" :messages="$errors->get('name')" />
        </div>

        <div>
            <x-input-label for="email" :value="__('Email')" />
            <x-text-input id="email" name="email" type="email" class="mt-1 block w-full"  required autocomplete="username" />
            <x-input-error class="mt-2" :messages="$errors->get('email')" />
        </div>

        <div>
            <x-input-label for="roleId" :value="__('Role')" />
            <x-select-input id="roleId" name="roleId" class="mt-1 block w-full" required>
                @foreach(\App\Models\SpotRoles::all() as $role)
                    <option value="{{ $role->roleId }}">{{ $role->roleName }}</option>
                @endforeach
            </x-select-input>
            <x-input-error class="mt-2" :messages="$errors->get('roleId')" />
        </div>

        <div class="flex items-center gap-4">
            <x-primary-button>{{ __('Save') }}</x-primary-button>

            @if (session('status') === 'User added successfully!')
                <p
                    x-data="{ show: true }"
                    x-show="show"
                    x-transition
                    x-init="setTimeout(() => show = false, 2000)"
                    class="text-sm text-gray-600 dark:text-gray-400"
                >{{ __('Saved.') }}</p>
            @endif
        </div>
    </form>
</section>
```

Go to resources >> views
Create file *addUser.blade.php*

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">
            {{ __('Add User') }}
        </h2>
    </x-slot>

    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="p-4 sm:p-8 bg-white dark:bg-gray-800 shadow sm:rounded-lg">
                <div class="max-w-xl">
                    @include('profile.partials.add-user-form')
                </div>
            </div>
        </div>
    </div>
</x-app-layout>
```

Update Routes to include new views

Go to routes
Edit web.php

```
<?php

use App\Http\Controllers\AddUserController;
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});

Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

Route::get('/addUser', function () {
    return view('addUser');
})->middleware(['auth', 'verified'])->name('addUser');

Route::post('/addUser', [AddUserController::class, 'store'])->name('addUser.create');

Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

require __DIR__.'/auth.php';
```

Step 7: Creating Controllers

Run Command

```
> php artisan make:controller AddUserController
```

Edit new Controller (in >> app >> Http >> Controllers)

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

use Illuminate\Http\Request;

class AddUserController extends Controller
{
    // Show the form for creating a new user
    public function create()
    {
        return view('addUser');
    }

    // Controller for adding a user
    public function store(Request $request)
    {
        // Validate and create the user
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'roleId' => 'required|integer|exists:roles,roleId',
        ]);

        $user = new \App\Models\AddUser();
        $user->name = ucwords($validated['name']);
        $user->email = strtolower($validated['email']);
        $user->roleId = $validated['roleId'];
        $user->save();

        return redirect()->route('addUser')->with('status', 'User added successfully!');
    }
}
```

Insert links (pointing to the views) in the main navigation section

Edit *navigation.blade.php* (in >> resources >> views >> layouts)

```
        <!-- Navigation Links -->
        <div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
                <x-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">
                        {{ __('Dashboard') }}
                </x-nav-link>
        </div>
        <div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
                <x-nav-link :href="route('addUser')" :active="request()->routeIs('addUser')">
                        {{ __('Add User') }}
                </x-nav-link>
        </div>
```

Step 8: Create a Blade File to view all Users Stored in a table

References

- https://laravel.com/docs/12.x