Laravel Project Practice

1. Project goal (what you'll build)

A small "Contacts" app where a user can:

- See a list of contacts (HTML page)
- Create a contact (HTML form)
- Edit & delete contacts (HTML pages)
  Everything served with Blade templates (Laravel's HTML templating).

2. Prerequisites (quick)

- PHP 8.2+ (or the version supported by your chosen Laravel release)
- Composer (PHP package manager)
- A database (SQLite for simplicity, or MySQL/Postgres)
- Terminal / command line

Laravel versions and support change; the commands below match current Laravel docs. Laravel+1

3. Create a new Laravel project

Open terminal and run:

```
# Using Composer (recommended)
composer create-project laravel/laravel contacts-app

cd contacts-app
```

Or (if you installed the laravel installer):

```
laravel new contacts-app
cd contacts-app
```

You can start a dev server with:

```
php artisan serve
# open http://127.0.0.1:8000
```

(Development server documented in Laravel docs). Laravel+1

4. Configure the database (SQLite easiest for practice)

Edit .env:

```
DB_CONNECTION=sqlite
DB_DATABASE=/absolute/path/to/contacts-app/database/database.sqlite
```

Create the sqlite file:

```
touch database/database.sqlite
```

(If you prefer MySQL, set DB_HOST, DB_DATABASE, DB_USERNAME, DB_PASSWORD in .env.)

5. Make a migration & model for Contact

Create model + migration:

```
php artisan make:model Contact -m
```

Open the migration file in database/migrations/*create_contacts_table.php and set columns:

```
public function up()
{
   Schema::create('contacts', function (Blueprint $table) {
      $table->id();
      $table->string('name');
      $table->string('email')->unique();
      $table->string('phone')->nullable();
      $table->timestamps();
   });
}
```

Run migrations:

```
php artisan migrate
```

6. Create a Controller and resource routes

Make a resource controller:

```
php artisan make:controller ContactController --resource --model=Contact
```

Register routes in routes/web.php:

```
use App\Http\Controllers\ContactController;
```

```
Route::get('/',  function () {
   return redirect()->route('contacts.index');
});

Route::resource('contacts', ContactController::class);
```

Resource routes create the standard index/create/store/show/edit/update/destroy endpoints.

7. Implement simple controller methods (example)

Open app/Http/Controllers/ContactController.php and implement:

```
namespace App\Http\Controllers;

use App\Models\Contact;
use Illuminate\Http\Request;

class ContactController extends Controller
{
   public function index()
   {
      $contacts = Contact::orderBy('name')->get();
      return view('contacts.index', compact('contacts'));
   }

   public function create()
   {
      return view('contacts.create');
   }

   public function store(Request $request)
   {
      $data = $request->validate([
         'name'  => 'required|string|max:255',
         'email' => 'required|email|unique:contacts,email',
         'phone' => 'nullable|string|max:50',
      ]);

      Contact::create($data);

      return redirect()->route('contacts.index')->with('success', 'Contact added.');
   }

   public function edit(Contact $contact)
   {
      return view('contacts.edit', compact('contact'));
   }

   public function update(Request $request, Contact $contact)
   {
      $data = $request->validate([
```

```
            'name'  => 'required|string|max:255',
            'email' => 'required|email|unique:contacts,email,'.$contact->id,
            'phone' => 'nullable|string|max:50',
        ]);

        $contact->update($data);

        return redirect()->route('contacts.index')->with('success', 'Contact updated.');
    }

    public function destroy(Contact $contact)
    {
        $contact->delete();
        return redirect()->route('contacts.index')->with('success', 'Contact deleted.');
    }
}
```

Also ensure your `Contact` model (`app/Models/Contact.php`) allows mass assignment:

```
protected $fillable = ['name','email','phone'];
```

8. Create Blade HTML views (plain HTML structure)

Create folder `resources/views/contacts/` and add 3 files: `index.blade.php`, `create.blade.php`, `edit.blade.php`.

`resources/views/layouts/app.blade.php` (basic layout):

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="csrf-token" content="{{ csrf_token() }}">
  <title>Contacts App</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/water.css@2/out/water.css">
</head>
<body>
  <header>
    <h1>Contacts App</h1>
    <nav><a href="{{ route('contacts.index') }}">Home</a>  | <a href="{{ route('contacts.create') }}">Add
Contact</a></nav>
    <hr>
  </header>

  <main>
    @if(session('success'))
      <p style="color:green">{{ session('success') }}</p>
    @endif

    @yield('content')
  </main>
```

```
</body>
</html>
```

index.blade.php:

```
@extends('layouts.app')

@section('content')
 <h2>All Contacts</h2>

 @if($contacts->isEmpty())
  <p>No contacts found. <a href="{{ route('contacts.create') }}">Add one</a>.</p>
 @else
  <table>
   <thead><tr><th>Name</th><th>Email</th><th>Phone</th><th>Actions</th></tr></thead>
   <tbody>
    @foreach($contacts as $c)
     <tr>
      <td>{{ $c->name }}</td>
      <td>{{ $c->email }}</td>
      <td>{{ $c->phone }}</td>
      <td>
       <a href="{{ route('contacts.edit', $c) }}">Edit</a>
       <form action="{{ route('contacts.destroy', $c) }}" method="POST" style="display:inline">
        @csrf
        @method('DELETE')
        <button type="submit" onclick="return confirm('Delete this contact?')">Delete</button>
       </form>
      </td>
     </tr>
    @endforeach
   </tbody>
  </table>
 @endif
@endsection
```

create.blade.php and edit.blade.php are simple HTML forms (remember @csrf token):

create.blade.php:

```
@extends('layouts.app')

@section('content')
 <h2>Add Contact</h2>

 <form action="{{ route('contacts.store') }}" method="POST">
  @csrf
  <label>Name <input type="text" name="name" value="{{ old('name') }}"></label>
  @error('name')<p style="color:red">{{ $message }}</p>@enderror

  <label>Email <input type="email" name="email" value="{{ old('email') }}"></label>
```

```
    @error('email')<p style="color:red">{{ $message }}</p>@enderror

    <label>Phone <input type="text" name="phone" value="{{ old('phone') }}"></label>
    @error('phone')<p style="color:red">{{ $message }}</p>@enderror

    <button type="submit">Save</button>
  </form>
@endsection
```

edit.blade.php is similar but prefilled and uses method="POST" with @method('PUT'):

```
@extends('layouts.app')

@section('content')
  <h2>Edit Contact</h2>

  <form action="{{ route('contacts.update', $contact) }}" method="POST">
    @csrf
    @method('PUT')

    <label>Name <input type="text" name="name" value="{{ old('name', $contact->name) }}"></label>
    @error('name')<p style="color:red">{{ $message }}</p>@enderror

    <label>Email <input type="email" name="email" value="{{ old('email', $contact->email) }}"></label>
    @error('email')<p style="color:red">{{ $message }}</p>@enderror

    <label>Phone <input type="text" name="phone" value="{{ old('phone', $contact->phone) }}"></label>
    @error('phone')<p style="color:red">{{ $message }}</p>@enderror

    <button type="submit">Update</button>
  </form>
@endsection
```

Notes:

- Blade {{ }} auto-escapes output (XSS protection).
- Use @csrf in all POST forms (CSRF protection).

9. Test it

- Run php artisan serve
- Open http://127.0.0.1:8000
- Add / edit / delete contacts.

10. Optional: Add basic auth (Laravel Breeze) for practice

If you want user login/registration quickly (optional):

```
composer require laravel/breeze --dev
php artisan breeze:install blade
npm install
npm run dev
php artisan migrate
```

Breeze scaffolds login, register, password reset with minimal HTML. (Docs: Breeze starter kit). Laravel+1

11. Security & best practices (beginner checklist)

- Keep `.env` secret and out of VCS.
- Use `{{ }}` in views (auto-escaped).
- Validate inputs server-side (`$request->validate()`).
- Use `@csrf` in forms.
- For file uploads: validate MIME, size, store using `Storage` facade.
- For auth tokens / APIs: use Sanctum or Passport if needed. Laravel

12. Next exercises for learners

1. Add search/filter on contacts (route + controller query).
2. Add pagination (`->paginate(10)`).
3. Upload a profile picture for a contact (validate & store).
4. Add authentication & restrict contacts to the logged-in user (associate `user_id` with contacts).
5. Deploy to a free host (Fly.io, Render, or shared host) — learn environment config.

Quick reference of useful commands

```
composer create-project laravel/laravel contacts-app
php artisan make:model Contact -m
php artisan migrate
php artisan make:controller ContactController --resource --model=Contact
php artisan serve
composer require laravel/breeze --dev
php artisan breeze:install blade
npm install
npm run dev
```