Laravel Tutorial: Step by Step Guide to Building Your Laravel Application

The goal with this Laravel tutorial to create a guide for those just learning Laravel. This guide will take you from the very beginning of an idea into a real deployable application.

Prerequisites

- A local PHP environment (Valet, Homestead, Vagrant, MAMP, etc.).
- A database (I'll be using MySQL)
- PHPUnit installed.
- Node JS installed.

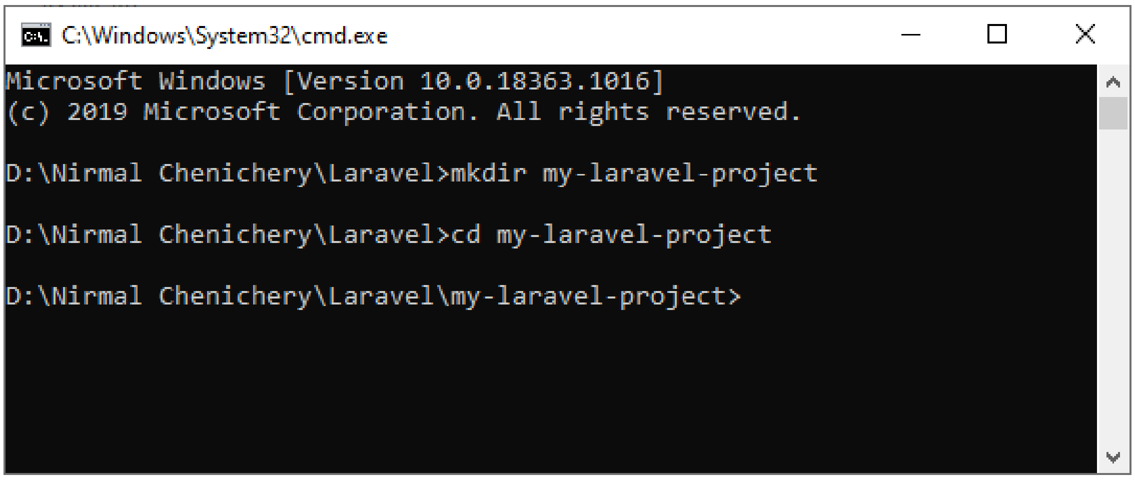
Let's get started!

1. The First Steps

It's time to create new empty project up and running. Create a directory and put all my projects in that directory

Open your terminal application and switch into this directory or you can create directory using GUI.

```
mkdir ~/Sites
cd ~/Sites
```



C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Nirmal Chenichery\Laravel>mkdir my-laravel-project

D:\Nirmal Chenichery\Laravel>cd my-laravel-project

D:\Nirmal Chenichery\Laravel\my-laravel-project>
```

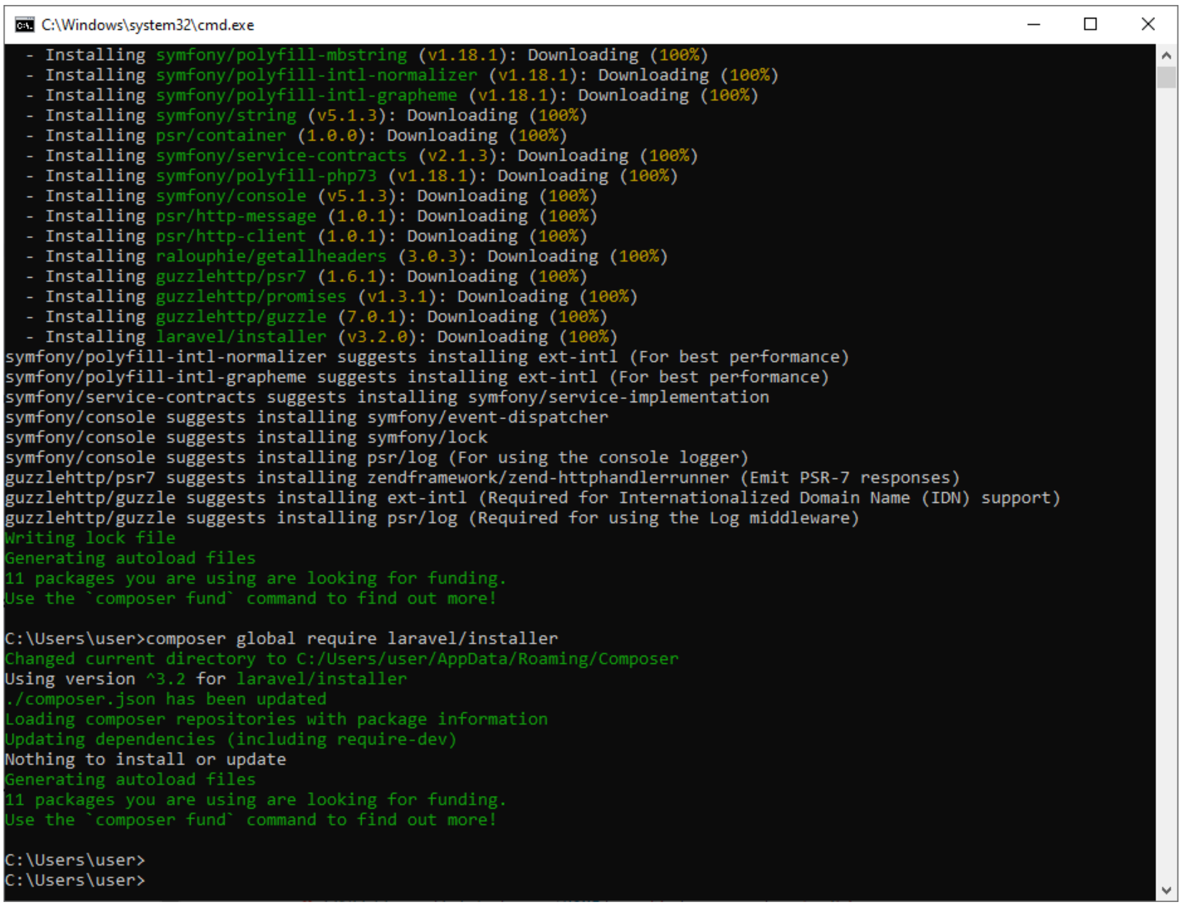
Installing Laravel

Laravel utilizes Composer to manage its dependencies. So, before using Laravel, make sure you have Composer installed on your machine.

Via Laravel Installer

First, download the Laravel installer using Composer:

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



```
C:\Windows\system32\cmd.exe
  - Installing symfony/polyfill-mbstring (v1.18.1): Downloading (100%)
  - Installing symfony/polyfill-intl-normalizer (v1.18.1): Downloading (100%)
  - Installing symfony/polyfill-intl-grapheme (v1.18.1): Downloading (100%)
  - Installing symfony/string (v5.1.3): Downloading (100%)
  - Installing psr/container (1.0.0): Downloading (100%)
  - Installing symfony/service-contracts (v2.1.3): Downloading (100%)
  - Installing symfony/polyfill-php73 (v1.18.1): Downloading (100%)
  - Installing symfony/console (v5.1.3): Downloading (100%)
  - Installing psr/http-message (1.0.1): Downloading (100%)
  - Installing psr/http-client (1.0.1): Downloading (100%)
  - Installing ralouphie/getallheaders (3.0.3): Downloading (100%)
  - Installing guzzlehttp/psr7 (1.6.1): Downloading (100%)
  - Installing guzzlehttp/promises (v1.3.1): Downloading (100%)
  - Installing guzzlehttp/guzzle (7.0.1): Downloading (100%)
  - Installing laravel/installer (v3.2.0): Downloading (100%)
symfony/polyfill-intl-normalizer suggests installing ext-intl (For best performance)
symfony/polyfill-intl-grapheme suggests installing ext-intl (For best performance)
symfony/service-contracts suggests installing symfony/service-implementation
symfony/console suggests installing symfony/event-dispatcher
symfony/console suggests installing symfony/lock
symfony/console suggests installing psr/log (For using the console logger)
guzzlehttp/psr7 suggests installing zendframework/zend-httphandlerrunner (Emit PSR-7 responses)
guzzlehttp/guzzle suggests installing ext-intl (Required for Internationalized Domain Name (IDN) support)
guzzlehttp/guzzle suggests installing psr/log (Required for using the Log middleware)
Writing lock file
Generating autoload files
11 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

C:\Users\user>composer global require laravel/installer
Changed current directory to C:/Users/user/AppData/Roaming/Composer
Using version ^3.2 for laravel/installer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Nothing to install or update
Generating autoload files
11 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

C:\Users\user>
C:\Users\user>
```

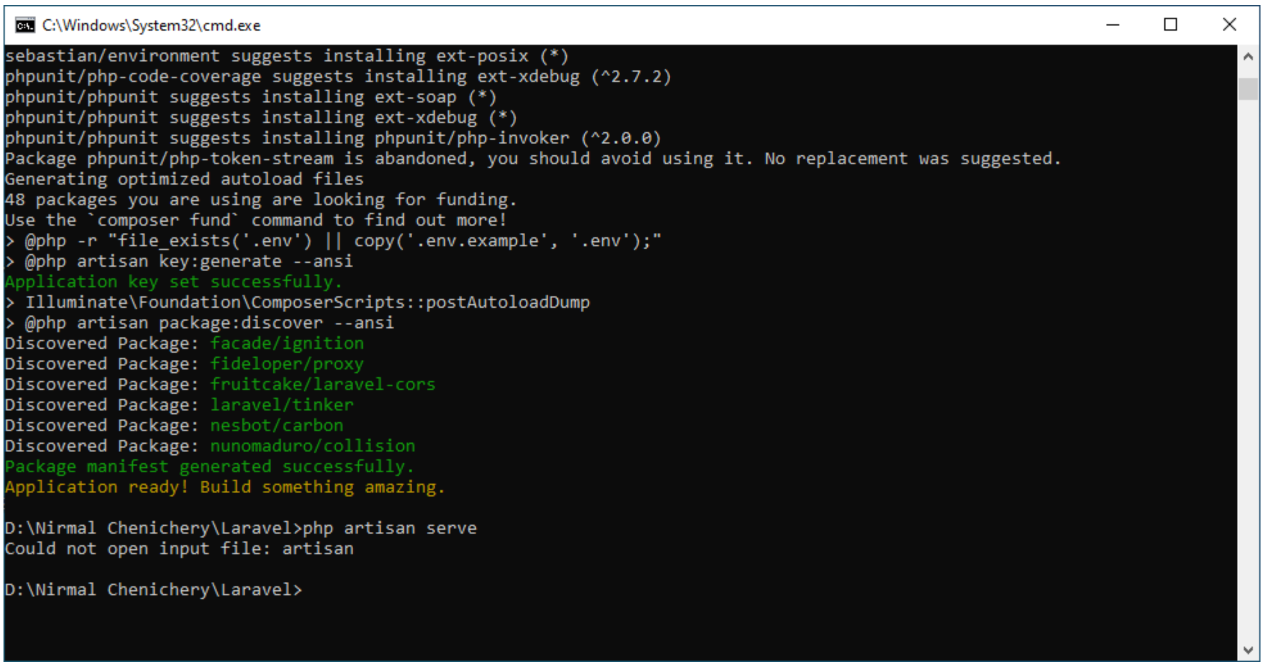
Make sure to place Composer's system-wide vendor bin directory in your **$PATH** so the laravel executable can be located by your system. This directory exists in different locations based on your operating system; however, some common locations include:

- macOS: `$HOME/.composer/vendor/bin`
- Windows: `%USERPROFILE%\AppData\Roaming\Composer\vendor\bin`
- GNU / Linux Distributions: `$HOME/.config/composer/vendor/bin` or `$HOME/.composer/vendor/bin`

You could also find the composer's global installation path by running **composer global about** and looking up from the first line.

Once installed, the laravel new command will create a fresh Laravel installation in the directory you specify. For instance, laravel new blog will create a directory named blog containing a fresh Laravel installation with all of Laravel's dependencies already installed:

```
laravel new blog
```

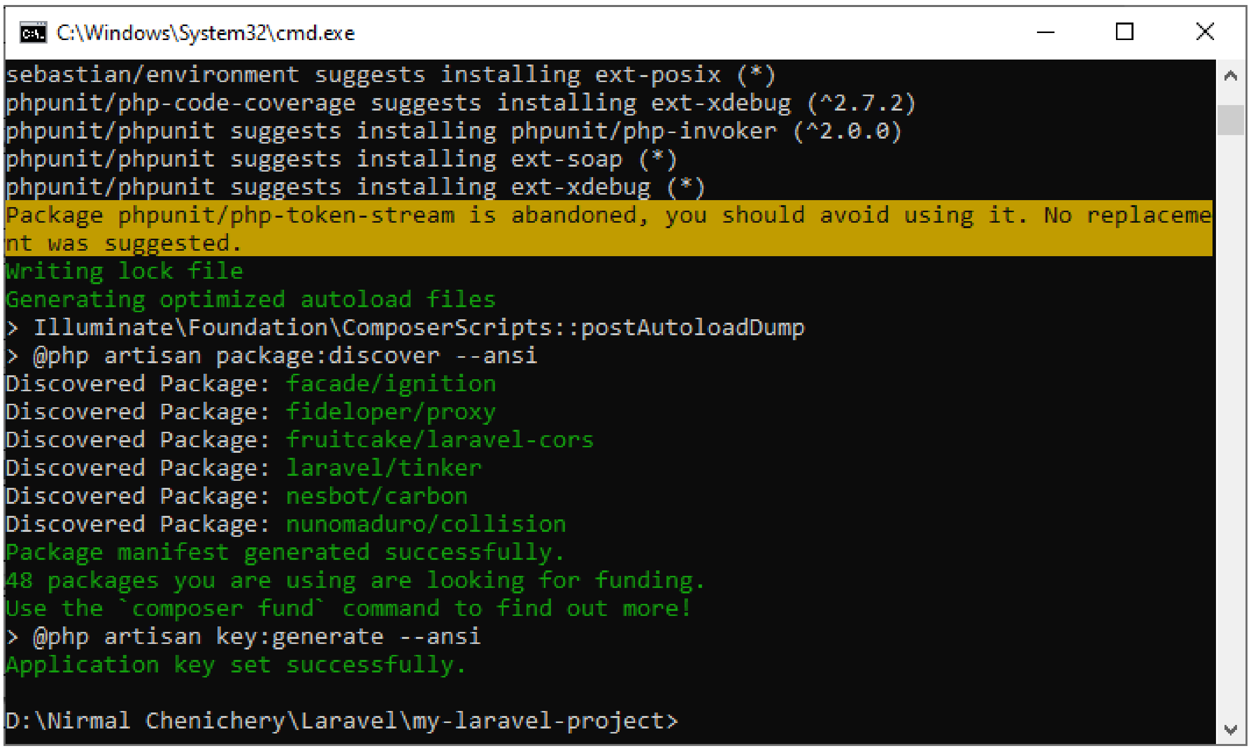


```
sebastian/environment suggests installing ext-posix (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.7.2)
phpunit/phpunit suggests installing ext-soap (*)
phpunit/phpunit suggests installing ext-xdebug (*)
phpunit/phpunit suggests installing phpunit/php-invoker (^2.0.0)
Package phpunit/php-token-stream is abandoned, you should avoid using it. No replacement was suggested.
Generating optimized autoload files
48 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
> @php artisan key:generate --ansi
Application key set successfully.
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
Application ready! Build something amazing.

D:\Nirmal Chenichery\Laravel>php artisan serve
Could not open input file: artisan

D:\Nirmal Chenichery\Laravel>
```

Via Composer Create-Project

Alternatively, you may also install Laravel by issuing the Composer create-project command in your terminal:

```
composer create-project --prefer-dist laravel/laravel blog
```

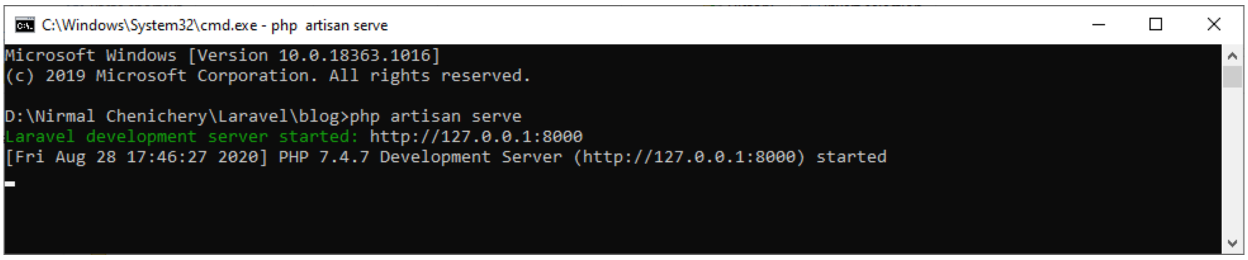


```
sebastian/environment suggests installing ext-posix (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.7.2)
phpunit/phpunit suggests installing phpunit/php-invoker (^2.0.0)
phpunit/phpunit suggests installing ext-soap (*)
phpunit/phpunit suggests installing ext-xdebug (*)
Package phpunit/php-token-stream is abandoned, you should avoid using it. No replaceme
nt was suggested.
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
48 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan key:generate --ansi
Application key set successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project>
```

Local Development Server

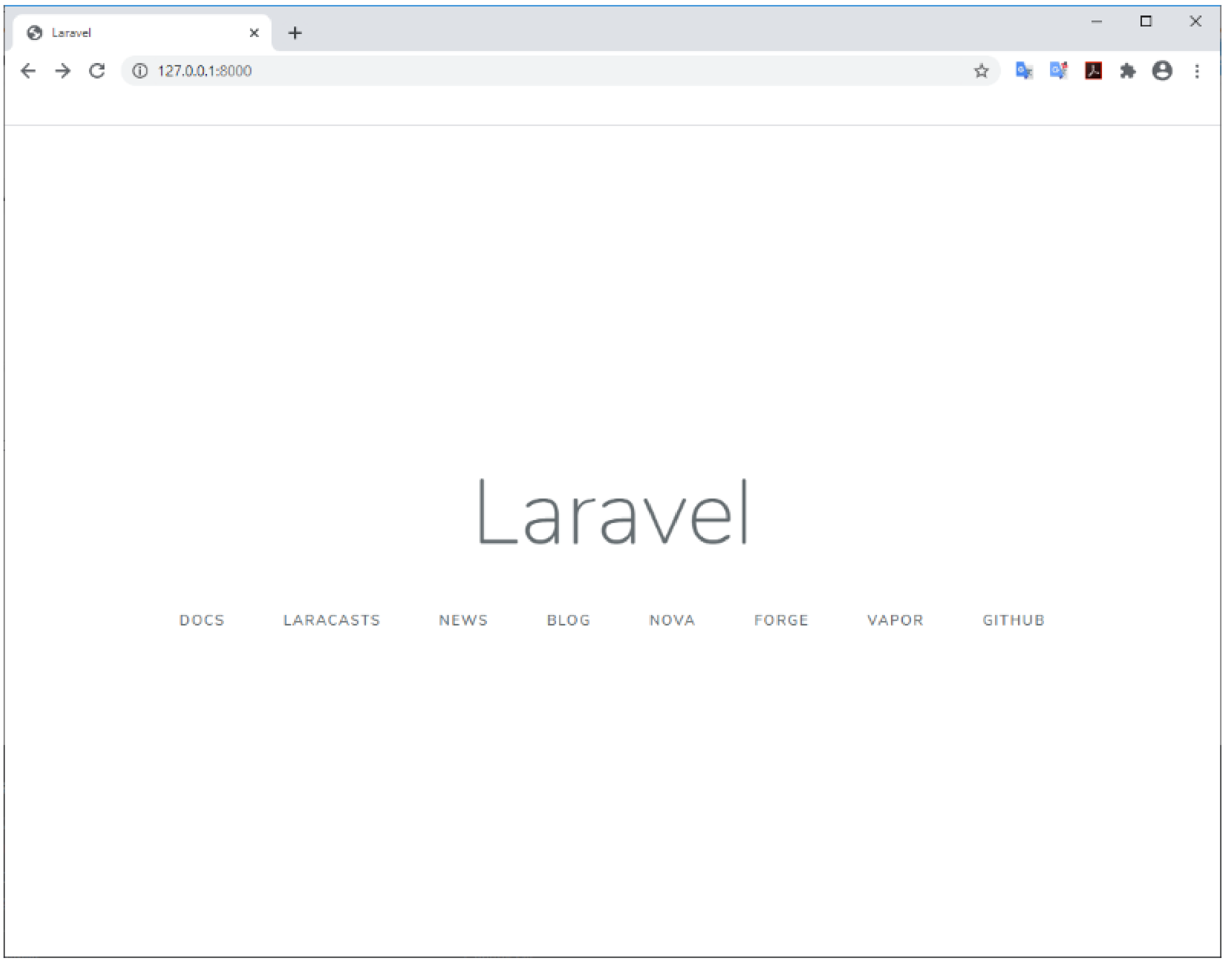
If you have PHP installed locally and you would like to use PHP's built-in development server to serve your application, you may use the serve Artisan command. This command will start a development server

```
php artisan serve
```



```
C:\Windows\System32\cmd.exe - php  artisan serve
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Nirmal Chenichery\Laravel\blog>php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Fri Aug 28 17:46:27 2020] PHP 7.4.7 Development Server (http://127.0.0.1:8000) started
```

Copy URL and Paste it in your browser



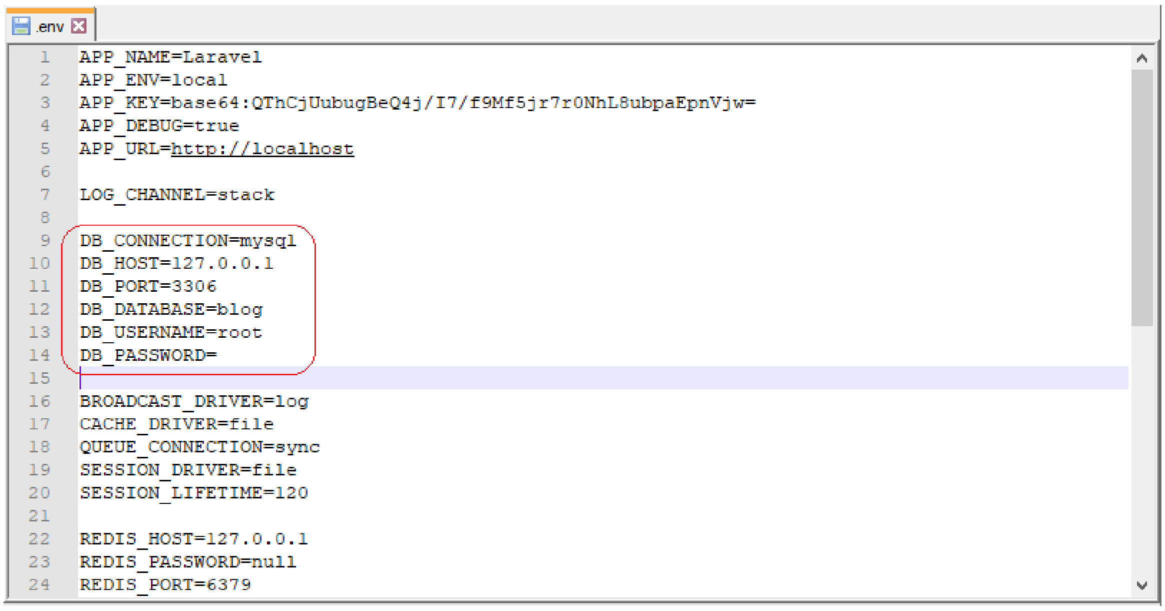
2. Database Setup

When you create a new Laravel project, the installation process automatically creates a .env file (copied from the .env.example file) for configuration and credentials.

- **Create new database**

```
Create database blog
```

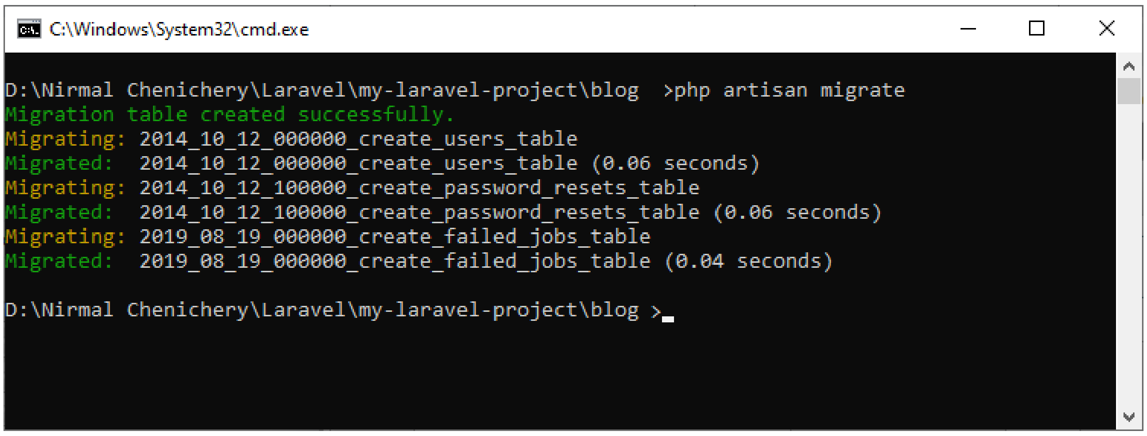
- You would then want to adjust the database configuration in .env: (check installed port)



```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:QThCjUubugBeQ4j/I7/f9Mf5jr7rONhL8ubpaEpnVjw=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=blog
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

- Test your database connection is running use migrate artisan command (optional step because , we will migrate later)

```
php artisan migrate
```

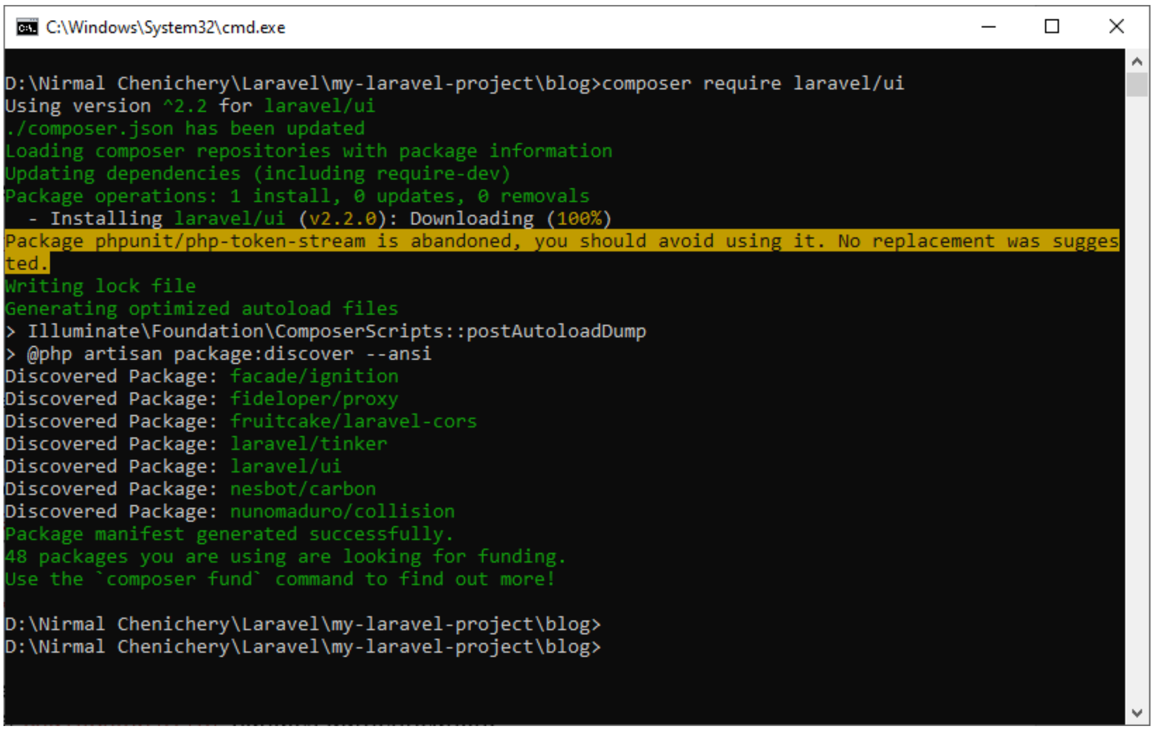


```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog  >php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.06 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (0.06 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.04 seconds)

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog >
```

3. Authentication

- Laravel's `laravel/ui` package provides a quick way to scaffold all of the routes and views you need for authentication using a few simple commands:

```
composer require laravel/ui
```



```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>composer require laravel/ui
Using version ^2.2 for laravel/ui
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing laravel/ui (v2.2.0): Downloading (100%)
Package phpunit/php-token-stream is abandoned, you should avoid using it. No replacement was sugges
ted.
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
48 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

This command should be used on fresh applications and will install a layout view, registration and login views, as well as routes for all authentication end-points. A `HomeController` will also be generated to handle post-login requests to your application's dashboard.

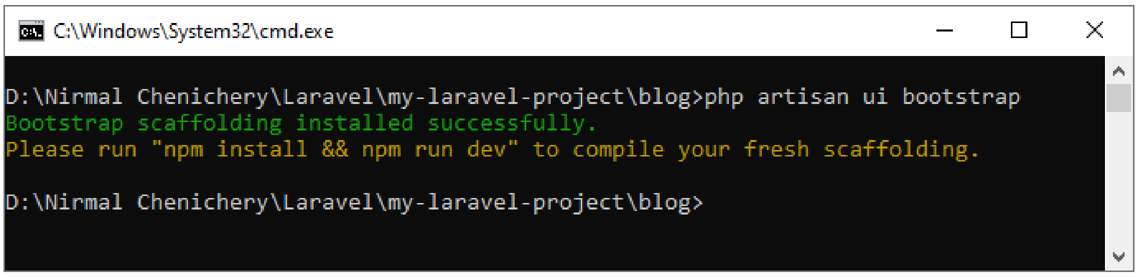
The `laravel/ui` package also generates several pre-built authentication controllers, which are located in the `App\Http\Controllers\Auth` namespace.

- The `RegisterController` handles new user registration.
- The `LoginController` handles authentication.
- The `ForgotPasswordController` handles e-mailing links for resetting passwords.
- The `ResetPasswordController` contains the logic to reset passwords.

Each of these controllers uses a trait to include their necessary methods. For many applications, you will not need to modify these controllers at all.

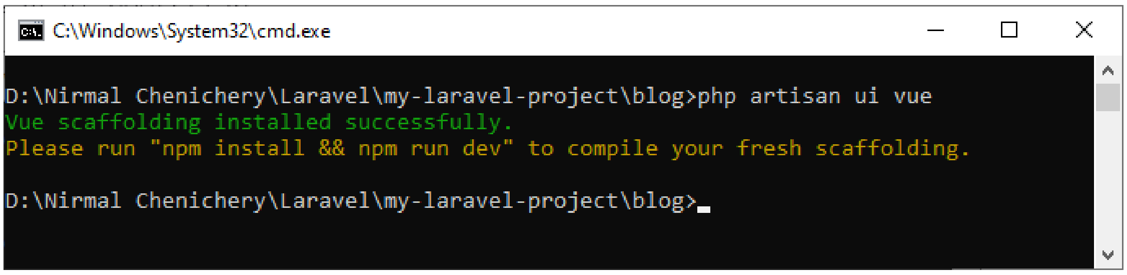
Once the `laravel/ui` package has been installed, you may install the frontend scaffolding using the `ui` Artisan command:

```
// Generate basic scaffolding...
php artisan ui bootstrap
```



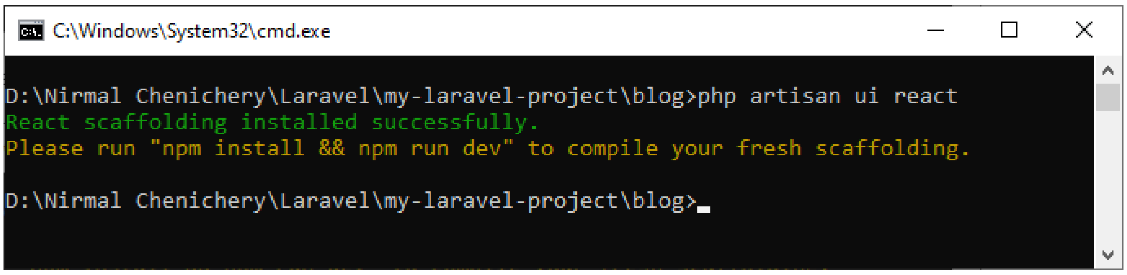
```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan ui bootstrap
Bootstrap scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

```
php artisan ui vue
```



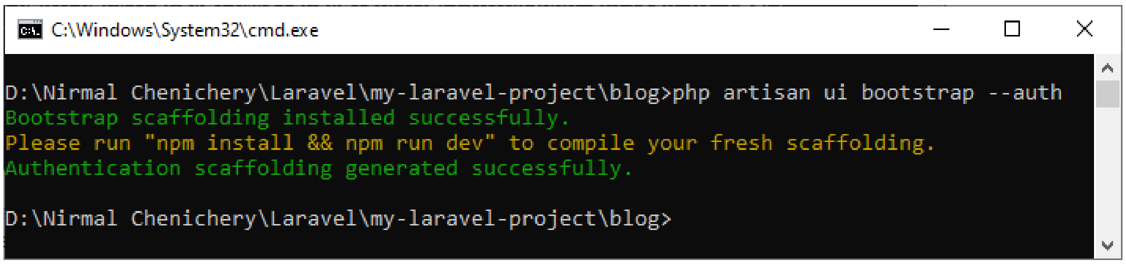
```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan ui vue
Vue scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

```
php artisan ui react
```



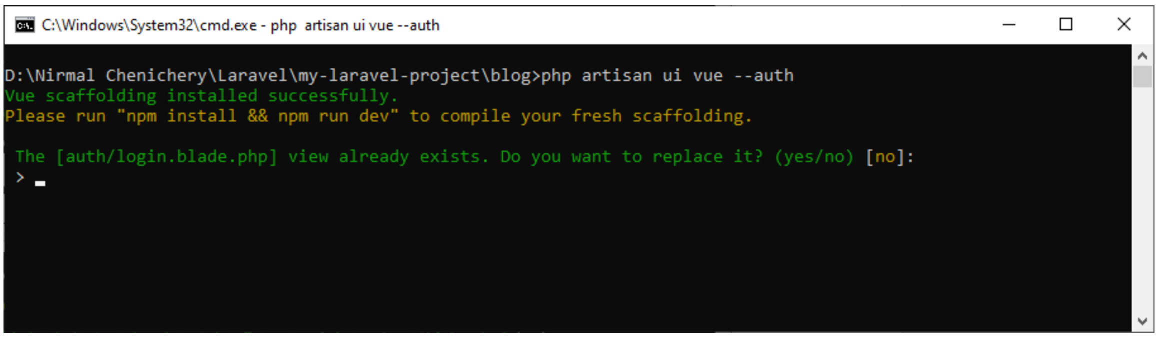
```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan ui react
React scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

```
// Generate login / registration scaffolding...
php artisan ui bootstrap --auth
```



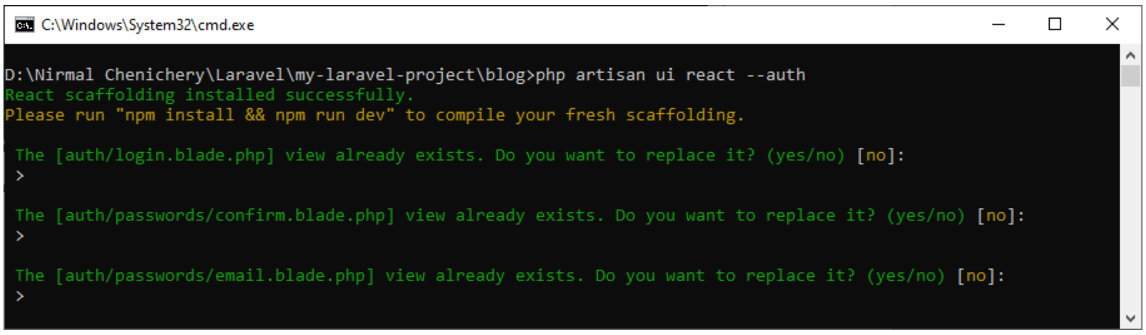
```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan ui bootstrap --auth
Bootstrap scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
Authentication scaffolding generated successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

```
php artisan ui vue --auth
```



```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan ui vue --auth
Vue scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.

 The [auth/login.blade.php] view already exists. Do you want to replace it? (yes/no) [no]:
 >
```

```
php artisan ui react --auth
```



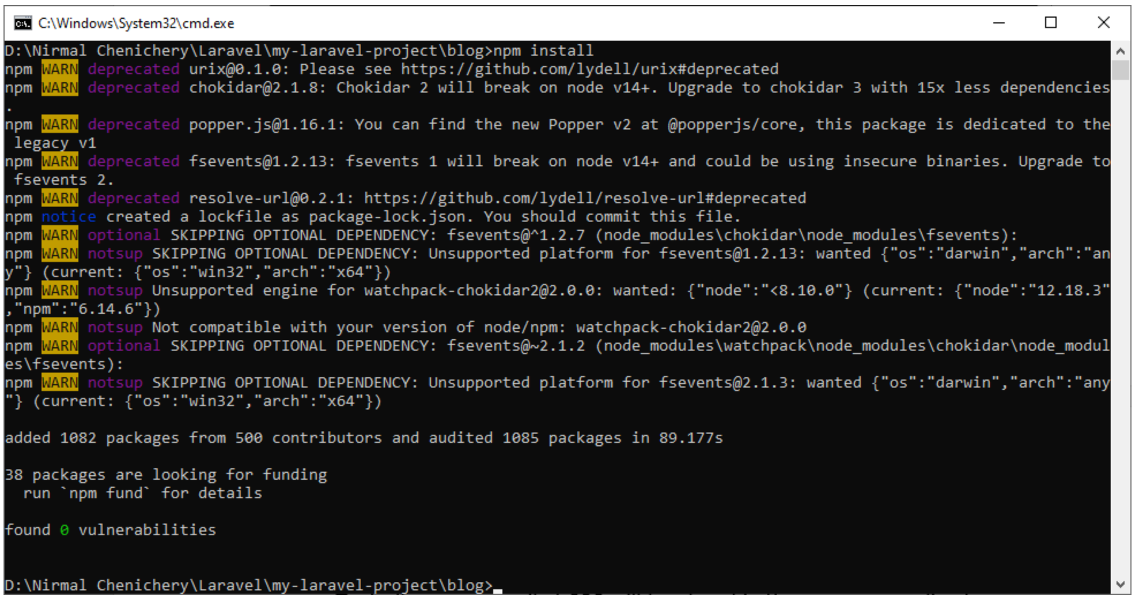
```
C:\Windows\System32\cmd.exe
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan ui react --auth
React scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.

 The [auth/login.blade.php] view already exists. Do you want to replace it? (yes/no) [no]:
 >

 The [auth/passwords/confirm.blade.php] view already exists. Do you want to replace it? (yes/no) [no]:
 >

 The [auth/passwords/email.blade.php] view already exists. Do you want to replace it? (yes/no) [no]:
 >
```

Compiling CSS

Before compiling your CSS, install your project's frontend dependencies using the Node package manager (NPM)

Download and install node.js it will include npm

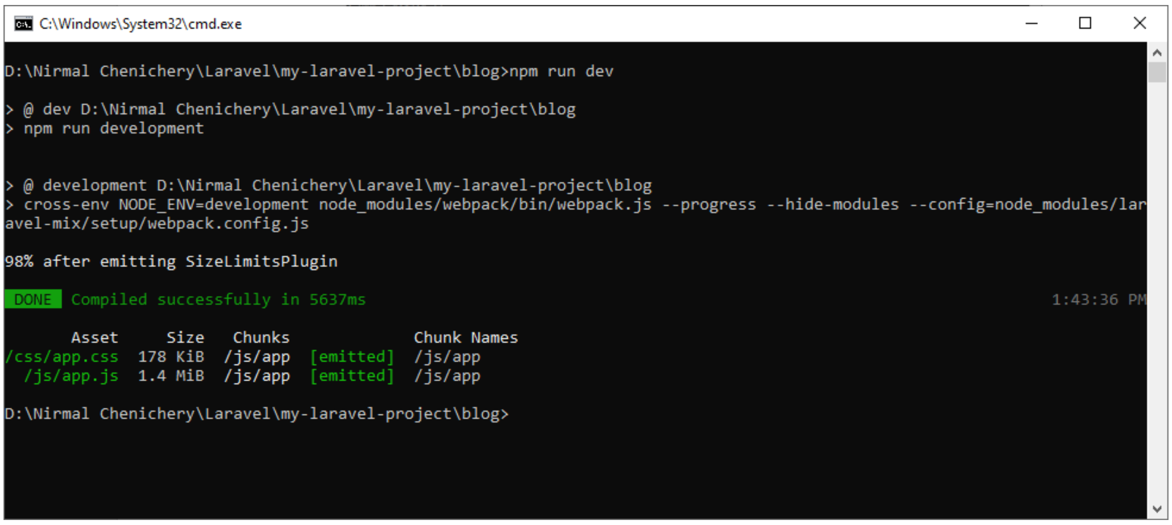
```
npm install
```



```
C:\Windows\System32\cmd.exe
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>npm install
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies
.
npm WARN deprecated popper.js@1.16.1: You can find the new Popper v2 at @popperjs/core, this package is dedicated to the
 legacy v1
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to
 fsevents 2.
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.2.7 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"an
y"} (current: {"os":"win32","arch":"x64"})
npm WARN notsup Unsupported engine for watchpack-chokidar2@2.0.0: wanted: {"node":"<8.10.0"} (current: {"node":"12.18.3"
,"npm":"6.14.6"})
npm WARN notsup Not compatible with your version of node/npm: watchpack-chokidar2@2.0.0
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.1.2 (node_modules\watchpack\node_modules\chokidar\node_modul
es\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os":"darwin","arch":"any
"} (current: {"os":"win32","arch":"x64"})

added 1082 packages from 500 contributors and audited 1085 packages in 89.177s

38 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities


D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>_
```

Once the dependencies have been installed using `npm install`, you can compile your SASS files to plain CSS using Laravel Mix. The `npm run dev` command will process the instructions in your `webpack.mix.js` file. Typically, your compiled CSS will be placed in the `public/css` directory:

```
npm run dev
```

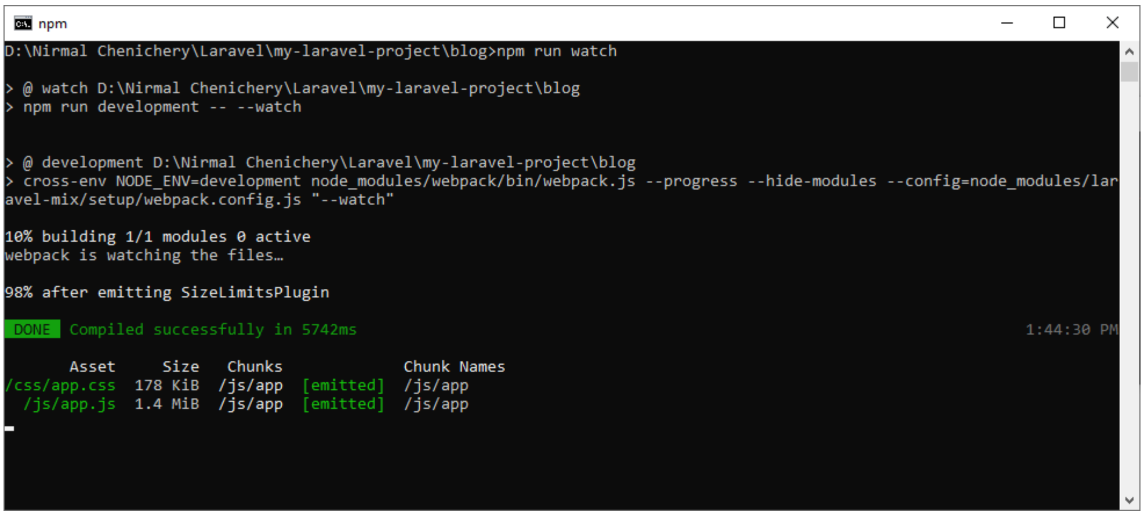


```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>npm run dev

> @ dev D:\Nirmal Chenichery\Laravel\my-laravel-project\blog
> npm run development


> @ development D:\Nirmal Chenichery\Laravel\my-laravel-project\blog
> cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/lar
avel-mix/setup/webpack.config.js

98% after emitting SizeLimitsPlugin

 DONE  Compiled successfully in 5637ms                                                                      1:43:36 PM

       Asset     Size   Chunks             Chunk Names
/css/app.css  178 KiB  /js/app  [emitted]  /js/app
  /js/app.js  1.4 MiB  /js/app  [emitted]  /js/app

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

The `webpack.mix.js` file included with Laravel's frontend scaffolding will compile the `resources/sass/app.scss` SASS file. This `app.scss` file imports a file of SASS variables and loads Bootstrap, which provides a good starting point for most applications. Feel free to customize the `app.scss` file however you wish or even use an entirely different pre-processor by configuring Laravel Mix.

```
npm run watch
```

The watch command will listen for files changes to JS and CSS files, and automatically update them. You probably want to have npm run watch running in a separate tab while developing.



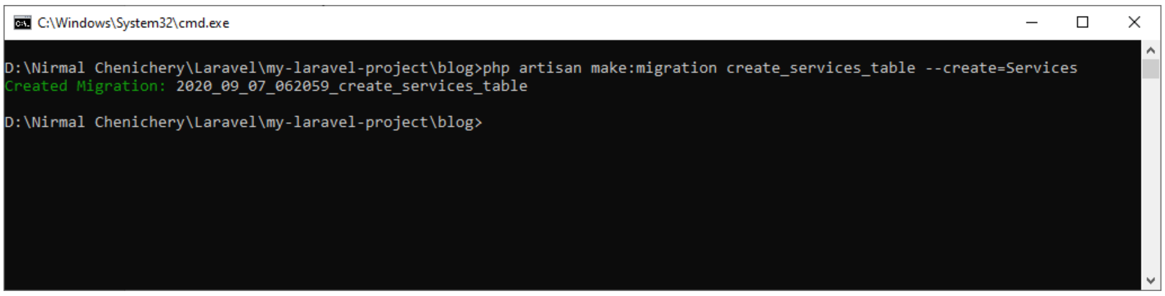
```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>npm run watch

> @ watch D:\Nirmal Chenichery\Laravel\my-laravel-project\blog
> npm run development -- --watch


> @ development D:\Nirmal Chenichery\Laravel\my-laravel-project\blog
> cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/lar
avel-mix/setup/webpack.config.js "--watch"

10% building 1/1 modules 0 active
webpack is watching the files…

98% after emitting SizeLimitsPlugin

 DONE  Compiled successfully in 5742ms                                                                      1:44:30 PM

       Asset     Size   Chunks             Chunk Names
/css/app.css  178 KiB  /js/app  [emitted]  /js/app
  /js/app.js  1.4 MiB  /js/app  [emitted]  /js/app
```

4. Building a List of Links

Even though showing a list of links sounds like a small task it still requires a database, a database table, data in the table, a database query, and a view file.

❖ Creating a migration will be the first step, and the Laravel Artisan command line tool can help us build that.

```
php artisan make:migration create_services_table --create=Services
```

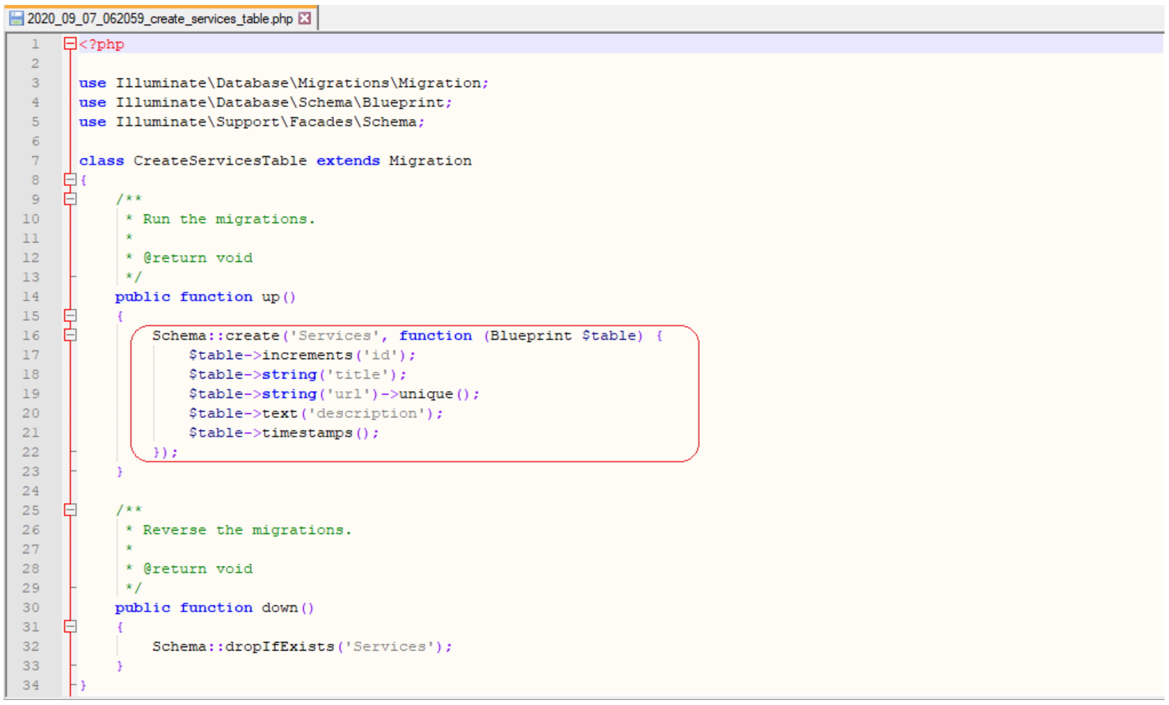


```
C:\Windows\System32\cmd.exe

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan make:migration create_services_table --create=Services
Created Migration: 2020_09_07_062059_create_services_table

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

Now, open the file this command created. It will be located at database/migrations/{{datetime}}_create_links_table.php. You'll notice a few other migrations in this folder as well, which the framework provides.

❖ Inside the "up()" method, add the following schema:

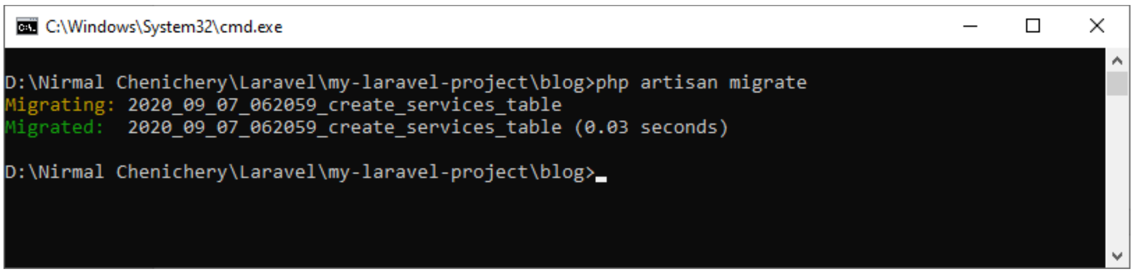
```
Schema::create('services', function (Blueprint $table) {
       $table->increments('id');
       $table->string('title');
       $table->string('url')->unique();
       $table->text('description');
       $table->timestamps();
})
```



```
2020_09_07_062059_create_services_table.php

<?php

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

class CreateServicesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('Services', function (Blueprint $table) {
            $table->increments('id');
            $table->string('title');
            $table->string('url')->unique();
            $table->text('description');
            $table->timestamps();
        });
    }

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

❖ Save the file and run the migration:

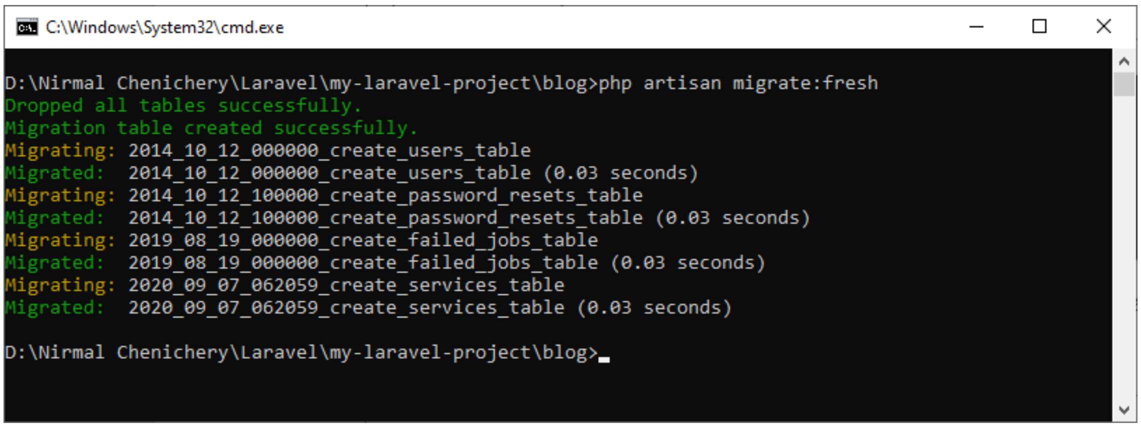
```
php artisan migrate
```



```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan migrate
Migrating: 2020_09_07_062059_create_services_table
Migrated:  2020_09_07_062059_create_services_table (0.03 seconds)

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>_
```

❖ While you are working with test data, you can quickly apply the schema:

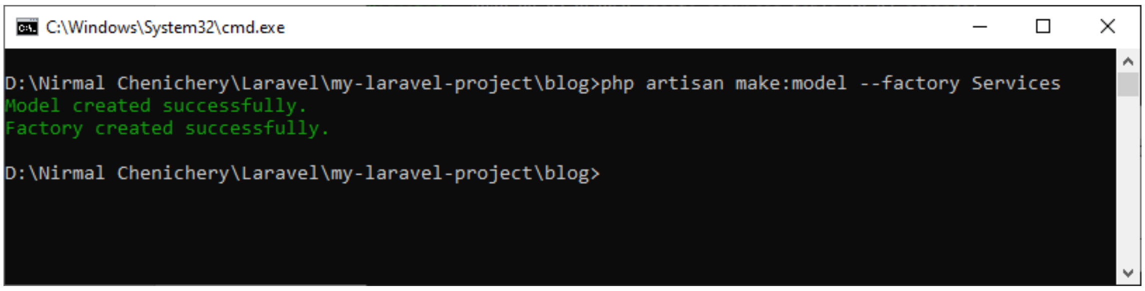
```
php artisan migrate:fresh
```



```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan migrate:fresh
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.03 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (0.03 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.03 seconds)
Migrating: 2020_09_07_062059_create_services_table
Migrated:  2020_09_07_062059_create_services_table (0.03 seconds)

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>_
```

❖ Next, we need some data and a model to work with our database table.

```
php artisan make:model --factory Services
```



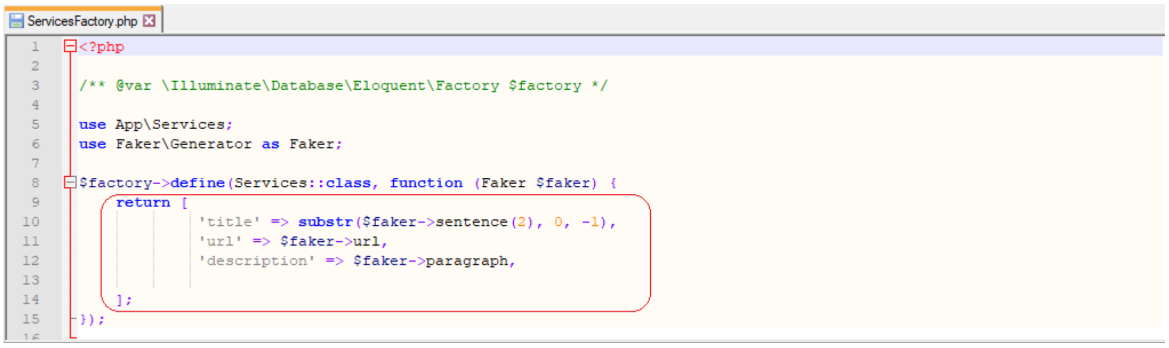
```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan make:model --factory Services
Model created successfully.
Factory created successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

The make:model command creates an app/Service.php model file. Laravel models provide a powerful database API called Eloquent.

The --factory flag will generate a new factory file in the database/factories path for generating app data. In our case, a new ServiceFactory file will include an empty factory definition for our Service model.

- Open the ServiceFactory.php file and fill in the following (database/factories).

```
$factory->define(Services::class, function (Faker $faker) {
    return [
                'title' => substr($faker->sentence(2), 0, -1),
                'url' => $faker->url,
                'description' => $faker->paragraph,
    ];
});
```

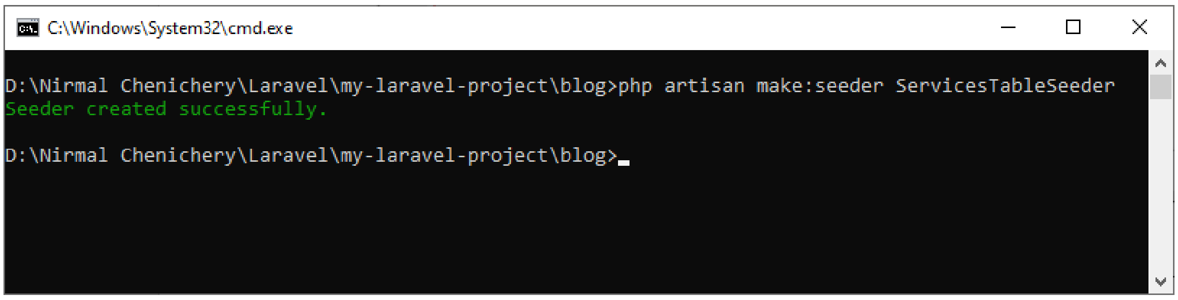


```
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\Services;
use Faker\Generator as Faker;

$factory->define(Services::class, function (Faker $faker) {
    return [
                'title' => substr($faker->sentence(2), 0, -1),
                'url' => $faker->url,
                'description' => $faker->paragraph,

    ];
});
```

We use the $faker->sentence() method to generate a title, and substr to remove the period at the end of the sentence.

- Next, create the service seeder, so we can easily add demo data to the table:

```
php artisan make:seeder ServicesTableSeeder
```

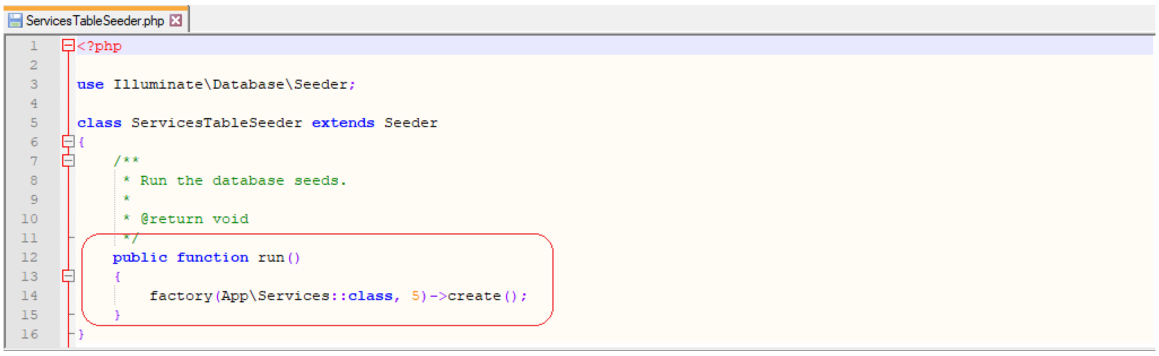


```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan make:seeder ServicesTableSeeder
Seeder created successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>_
```

The make:seeder command generates a new database class to seed our Services table with data.

- Open the database/seeds/ServiceTableSeeder.php file and add the following:

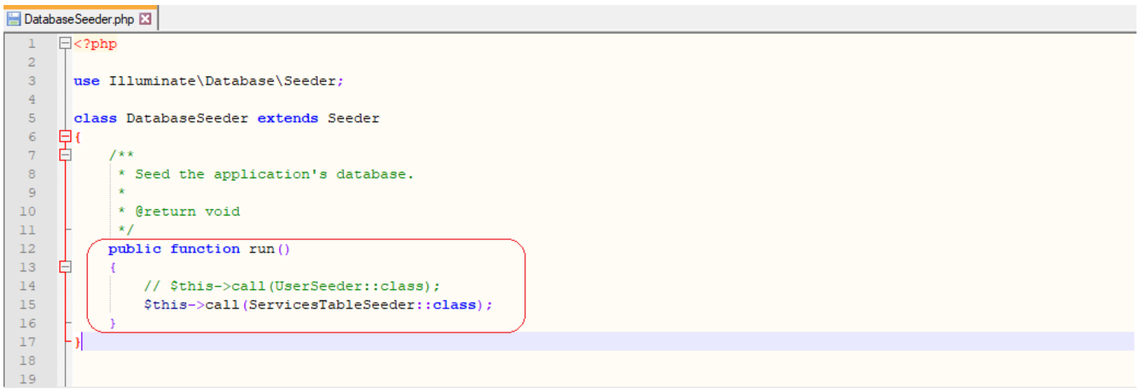
```
public function run()
{
  factory(App\Service::class, 5)->create();
}
```


ServicesTableSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class ServicesTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        factory(App\Services::class, 5)->create();
    }
}
```

❖ In order to “activate” the ServiceTableSeeder, we need to call it from the main database/seeds/DatabaseSeeder.php run method:

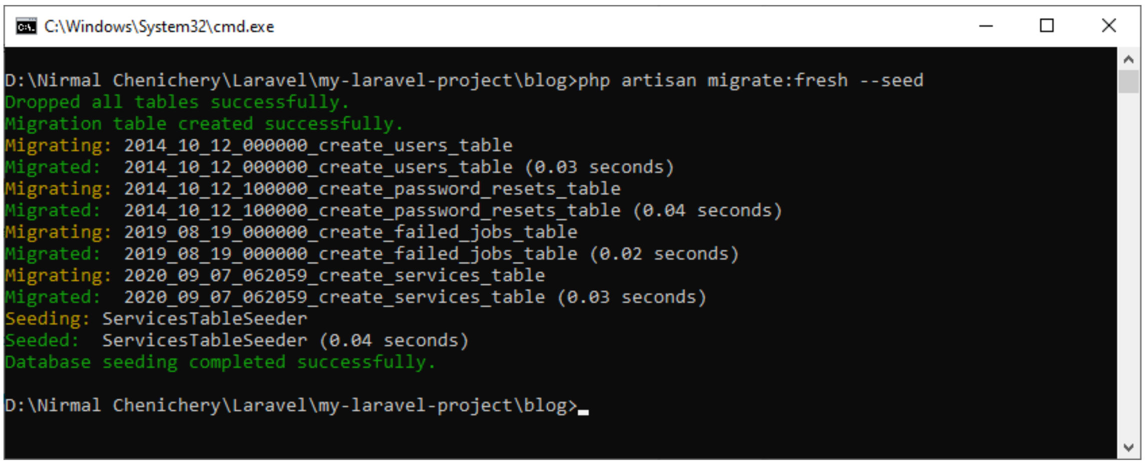
```php
public function run()
    {
        // $this->call(UserSeeder::class);
        $this->call(ServicesTableSeeder::class);
    }
```


DatabaseSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        // $this->call(UserSeeder::class);
        $this->call(ServicesTableSeeder::class);
    }
}
```

❖ You can now run the migrations and seeds to add data to the table automatically. Using the migrate:fresh command, we can get a clean schema that applies all migrations and then seeds the database:

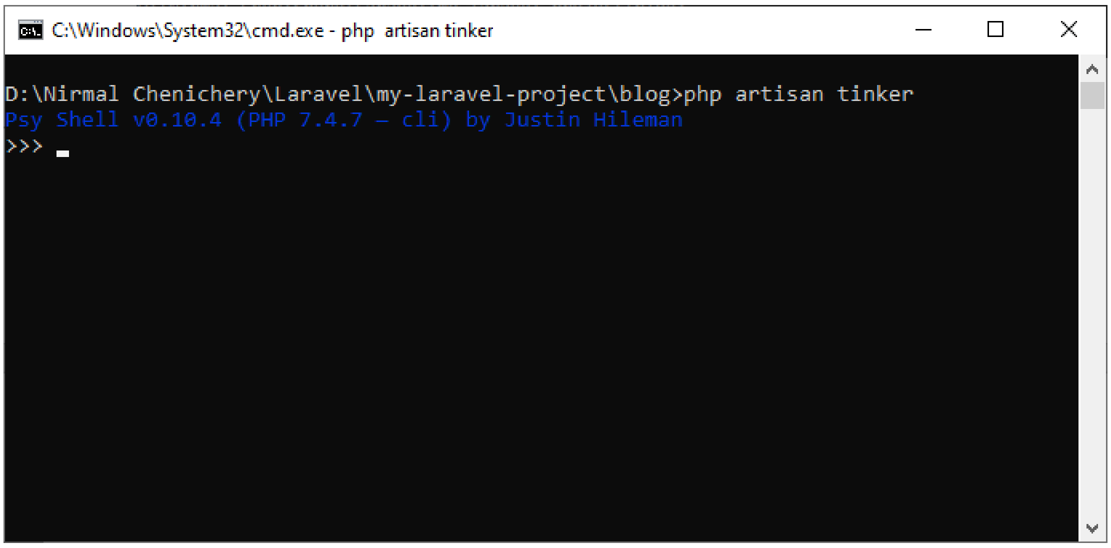
```
php artisan migrate:fresh --seed
```


C:\Windows\System32\cmd.exe

```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.03 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (0.04 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.02 seconds)
Migrating: 2020_09_07_062059_create_services_table
Migrated:  2020_09_07_062059_create_services_table (0.03 seconds)
Seeding: ServicesTableSeeder
Seeded:  ServicesTableSeeder (0.04 seconds)
Database seeding completed successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

- Using the **tinker shell** you can start playing around with the model data:

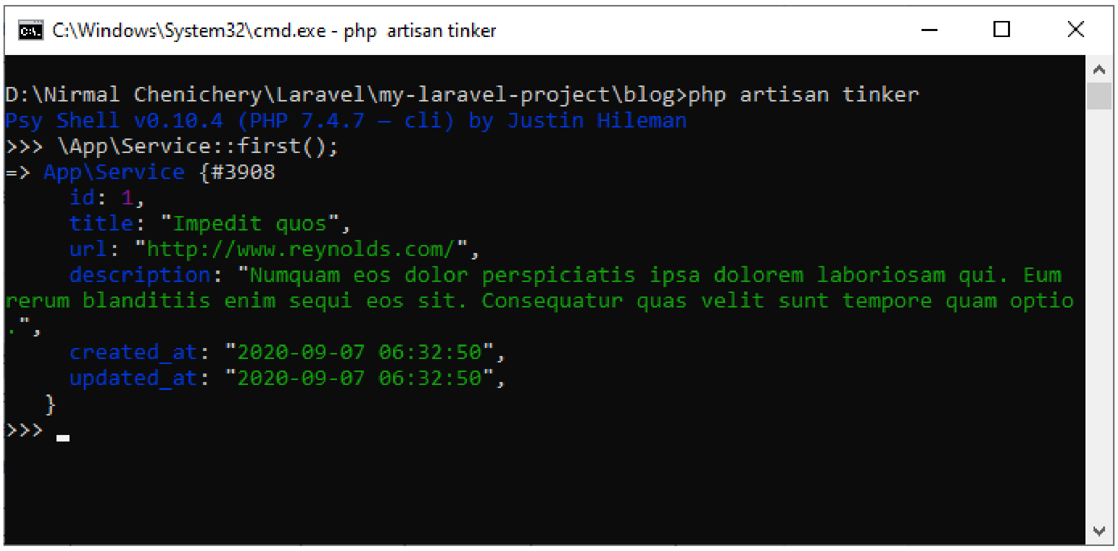
```
php artisan tinker
```



C:\Windows\System32\cmd.exe - php artisan tinker

```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan tinker
Psy Shell v0.10.4 (PHP 7.4.7 — cli) by Justin Hileman
>>> 
```

```
>>> \App\Service::first();
```



C:\Windows\System32\cmd.exe - php artisan tinker

```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan tinker
Psy Shell v0.10.4 (PHP 7.4.7 — cli) by Justin Hileman
>>> \App\Service::first();
=> App\Service {#3908
     id: 1,
     title: "Impedit quos",
     url: "http://www.reynolds.com/",
     description: "Numquam eos dolor perspiciatis ipsa dolorem laboriosam qui. Eum
rerum blanditiis enim sequi eos sit. Consequatur quas velit sunt tempore quam optio
.",
     created_at: "2020-09-07 06:32:50",
     updated_at: "2020-09-07 06:32:50",
   }
>>> 
```

We have the data place and a model to interact with the database! Let’s start building the UI to add new links to the application.

5. Routing and Views

To build out a view showing the list of services, we need to update the main project route and also define a new route that will display our submission form. We can add new routes to our application in the routes/web.php file.

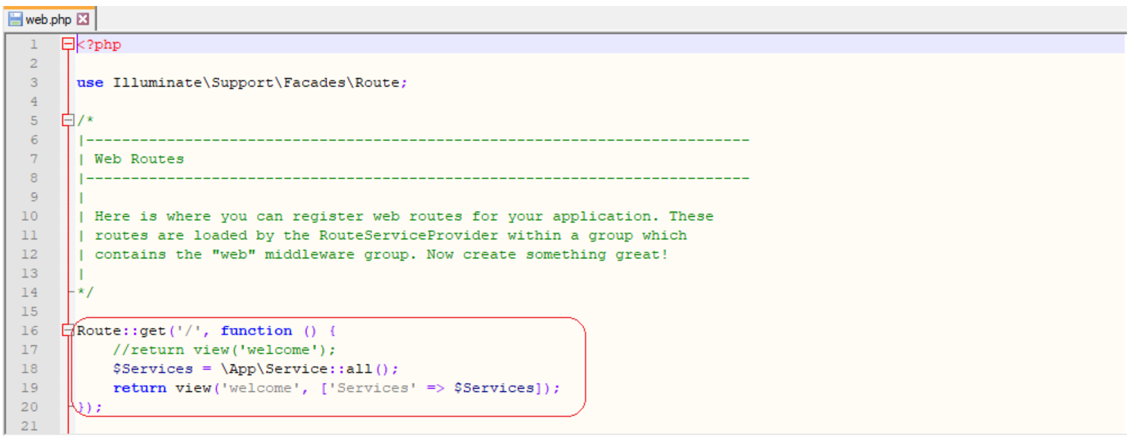
In the web routes file, you should see the default route below:

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

To create a new route, we can either use a route closure or a dedicated controller class. We will use closures for our submission and index routes.

- ❖ First, let’s update the home route by getting a collection of links from the database and passing them to the view:

```
Route::get('/', function () {
    //return view('welcome');
    $Services = \App\Service::all();
    return view('welcome', ['Services' => $Services]);
});
```

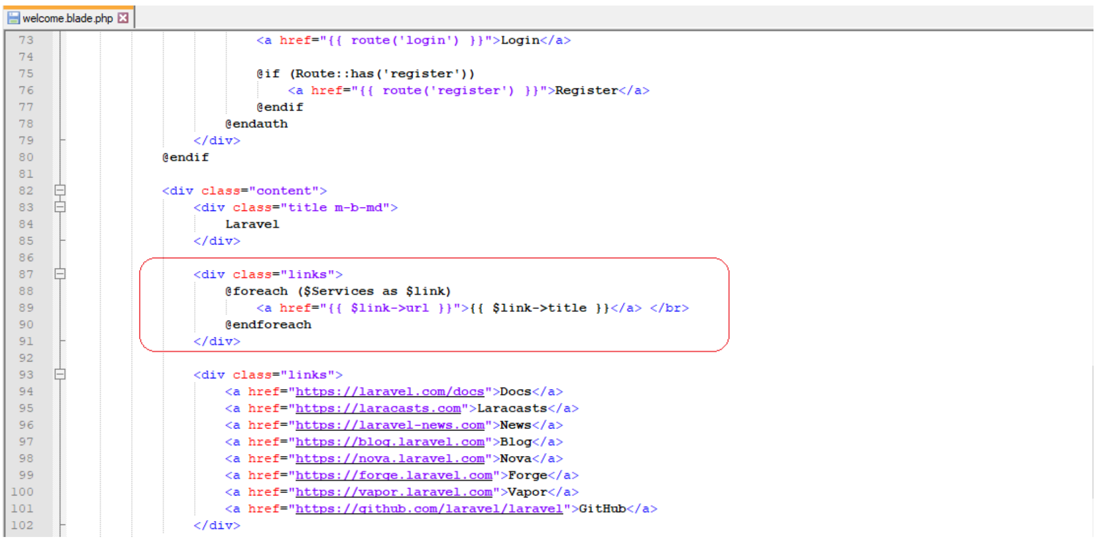


web.php

```
<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    //return view('welcome');
    $Services = \App\Service::all();
    return view('welcome', ['Services' => $Services]);
});
```

- ❖ Next, edit the welcome.blade.php(resources\views) file and add a simple foreach to show all the links:

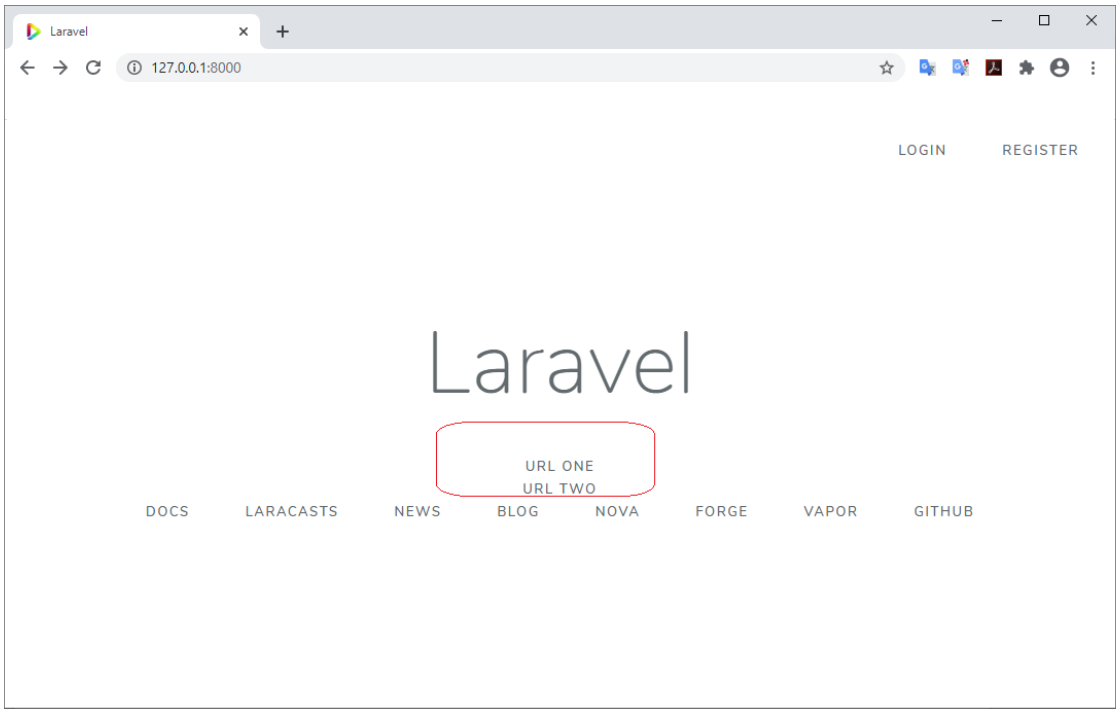
```
@foreach ($Services as $link)
  <a href="{{ $link->url }}">{{ $link->title }}</a>
@endforeach
```



welcome.blade.php

```
                <a href="{{ route('login') }}">Login</a>

                @if (Route::has('register'))
                    <a href="{{ route('register') }}">Register</a>
                @endif
            @endauth
        </div>
    @endif

    <div class="content">
        <div class="title m-b-md">
            Laravel
        </div>

        <div class="links">
            @foreach ($Services as $link)
                <a href="{{ $link->url }}">{{ $link->title }}</a> </br>
            @endforeach
        </div>

        <div class="links">
            <a href="https://laravel.com/docs">Docs</a>
            <a href="https://laracasts.com">Laracasts</a>
            <a href="https://laravel-news.com">News</a>
            <a href="https://blog.laravel.com">Blog</a>
            <a href="https://nova.laravel.com">Nova</a>
            <a href="https://forge.laravel.com">Forge</a>
            <a href="https://vapor.laravel.com">Vapor</a>
            <a href="https://github.com/laravel/laravel">GitHub</a>
        </div>
```

- If you refresh your browser, you should now see the list of all the links added. With that all set, let's move to submitting links.



Insert Update and Delete record from MySQL

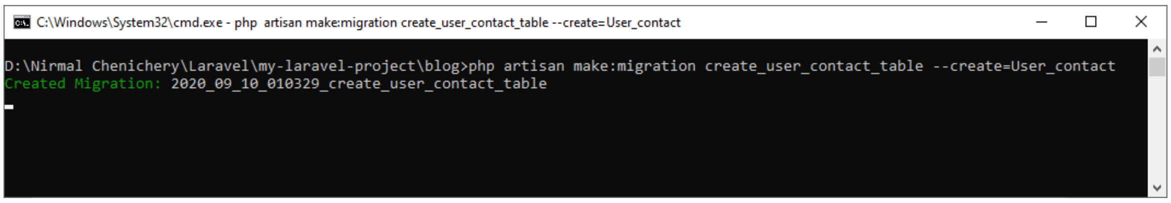
- **Creating database table using migration**

  Now after the database is setup, we will create a table using artisan make command, when we run make command, it will create a new migration file (/database/migrations/tablename.php)

  Table Structure

```
CREATE TABLE user_contacts (
  id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  name varchar(80) NOT NULL,
  address varchar(1024) NOT NULL
);
```

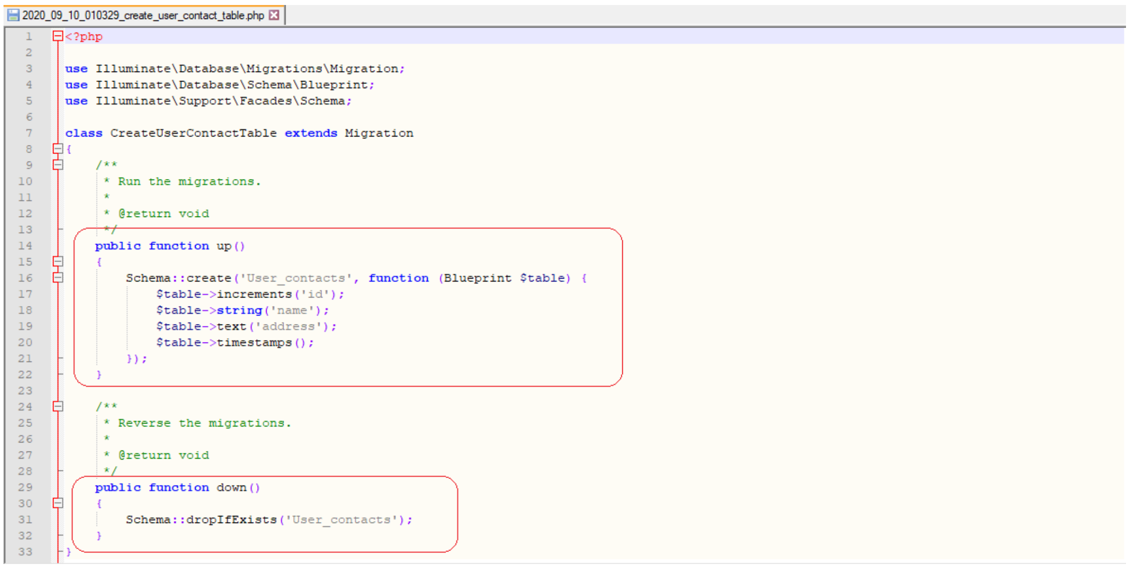
```
php artisan make:migration create_user_contact_table --create=User_contact
```



```
C:\Windows\System32\cmd.exe - php  artisan make:migration create_user_contact_table --create=User_contact

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan make:migration create_user_contact_table --create=User_contact
Created Migration: 2020_09_10_010329_create_user_contact_table
```

- Open newly created file and add code for creating a table in function up(), function down() is used to reverse the migrations, drop the table.

```
public function up()
{
     Schema::create('User_contacts', function (Blueprint $table) {
         $table->increments('id');
         $table->string('name');
         $table->text('address');
         $table->timestamps();
     });
}

public function down()
{
   Schema::dropIfExists('User_contacts');
}
```

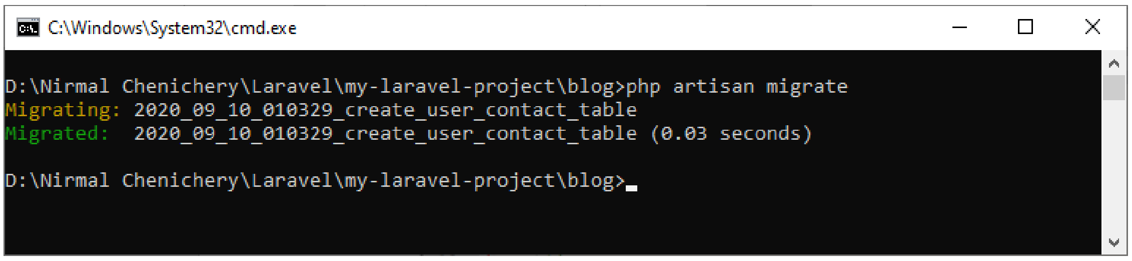


2020_09_10_010329_create_user_contact_table.php

```php
<?php

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

class CreateUserContactTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('User_contacts', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->text('address');
            $table->timestamps();
        });
    }

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

❖ Save the file and run the migration

```
php artisan migrate
```

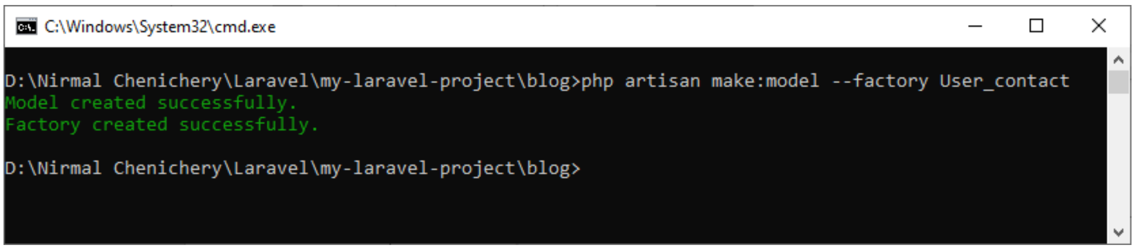


C:\Windows\System32\cmd.exe

```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan migrate
Migrating: 2020_09_10_010329_create_user_contact_table
Migrated:  2020_09_10_010329_create_user_contact_table (0.03 seconds)

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

❖ Next, we need to create model to work with our database table. This will create a new file User_contact.php in app/ directory

```
php artisan make:model --factory User_contact
```

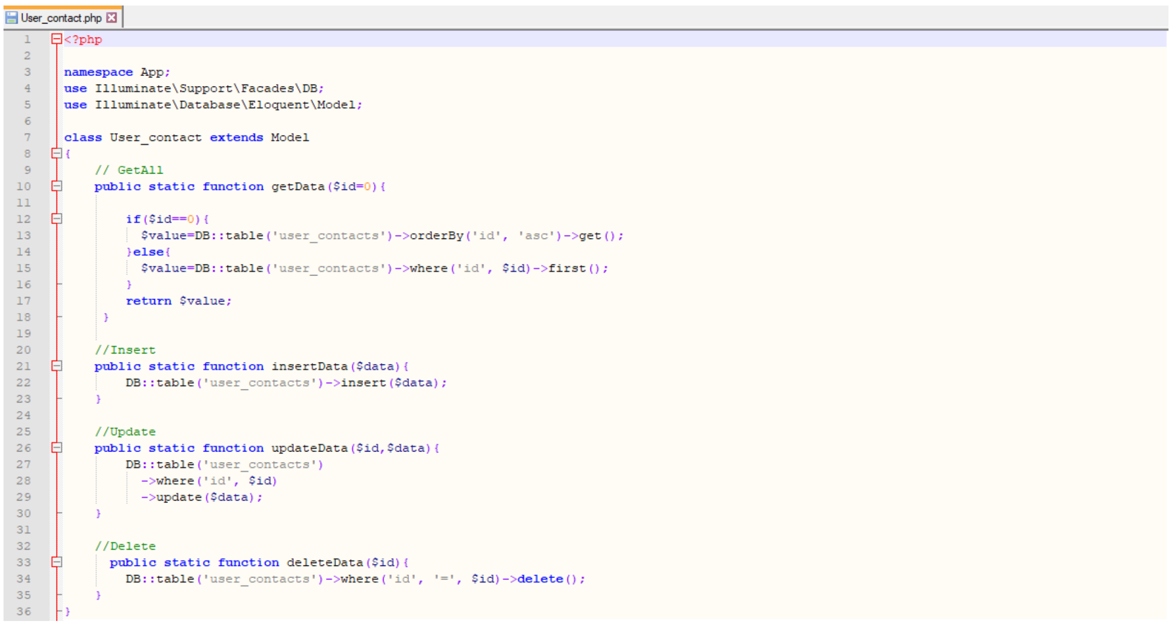


C:\Windows\System32\cmd.exe

```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan make:model --factory User_contact
Model created successfully.
Factory created successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

❖ Write method in model for insert, update and delete (sample code is below)

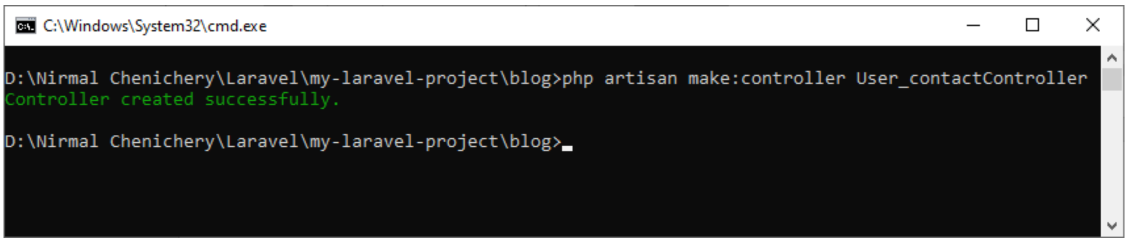
```php
<?php
namespace App;
use Illuminate\Support\Facades\DB;
use Illuminate\Database\Eloquent\Model;

class User_contact extends Model
{
    // GetAll
    public static function getData($id=0){

        if($id==0){
          $value=DB::table('user_contacts')->orderBy('id', 'asc')->get();
        }else{
          $value=DB::table('user_contacts')->where('id', $id)->first();
        }
        return $value;
    }
    //Insert
    public static function insertData($data){
        DB::table('user_contacts')->insert($data);
    }
    //Update
    public static function updateData($id,$data){
        DB::table('user_contacts')
          ->where('id', $id)
          ->update($data);
    }
    //Delete
      public static function deleteData($id){
        DB::table('user_contacts')->where('id', '=', $id)->delete();
    }
}
```


User_contact.php

```php
<?php

namespace App;
use Illuminate\Support\Facades\DB;
use Illuminate\Database\Eloquent\Model;

class User_contact extends Model
{
    // GetAll
    public static function getData($id=0){

        if($id==0){
          $value=DB::table('user_contacts')->orderBy('id', 'asc')->get();
        }else{
          $value=DB::table('user_contacts')->where('id', $id)->first();
        }
        return $value;
    }

    //Insert
    public static function insertData($data){
        DB::table('user_contacts')->insert($data);
    }

    //Update
    public static function updateData($id,$data){
        DB::table('user_contacts')
          ->where('id', $id)
          ->update($data);
    }

    //Delete
      public static function deleteData($id){
        DB::table('user_contacts')->where('id', '=', $id)->delete();
    }
}
```

- Next, we need to create controller using `php artisan make:controller`. This will create a new file `app/Http/Controllers/` directory

```
php artisan make:controller User_contactController
```



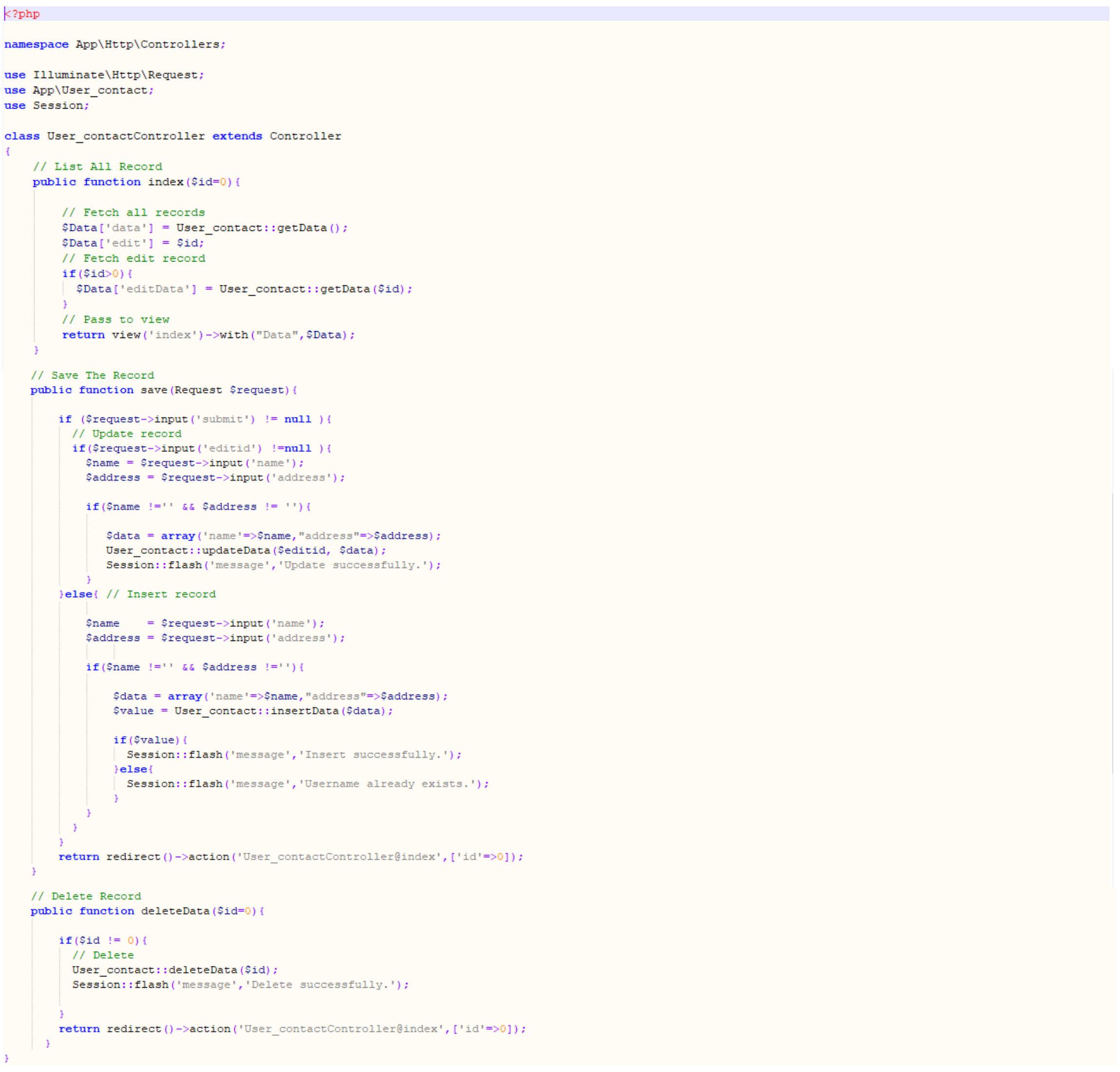
```
C:\Windows\System32\cmd.exe

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan make:controller User_contactController
Controller created successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>_
```

Navigate to app/Http/Controllers/ directory and open User_contactController.php and Import User_contact Model from App namespace with use App\User_contact outside of class.

```
use App\User_contact;
```

- Create method for data manipulations (Fetch, Save, Update, Delete)- Sample code is below.

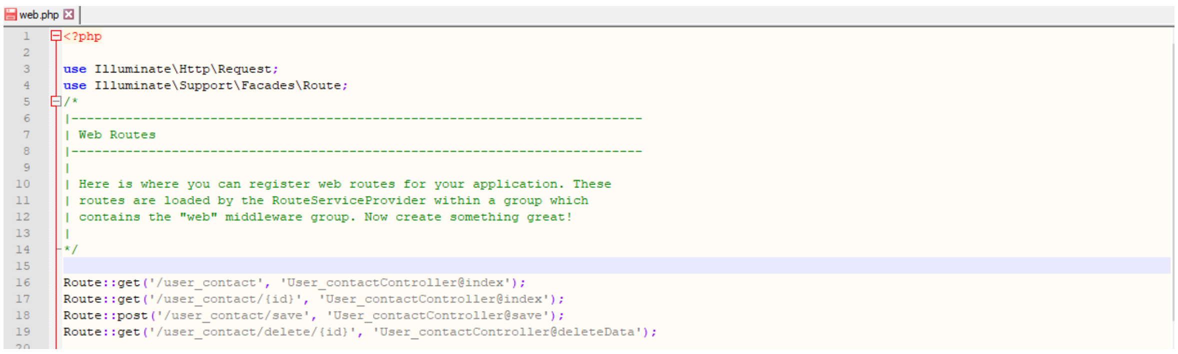
```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\User_contact;
use Session;

class User_contactController extends Controller
{
    // List All Record
    public function index($id=0){

        // Fetch all records
        $Data['data'] = User_contact::getData();
        $Data['edit'] = $id;
        // Fetch edit record
        if($id>0){
          $Data['editData'] = User_contact::getData($id);
        }
        // Pass to view
        return view('index')->with("Data",$Data);
    }

    // Save The Record
    public function save(Request $request){

        if ($request->input('submit') != null ){
          // Update record
          if($request->input('editid') !=null ){
            $name = $request->input('name');
            $address = $request->input('address');

            if($name !='' && $address != ''){

               $data = array('name'=>$name,"address"=>$address);
               User_contact::updateData($editid, $data);
               Session::flash('message','Update successfully.');
            }
        }else{ // Insert record

           $name    = $request->input('name');
           $address = $request->input('address');

           if($name !='' && $address !=''){

               $data = array('name'=>$name,"address"=>$address);
               $value = User_contact::insertData($data);

               if($value){
                 Session::flash('message','Insert successfully.');
               }else{
                 Session::flash('message','Username already exists.');
              }
          }
        }
      }
      return redirect()->action('User_contactController@index',['id'=>0]);
    }

    // Delete Record
    public function deleteData($id=0){

      if($id != 0){
          // Delete
          User_contact::deleteData($id);
          Session::flash('message','Delete successfully.');

      }
      return redirect()->action('User_contactController@index',['id'=>0]);
      }
}
```



```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\User_contact;
use Session;

class User_contactController extends Controller
{
    // List All Record
    public function index($id=0){

        // Fetch all records
        $Data['data'] = User_contact::getData();
        $Data['edit'] = $id;
        // Fetch edit record
        if($id>0){
          $Data['editData'] = User_contact::getData($id);
        }
        // Pass to view
        return view('index')->with("Data",$Data);
    }

    // Save The Record
    public function save(Request $request){

        if ($request->input('submit') != null ){
          // Update record
          if($request->input('editid') !=null ){
            $name = $request->input('name');
            $address = $request->input('address');

            if($name !='' && $address != ''){

               $data = array('name'=>$name,"address"=>$address);
               User_contact::updateData($editid, $data);
               Session::flash('message','Update successfully.');
            }
        }else{ // Insert record

            $name    = $request->input('name');
            $address = $request->input('address');

            if($name !='' && $address !=''){

                $data = array('name'=>$name,"address"=>$address);
                $value = User_contact::insertData($data);

                if($value){
                  Session::flash('message','Insert successfully.');
                }else{
                  Session::flash('message','Username already exists.');
                }

            }
          }
        }
        return redirect()->action('User_contactController@index',['id'=>0]);
    }

    // Delete Record
    public function deleteData($id=0){

        if($id != 0){
          // Delete
          User_contact::deleteData($id);
          Session::flash('message','Delete successfully.');

        }
        return redirect()->action('User_contactController@index',['id'=>0]);
      }
}
```

- Route the pages, Open `routes/web.php` and add below code

```php
Route::get('/user_contact', 'User_contactController@index');
Route::get('/user_contact/{id}', 'User_contactController@index');
Route::post('/user_contact/save', 'User_contactController@save');
Route::get('/user_contact/delete/{id}', 'User_contactController@deleteData
')
```

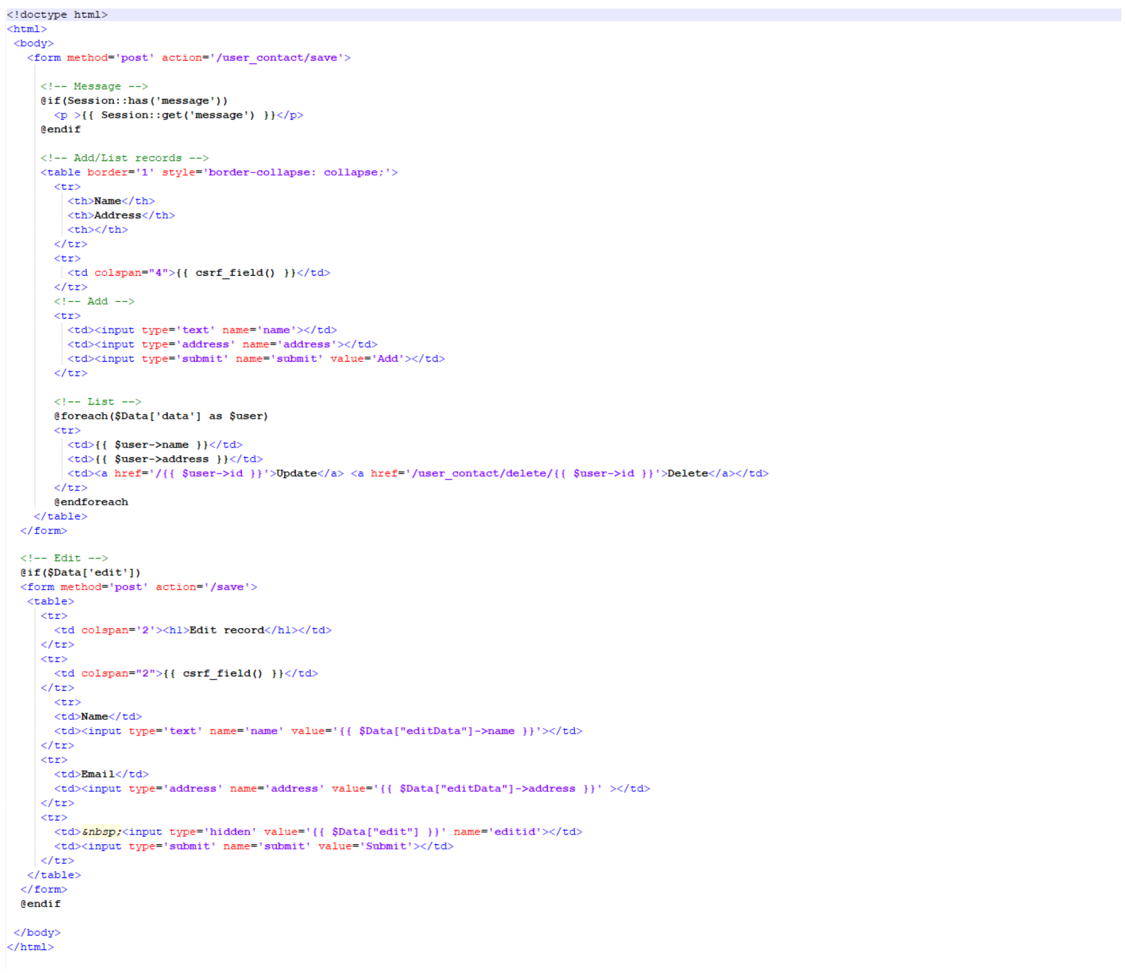


```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/user_contact', 'User_contactController@index');
Route::get('/user_contact/{id}', 'User_contactController@index');
Route::post('/user_contact/save', 'User_contactController@save');
Route::get('/user_contact/delete/{id}', 'User_contactController@deleteData');
```

❖ Create View, A new index.blade.php file in resources/views/ directory. Sample Code is below

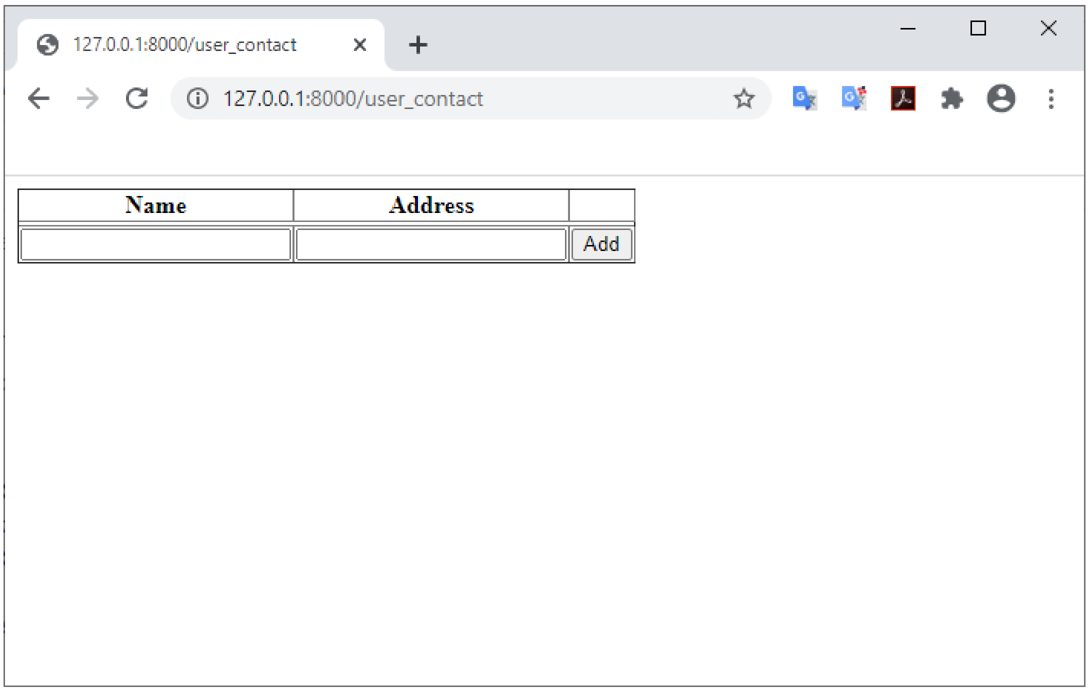
```
<!doctype html>
<html>
 <body>
   <form method='post' action='/user_contact/save'>

     <!-- Message -->
     @if(Session::has('message'))
       <p >{{ Session::get('message') }}</p>
     @endif

     <!-- Add/List records -->
     <table border='1' style='border-collapse: collapse;'>
       <tr>
         <th>Name</th>
         <th>Address</th>
         <th></th>
       </tr>
       <tr>
         <td colspan="4">{{ csrf_field() }}</td>
       </tr>
       <!-- Add -->
       <tr>
         <td><input type='text' name='name'></td>
         <td><input type='address' name='address'></td>
         <td><input type='submit' name='submit' value='Add'></td>
       </tr>

      <!-- List -->
       @foreach($Data['data'] as $user)
       <tr>
         <td>{{ $user->name }}</td>
         <td>{{ $user->address }}</td>
         <td><a href='/{{ $user-
         >id }}'>Update</a> <a href='/user_contact/delete/{{ $user-
         >id }}'>Delete</a></td>
       </tr>
       @endforeach
    </table>
  </form>

  <!-- Edit -->
  @if($Data['edit'])
  <form method='post' action='/save'>
   <table>
     <tr>
       <td colspan='2'><h1>Edit record</h1></td>
```

```
        </tr>
        <tr>
          <td colspan="2">{{ csrf_field() }}</td>
        </tr>
          <tr>
          <td>Name</td>
         <td><input type='text' name='name' value='{{ $Data["editData"]-
         >name }}'></td>
        </tr>
        <tr>
         <td>Email</td>
         <td><input type='address' name='address' value='{{ $Data["editData"]
         ->address }}' ></td>
        </tr>
        <tr>
         <td>&nbsp;<input type='hidden' value='{{ $Data["edit"] }}' name='edi
         tid'></td>
          <td><input type='submit' name='submit' value='Submit'></td>
        </tr>
      </table>
    </form>
    @endif

  </body>
</html>
```

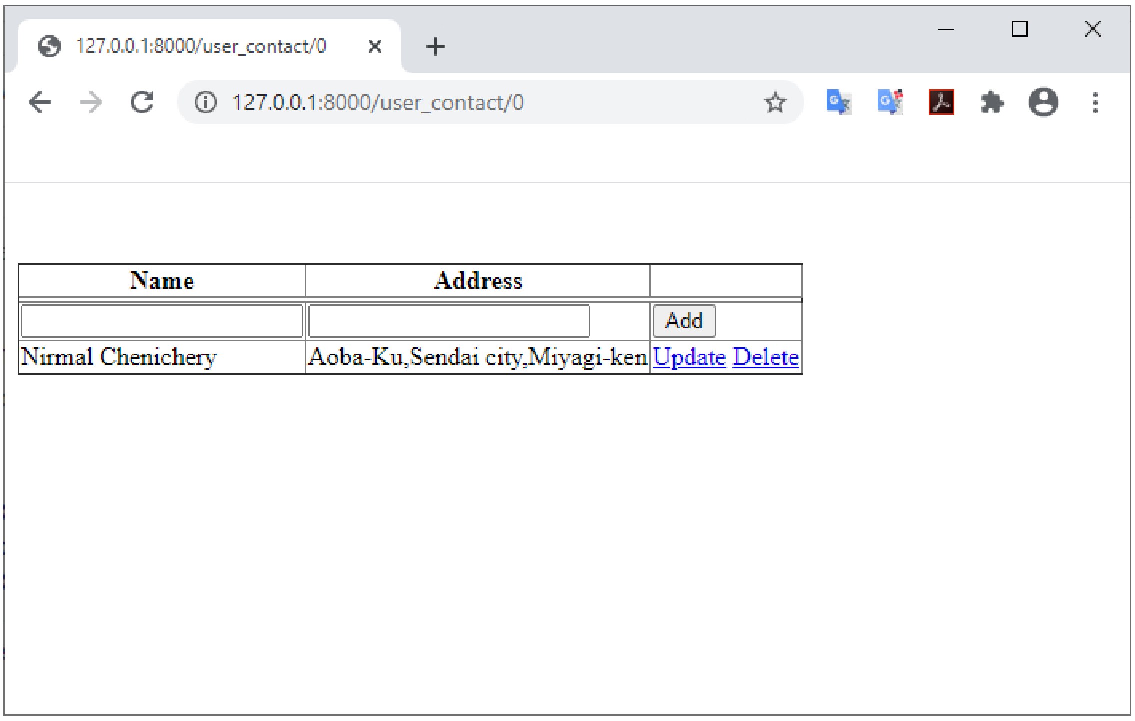


```
<!doctype html>
<html>
 <body>
   <form method='post' action='/user_contact/save'>

     <!-- Message -->
     @if(Session::has('message'))
       <p >{{ Session::get('message') }}</p>
     @endif

     <!-- Add/List records -->
     <table border='1' style='border-collapse: collapse;'>
       <tr>
         <th>Name</th>
         <th>Address</th>
         <th></th>
       </tr>
       <tr>
         <td colspan="4">{{ csrf_field() }}</td>
       </tr>
       <!-- Add -->
       <tr>
         <td><input type='text' name='name'></td>
         <td><input type='address' name='address'></td>
         <td><input type='submit' name='submit' value='Add'></td>
       </tr>

       <!-- List -->
       @foreach($Data['data'] as $user)
       <tr>
         <td>{{ $user->name }}</td>
         <td>{{ $user->address }}</td>
         <td><a href='/{{ $user->id }}'>Update</a> <a href='/user_contact/delete/{{ $user->id }}'>Delete</a></td>
       </tr>
       @endforeach
    </table>
  </form>

  <!-- Edit -->
  @if($Data['edit'])
  <form method='post' action='/save'>
   <table>
     <tr>
       <td colspan='2'><h1>Edit record</h1></td>
     </tr>
     <tr>
       <td colspan="2">{{ csrf_field() }}</td>
     </tr>
       <tr>
       <td>Name</td>
       <td><input type='text' name='name' value='{{ $Data["editData"]->name }}'></td>
     </tr>
     <tr>
       <td>Email</td>
       <td><input type='address' name='address' value='{{ $Data["editData"]->address }}' ></td>
     </tr>
     <tr>
       <td>&nbsp;<input type='hidden' value='{{ $Data["edit"] }}' name='editid'></td>
       <td><input type='submit' name='submit' value='Submit'></td>
     </tr>
   </table>
  </form>
  @endif

 </body>
</html>
```

❖ Run the application, `http://127.0.0.1:8000/user_contact`



Name	Address	
		Add

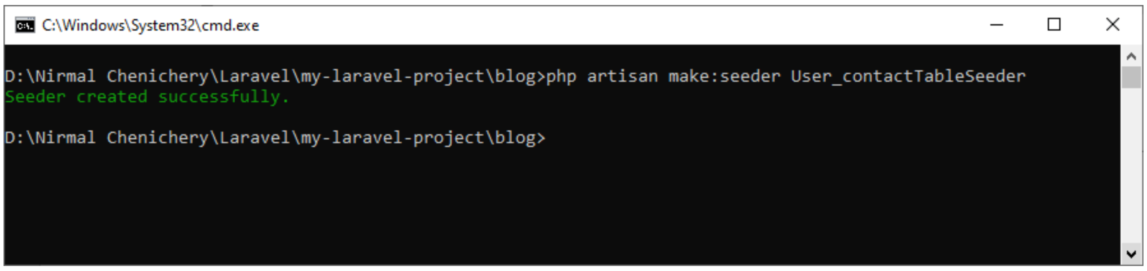


Name	Address	
		Add
Nirmal Chenichery	Aoba-Ku,Sendai city,Miyagi-ken	Update Delete

Inserting default / seed data into table

- The `make:seeder` command will create seed file `database/seeds/` directory.

```
php artisan make:seeder User_contactTableSeeder
```



```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan make:seeder User_contactTableSeeder
Seeder created successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```

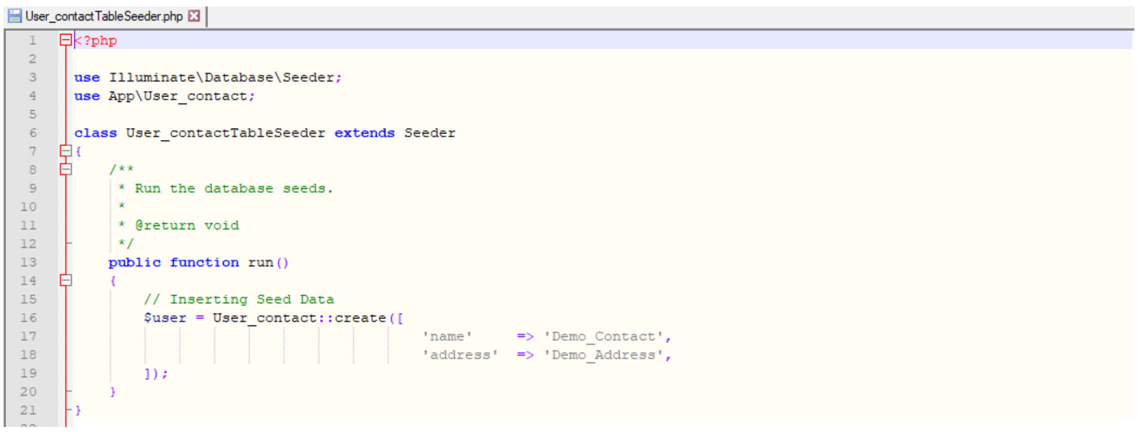
database > seeds > User_contactTableSeeder.php > ...

```
<?php

use Illuminate\Database\Seeder;

class User_contactTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //
    }
}
```

❖ Open `User_contactTableSeeder` and write code

```
use App\User_contact;  // in above class

public function run()
     {
          // Inserting Seed Data
          $user = User_contact::create([
                         'name'      => 'Demo_Contact',
                         'address'   => 'Demo_Address',
          ]);
     }
```

Note: - if migrate is not done, please run `php artisan migrate`

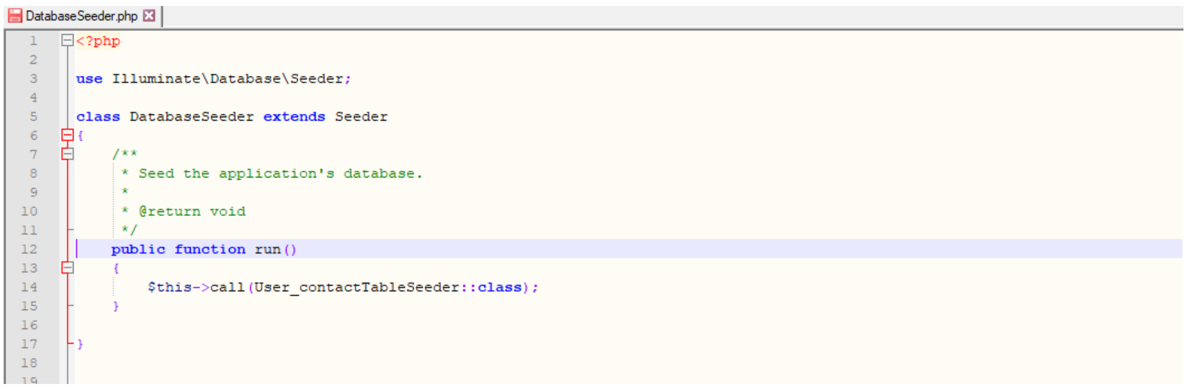


User_contactTableSeeder.php

```
<?php

use Illuminate\Database\Seeder;
use App\User_contact;

class User_contactTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        // Inserting Seed Data
        $user = User_contact::create([
                                    'name'      => 'Demo_Contact',
                                    'address'   => 'Demo_Address',
        ]);
    }
}
```

❖ In order to "activate" the `User_contactTableSeeder`, we need to call it from the main database/seeds/DatabaseSeeder.php run method:

```
public function run()
     {
          $this->call(User_contactTableSeeder::class);
     }
```

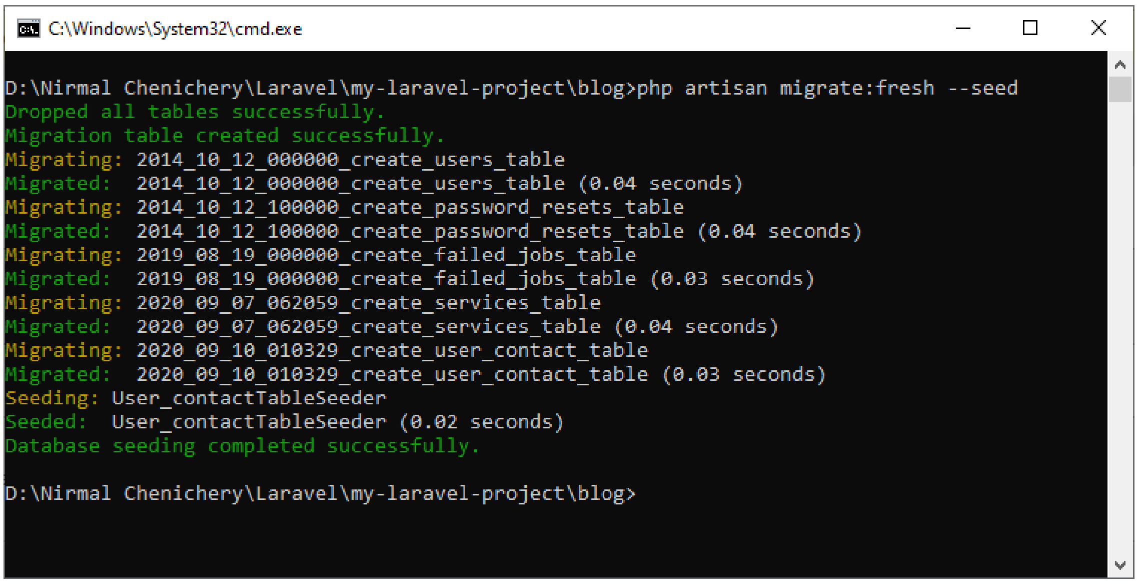


DatabaseSeeder.php

```
<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call(User_contactTableSeeder::class);
    }
}
```

- You can now run the migrations and seeds to add data to the table automatically. Using the migrate:fresh command, we can get a clean schema that applies all migrations and then seeds the database.

```
php artisan migrate:fresh -seed
```



```
D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.04 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (0.04 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.03 seconds)
Migrating: 2020_09_07_062059_create_services_table
Migrated:  2020_09_07_062059_create_services_table (0.04 seconds)
Migrating: 2020_09_10_010329_create_user_contact_table
Migrated:  2020_09_10_010329_create_user_contact_table (0.03 seconds)
Seeding: User_contactTableSeeder
Seeded:  User_contactTableSeeder (0.02 seconds)
Database seeding completed successfully.

D:\Nirmal Chenichery\Laravel\my-laravel-project\blog>
```