

Doptor Organogram Package Setup Guide

This guide will walk you through the steps required to manually or automatically set up the **Doptor Organogram Package** in a Laravel project. It also provides information on how to improve and extend the package to suit your requirements.

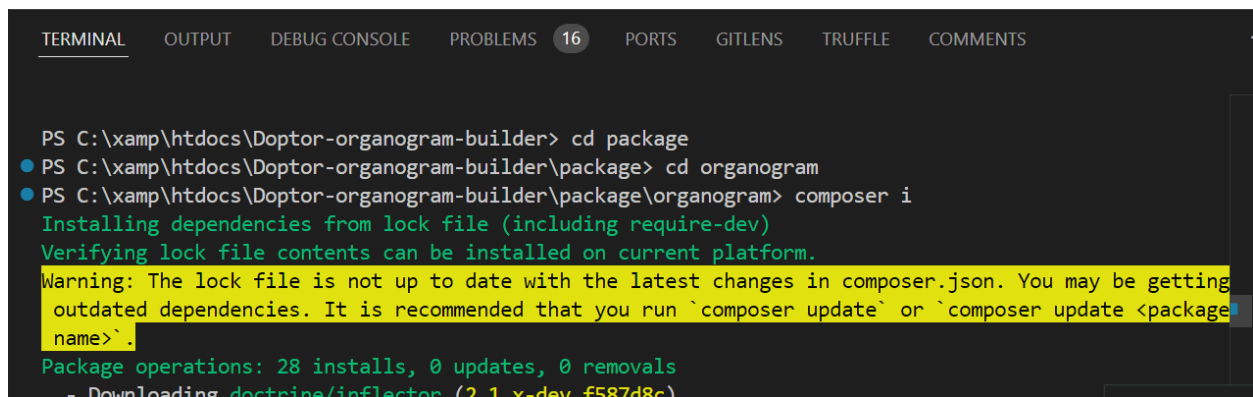
Manual Setup

Follow these steps for the manual setup of the Doptor Organogram package:

Step 1: Instal dependencies

Install all dependencies on (organogram builder) in `package/organogram`

Run `composer i`



```
TERMINAL  OUTPUT  DEBUG CONSOLE  PROBLEMS  16  PORTS  GITLENS  TRUFFLE  COMMENTS

PS C:\xamp\htdocs\Doptor-organogram-builder> cd package
● PS C:\xamp\htdocs\Doptor-organogram-builder\package> cd organogram
● PS C:\xamp\htdocs\Doptor-organogram-builder\package\organogram> composer i
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Warning: The lock file is not up to date with the latest changes in composer.json. You may be getting
outdated dependencies. It is recommended that you run `composer update` or `composer update <package
name>`.
Package operations: 28 installs, 0 updates, 0 removals
- Downloading doctrine/inflector (2.1.x-dev f587d8c)
```

Install the Laravel UI Package

1. Open your terminal or command line interface.
2. Navigate to your Laravel project directory.
3. Run the following command to install the `laravel/ui` package:
`composer require laravel/ui`

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\xampp\htdocs\Doptor-organogram-builder> composer require laravel/ui
./composer.json has been updated
Running composer update laravel/ui
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Package fruitcake/laravel-cors is abandoned, you should avoid using it. No replacement was suggested.
Package laravelcollective/html is abandoned, you should avoid using it. Use spatie/laravel-html instead.
```

Step 2: Configure `composer.json`

1. Open the `composer.json` file in the root of your project.
2. In the `autoload-dev` section, under `"psr-4"`, add the following line:
`"a2i\organogram\":`
`"../../Doptor-organogram-builder/package/organogram/src"`

```
{
    "autoload-dev": {
        "psr-4": {
            "Tests\\": "tests/",
            "a2i\\organogram\\": "../../Doptor-organogram-builder/package/organogram/src"
        }
    },
    "autoload": {
        "psr-4": {
            "a2i\\organogram\\": "src/"
        }
    }
}
```

- **Note:** The path must be the **relative path** from your project folder to the package folder. Adjust the path accordingly based on where the `Doptor-organogram-builder` package is located in your system.
3. Save the changes to the `composer.json` file.

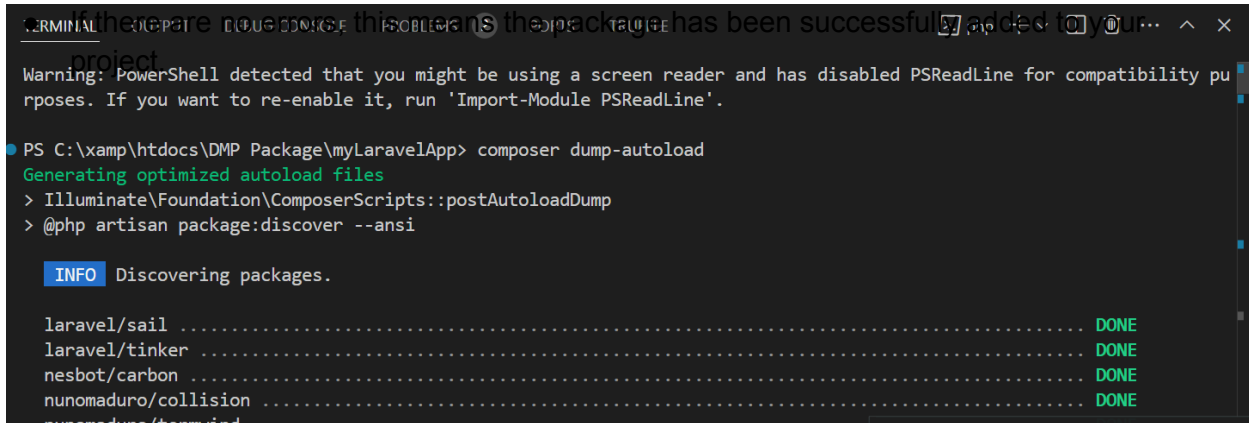
Step 3: Register the Service Provider

1. Open the `bootstrap/providers.php` file.
2. Add the following line under the `providers` section to register the service provider:
`a2i\organogram\OrganogramServiceProvider::class,`
3. Save the file after making the changes.

Step 4: Dump Autoload

To ensure the package is correctly recognized, run the following command in your terminal:

```
composer dump-autoload
```



```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\xampp\htdocs\DMP Package\myLaravelApp> composer dump-autoload
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

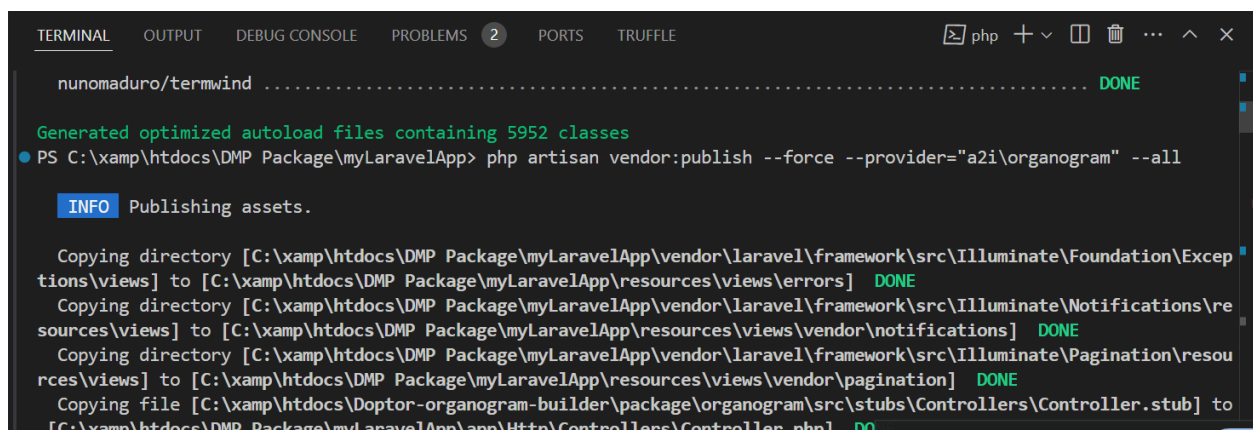
laravel/sail ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind .....
```

Step 5: Publish Vendor Files

Run the following command to publish the package files:

```
php artisan vendor:publish --force --provider="a2i\organogram" --all
```

- This command will add some new files to your project and override others.



```
nunomaduro/termwind ..... DONE

Generated optimized autoload files containing 5952 classes
PS C:\xampp\htdocs\DMP Package\myLaravelApp> php artisan vendor:publish --force --provider="a2i\organogram" --all

 INFO  Publishing assets.

Copying directory [C:\xampp\htdocs\DMP Package\myLaravelApp\vendor\laravel\framework\src\Illuminate\Foundation\Exceptions\views] to [C:\xampp\htdocs\DMP Package\myLaravelApp\resources\views\errors] DONE
Copying directory [C:\xampp\htdocs\DMP Package\myLaravelApp\vendor\laravel\framework\src\Illuminate\Notifications\resources\views] to [C:\xampp\htdocs\DMP Package\myLaravelApp\resources\views\vendor\notifications] DONE
Copying directory [C:\xampp\htdocs\DMP Package\myLaravelApp\vendor\laravel\framework\src\Illuminate\Pagination\resources\views] to [C:\xampp\htdocs\DMP Package\myLaravelApp\resources\views\vendor\pagination] DONE
Copying file [C:\xampp\htdocs\Doctor-organogram-builder\package\organogram\src\stubs\Controllers\Controller.stub] to [C:\xampp\htdocs\DMP Package\myLaravelApp\app\Http\Controllers\Controller.php] DO
```

Step 6: Review the New Files

The `php artisan vendor:publish` command will make the following changes to your project:

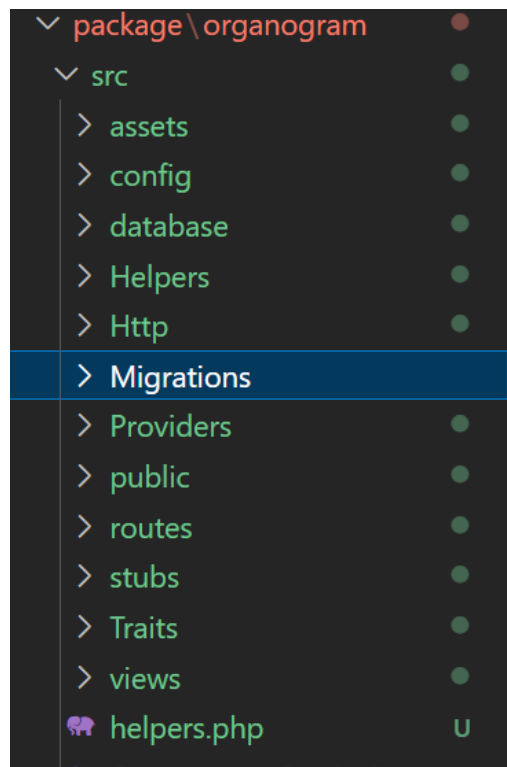
- **Files that will be added/overridden:**
 - `routes/web.php`: This file will be modified to include the necessary routes for the organogram.
 - `User.php`: This file will be updated with relevant traits.
 - `Controller.php`: Controllers related to the organogram will be added.
 - `Traits` folder: A new folder named `Traits` will be created with useful functionalities.
 - `Auth` folder inside `Controller` and `View` folders.
 - **Assets**: CSS, JavaScript, fonts, images, and other asset folders will be added inside the `public` folder.

Step 7: Migrations Setup

To set up the database tables required by the Doptor Organogram package, follow these steps:

1. Place the migration files inside the following directory:

`package/organogram/src/migrations`



2. In the `OrganogramServiceProvider.php` file, ensure that the line to load migrations is uncommented:

```
$this->loadMigrationsFrom(__DIR__.'/migrations');
```

```
$this->loadRoutesFrom(__DIR__.'/routes/web.php');  
$this->loadViewsFrom(__DIR__.'/views', 'organogrampkg');  
// To run migrations put all the migrations file inside src/migrations folder  
// below line will run all your migrations when you run php artisan migrate command  
// $this->loadMigrationsFrom(__DIR__.'/migrations'); uncomment this line for doptor migration
```

3. Finally, run the migration command to apply the database changes:

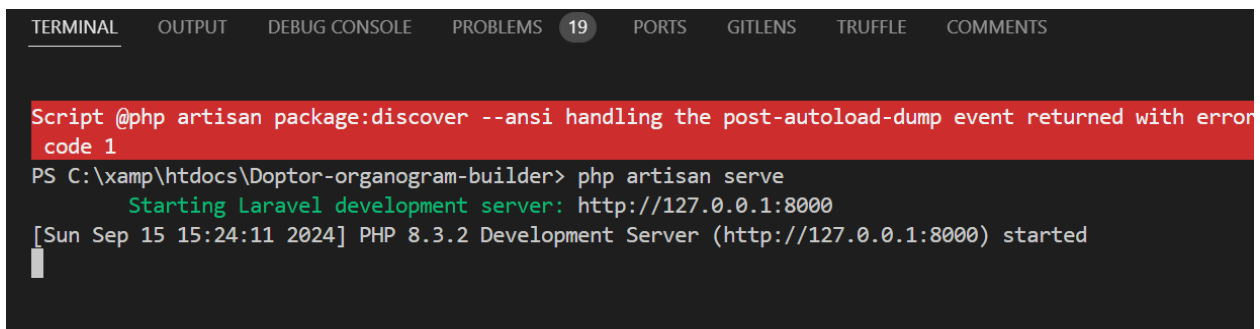
```
php artisan migrate
```

This will set up the necessary database tables for the organogram package.

Step 8: Start the Laravel Server

Once the setup is complete, start the Laravel development server:

```
php artisan serve
```



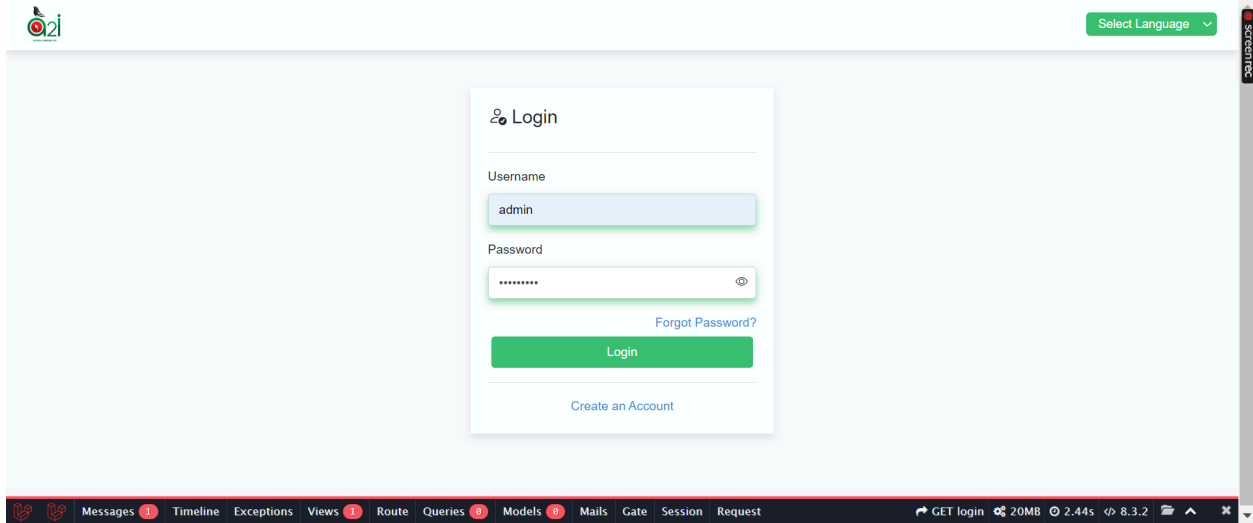
The screenshot shows a terminal window with the following content:

```
Script @php artisan package:discover --ansi handling the post-autoload-dump event returned with error code 1  
PS C:\xampp\htdocs\Doctor-organogram-builder> php artisan serve  
Starting Laravel development server: http://127.0.0.1:8000  
[Sun Sep 15 15:24:11 2024] PHP 8.3.2 Development Server (http://127.0.0.1:8000) started
```

Step 9: Access the Login Page

You can now navigate to the `/login` route in your browser:

<http://localhost:8000/login>



You should see the Doptor login page at this point.

Automatic Setup (After Publishing to Packagist)

Once the Doptor Organogram package is published on [Packagist](#), follow these steps for an automatic setup:

Step 1: Install the Laravel UI Package

Just like in the manual setup, you will need to install the `laravel/ui` package. In your Laravel project directory, run:

```
composer require laravel/ui
```

Step 2: Install the Organogram Package via Composer

In your Laravel project directory, run the following command to require the organogram package:

```
composer require a2i/organogram
```

Step 3: Dump Autoload

After requiring the package, run:

```
composer dump-autoload
```

This will load the new package into your Laravel project.

Step 4: Publish Vendor Files

To publish the package's vendor files, run the following command:

```
php artisan vendor:publish --force --provider="a2i\organogram" --all
```

Step 4: Run Migrations

Now, run the migrations to set up the database:

```
php artisan migrate
```

Step 6: Start the Server

Run the Laravel server:

```
php artisan serve
```

Improving the Doptor Organogram Package

Here are some ways to improve the package and customize it according to your specific needs:

1. Understanding Requirements

Before making improvements, clarify your project's requirements. The current structure of the Doptor Organogram package is a base that you can build upon. Assess what additional features or customizations are needed for your specific use case.

2. Migration Files

To make sure migrations work properly, you need to manage them in the package structure:

Place migration files inside the following directory:

```
package/organogram/src/migrations
```

Then, in the `OrganogramServiceProvider.php` file, uncomment the line that loads migrations:

```
$this->loadMigrationsFrom(__DIR__.'/migrations');
```

By doing this, Laravel will automatically load the necessary migration files from the package.

3. Extending the Functionality

At present, only the login and dashboard pages are functional. You can extend the package by:

- Creating additional routes and pages.
- Enhancing the dashboard to show dynamic data.
- Adding more detailed organizational structures and hierarchy features.
- Implementing user roles and permissions for better access control.

4. Referencing Laravel/UI Package

The **Laravel/UI** package is a great resource for building packages. By studying its structure, you can gain insights into how to further enhance your organogram package.

5. Documentation

As you extend and improve the package, ensure that you write clear and comprehensive documentation. This will make it easier for others (and yourself) to understand how to set up, use, and modify the package.

By following the steps outlined above, you can successfully set up and improve the **Doptor Organogram Package** in your Laravel project. With further customization, the package can be tailored to meet your specific requirements, ensuring that it integrates seamlessly into your workflow.