Laravel 9 Livewire CRUD Operations using Jetstream Example

- **Step 1:** Set Up Laravel Project
- **Step 2:** Add Database Details in ENV
- **Step 3:** Create Model and Migration
- **Step 4:** Install Livewire and Jetstream
- **Step 5:** Create CRUD Components
- **Step 6:** Frame Up CRUD Component
- **Step 7:** Create Routes
- **Step 8:** Prepare Blade View
- **Step 9:** Start Development Server

Set Up Laravel Project

Initiate the project by installing a fresh new laravel application, so head over to the terminal, type the command, and create a new laravel app.

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

Next, move to the project folder:

```
cd laravel_livewire_crud
```

Add Database Details in ENV

Next, you have to connect the laravel app to the database, hence open the **.env** configuration file and add the database credentials as suggested below.

```
DB_CONNECTION=mysql
```

```
DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=database_name

DB_USERNAME=database_user_name

DB_PASSWORD=database_password
```

Create Model and Migration

In the terminal screen, type the recommended command and execute it to generate model and migration files.

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

You need to add the $fillable array and add the table values such as name, email and mobile in the **app/Models/Student.php** file.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;
```

```php
class Student extends Model


{


    use HasFactory;




    protected $fillable = [


        'name',


        'email',


        'mobile'


    ];


}
```

Subsequently, we need to configure the migration table for students, so add the table properties in the **database/migrations/create_students_table.php** file .

```php
<?php


use Illuminate\Database\Migrations\Migration;
```

```php
use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateStudentsTable extends Migration

{

    /**

     * Run the migrations.

     *

     * @return void

     */

    public function up()

    {

        Schema::create('students', function (Blueprint $table) {

            $table->id();

            $table->string('name', 100);
```

```
            $table->string('email');

            $table->string('mobile');

            $table->timestamps();

        });

    }

    /**

     * Reverse the migrations.

     *

     * @return void

     */

    public function down()

    {

        Schema::dropIfExists('students');

    }
```

```
}
```

Install Livewire and Jetstream

Now, you need to run the suggested command to install jetstream and livewire packages in the laravel app.

```
composer require laravel/jetstream



php artisan jetstream:install livewire
```

Further, you have to compile the assets, accordingly, use both the npm commands simultaneously to get done the build compilation task.

```
npm install && npm run dev
```

Now, open another terminal screen and run the migration using the suggested php artisan command.

```
php artisan migrate
```

Create Livewire CRUD Components

This step tells how to generate the Livewire crud component, so type the command in the terminal and run the command.

```
php artisan make:livewire crud
```

The execution of the above command generated two files, one in Http and another in resources directories.

```
CLASS: app/Http/Livewire/Crud.php


VIEW:  resources/views/livewire/crud.blade.php
```

Frame Up CRUD Component

Next, you have to define the crud methods inside the livewire crud component file; hence open and update the given code in the **app/Http/Livewire/Crud.php** file.

```
<?php

namespace App\Http\Livewire;

use Livewire\Component;

use App\Models\Student;




class Crud extends Component

{

    public $students, $name, $email, $mobile, $student_id;
```

```php
    public $isModalOpen = 0;

    public function render()

    {

        $this->students = Student::all();

        return view('livewire.crud');

    }

    public function create()

    {

        $this->resetCreateForm();

        $this->openModalPopover();

    }

    public function openModalPopover()

    {

        $this->isModalOpen = true;
```

```
    }

    public function closeModalPopover()

    {

        $this->isModalOpen = false;

    }

    private function resetCreateForm(){

        $this->name = '';

        $this->email = '';

        $this->mobile = '';

    }




    public function store()

    {

        $this->validate([
```

```
            'name' => 'required',

            'email' => 'required',

            'mobile' => 'required',

        ]);




        Student::updateOrCreate(['id' => $this->student_id], [

            'name' => $this->name,

            'email' => $this->email,

            'mobile' => $this->mobile,

        ]);

        session()->flash('message', $this->student_id ? 'Student updated.' :
'Student created.');

        $this->closeModalPopover();

        $this->resetCreateForm();

    }
```

```php
    public function edit($id)

    {

        $student = Student::findOrFail($id);

        $this->student_id = $id;

        $this->name = $student->name;

        $this->email = $student->email;

        $this->mobile = $student->mobile;




        $this->openModalPopover();

    }




    public function delete($id)

    {

        Student::find($id)->delete();
```

```
        session()->flash('message', 'Studen deleted.');


    }


}
```

Create Routes

To enable navigation for the livewire crud app, define the routes. Import Crud class and define the route method, so open **routes/web.php** file.

```
<?php


use Illuminate\Support\Facades\Route;


use App\Http\Livewire\Crud;


/*


|--------------------------------------------------------------------------


| 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('students', Crud::class);
```

If you are getting a **“Route [dashboard] not defined.”** error, open the **resources/views/navigation-menu.blade.php** file and comment entire code available in the file.

However, in a real app, you enable authentication; in that case, you have to skip this step.

Prepare Blade View

Lastly, we have to create the blade views; our crud operations layout is simple. It comprises two-blade view files. In one view, we will show data in tabular forms where users can take actions to update or edit student data. At the same time, another view contains the modal popup, which creates and updates the data.

Open and insert code in **resources/views/livewire/crud.blade.php** file.

```
<x-slot name="header">

    <h2 class="text-center">Laravel 9 Livewire CRUD Demo</h2>

</x-slot>

<div class="py-12">

    <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

        <div class="bg-white overflow-hidden shadow-xl sm:rounded-lg px-4 py-4">

            @if (session()->has('message'))

            <div class="bg-teal-100 border-t-4 border-teal-500 rounded-b text-
teal-900 px-4 py-3 shadow-md my-3"

                role="alert">

                <div class="flex">

                    <div>

                        <p class="text-sm">{{ session('message') }}</p>

                    </div>

                </div>
```

```
            </div>


            @endif


            <button wire:click="create()"


                class="my-4 inline-flex justify-center w-full rounded-md border
border-transparent px-4 py-2 bg-red-600 text-base font-bold text-white shadow-sm
hover:bg-red-700">


                Create Student


            </button>


            @if($isModalOpen)


            @include('livewire.create')


            @endif


            <table class="table-fixed w-full">


                <thead>


                    <tr class="bg-gray-100">


                        <th class="px-4 py-2 w-20">No.</th>
```

```
                <th class="px-4 py-2">Name</th>

                <th class="px-4 py-2">Email</th>

                <th class="px-4 py-2">Mobile</th>

                <th class="px-4 py-2">Action</th>

            </tr>

        </thead>

        <tbody>

            @foreach($students as $student)

            <tr>

                <td class="border px-4 py-2">{{ $student->id }}</td>

                <td class="border px-4 py-2">{{ $student->name }}</td>

                <td class="border px-4 py-2">{{ $student->email}}</td>

                <td class="border px-4 py-2">{{ $student->mobile}}</td>

                <td class="border px-4 py-2">
```

```
                            <button wire:click="edit({{ $student->id }})"

                                class="flex px-4 py-2 bg-gray-500 text-gray-900
cursor-pointer">Edit</button>

                            <button wire:click="delete({{ $student->id }})"

                                class="flex px-4 py-2 bg-red-100 text-gray-900
cursor-pointer">Delete</button>

                        </td>

                    </tr>

                    @endforeach

                </tbody>

            </table>

        </div>

    </div>

</div>
```

Please create a new file name it create.blade.php file inside the resources/views/livewire/ then add the below

code in
the **resources/views/livewire/create.blade.php** file.

```
<div class="fixed z-10 inset-0 overflow-y-auto ease-out duration-400">

    <div class="flex items-end justify-center min-h-screen pt-4 px-4 pb-20 text-
center sm:block sm:p-0">

        <div class="fixed inset-0 transition-opacity">

            <div class="absolute inset-0 bg-gray-500 opacity-75"></div>

        </div>

        <div class="inline-block align-bottom bg-white rounded-lg text-left
overflow-hidden shadow-xl transform transition-all sm:my-8 sm:align-middle
sm:max-w-lg sm:w-full"

            role="dialog" aria-modal="true" aria-labelledby="modal-headline">

            <form>

                <div class="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4">

                    <div class="">

                        <div class="mb-4">

                            <label for="exampleFormControlInput1"
```

```
                                class="block text-gray-700 text-sm font-bold mb-
2">Name</label>



                            <input type="text"



                                class="shadow appearance-none border rounded w-
full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-
outline"



                                id="exampleFormControlInput1" placeholder="Enter
Name" wire:model="name">



                            @error('name') <span class="text-red-500">{{
$message }}</span>@enderror



                        </div>



                        <div class="mb-4">



                            <label for="exampleFormControlInput2"



                                class="block text-gray-700 text-sm font-bold mb-
2">Email:</label>



                            <textarea



                                class="shadow appearance-none border rounded w-
full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-
outline"



                                id="exampleFormControlInput2" wire:model="email"
```

```
                                placeholder="Enter Email"></textarea>

                            @error('email') <span class="text-red-500">{{
$message }}</span>@enderror

                        </div>

                        <div class="mb-4">

                            <label for="exampleFormControlInput2"

                                class="block text-gray-700 text-sm font-bold mb-
2">Mobile:</label>

                            <textarea

                                class="shadow appearance-none border rounded w-
full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-
outline"

                                id="exampleFormControlInput2"
wire:model="mobile"

                                placeholder="Enter Mobile"></textarea>

                            @error('mobile') <span class="text-red-500">{{
$message }}</span>@enderror

                        </div>
```

```
                    </div>


                </div>


                <div class="bg-gray-50 px-4 py-3 sm:px-6 sm:flex sm:flex-row-
reverse">


                    <span class="flex w-full rounded-md shadow-sm sm:ml-3 sm:w-
auto">


                        <button wire:click.prevent="store()" type="button"


                            class="inline-flex justify-center w-full rounded-md
border border-transparent px-4 py-2 bg-red-600 text-base leading-6 font-bold
text-white shadow-sm hover:bg-red-700 focus:outline-none focus:border-green-700
focus:shadow-outline-green transition ease-in-out duration-150 sm:text-sm
sm:leading-5">


                            Store


                        </button>


                    </span>


                    <span class="mt-3 flex w-full rounded-md shadow-sm sm:mt-0
sm:w-auto">


                        <button wire:click="closeModalPopover()" type="button"


                            class="inline-flex justify-center w-full rounded-md
border border-gray-300 px-4 py-2 bg-white text-base leading-6 font-bold text-
gray-700 shadow-sm hover:text-gray-700 focus:outline-none focus:border-blue-300
```

```
focus:shadow-outline-blue transition ease-in-out duration-150 sm:text-sm
sm:leading-5">


                            Close


                        </button>


                    </span>


                </div>


            </form>


        </div>


    </div>


</div>
```

*In order to run the app, we have to make a little tweak, just go to **resources/views/navigation-menu.blade.php** file, and remove or comment out the entire code of this file.*

Start Development Server

We have almost completed the crud app development and now only left with checking how it works. So, use the

php artisan command along with the serve tag to invoke the laravel development server.

```
php artisan serve
```

You can eventually test the app by using the given below url on the browser's address bar.

```
http://127.0.0.1:8000/students
```

OUTPUT