Laravel 10 CRUD Tutorial

1. Setup:

- **Install Laravel 10:** Use `composer create-project laravel/laravel your_project_name`
- **Configure database:** Set your DB credentials in `.env` file

2. Model and Migration:

- **Create Model:** Use `php artisan make:model Post` (Replace "Post" with your model name)
- **Define fields:** Edit `App\Models\Post.php` and add fields (e.g., title, content)
- **Create Migration:** Use `php artisan make:migration create_posts_table`
- **Edit Migration:** Edit `database/migrations/2024_01_27_084208_create_posts_table.php` and define table schema based on your model.
- **Migrate schema:** Use `php artisan migrate`

3. Controller:

- **Create Controller:** Use `php artisan make:controller PostController`
- **Define methods:** Edit `App\Http\Controllers\PostController.php` and add CRUD methods (e.g., index, create, store, edit, update, destroy)
- **Use Model and methods:** Each method should:
  - Use `Post::all()` for `index` to list posts.
  - Use `Post::create()` and form request validation for `store` to create a new post.
  - Use `Post::find($id)` for `edit` and `update` to find and update a specific post.
  - Use `Post::destroy($id)` for `destroy` to delete a post.

4. Routes:

- **Define routes:** Edit routes/web.php and define routes for each action:
  - Route::get('posts', 'PostController@index');
  - Route::get('posts/create', 'PostController@create');
  - Route::post('posts', 'PostController@store');
  - Route::get('posts/{id}/edit', 'PostController@edit');
  - Route::put('posts/{id}', 'PostController@update');
  - Route::delete('posts/{id}', 'PostController@destroy');

5. Views:

- **Create blade files:** Create folders for index, create, edit and put corresponding .blade.php files inside.
- **Use Blade syntax:** Each view should:
  - Display list of posts in index.blade.php with links to edit/delete buttons.
  - Show form for creating new post in create.blade.php.
  - Pre-populate form with existing data for editing in edit.blade.php.
  - Show confirmation message or redirect after CRUD actions.

6. Run the Application:

- Start development server: php artisan serve
- Access routes in your browser (e.g., http://localhost:8000/posts)

Explanations:

- **Artisan:** Laravel's command-line tool for tasks like creating models, controllers, etc.
- **Migrations:** Define changes to your database schema.
- **Controllers:** Handle HTTP requests and business logic.
- **Routes:** Map URLs to controller methods.

- **Blade templates:** Dynamically generate HTML views.

This is a basic example. You can further enhance your CRUD application by:

- Adding form validation and error handling.
- Implementing search functionality.
- Adding user authentication and authorization.
- Using Bootstrap or other front-end frameworks for styling.

I hope this explanation helps you get started with building a CRUD application in Laravel 10!

Laravel 10 CRUD with Detailed Implementation

Here's a more detailed implementation of the Laravel 10 CRUD, breaking down each step with code examples:

1. Setup:

1. **Install Laravel 10:** Run `composer create-project laravel/laravel your_project_name` in your terminal.
2. **Configure database:** Set your database credentials in the `.env` file located in the project root. You'll need database name, username, password, and host.

2. Model and Migration:

1. **Create Model:** Run `php artisan make:model Post` to generate a `Post` model in the `App\Models` directory.
2. **Define fields:** Edit the `App\Models\Post.php` file and add your fields. For example:

```
<?php

namespace App\Models;
```

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

class Post extends Model
{
    protected $fillable = ['title', 'content'];

    // You can define additional methods here
}
```

1. **Create Migration:** Run `php artisan make:migration create_posts_table` to generate a migration file in the `database/migrations` directory.
2. **Edit Migration:** Open the generated migration file (`2024_01_27_084208_create_posts_table.php`) and define the table schema. Replace the default content with:

```
<?php

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

class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('content');
            $table->timestamps();
        });
    }

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

```
    }
}
```

1. **Migrate schema:** Run `php artisan migrate` to apply the migration and create the `posts` table in your database.

3. Controller:

1. **Create Controller:** Run `php artisan make:controller PostController` to generate a `PostController` in the `App\Http\Controllers` directory.
2. **Define methods:** Edit the `App\Http\Controllers\PostController.php` file and add CRUD methods. Here's an example structure:

```
<?php

namespace App\Http\Controllers;

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

class PostController extends Controller
{
    public function index()
    {
        $posts = Post::all();
        return view('posts.index', compact('posts'));
    }

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

    public function store(Request $request)
    {
        $request->validate([
            'title' => 'required|string|max:255',
            'content' => 'required|string',
        ]);

        $post = Post::create($request->all());

        return redirect()->route('posts.index')->with('success', 'Post created successfully!');
    }
```

```
    // Implement remaining methods (edit, update, destroy) similarly

}
```

4. Routes:

1. **Define routes:** Open the routes/web.php file and add routes for each action:

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

Route::resource('posts', 'PostController');

// Alternatively, define individual routes:
// Route::get('posts', 'PostController@index');
// Route::get('posts/create', 'PostController@create');
// ... and so on
```

5. Views:

1. **Create blade files:** Create folders for index, create, edit, and destroy within the resources/views directory. Put corresponding .blade.php files inside each folder.
2. **Use Blade syntax:** Edit each view file and write HTML with Blade directives to display content and forms. Here's an example for index.blade.php:

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

@section('content')
    <h1>Posts</h1>

    @if (session('success'))
        <div class="alert alert-success">

        </div>
    @endif

    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Title</th>
                <th>Content</th>
                <th>Actions</th>
```

```
        </tr>
      </thead>
      <tbody>
        @foreach ($posts as $post)
          <tr>
            <td></td>
            <td></td>
            <td></td> <td>
              <a href="" class="btn btn-primary">Edit</a>
              <form action="" method="POST" class="d-inline">
                @csrf
                @method('DELETE')
                <button type="submit" class="btn btn-danger">Delete</button>
              </form>
            </td>
          </tr>
        @endforeach
      </tbody>
    </table>

    <a href="" class="btn btn-success">Create Post</a>
@endsection
```