Create Functions

Next, create the functions that store, index, update, destroy, create, show, and edit the data.

You can add them to the file **app/Http/Controller/PostController.php** shown below.

1. The `store` Function

The `store` function adds a post to the database.

Scroll to the `store` function and add the following code inside the empty curly braces:

```
$request->validate([
  'title' => 'required|max:255',
  'body' => 'required',
]);
Post::create($request->all());
return redirect()->route('posts.index')
  ->with('success','Post created successfully.');
```

This code takes an object containing the post's title and body, validates the data, adds a new post to the database if the data is valid, and redirects the user to the homepage with a success message.

2. The index Function

The `index` function fetches all the posts from the database and sends the data to the **posts.index** page.

3. The update Function

The `update` function contains the `id` of the post to update, the new post `title`, and the body. After validating the data, it finds the post with the same `id`. If found, the `update` function updates the post in the database with the new `title` and body. Then, it redirects the user to the homepage with a success message.

4. The destroy Function

The `destroy` function finds a post with the given `id` and deletes it from the database, then redirects the user to the homepage with a success message.

The functions above are the functions used to CRUD posts from the database. However, you must define more functions inside the controller to render the necessary pages in **resources/views/posts/**.

5. The create Function

The `create` function renders the **resources/views/posts/create.blade.php** page, which contains the form for adding posts to the database.

6. The show Function

The `show` function finds a post with the given `id` in the database and renders the **resources/views/posts/show.blade.php** file with the post.

7. The edit Function

The `edit` function finds a post with the given `id` in the database and renders the **resources/views/posts/edit.blade.php** file with the post details inside a form.

Set Up the Model

The `Post` model interacts with the **posts** table in the database.

1. To create a model with Artisan, run:

```
php artisan make:model Post
```

This code creates a **Post.php** file inside the **App/Models** folder.

2. Create a `fillable` array. Add the following code inside the `Post` class and below the `use HasFactory;` line

```
protected $fillable = [
  'title',
  'body',
];
```

This code creates a `fillable` array that allows you to add items to the database from your Laravel application.

3. Connect the `Post` model to the **PostController.php** file.
   Open **PostController.php** and add the line below under `use Illuminate\Http\Request;`. It looks like:

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

The **PostController.php** file should now look like this:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Post;
class PostController extends Controller
{
  /**
   * Display a listing of the resource.
   *
   * @return \Illuminate\Http\Response
   */
  public function index()
  {
    $posts = Post::all();
    return view('posts.index', compact('posts'));
  }
  /**
   * Store a newly created resource in storage.
   *
   * @param  \Illuminate\Http\Request  $request
   * @return \Illuminate\Http\Response
   */
  public function store(Request $request)
  {
    $request->validate([
      'title' => 'required|max:255',
      'body' => 'required',
```

```
    ]);
    Post::create($request->all());
    return redirect()->route('posts.index')
      ->with('success', 'Post created successfully.');
  }
  /**
   * Update the specified resource in storage.
   *
   * @param  \Illuminate\Http\Request  $request
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function update(Request $request, $id)
  {
    $request->validate([
      'title' => 'required|max:255',
      'body' => 'required',
    ]);
    $post = Post::find($id);
    $post->update($request->all());
    return redirect()->route('posts.index')
      ->with('success', 'Post updated successfully.');
  }
  /**
   * Remove the specified resource from storage.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function destroy($id)
  {
    $post = Post::find($id);
    $post->delete();
    return redirect()->route('posts.index')
      ->with('success', 'Post deleted successfully');
  }
  // routes functions
  /**
   * Show the form for creating a new post.
   *
```

```
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('posts.create');
    }
    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $post = Post::find($id);
        return view('posts.show', compact('post'));
    }
    /**
     * Show the form for editing the specified post.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $post = Post::find($id);
        return view('posts.edit', compact('post'));
    }
}
```

Add Routes

After creating the controller functions and the `Post` model, you must add routes for your controller functions.

1. Open **routes/web.php** and delete the boilerplate route that the application generated. Replace it with the code below to connect the controller functions to their respective routes:

```
// returns the home page with all posts
```

```
Route::get('/', PostController::class .'@index')->name('posts.index');
// returns the form for adding a post
Route::get('/posts/create', PostController::class . '@create')-
>name('posts.create');
// adds a post to the database
Route::post('/posts', PostController::class .'@store')-
>name('posts.store');
// returns a page that shows a full post
Route::get('/posts/{post}', PostController::class .'@show')-
>name('posts.show');
// returns the form for editing a post
Route::get('/posts/{post}/edit', PostController::class .'@edit')-
>name('posts.edit');
// updates a post
Route::put('/posts/{post}', PostController::class .'@update')-
>name('posts.update');
// deletes a post
Route::delete('/posts/{post}', PostController::class .'@destroy')-
>name('posts.destroy');
```

2. To connect the routes, open **app/Http/Controllers/PostController.php** and add the line below under the line `use Illuminate\Support\Facades\Route;`:

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PostController;
```

The **routes/web.php** file should now look like this:

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PostController;
// returns the home page with all posts
Route::get('/', PostController::class .'@index')->name('posts.index');
// returns the form for adding a post
Route::get('/posts/create', PostController::class . '@create')-
>name('posts.create');
// adds a post to the database
Route::post('/posts', PostController::class .'@store')-
>name('posts.store');
// returns a page that shows a full post
```

```
Route::get('/posts/{post}', PostController::class .'@show')-
>name('posts.show');
// returns the form for editing a post
Route::get('/posts/{post}/edit', PostController::class .'@edit')-
>name('posts.edit');
// updates a post
Route::put('/posts/{post}', PostController::class .'@update')-
>name('posts.update');
// deletes a post
Route::delete('/posts/{post}', PostController::class .'@destroy')-
>name('posts.destroy');
```

Generate Blade Files

Now that you have the routes, you can create the Laravel Blade files. Before using Artisan to generate the Blade files, create the `make:view` command, with which you can generate **blade.php** files.

1. Run the following code in the CLI to create a **MakeViewCommand** command file inside the **app/Console/Commands** folder:

```
php artisan make:command MakeViewCommand
```

2. Create a command for generating **.blade.php** files from the CLI by replacing the code in the **MakeViewCommand** file with the following:

```
<?php
namespace App\Console\Commands;
use Illuminate\Console\Command;
use File;
class MakeViewCommand extends Command
{
  /**
   * The name and signature of the console command.
   *
   * @var string
   */
  protected $signature = 'make:view {view}';
  /**
   * The console command description.
```

```
     *
     * @var string
     */
    protected $description = 'Create a new blade template.';
    /**
     * Execute the console command.
     *
     * @return mixed
     */
    public function handle()
    {
      $view = $this->argument('view');
      $path = $this->viewPath($view);
      $this->createDir($path);
      if (File::exists($path))
      {
          $this->error("File {$path} already exists!");
          return;
      }
      File::put($path, $path);
      $this->info("File {$path} created.");
    }
    /**
     * Get the view full path.
     *
     * @param string $view
     *
     * @return string
     */
    public function viewPath($view)
    {
      $view = str_replace('.', '/', $view) . '.blade.php';
      $path = "resources/views/{$view}";
      return $path;
    }
    /**
     * Create a view directory if it does not exist.
     *
     * @param $path
     */
```

```
  public function createDir($path)
  {
    $dir = dirname($path);
    if (!file_exists($dir))
    {
        mkdir($dir, 0777, true);
    }
  }
}
```

Create a Homepage

Next, create your homepage. The homepage is the **index.blade.php** file, which lists all the posts.

1. To create the homepage, run:

```
php artisan make:view posts.index
```

This creates a **posts** folder inside the **/resources/views** folder and an **index.blade.php** file underneath it. The resulting path is **/resources/views/posts/index.blade.php**.

2. Add the following code inside the **index.blade.php** file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">
  <title>Posts</title>
</head>
<body>
```

```
  <nav class="navbar navbar-expand-lg navbar-light bg-warning">
    <div class="container-fluid">
      <a class="navbar-brand h1" href={{ route('posts.index')
}}>CRUDPosts</a>
      <div class="justify-end ">
        <div class="col ">
          <a class="btn btn-sm btn-success" href={{
route('posts.create') }}>Add Post</a>
        </div>
      </div>
    </div>
  </nav>
  <div class="container mt-5">
    <div class="row">
      @foreach ($posts as $post)
        <div class="col-sm">
          <div class="card">
            <div class="card-header">
              <h5 class="card-title">{{ $post->title }}</h5>
            </div>
            <div class="card-body">
              <p class="card-text">{{ $post->body }}</p>
            </div>
            <div class="card-footer">
              <div class="row">
                <div class="col-sm">
                  <a href="{{ route('posts.edit', $post->id) }}"
            class="btn btn-primary btn-sm">Edit</a>
                </div>
                <div class="col-sm">
                    <form action="{{ route('posts.destroy', $post->id)
}}" method="post">
                      @csrf
                      @method('DELETE')
                      <button type="submit" class="btn btn-danger btn-
sm">Delete</button>
                    </form>
                </div>
              </div>
            </div>
```

```
          </div>
        </div>
      @endforeach
    </div>
  </div>
</body>
</html>
```

The code above creates a simple HTML page that uses Bootstrap for styling. It establishes a navigation bar and a grid template that lists all the posts in the database with details and two action buttons — edit and delete — using the `@foreach` Blade helper.

The **Edit** button takes a user to the **Edit post** page, where they can edit the post. The **Delete** button deletes the post from the database using `{{ route('posts.destroy', $post->id) }}` with a `DELETE` method.

Note: The navbar code for all the files is the same as the previous file.

3. Create the **create.blade.php** page. The Blade file called **create** adds posts to the database. Use the following command to generate the file:

```
php artisan make:view posts.create
```

This generates a **create.blade.php** file inside the **/resources/views/posts** folder.

4. Add the following code to the **create.blade.php** file:

```
// same as the previous file. Add the following after the nav tag and
before the closing body tag.
<div class="container h-100 mt-5">
  <div class="row h-100 justify-content-center align-items-center">
    <div class="col-10 col-md-8 col-lg-6">
      <h3>Add a Post</h3>
      <form action="{{ route('posts.store') }}" method="post">
        @csrf
        <div class="form-group">
          <label for="title">Title</label>
```

```
          <input type="text" class="form-control" id="title"
name="title" required>
        </div>
        <div class="form-group">
          <label for="body">Body</label>
          <textarea class="form-control" id="body" name="body"
rows="3" required></textarea>
        </div>
        <br>
        <button type="submit" class="btn btn-primary">Create
Post</button>
      </form>
    </div>
  </div>
</div>
```

The code above creates a form with `title` and body fields and a `submit` button for adding a post to the database through the `{{ route('posts.store') }}` action with a POST method.

5. Create the **Edit post** page to edit posts in the database. Use the following command to generate the file:

```
php artisan make:view posts.edit
```

This creates an **edit.blade.php** file inside the **/resources/views/posts** folder.

6. Add the following code to the **edit.blade.php** file:

```
<div class="container h-100 mt-5">
  <div class="row h-100 justify-content-center align-items-center">
    <div class="col-10 col-md-8 col-lg-6">
      <h3>Update Post</h3>
      <form action="{{ route('posts.update', $post->id) }}"
method="post">
        @csrf
        @method('PUT')
        <div class="form-group">
          <label for="title">Title</label>
```

```
          <input type="text" class="form-control" id="title"
name="title"
            value="{{ $post->title }}" required>
        </div>
        <div class="form-group">
          <label for="body">Body</label>
          <textarea class="form-control" id="body" name="body"
rows="3" required>{{ $post->body }}</textarea>
        </div>
        <button type="submit" class="btn mt-3 btn-primary">Update
Post</button>
      </form>
    </div>
  </div>
</div>
```

The code above creates a form with `title` and body fields and a submit button for editing a post with the specified `id` in the database through the `{{ route('posts.update') }}` action with a PUT method.

7. Then, restart your application server using the code below:

```
php artisan serve
```

Visit `http://127.0.0.1:8000` on your browser to view your new blog. Click the **Add Post** button to add new posts.