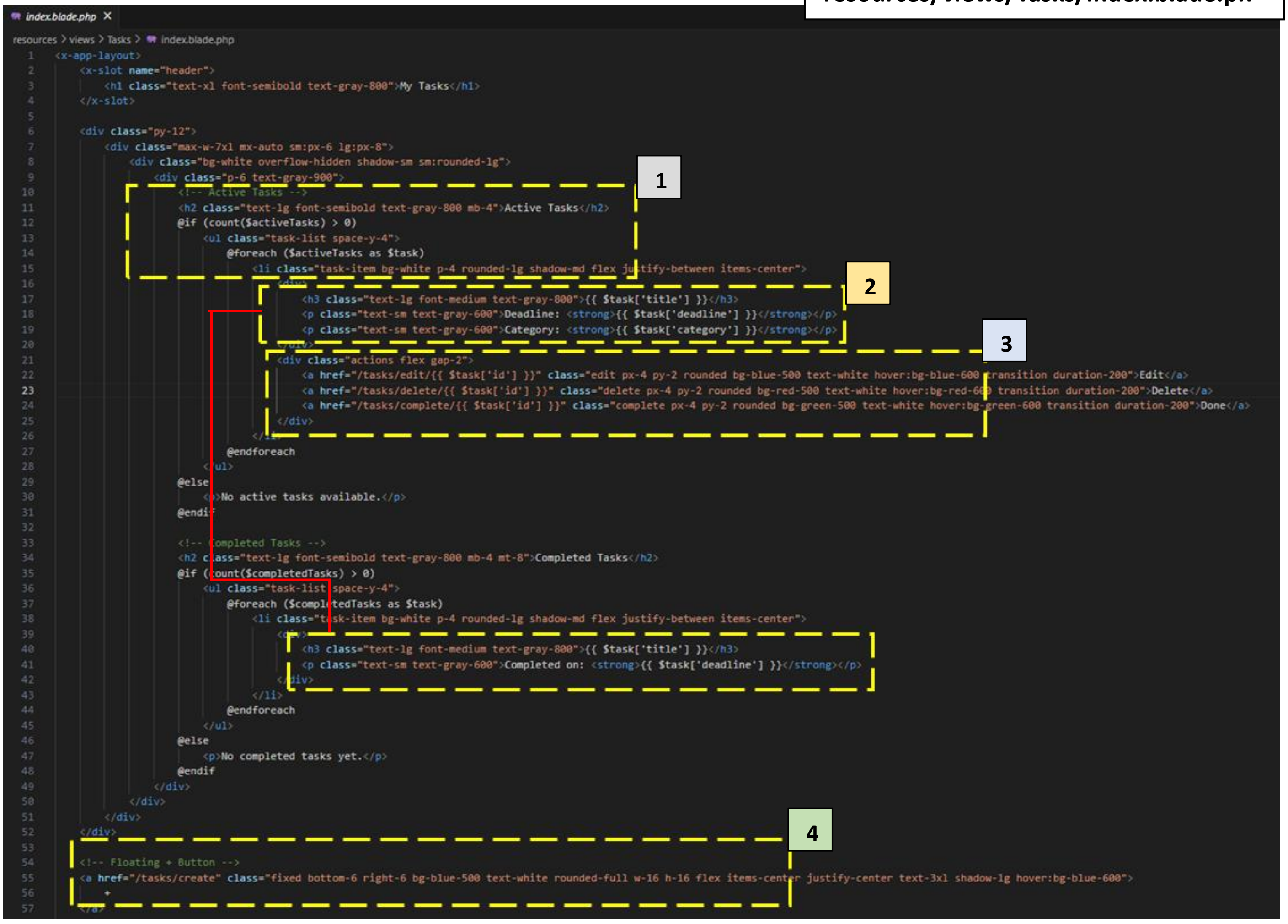
ANGELA B. ABAD
BSIT-3C

WEB DEVELOPMENT

DOCUMENTATION: Laboratory 2 – Populating Pages

I. Blade Files

resources/views/Tasks/index.blade.ph



```
<x-app-layout>
    <x-slot name="header">
        <h1 class="text-xl font-semibold text-gray-800">My Tasks</h1>
    </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-sm sm:rounded-lg">
                <div class="p-6 text-gray-900">
                    <!-- Active Tasks -->
                    <h2 class="text-lg font-semibold text-gray-800 mb-4">Active Tasks</h2>
                    @if (count($activeTasks) > 0)
                        <ul class="task-list space-y-4">
                            @foreach ($activeTasks as $task)
                                <li class="task-item bg-white p-4 rounded-lg shadow-md flex justify-between items-center">
                                    <div>
                                        <h3 class="text-lg font-medium text-gray-800">{{ $task['title'] }}</h3>
                                        <p class="text-sm text-gray-600">Deadline: <strong>{{ $task['deadline'] }}</strong></p>
                                        <p class="text-sm text-gray-600">Category: <strong>{{ $task['category'] }}</strong></p>
                                    </div>
                                    <div class="actions flex gap-2">
                                        <a href="/tasks/edit/{{ $task['id'] }}" class="edit px-4 py-2 rounded bg-blue-500 text-white hover:bg-blue-600 transition duration-200">Edit</a>
                                        <a href="/tasks/delete/{{ $task['id'] }}" class="delete px-4 py-2 rounded bg-red-500 text-white hover:bg-red-600 transition duration-200">Delete</a>
                                        <a href="/tasks/complete/{{ $task['id'] }}" class="complete px-4 py-2 rounded bg-green-500 text-white hover:bg-green-600 transition duration-200">Done</a>
                                    </div>
                                </li>
                            @endforeach
                        </ul>
                    @else
                        <p>No active tasks available.</p>
                    @endif

                    <!-- Completed Tasks -->
                    <h2 class="text-lg font-semibold text-gray-800 mb-4 mt-8">Completed Tasks</h2>
                    @if (count($completedTasks) > 0)
                        <ul class="task-list space-y-4">
                            @foreach ($completedTasks as $task)
                                <li class="task-item bg-white p-4 rounded-lg shadow-md flex justify-between items-center">
                                    <div>
                                        <h3 class="text-lg font-medium text-gray-800">{{ $task['title'] }}</h3>
                                        <p class="text-sm text-gray-600">Completed on: <strong>{{ $task['deadline'] }}</strong></p>
                                    </div>
                                </li>
                            @endforeach
                        </ul>
                    @else
                        <p>No completed tasks yet.</p>
                    @endif
                </div>
            </div>
        </div>
    </div>

    <!-- Floating + Button -->
    <a href="/tasks/create" class="fixed bottom-6 right-6 bg-blue-500 text-white rounded-full w-16 h-16 flex items-center justify-center text-3xl shadow-lg hover:bg-blue-600">
        +
    </a>
```

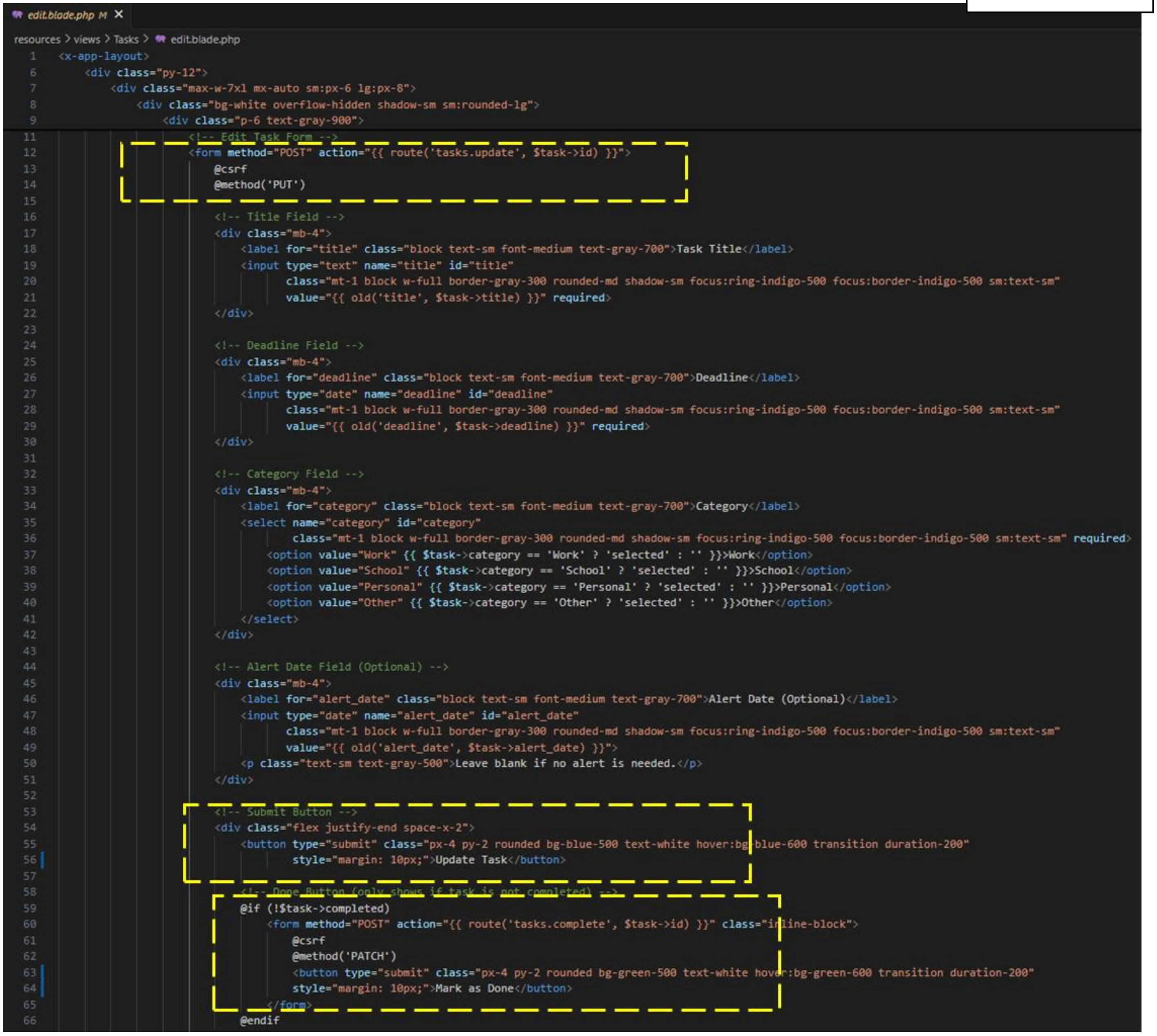
Task display logic: The logic for displaying tasks involves checking the ***$activeTasks*** array. If it's empty, a placeholder message is shown. Otherwise, it loops through the array to render each task. **1**

Dynamic Task Information: Dynamically presents task details, such as the title, deadline, and category, utilizing Blade syntax. **2**

Action Buttons (Edit, Delete, Complete): Includes links for editing, deleting, or marking tasks as complete, with each action tied to the specific task's ID. **3**

Floating Button: **Purpose**: This floating button serves as a quick-access feature for creating a new task. By clicking the button, the user is directed to a form or page to input details for the new task. **4**

edit.blade.php



edit.blade.php M ×

resources > views > Tasks > edit.blade.php

```
<x-app-layout>
    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
                <div class="p-6 text-gray-900">
                    <!-- Edit Task Form -->
                    <form method="POST" action="{{ route('tasks.update', $task->id) }}">
                        @csrf
                        @method('PUT')

                        <!-- Title Field -->
                        <div class="mb-4">
                            <label for="title" class="block text-sm font-medium text-gray-700">Task Title</label>
                            <input type="text" name="title" id="title"
                                   class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
                                   value="{{ old('title', $task->title) }}" required>
                        </div>

                        <!-- Deadline Field -->
                        <div class="mb-4">
                            <label for="deadline" class="block text-sm font-medium text-gray-700">Deadline</label>
                            <input type="date" name="deadline" id="deadline"
                                   class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
                                   value="{{ old('deadline', $task->deadline) }}" required>
                        </div>

                        <!-- Category Field -->
                        <div class="mb-4">
                            <label for="category" class="block text-sm font-medium text-gray-700">Category</label>
                            <select name="category" id="category"
                                    class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm" required>
                                <option value="Work" {{ $task->category == 'Work' ? 'selected' : '' }}>Work</option>
                                <option value="School" {{ $task->category == 'School' ? 'selected' : '' }}>School</option>
                                <option value="Personal" {{ $task->category == 'Personal' ? 'selected' : '' }}>Personal</option>
                                <option value="Other" {{ $task->category == 'Other' ? 'selected' : '' }}>Other</option>
                            </select>
                        </div>

                        <!-- Alert Date Field (Optional) -->
                        <div class="mb-4">
                            <label for="alert_date" class="block text-sm font-medium text-gray-700">Alert Date (Optional)</label>
                            <input type="date" name="alert_date" id="alert_date"
                                   class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
                                   value="{{ old('alert_date', $task->alert_date) }}">
                            <p class="text-sm text-gray-500">Leave blank if no alert is needed.</p>
                        </div>

                        <!-- Submit Button -->
                        <div class="flex justify-end space-x-2">
                            <button type="submit" class="px-4 py-2 rounded bg-blue-500 text-white hover:bg-blue-600 transition duration-200"
                                    style="margin: 10px;">Update Task</button>

                            <!-- Done Button (only shows if task is not completed) -->
                            @if (!$task->completed)
                                <form method="POST" action="{{ route('tasks.complete', $task->id) }}" class="inline-block">
                                    @csrf
                                    @method('PATCH')
                                    <button type="submit" class="px-4 py-2 rounded bg-green-500 text-white hover:bg-green-600 transition duration-200"
                                    style="margin: 10px;">Mark as Done</button>
                                </form>
                            @endif
```

Form Setup:

• Sends an HTTP POST request to the tasks.update route, including the task's ID to update a particular task.

• Uses @csrf for security and @method('PUT') to indicate the HTTP PUT method, commonly used for updating resources.

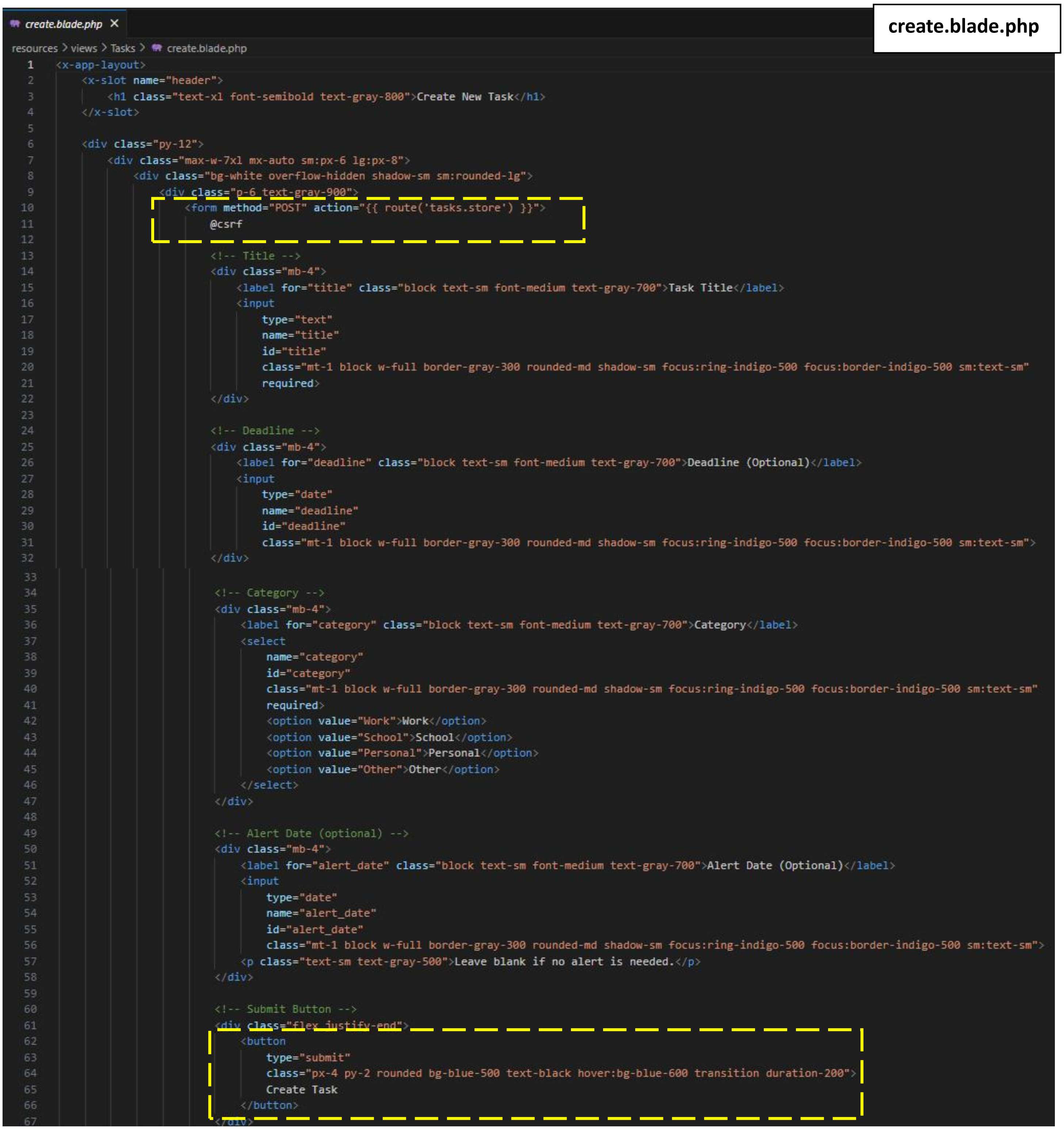
Update Task Button: Submits the form to save any changes made to the task, designed for easy visibility and use.

Mark as Done Button (Conditional):

• Shows a button to mark the task as complete if it hasn't been completed yet.

• Sends a PATCH request to the tasks.complete route, passing the task's ID.

create.blade.php



create.blade.php ×

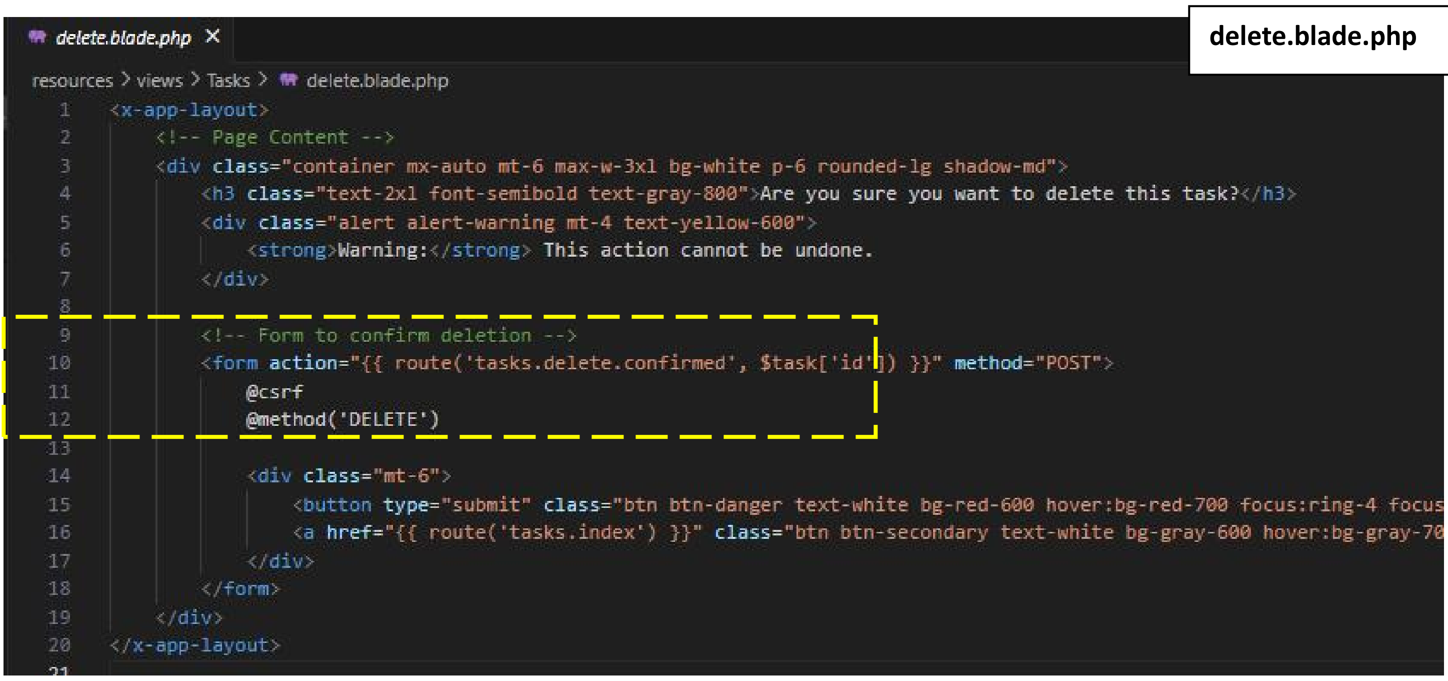
resources > views > Tasks > create.blade.php

```
<x-app-layout>
    <x-slot name="header">
        <h1 class="text-xl font-semibold text-gray-800">Create New Task</h1>
    </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-sm sm:rounded-lg">
                <div class="p-6 text-gray-900">
                    <form method="POST" action="{{ route('tasks.store') }}">
                        @csrf

                        <!-- Title -->
                        <div class="mb-4">
                            <label for="title" class="block text-sm font-medium text-gray-700">Task Title</label>
                            <input
                                type="text"
                                name="title"
                                id="title"
                                class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
                                required>
                        </div>

                        <!-- Deadline -->
                        <div class="mb-4">
                            <label for="deadline" class="block text-sm font-medium text-gray-700">Deadline (Optional)</label>
                            <input
                                type="date"
                                name="deadline"
                                id="deadline"
                                class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm">
                        </div>

                        <!-- Category -->
                        <div class="mb-4">
                            <label for="category" class="block text-sm font-medium text-gray-700">Category</label>
                            <select
                                name="category"
                                id="category"
                                class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
                                required>
                                <option value="Work">Work</option>
                                <option value="School">School</option>
                                <option value="Personal">Personal</option>
                                <option value="Other">Other</option>
                            </select>
                        </div>

                        <!-- Alert Date (optional) -->
                        <div class="mb-4">
                            <label for="alert_date" class="block text-sm font-medium text-gray-700">Alert Date (Optional)</label>
                            <input
                                type="date"
                                name="alert_date"
                                id="alert_date"
                                class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm">
                            <p class="text-sm text-gray-500">Leave blank if no alert is needed.</p>
                        </div>

                        <!-- Submit Button -->
                        <div class="flex justify-end">
                            <button
                                type="submit"
                                class="px-4 py-2 rounded bg-blue-500 text-black hover:bg-blue-600 transition duration-200">
                                Create Task
                            </button>
                        </div>
```

Form Setup: Configures the form to submit data using the POST method to the tasks.store route, with the @csrf directive ensuring protection against cross-site request forgery.

The form enables users to create a new task by providing details like the title, optional deadline, category, and alert date. All fields are styled for clear visibility and ease of use, with data securely sent via a POST request to the designated route.

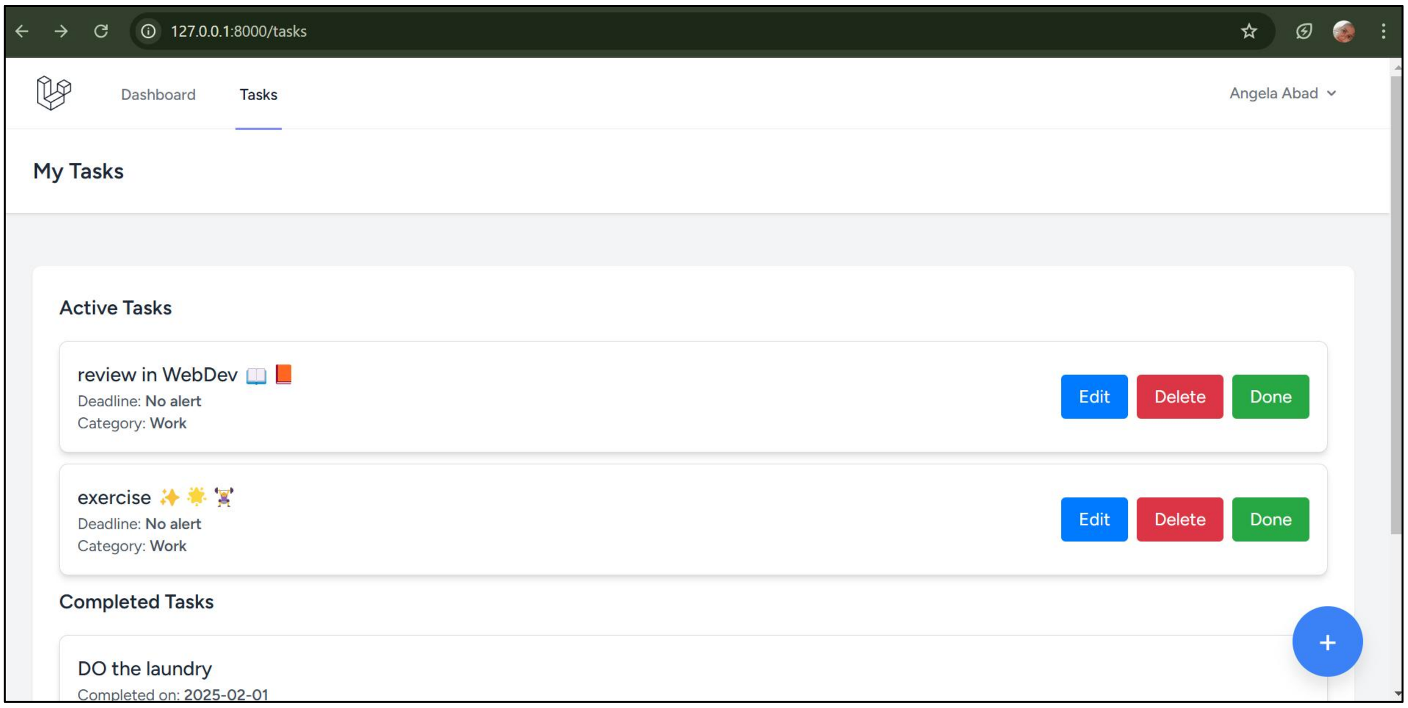
delete.blade.php



```
<x-app-layout>
    <!-- Page Content -->
    <div class="container mx-auto mt-6 max-w-3xl bg-white p-6 rounded-lg shadow-md">
        <h3 class="text-2xl font-semibold text-gray-800">Are you sure you want to delete this task?</h3>
        <div class="alert alert-warning mt-4 text-yellow-600">
            <strong>Warning:</strong> This action cannot be undone.
        </div>

        <!-- Form to confirm deletion -->
        <form action="{{ route('tasks.delete.confirmed', $task['id']) }}" method="POST">
            @csrf
            @method('DELETE')

            <div class="mt-6">
                <button type="submit" class="btn btn-danger text-white bg-red-600 hover:bg-red-700 focus:ring-4 focus
                <a href="{{ route('tasks.index') }}" class="btn btn-secondary text-white bg-gray-600 hover:bg-gray-70
            </div>
        </form>
    </div>
</x-app-layout>
```

The **delete method** manages the process of permanently removing a task from the system, executing the action only after the user confirms the deletion.

II. Rendered Pages



My Tasks

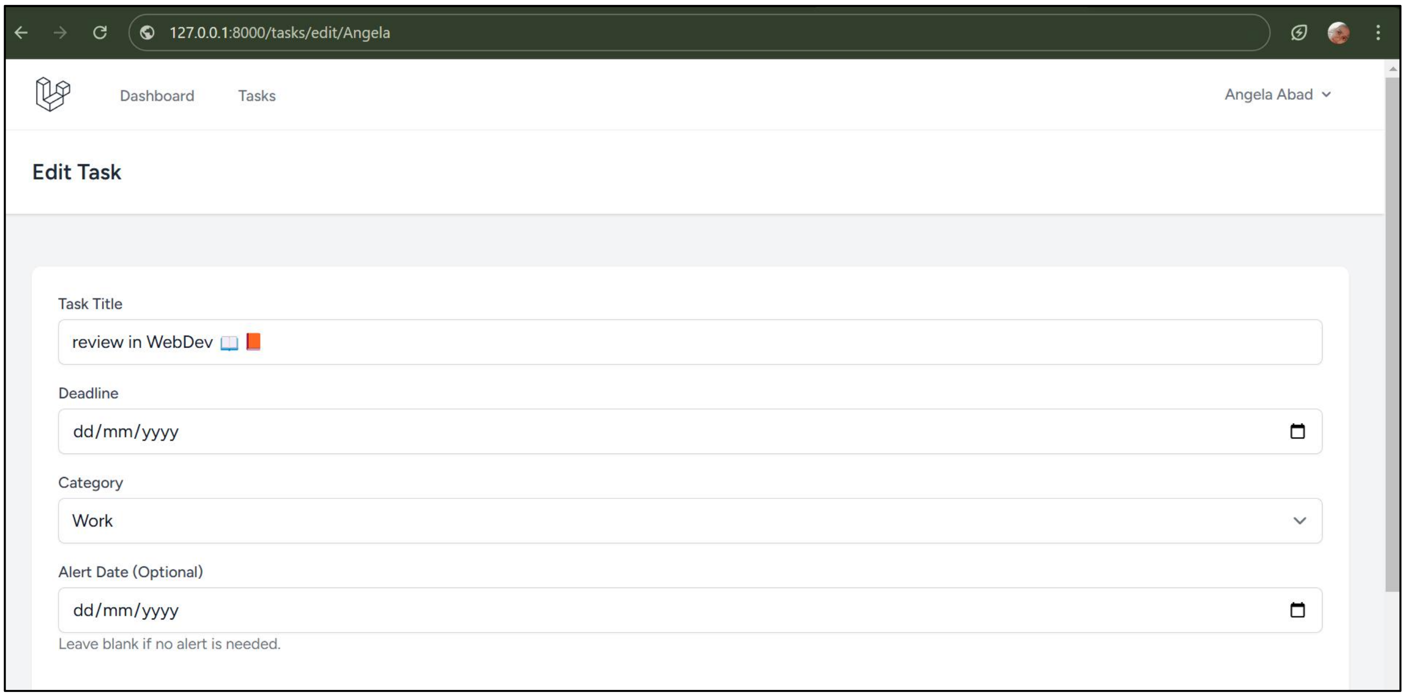
Active Tasks

review in WebDev 📖 📕
Deadline: No alert
Category: Work
Edit Delete Done

exercise ✨ 🌟 🏋
Deadline: No alert
Category: Work
Edit Delete Done

Completed Tasks

DO the laundry
Completed on: 2025-02-01


127.0.0.1:8000/tasks/edit/Angela

Edit Task

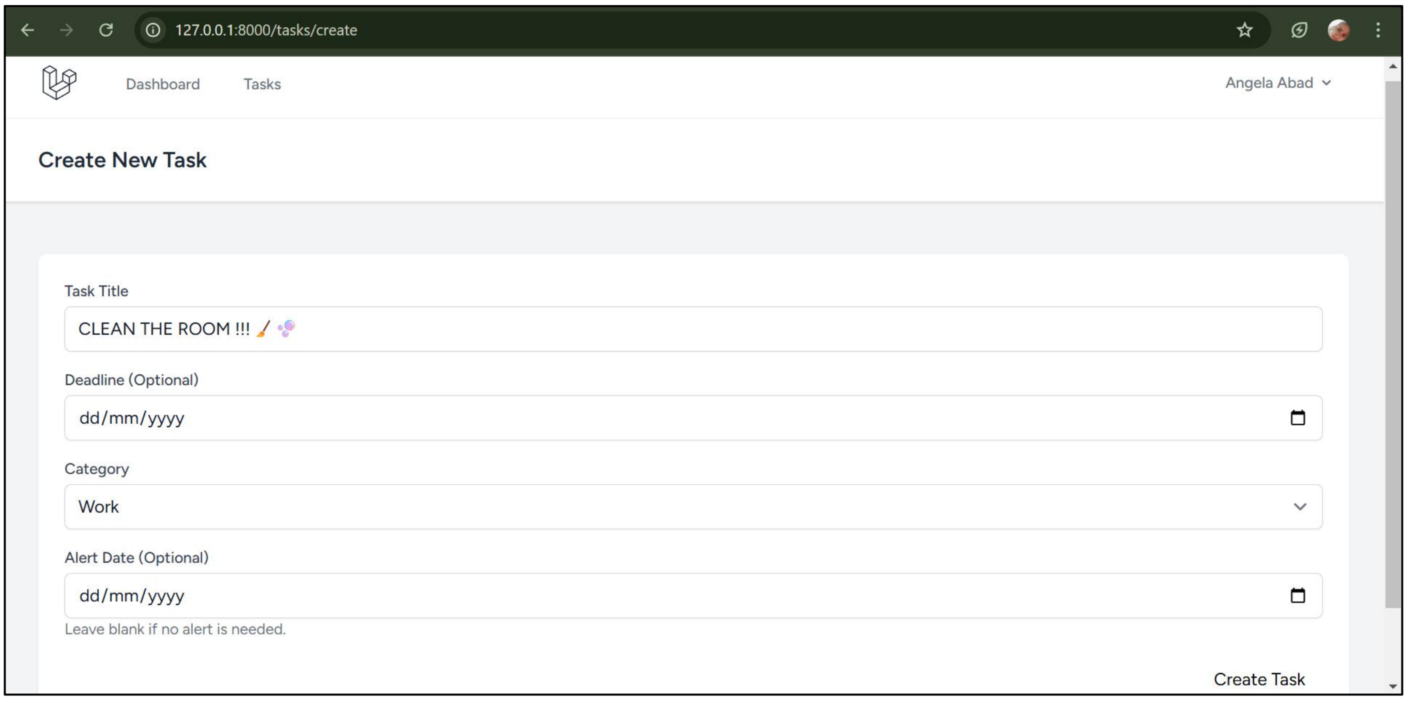
Task Title: review in WebDev 📖 📕

Deadline: dd/mm/yyyy

Category: Work

Alert Date (Optional): dd/mm/yyyy

Leave blank if no alert is needed.


127.0.0.1:8000/tasks/create

Create New Task

Task Title: CLEAN THE ROOM !!! 🧹 🫧

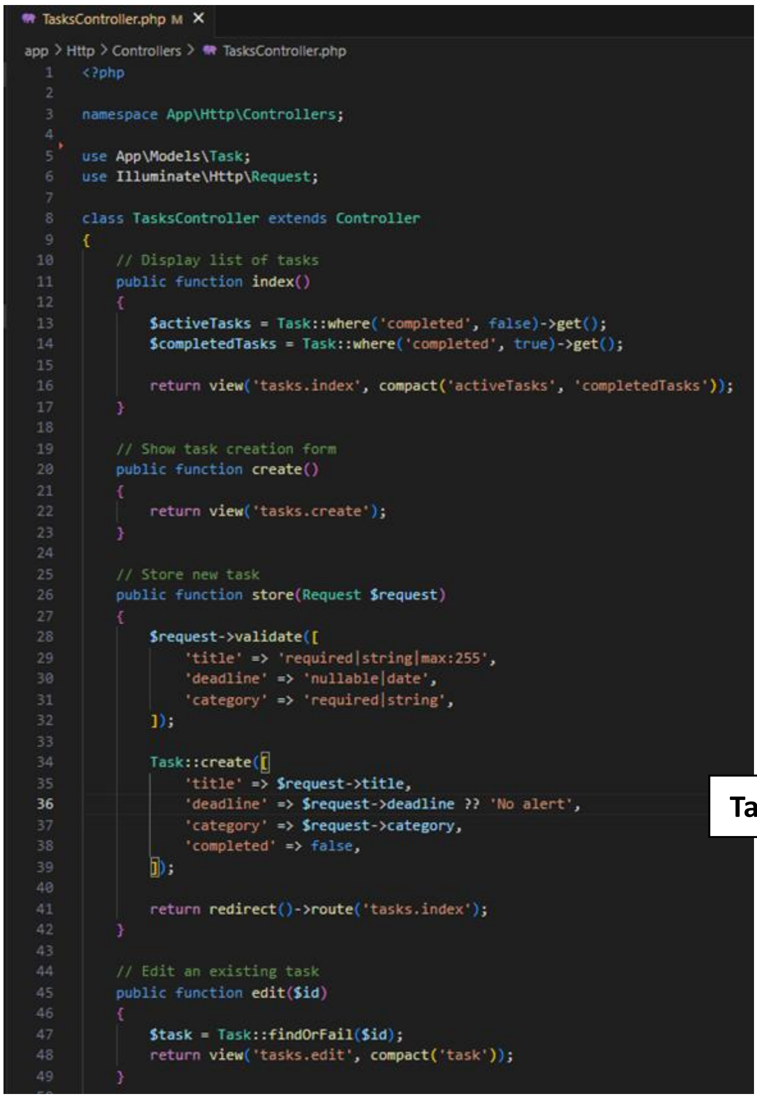
Deadline (Optional): dd/mm/yyyy

Category: Work

Alert Date (Optional): dd/mm/yyyy

Leave blank if no alert is needed.

III. Controllers Logic



```php
<?php

namespace App\Http\Controllers;

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

class TasksController extends Controller
{
    // Display list of tasks
    public function index()
    {
        $activeTasks = Task::where('completed', false)->get();
        $completedTasks = Task::where('completed', true)->get();

        return view('tasks.index', compact('activeTasks', 'completedTasks'));
    }

    // Show task creation form
    public function create()
    {
        return view('tasks.create');
    }

    // Store new task
    public function store(Request $request)
    {
        $request->validate([
            'title' => 'required|string|max:255',
            'deadline' => 'nullable|date',
            'category' => 'required|string',
        ]);

        Task::create([
            'title' => $request->title,
            'deadline' => $request->deadline ?? 'No alert',
            'category' => $request->category,
            'completed' => false,
        ]);

        return redirect()->route('tasks.index');
    }

    // Edit an existing task
    public function edit($id)
    {
        $task = Task::findOrFail($id);
        return view('tasks.edit', compact('task'));
    }
```

1. index Method: Displays a list of tasks, categorized into active and completed tasks.
Logic: Fetches active tasks where completed = false from the database. Fetches completed tasks where completed = true from the database.
Returns: Renders the tasks.index Blade view with activeTasks and completedTasks variables.

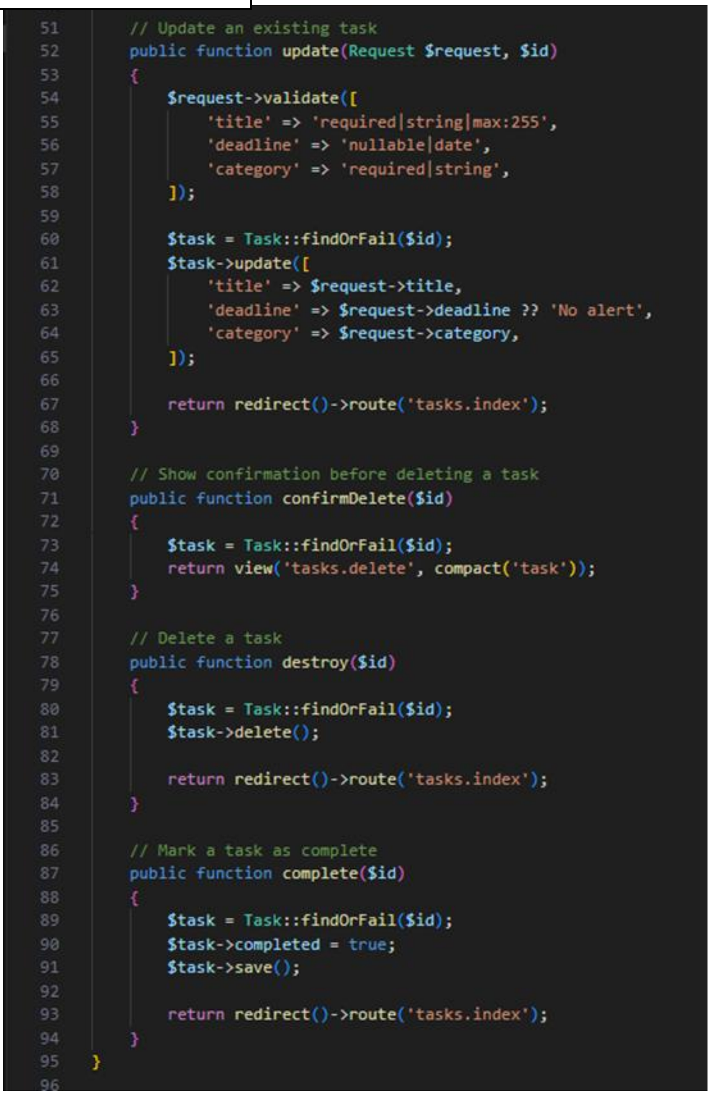
2. create Method: Displays a form for creating a new task.
Logic: Simply returns the tasks.create view.
Returns: The Blade view with the task creation form.

3. edit Method: Displays a form for editing an existing task.
Logic: Fetches the task by its $id using findOrFail() (throws an exception if the task is not found).
Returns: The tasks.edit view with the task to be edited.

TasksController.php

4. store Method: Saves a new task in the database based on user input from the task creation form.
Logic:
> Validates the request:
- <u>*title*</u> is required and must be a string of max 255 characters.
- <u>*deadline*</u> is optional and must be a valid date.
- <u>*category*</u> is required and must be a string.
> Creates a new task using Task::create().
- Defaults completed to false.
> Redirects the user back to the tasks.index page.

5. update Method: Updates an existing task's details in the database.
Logic:
> Validates the request:
- Same validation rules as the store method.
> Fetches the task using its $id.
> Updates the task with the provided data.
Returns: Redirects the user back to the tasks.index page.



```php
    // Update an existing task
    public function update(Request $request, $id)
    {
        $request->validate([
            'title' => 'required|string|max:255',
            'deadline' => 'nullable|date',
            'category' => 'required|string',
        ]);

        $task = Task::findOrFail($id);
        $task->update([
            'title' => $request->title,
            'deadline' => $request->deadline ?? 'No alert',
            'category' => $request->category,
        ]);

        return redirect()->route('tasks.index');
    }

    // Show confirmation before deleting a task
    public function confirmDelete($id)
    {
        $task = Task::findOrFail($id);
        return view('tasks.delete', compact('task'));
    }

    // Delete a task
    public function destroy($id)
    {
        $task = Task::findOrFail($id);
        $task->delete();

        return redirect()->route('tasks.index');
    }

    // Mark a task as complete
    public function complete($id)
    {
        $task = Task::findOrFail($id);
        $task->completed = true;
        $task->save();

        return redirect()->route('tasks.index');
    }
}
```

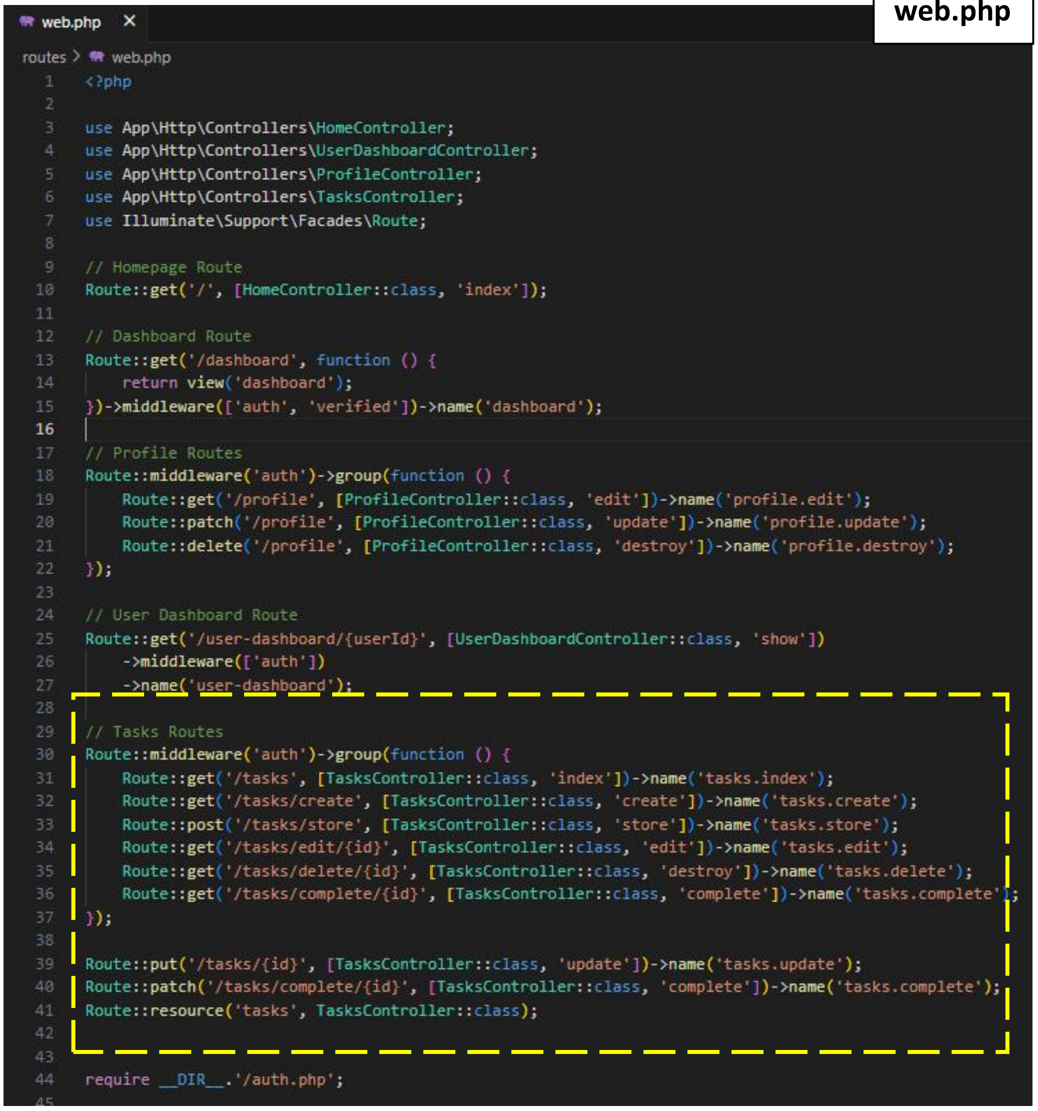
6. confirmDelete Method: Shows a confirmation view before deleting a task.
Logic: Fetches the task by its $id.
Returns: The tasks.delete view with the task to be deleted.

7. destroy Method: Deletes a task from the database.
Logic:
> Fetches the task by its $id.
> Deletes the task using the delete() method.
Returns: Redirects back to the tasks.index page.

8. complete Method: Marks a task as completed.
Logic:
> Fetches the task by its $id.
> Updates the completed attribute to true.
> Saves the task back to the database.
Returns: Redirects back to the tasks.index page.

IV. Routes

web.php



```php
<?php

use App\Http\Controllers\HomeController;
use App\Http\Controllers\UserDashboardController;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\TasksController;
use Illuminate\Support\Facades\Route;

// Homepage Route
Route::get('/', [HomeController::class, 'index']);

// Dashboard Route
Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

// Profile Routes
Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

// User Dashboard Route
Route::get('/user-dashboard/{userId}', [UserDashboardController::class, 'show'])
    ->middleware(['auth'])
    ->name('user-dashboard');

// Tasks Routes
Route::middleware('auth')->group(function () {
    Route::get('/tasks', [TasksController::class, 'index'])->name('tasks.index');
    Route::get('/tasks/create', [TasksController::class, 'create'])->name('tasks.create');
    Route::post('/tasks/store', [TasksController::class, 'store'])->name('tasks.store');
    Route::get('/tasks/edit/{id}', [TasksController::class, 'edit'])->name('tasks.edit');
    Route::get('/tasks/delete/{id}', [TasksController::class, 'destroy'])->name('tasks.delete');
    Route::get('/tasks/complete/{id}', [TasksController::class, 'complete'])->name('tasks.complete');
});

Route::put('/tasks/{id}', [TasksController::class, 'update'])->name('tasks.update');
Route::patch('/tasks/complete/{id}', [TasksController::class, 'complete'])->name('tasks.complete');
Route::resource('tasks', TasksController::class);


require __DIR__.'/auth.php';
```

GET /tasks (tasks.index)
Displays a list of all tasks categorized as active or completed.

GET /tasks/create (tasks.create)
Shows a form for creating a new task.

POST /tasks/store (tasks.store)
Handles the form submission to store a new task in the database.

GET /tasks/edit/{id} (tasks.edit)
Displays a form to edit an existing task.

GET /tasks/delete/{id}
(tasks.delete**)**
Deletes a specific task from the database.

GET /tasks/complete/{id}
(tasks.complete**)**
Marks a specific task as completed.

PUT /tasks/{id} (tasks.update)
Updates an existing task's details after form submission.

PATCH /tasks/complete/{id}
(tasks.complete**)**
Marks a task as completed using the PATCH method.

Route::resource('tasks', TasksController::class)
Automatically generates standard CRUD routes for managing tasks.

LEARNINGS:

Simulating a Database

I learned how to simulate a database using Laravel's Eloquent ORM without needing an actual database connection. This made it easier for me to work on features like creating, editing, and deleting tasks, allowing me to focus more on the logic and functionality instead of worrying about setting up a database. I also discovered how useful the compact function is for passing variables to Blade views. For example, in the index method, I used compact*('activeTasks', 'completedTasks')*, which helped simplify the process of sending task data to the view.