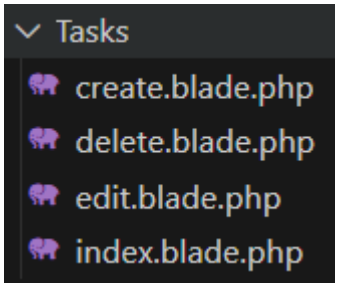
I. BLADE FILES

resources/views/Tasks



index.blade.php

```
resources > views > Tasks > index.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">
                    <!-- 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">
```

Task display logic: Checks if there are tasks in the $activeTasks array. If none, displays a placeholder message. Otherwise, iterates through and

Action Buttons (Edit, Delete, Complete): Provides links to edit, delete, or mark tasks as complete. Each action uses the id of the task.

Dynamic Task Information: Displays task details like title, deadline, and category dynamically using Blade syntax.

Action Buttons (Edit, Delete, Complete): Provides links to edit, delete, or mark tasks as complete. Each action uses the id of the task.

create.blade.php

resources > views > Tasks > create.blade.php

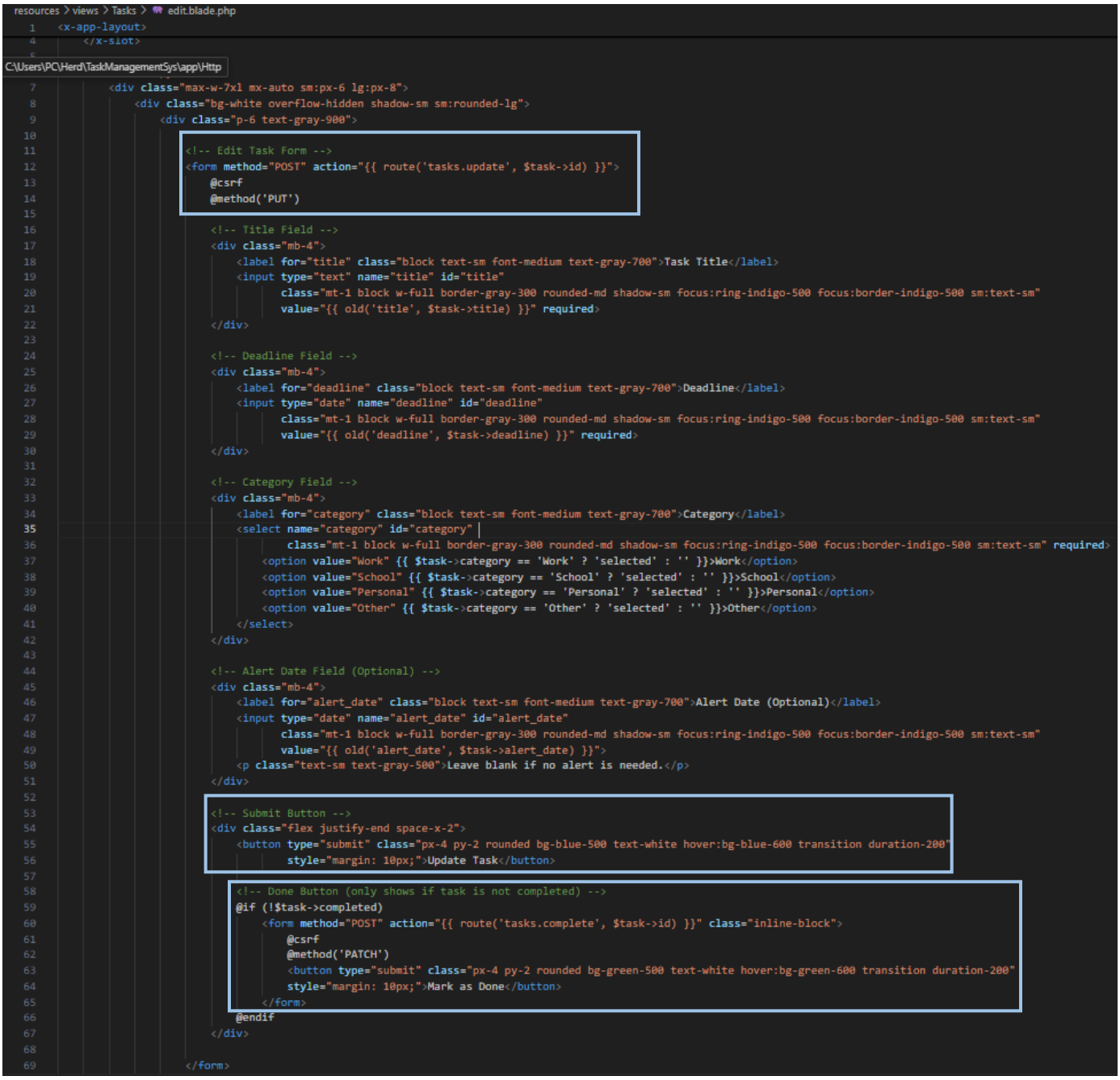
```blade
<x-app-layout>
    <x-slot name="header">
        <h1 class="text-xl font-semibold text-gray-800">Create New Task</h1>
    </x-slot>

    <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:

Sets up the form to submit data via the POST method to the tasks.store route. The @csrf directive protects against cross-site request forgery.

The form allows users to create a new task by entering details such as the title, optional deadline, category, and alert date, with all fields styled for clarity and usability, and data securely submitted via a **POST request to the specified route.**

Edit.blade.php



```
resources > views > Tasks > edit.blade.php
<x-app-layout>
    </x-slot>
C:\Users\PC\Herd\TaskManagementSys\app\Http
        <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
                        </div>

                    </form>
```

Form Setup:

• Initiates an HTTP POST request to the tasks.update route, passing the task's id to update a specific task.

• Includes @csrf for security and @method('PUT') to specify the HTTP PUT method, which is typically used for updating resources.

Update Task Button:

Submits the form to save changes made to the task and styled for visibility and ease of use.

Mark as Done Button (Conditional):

• Displays a button to mark the task as completed if the task is not already completed.

• Sends a PATCH request to the tasks.complete route with the task's id.

delete.blade.php

resources > views > Tasks > delete.blade.php

```blade
<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:ring-red-300 font-medium rounded-lg text-sm px-5 py-2.5 text-center">Yes, Delete</button>
                <a href="{{ route('tasks.index') }}" class="btn btn-secondary text-white bg-gray-600 hover:bg-gray-700 focus:ring-4 focus:ring-gray-300 font-medium rounded-lg text-sm px-5 py-2.5 text-center ml-4">No, Cancel</a>
            </div>
        </form>
    </div>
</x-app-layout>
```

The delete method handles the logic for permanently removing a task from the system after the user confirms the deletion.

II. **CONTROLLER LOGIC**

TaskController.php

app > Http > Controllers > TasksController.php

```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'));
    }

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

app > Http > Controllers > TasksController.php

```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');
    }
}
```

index()

Purpose: Fetches and organizes tasks into active and completed categories for the task list view.

Key Logic:

- Retrieves tasks marked as active (completed = false) and completed (completed = true) from the simulated database.
- Passes the data to the tasks.index view using compact.

create()

Purpose: Displays the form for creating a new task.

Key Logic: Directly returns the tasks.create view without any data since it is for input only

store(Request $request)

Purpose: Validates and saves a new task to the simulated database.

Key Logic: Validates form input for required fields like title and category

edit($id)

Purpose: Retrieves a task for editing and displays the edit form.

Key Logic: Finds the task by ID using Task::findOrFail**.**

complete($id)

Purpose: Marks a task as completed.

Key Logic: Finds the task by ID and sets its completed attribute to true

confirmDelete($id)

Purpose: Displays a confirmation page before deleting a task.

Key Logic: Retrieves the task to confirm the details before deletion.

update(Request $request, $id)

Purpose: Updates an existing task with the modified data.

Key Logic: Validates the input fields to ensure data integrity.

destroy($id)

Purpose: Permanently deletes a task from the simulated database.

Key Logic: Finds the task by ID using Task::findOrFail and deletes it using the delete method

III. **ROUTES**

web.php

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';
```

Task Listing Route

Functionality: Displays a list of tasks, separated into active and completed categories.

Parameter Handling: No parameters required**.**

Task Creation Form Route

Functionality: Shows a form for creating a new task.

Parameter Handling: No parameters required.

Task Store Route

Functionality: Processes the form submission to add a new task to the database.

Parameter Handling: Accepts form inputs (title, deadline, category) via a POST request.

Task Editing Route

Functionality: Displays a form pre-filled with the details of an existing task for editing.

Parameter Handling: id is passed as a URL parameter to identify the task to edit.

Task Update Route

Functionality: Saves updates to an existing task in the database.

Parameter Handling: Accepts id as a URL parameter and form inputs (title, deadline, category) via a PUT request.

Task Deletion Confirmation Route

Functionality: Shows a confirmation page before deleting a task.

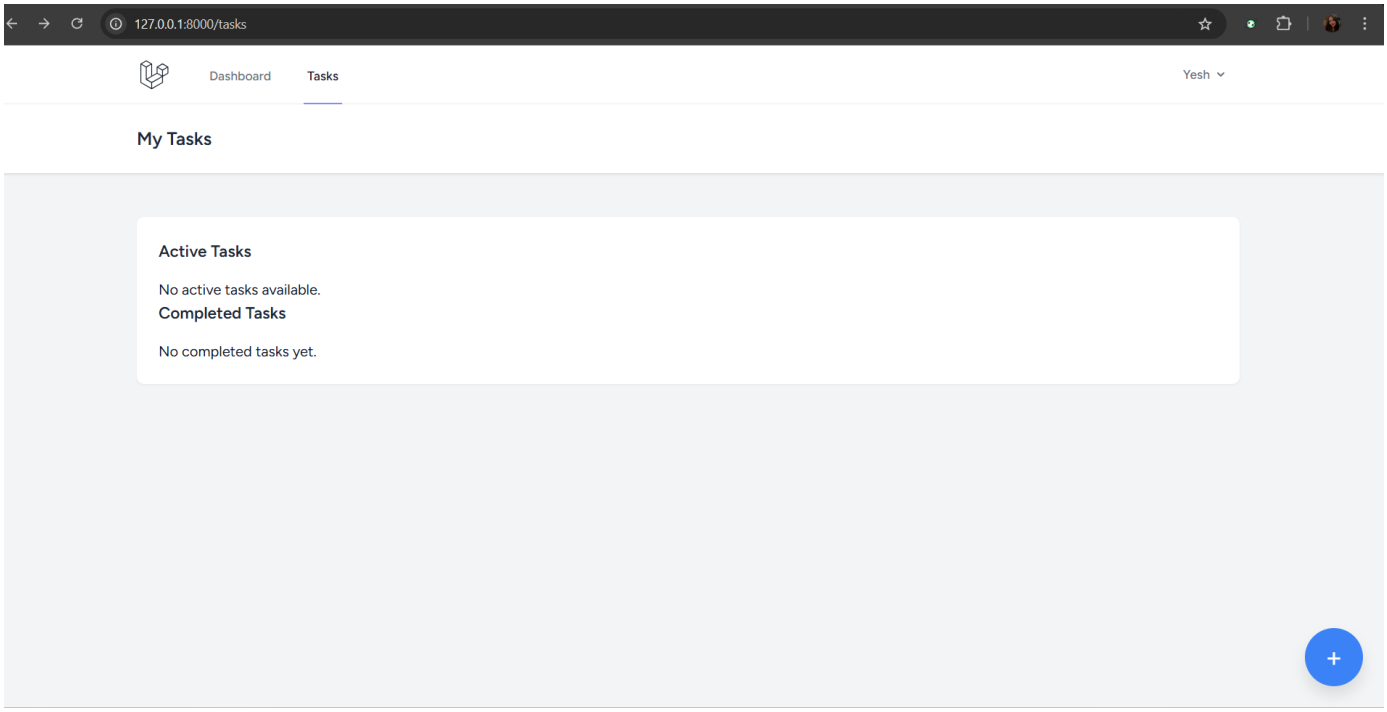
Parameter Handling: id is passed as a URL parameter to identify the task to delete.

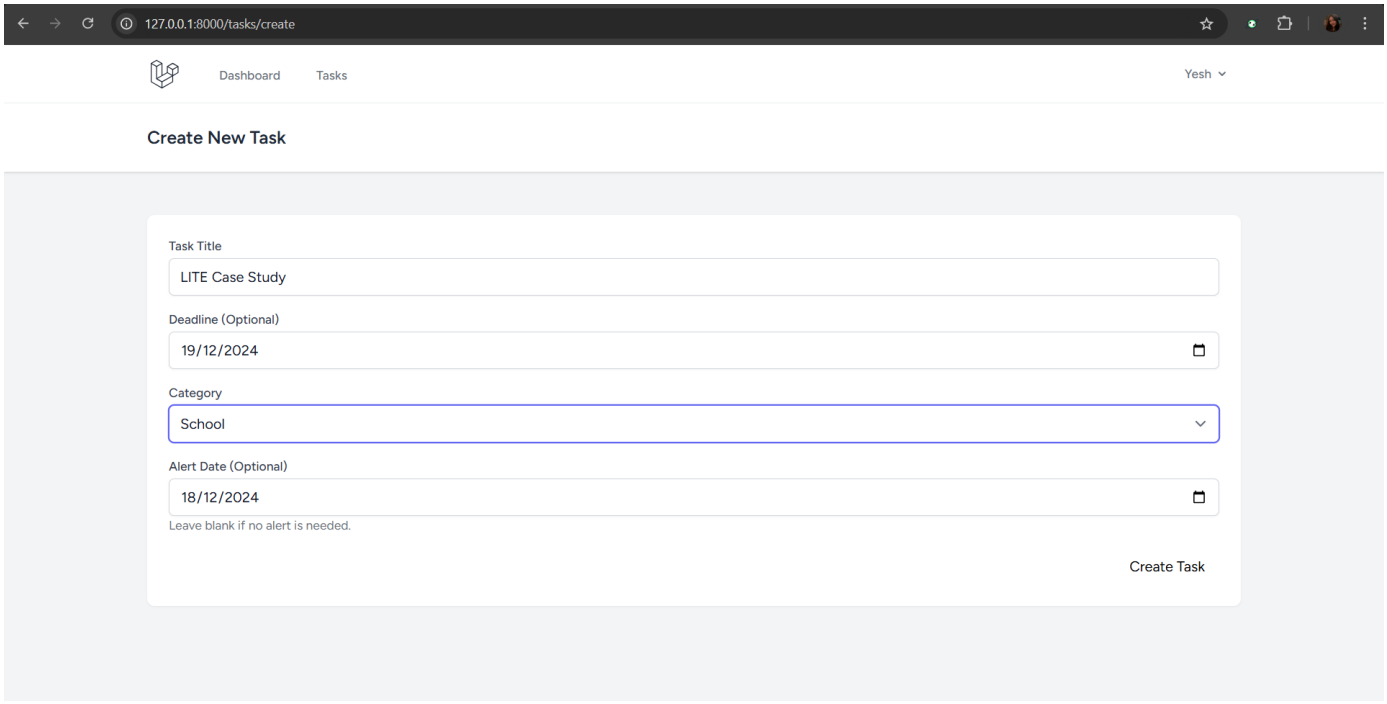
Task Completion Route

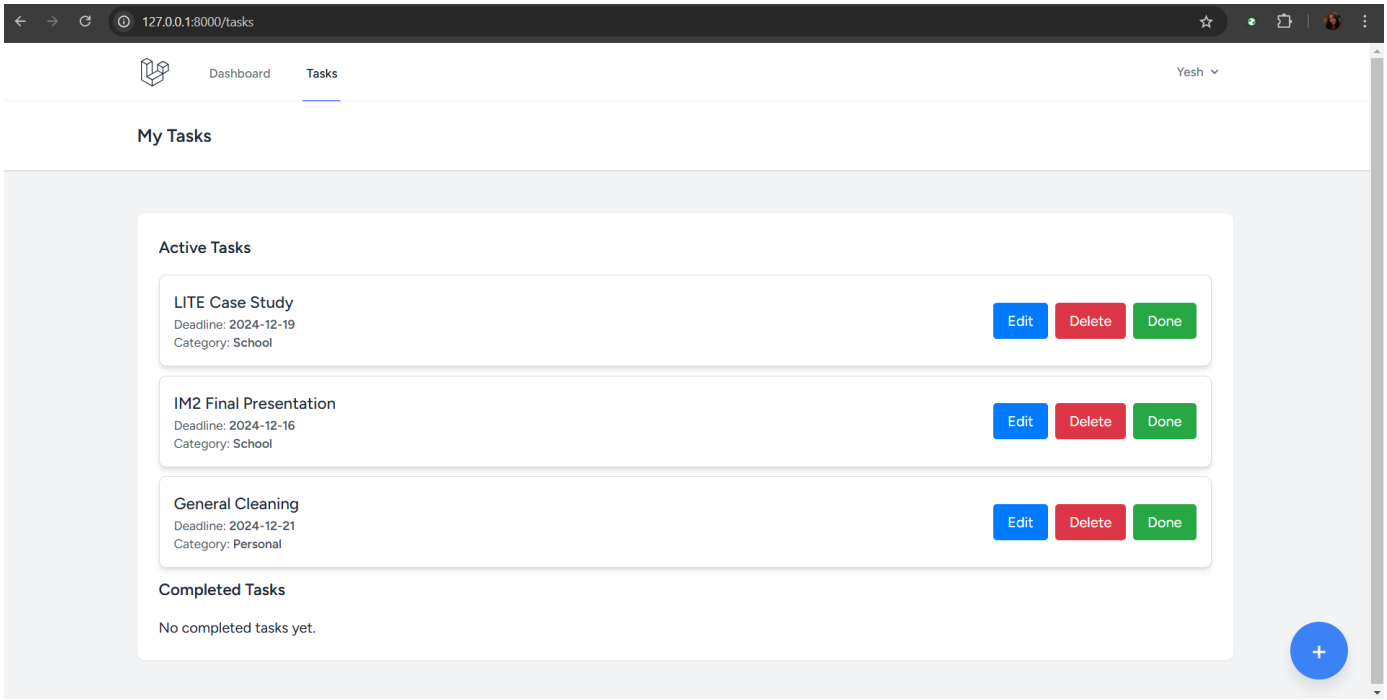
Functionality: Marks a task as completed and updates its status in the database.

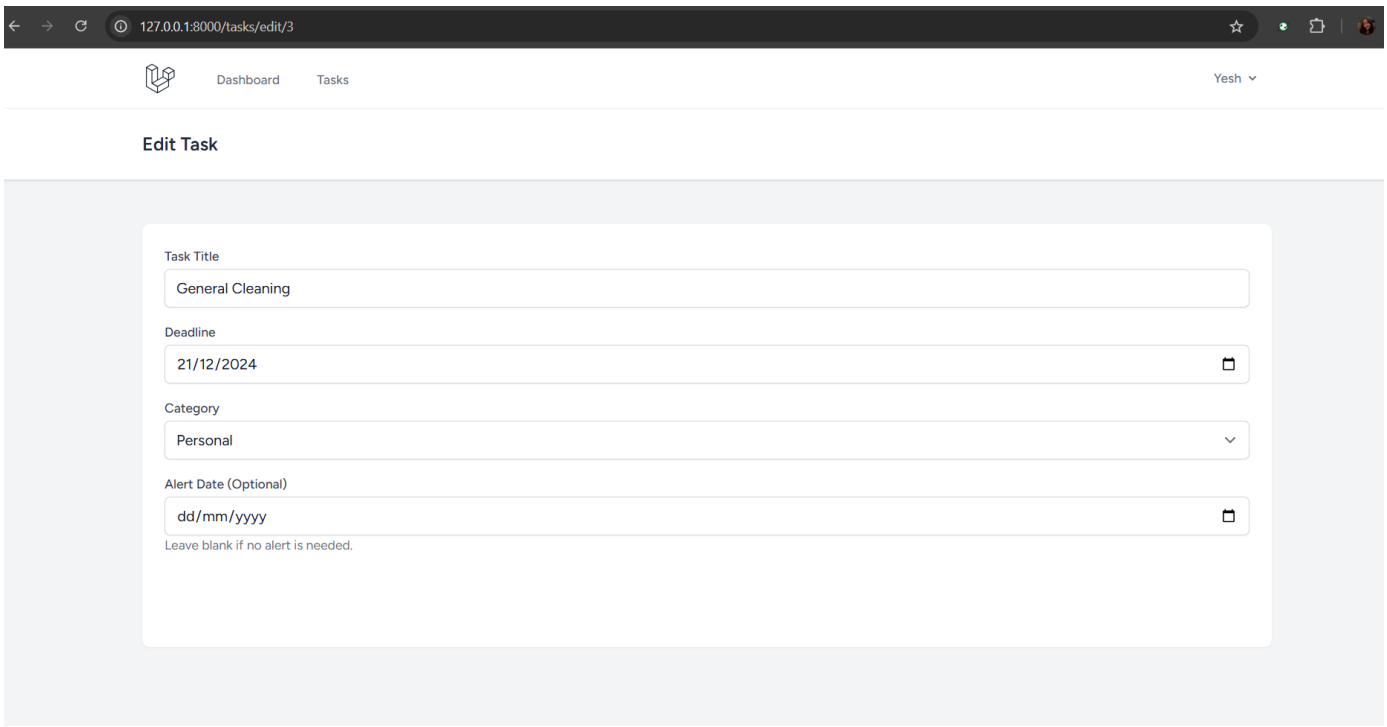
Parameter Handling: id is passed as a URL parameter to identify the task to mark as completed.

IV. **RENDERED PAGES**






127.0.0.1:8000/tasks
Dashboard
Tasks
Yesh
My Tasks
Active Tasks
LITE Case Study
Deadline: 2024-12-19
Category: School
Edit
Delete
Done
IM2 Final Presentation
Deadline: 2024-12-16
Category: School
Edit
Delete
Done
General Cleaning
Deadline: 2024-12-21
Category: Personal
Edit
Delete
Done
Completed Tasks
No completed tasks yet.


127.0.0.1:8000/tasks/edit/3
Dashboard
Tasks
Yesh
Edit Task
Task Title
General Cleaning
Deadline
21/12/2024
Category
Personal
Alert Date (Optional)
dd/mm/yyyy
Leave blank if no alert is needed.

v. WHAT I LEARNED

I learned how to simulate a database in Laravel using Eloquent ORM without requiring a physical connection, which allowed me to focus on core features like task creation, editing, and deletion. This approach simplified early development by enabling data mocking and testing without database setup. I also discovered the efficiency of the compact function for passing multiple variables to Blade views, making the code cleaner and more readable. Additionally, I worked with controllers to manage routes and integrated middleware to restrict access based on user authentication. Lastly, I enhanced the application's flexibility by using dynamic route parameters to filter tasks and display user-specific information.