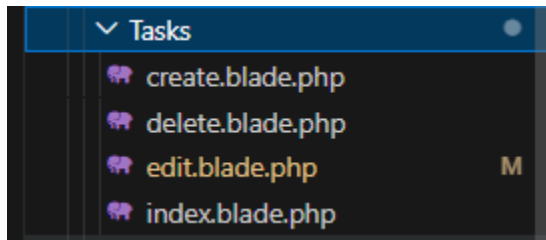


## DOCUMENTATION: Laboratory 2 – Populating Pages

### 1. Blade Files



Resources/Views/Task

```

index.blade.php
resources > views > Tasks > index.blade.php
1 <x-app-layout>
2 <x-slot name="header">
3 <h1 class="text-xl font-semibold text-gray-800">My Tasks</h1>
4 </x-slot>
5
6 <div class="py-12">
7 <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
8 <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
9 <div class="p-6 text-gray-800">
10 <!-- Active Tasks -->
11 <h2 class="text-lg font-semibold text-gray-800 mb-4">Active Tasks</h2>
12 <@if (count($activeTasks) > 0)>
13 <ul class="task-list space-y-4">
14 <@foreach ($activeTasks as $task)>
15 <li class="task-item bg-white p-4 rounded-lg shadow-md flex justify-between items-center">
16 <div>
17 <h3 class="text-lg font-medium text-gray-800">{{ $task['title'] }}</h3>
18 <p class="text-sm text-gray-600">Deadline: <strong>{{ $task['deadline'] }}</strong></p>
19 <p class="text-sm text-gray-600">Category: <strong>{{ $task['category'] }}</strong></p>
20 </div>
21 <div class="actions flex gap-2">
22 <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>
23 <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>
24 <a href="/tasks/complete/{{ $task['id'] }}" class="complete px-4 py-2 rounded bg-green-500 text-white hover:bg-green-600 transition duration-200">Done</a>
25 </div>
26 </li>
27 </@foreach>
28 </ul>
29 <@else>
30 <p>No active tasks available.</p>
31 <@endif>
32
33 <!-- Completed Tasks -->
34 <h2 class="text-lg font-semibold text-gray-800 mb-4 mt-8">Completed Tasks</h2>
35 <@if (count($completedTasks) > 0)>
36 <ul class="task-list space-y-4">
37 <@foreach ($completedTasks as $task)>
38 <li class="task-item bg-white p-4 rounded-lg shadow-md flex justify-between items-center">
39 <div>
40 <h3 class="text-lg font-medium text-gray-800">{{ $task['title'] }}</h3>
41 <p class="text-sm text-gray-600">Completed on: <strong>{{ $task['deadline'] }}</strong></p>
42 </div>
43 </li>
44 </@foreach>
45 </ul>
46 <@else>
47 <p>No completed tasks yet.</p>
48 <@endif>
49 </div>
50 </div>
51 </div>
52
53 <!-- Floating + Button -->
54 <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>
55
56 </x-app-layout>
57

```

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

**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 X
resources > views > Tasks > create.blade.php

1 <x-app-layout>
2   <x-slot name="header">
3     <h1 class="text-xl font-semibold text-gray-800">Create New Task</h1>
4   </x-slot>
5
6   <div class="py-12">
7     <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
8       <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
9         <div class="p-6 text-gray-900">
10          <form method="POST" action="{{ route('tasks.store') }}">
11            @csrf
12          </form>
13        </div>
14        <!-- Title -->
15        <div class="mb-4">
16          <label for="title" class="block text-sm font-medium text-gray-700">Task Title</label>
17          <input
18            type="text"
19            name="title"
20            id="title"
21            class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
22            required>
23        </div>
24        <!-- Deadline -->
25        <div class="mb-4">
26          <label for="deadline" class="block text-sm font-medium text-gray-700">Deadline (Optional)</label>
27          <input
28            type="date"
29            name="deadline"
30            id="deadline"
31            class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm">
32        </div>
33        <!-- Category -->
34        <div class="mb-4">
35          <label for="category" class="block text-sm font-medium text-gray-700">Category</label>
36          <select
37            name="category"
38            id="category"
39            class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
40            required>
41            <option value="Work">Work</option>
42            <option value="School">School</option>
43            <option value="Personal">Personal</option>
44            <option value="Other">Other</option>
45          </select>
46        </div>
47        <!-- Alert Date (optional) -->
48        <div class="mb-4">
49          <label for="alert_date" class="block text-sm font-medium text-gray-700">Alert Date (Optional)</label>
50          <input
51            type="date"
52            name="alert_date"
53            id="alert_date"
54            class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm">
55          <p class="text-sm text-gray-500">Leave blank if no alert is needed.</p>
56        </div>
57        <!-- Submit Button -->
58        <div class="flex justify-end">
59          <button
60            type="submit"
61            class="px-4 py-2 rounded bg-blue-500 text-black hover:bg-blue-600 transition duration-200">
62            Create Task
63          </button>
64        </div>
65      </div>
66    </div>
67  </x-app-layout>
```

**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 M X
resources > views > Tasks > edit.blade.php
1 <x-app-layout>
2 <div class="py-12">
3 <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
4 <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
5 <div class="p-6 text-gray-900">
6
7 <!-- Edit Task Form -->
8 <form method="POST" action="{{ route('tasks.update', $task->id) }}">
9 @csrf
10 @method('PUT')
11
12 <!-- Title Field -->
13 <div class="mb-4">
14 <label for="title" class="block text-sm font-medium text-gray-700">Task Title</label>
15 <input type="text" name="title" id="title"
16 class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
17 value="{{ old('title', $task->title) }}" required>
18 </div>
19
20 <!-- Deadline Field -->
21 <div class="mb-4">
22 <label for="deadline" class="block text-sm font-medium text-gray-700">Deadline</label>
23 <input type="date" name="deadline" id="deadline"
24 class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
25 value="{{ old('deadline', $task->deadline) }}" required>
26 </div>
27
28 <!-- Category Field -->
29 <div class="mb-4">
30 <label for="category" class="block text-sm font-medium text-gray-700">Category</label>
31 <select name="category" id="category"
32 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>
33 <option value="Work" {{ $task->category == 'Work' ? 'selected' : '' }}>Work</option>
34 <option value="School" {{ $task->category == 'School' ? 'selected' : '' }}>School</option>
35 <option value="Personal" {{ $task->category == 'Personal' ? 'selected' : '' }}>Personal</option>
36 <option value="Other" {{ $task->category == 'Other' ? 'selected' : '' }}>Other</option>
37 </select>
38 </div>
39
40 <!-- Alert Date Field (Optional) -->
41 <div class="mb-4">
42 <label for="alert_date" class="block text-sm font-medium text-gray-700">Alert Date (Optional)</label>
43 <input type="date" name="alert_date" id="alert_date"
44 class="mt-1 block w-full border-gray-300 rounded-md shadow-sm focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
45 value="{{ old('alert_date', $task->alert_date) }}">
46 <p class="text-sm text-gray-500">Leave blank if no alert is needed.</p>
47 </div>
48
49 <!-- Submit Button -->
50 <div class="flex justify-end space-x-2">
51 <button type="submit" class="px-4 py-2 rounded bg-blue-500 text-white hover:bg-blue-600 transition duration-200"
52 style="margin: 10px;">Update Task</button>
53
54 <!-- Done Button (only shows if task is not completed) -->
55 @if (!$task->completed)
56 <form method="POST" action="{{ route('tasks.complete', $task->id) }}" class="inline-block">
57 @csrf
58 @method('PATCH')
59 <button type="submit" class="px-4 py-2 rounded bg-green-500 text-white hover:bg-green-600 transition duration-200"
60 style="margin: 10px;">Mark as Done</button>
61 </form>
62 @endif
63
64 </div>
65 </div>
66 </div>
67 </div>
68 </x-app-layout>
```

### 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. 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.

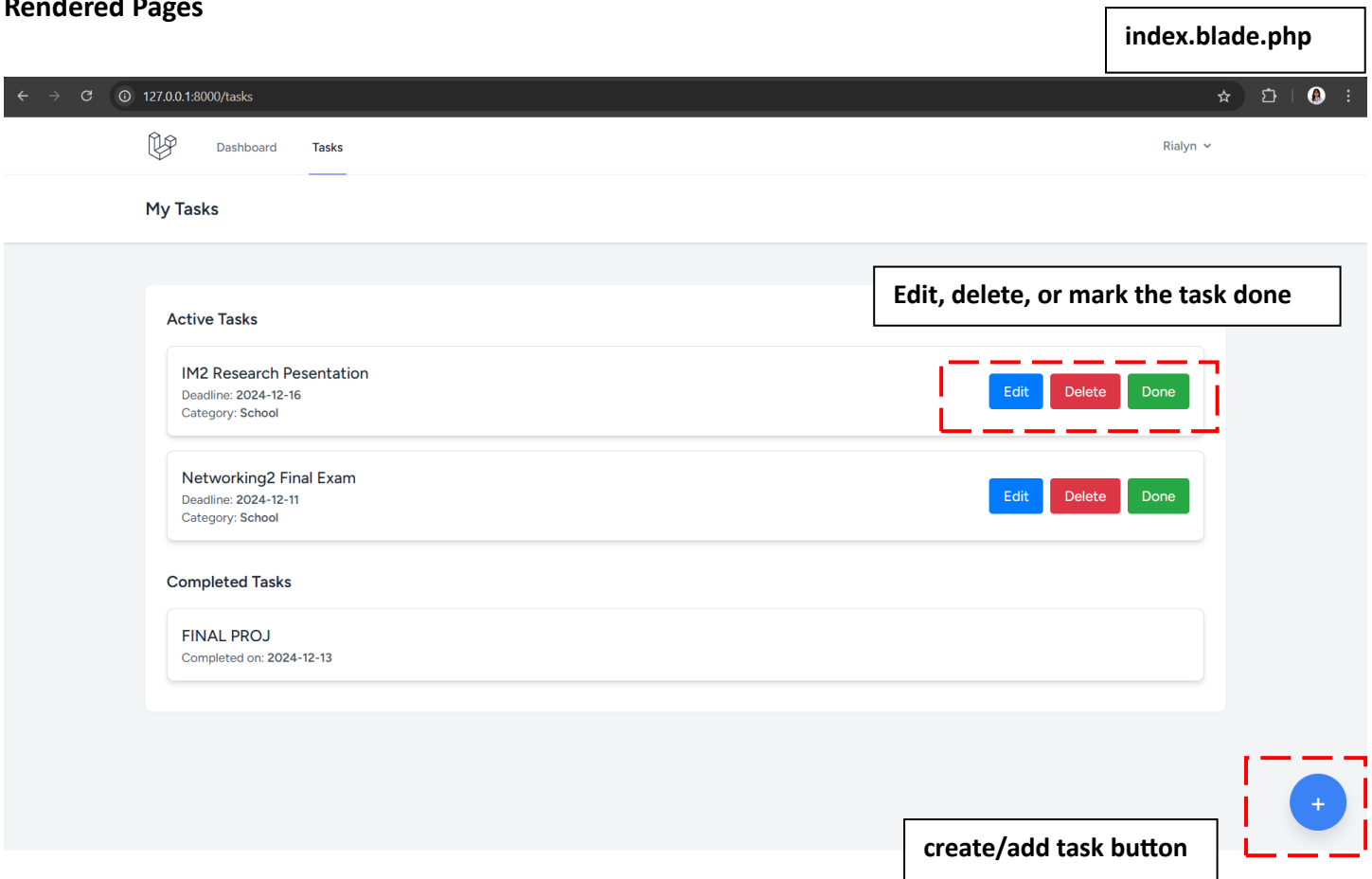
delete.blade.php X

delete.blade.php

```
resources > views > Tasks > delete.blade.php
1 <x-app-layout>
2 <!-- Page Content -->
3 <div class="container mx-auto mt-6 max-w-3xl bg-white p-6 rounded-lg shadow-md">
4   <h3 class="text-2xl font-semibold text-gray-800">Are you sure you want to delete this task?</h3>
5   <div class="alert alert-warning mt-4 text-yellow-600">
6     <strong>Warning:</strong> This action cannot be undone.
7   </div>
8
9   <!-- Form to confirm deletion -->
10  <form action="{{ route('tasks.delete.confirmed', $task['id']) }}" method="POST">
11    @csrf
12    @method('DELETE')
13
14    <div class="mt-6">
15      <button type="submit" class="btn btn-danger text-white bg-red-600 hover:bg-red-700 focus:ring-4 focus:ring-red-300">Delete</button>
16      <a href="{{ route('tasks.index') }}" class="btn btn-secondary text-white bg-gray-600 hover:bg-gray-700">Cancel</a>
17    </div>
18  </form>
19 </div>
20 </x-app-layout>
21
```

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

## Rendered Pages





## Create New Task

Task Title

Deadline (Optional)

dd/mm/yyyy



Category

Work



Alert Date (Optional)

dd/mm/yyyy



Leave blank if no alert is needed.

Create Task



## Edit Task

Task Title

IM2 Research Pesentation

Deadline

16/12/2024



Category

School



Alert Date (Optional)

dd/mm/yyyy



Leave blank if no alert is needed.

Update Task

Mark as Done

## 2. Controller Logic

### TasksController.php

```
TasksController.php M X
app > Http > Controllers > TasksController.php
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Task;
6 use Illuminate\Http\Request;
7
8 class TasksController extends Controller
9 {
10     // Display list of tasks
11     public function index()
12     {
13         $activeTasks = Task::where('completed', false)->get();
14         $completedTasks = Task::where('completed', true)->get();
15
16         return view('tasks.index', compact('activeTasks', 'completedTasks'));
17     }
18
19     // Show task creation form
20     public function create()
21     {
22         return view('tasks.create');
23     }
24
25     // Store new task
26     public function store(Request $request)
27     {
28         $request->validate([
29             'title' => 'required|string|max:255',
30             'deadline' => 'nullable|date',
31             'category' => 'required|string',
32         ]);
33
34         Task::create([
35             'title' => $request->title,
36             'deadline' => $request->deadline ?? 'No alert',
37             'category' => $request->category,
38             'completed' => false,
39         ]);
40
41         return redirect()->route('tasks.index');
42     }
43
44     // Edit an existing task
45     public function edit($id)
46     {
47         $task = Task::findOrFail($id);
48         return view('tasks.edit', compact('task'));
49     }
50
51     // Update an existing task
52     public function update(Request $request, $id)
53     {
54         $request->validate([
55             'title' => 'required|string|max:255',
56             'deadline' => 'nullable|date',
57             'category' => 'required|string',
58         ]);
59
60         $task = Task::findOrFail($id);
61         $task->update([
62             'title' => $request->title,
63             'deadline' => $request->deadline ?? 'No alert',
64             'category' => $request->category,
65         ]);
66
67         return redirect()->route('tasks.index');
68     }
69
70     // Show confirmation before deleting a task
71     public function confirmDelete($id)
72     {
73         $task = Task::findOrFail($id);
74         return view('tasks.delete', compact('task'));
75     }
76
77     // Delete a task
78     public function destroy($id)
79     {
80         $task = Task::findOrFail($id);
81         $task->delete();
82
83         return redirect()->route('tasks.index');
84     }
85
86     // Mark a task as complete
87     public function complete($id)
88     {
89         $task = Task::findOrFail($id);
90         $task->completed = true;
91         $task->save();
92
93         return redirect()->route('tasks.index');
94     }
95 }
96
```

#### 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.

#### **update(Request \$request, \$id)**

- **Purpose:** Updates an existing task with the modified data.
- **Key Logic:**
  - Validates the input fields to ensure data integrity.

#### **confirmDelete(\$id)**

- **Purpose:** Displays a confirmation page before deleting a task.
- **Key Logic:**
  - Retrieves the task to confirm the details before deletion.

#### **confirmDelete(\$id)**

- **Purpose:** Displays a confirmation page before deleting a task.
- **Key Logic:**
  - Retrieves the task to confirm the details before deletion.

#### **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.

#### **complete(\$id)**

- **Purpose:** Marks a task as completed.
- **Key Logic:**
  - Finds the task by ID and sets its completed attribute to true.

### 3. Routes

```
web.php X web.php
routes > web.php
1 <?php
2
3 use App\Http\Controllers\HomeController;
4 use App\Http\Controllers\UserDashboardController;
5 use App\Http\Controllers\ProfileController;
6 use App\Http\Controllers\TasksController;
7 use Illuminate\Support\Facades\Route;
8
9 // Homepage Route
10 Route::get('/', [HomeController::class, 'index']);
11
12 // Dashboard Route
13 Route::get('/dashboard', function () {
14     return view('dashboard');
15 })->middleware(['auth', 'verified'])->name('dashboard');
16
17 // Profile Routes
18 Route::middleware('auth')->group(function () {
19     Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
20     Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
21     Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
22 });
23
24 // User Dashboard Route
25 Route::get('/user-dashboard/{userId}', [UserDashboardController::class, 'show'])
26     ->middleware(['auth'])
27     ->name('user-dashboard');
28
29 // Tasks Routes
30 Route::middleware('auth')->group(function () {
31     Route::get('/tasks', [TasksController::class, 'index'])->name('tasks.index');
32     Route::get('/tasks/create', [TasksController::class, 'create'])->name('tasks.create');
33     Route::post('/tasks/store', [TasksController::class, 'store'])->name('tasks.store');
34     Route::get('/tasks/edit/{id}', [TasksController::class, 'edit'])->name('tasks.edit');
35     Route::get('/tasks/delete/{id}', [TasksController::class, 'destroy'])->name('tasks.delete');
36     Route::get('/tasks/complete/{id}', [TasksController::class, 'complete'])->name('tasks.complete');
37 });
38
39 Route::put('/tasks/{id}', [TasksController::class, 'update'])->name('tasks.update');
40 Route::patch('/tasks/complete/{id}', [TasksController::class, 'complete'])->name('tasks.complete');
41 Route::resource('tasks', TasksController::class);
42
43
44 require __DIR__.'/auth.php';
45
```

#### 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.

## Database Simulation

I learned how to simulate a database using Laravel's Eloquent ORM without a physical database connection. This allowed me to work on features like task creation, editing, and deletion while focusing on functionality and logic rather than database setup.

## Using the compact Function

The compact function was an efficient way to pass multiple variables to Blade views. For instance, in the index method, I used compact('activeTasks', 'completedTasks') to simplify the process of sending task data to the view.