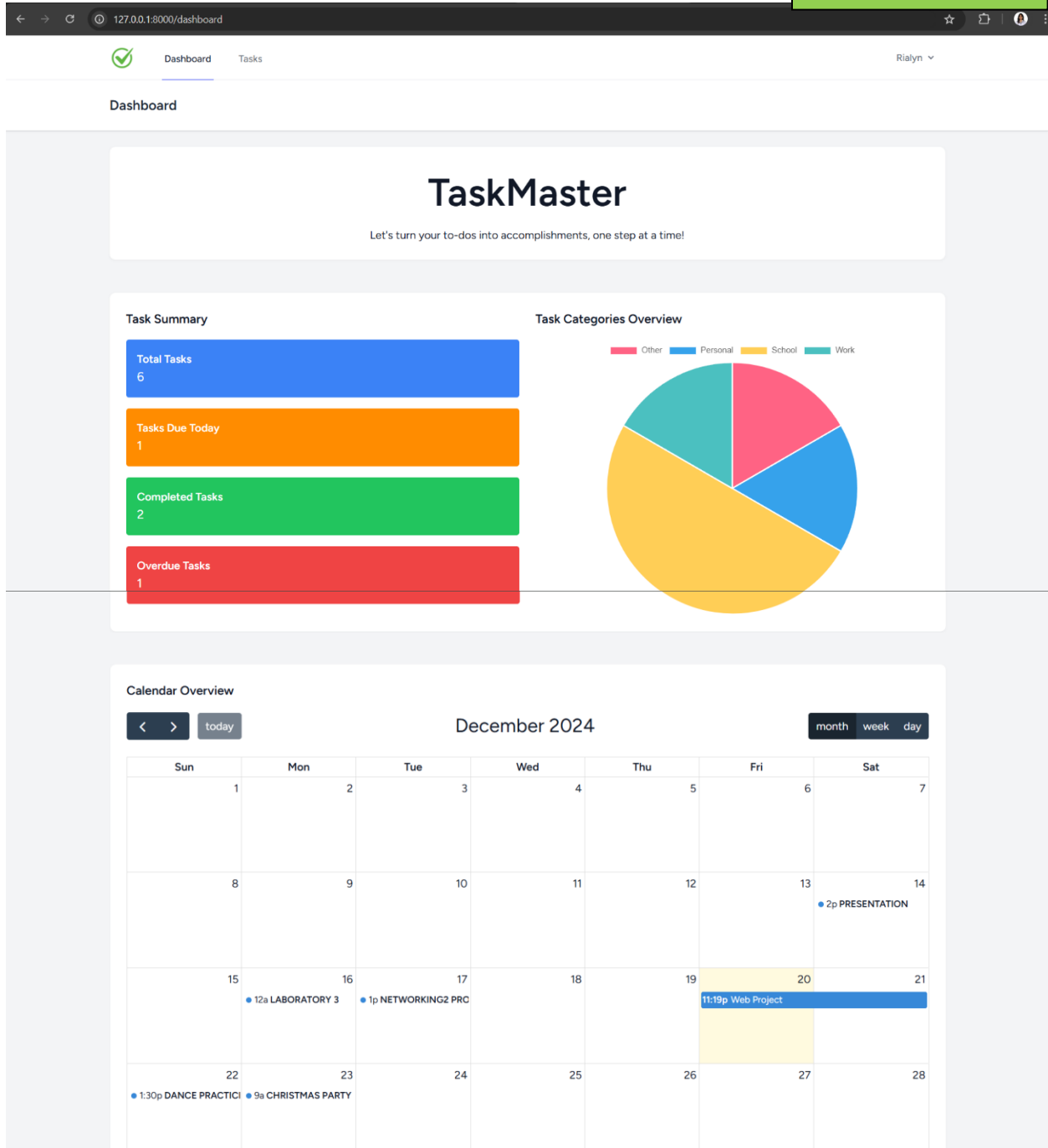


DOCUMENTATION: Laboratory 3: Populating From a Database

1. Rendered Pages and Accompanying Code

dashboard.blade.php



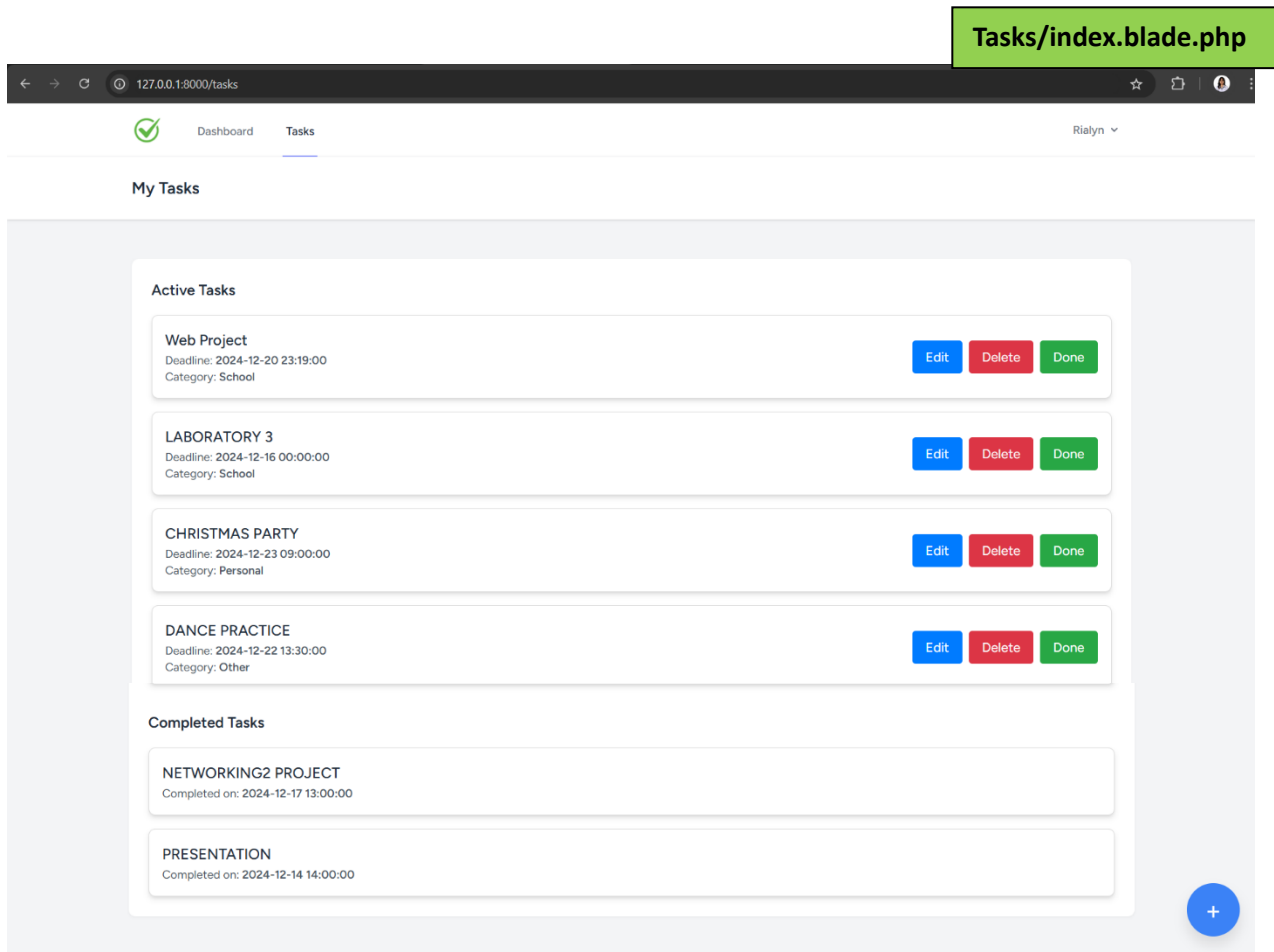
```
resources > views > dashboard.blade.php
1 @section('title', 'TaskMaster - Dashboard')
2 <x-app-layout>
3     <x-slot name="header">
4         <h2 class="font-semibold text-xl text-gray-800 leading-tight">
5             {{ __('Dashboard') }}
6         </h2>
7     </x-slot>
8
9     <div class="py-6">
10         <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
11             <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
12                 <div class="p-6 text-gray-900 flex flex-col justify-center items-center text-center">
13                     <h1 class="text-gray-900 mb-2" style="font-size: 3.5rem; font-weight: 900;">TaskMaster</h1>
14                     <p>{{ __('Let's turn your to-dos into accomplishments, one step at a time!') }}</p>
15                 </div>
16             </div>
17         </div>
18     </div>
19
20     <!-- Task Summary and Task Categories Overview -->
21     <div class="py-6">
22         <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
23             <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
24                 <div class="p-6 text-gray-900 grid grid-cols-2 gap-6">
25
26                     <!-- Task Summary -->
27                     <div>
28                         <h3 class="text-lg font-semibold mb-4">Task Summary</h3>
29                         <div class="grid grid-cols-1 gap-4">
30                             <div class="bg-blue-500 text-white p-4 rounded flex flex-col justify-between">
31                                 <h3 class="font-bold">Total Tasks</h3>
32                                 <p class="text-lg">{{ $totalTasks }}</p>
33                             </div>
34                             <div class="p-4 rounded flex flex-col justify-between" style="background-color: #FF8C00; color: #FFFFFF;>
35                                 <h3 class="font-bold">Tasks Due Today</h3>
36                                 <p class="text-lg">{{ $tasksDueToday }}</p>
37                             </div>
38                             <div class="bg-green-500 text-white p-4 rounded flex flex-col justify-between">
39                                 <h3 class="font-bold">Completed Tasks</h3>
40                                 <p class="text-lg">{{ $completedTasks }}</p>
41                             </div>
42                             <div class="bg-red-500 text-white p-4 rounded flex flex-col justify-between">
43                                 <h3 class="font-bold">Overdue Tasks</h3>
44                                 <p class="text-lg">{{ $overdueTasks }}</p>
45                             </div>
46                         </div>
47                     </div>
48
49                     <!-- Task Categories Overview -->
50                     <div>
51                         <h3 class="text-lg font-semibold mb-4">Task Categories Overview</h3>
52                         <div style="width: 100%; height: 400px;">
53                             <canvas id="categoryChart"></canvas>
54                         </div>
55                     </div>
56                 </div>
57             </div>
58         </div>
59     </div>
60
61     <script>
62         const ctx = document.getElementById('categoryChart').getContext('2d');
63         const categoryChart = new Chart(ctx, {
64             type: 'pie',
65             data: {
66                 labels: @json($tasksByCategory->pluck('category')),
67                 datasets: [{
68                     label: 'Tasks by Category',
69                     data: @json($tasksByCategory->pluck('count')),
70                     backgroundColor: ['FF6384', '36A2EB', 'FFCE56', '4BC0C0'],
71                 }]
72             },
73             options: {
74                 responsive: true,
75                 maintainAspectRatio: false
76             }
77         });
78     </script>
79
80     <!-- Calendar Overview -->
81     <div class="py-6">
82         <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
83             <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
84                 <div class="p-6 text-gray-900">
85                     <h3 class="text-lg font-semibold mb-4">Calendar Overview</h3>
86                     <div id="calendar" style="width: 100%; height: 500px;">
87                     </div>
88                 </div>
89             </div>
90         </div>
91     </div>
92
93     <script>
94         document.addEventListener('DOMContentLoaded', function () {
95             var calendarEl = document.getElementById('calendar');
96
97             var calendar = new FullCalendar.Calendar(calendarEl, {
98                 initialView: 'dayGridMonth',
99                 headerToolbar: {
100                     left: 'prev,next today',
101                     center: 'title',
102                     right: 'dayGridMonth,dayGridWeek,timeGridDay'
103                 },
104                 events: @json($tasks->map(function ($task) {
105                     return {
106                         'title' => $task->title,
107                         'start' => $task->deadline ? $task->deadline->toIso8601String() : null
108                     };
109                 })).filter(event => event.start !== null)
110             });
111
112             calendar.render();
113         });
114     </script>
115
116 </x-app-layout>
```

dashboard.blade.php

Task Summary

Task Categories
Overview

Calendar Overview



2. Controller Logic

2.1 Dashboard Logic

TasksController.php

```
90 // Dashboard
91 public function dashboard()
92 {
93     $tasks = Task::all(); // Fetch all tasks
94     $now = now(); // Current date and time
95     // Filter tasks due today
96     $tasksDueToday = $tasks->filter(function ($task) use ($now) {
97         if (!$task->deadline) {
98             return false; // Skip tasks without a deadline
99         }
100         $deadline = Carbon::parse($task->deadline);
101         // Debugging: Check task details
102         \Log::info("Task ID {$task->id}: Deadline {$task->deadline}, Now {$now}");
103         return $deadline->isToday() && $deadline->gte($now) && !$task->completed;
104     }->count());
105     // Filter overdue tasks
106     $overdueTasks = $tasks->filter(function ($task) use ($now) {
107         if (!$task->deadline) {
108             return false; // Skip tasks without a deadline
109         }
110         $deadline = Carbon::parse($task->deadline);
111         // Debugging: Check task details
112         \Log::info("Task ID {$task->id}: Deadline {$task->deadline}, Now {$now}");
113         return $deadline->lt($now) && !$task->completed;
114     }->count());
115     // Total tasks and completed tasks
116     $totalTasks = $tasks->count();
117     $completedTasks = $tasks->where('completed', true)->count();
118     // Tasks by category
119     $tasksByCategory = Task::select('category', \DB::raw('count(*) as count'))
120         ->groupBy('category')
121         ->get();
122     // Return the view with data
123     return view('dashboard', compact('tasks', 'totalTasks', 'tasksDueToday', 'completedTasks', 'overdueTasks', 'tasksByCategory'));
124 }
```

\$tasks = Task::all(); Retrieves all tasks from the database to analyze and display
\$now = now(); Captures the current date and time for comparisons

Tasks Due Today

- Filters tasks with:
 - A deadline set for today.
 - A deadline time in the future.
 - A status of not completed.
- Uses **Carbon::isToday()** and **gte()** for accurate filtering.

Overdue Tasks

- Filters tasks with:
 - A past deadline.
 - A status of not completed.
- Uses **Carbon::lt()** to identify overdue tasks.

Total and Completed Tasks

- **\$totalTasks:** Counts all tasks in the system.
- **\$completedTasks:** Counts tasks marked as completed.

Tasks by Category

- Groups tasks by their category and counts each group.

Return Data to View

- **return view('dashboard', compact(...));**
 - Purpose: Passes all calculated data (**\$tasks**, **\$totalTasks**, **\$tasksDueToday**, **\$completedTasks**, **\$overdueTasks**, **\$tasksByCategory**) to the dashboard view for rendering.

2.2 Task Management Logic

```
TasksController.php
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',
            'deadline' => 'nullable|date_format:Y-m-d\TH:i',
            'category' => 'required|string',
        ]);

        Task::create([
            'title' => $request->title,
            'deadline' => $request->deadline,
            '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',
            'deadline' => 'nullable|date',
            'deadline' => 'nullable|date_format:Y-m-d\TH:i',
            'category' => 'required|string',
        ]);

        $task = Task::findOrFail($id);
        $task->update([
            'title' => $request->title,
            'deadline' => $request->deadline,
            'category' => $request->category,
        ]);

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

    // 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');
    }
}
```

Display List of Tasks

- **\$activeTasks = Task::where('completed', false)->get();**
 - Retrieves tasks not marked as completed for the active list.
- **\$completedTasks = Task::where('completed', true)->get();**
 - Retrieves tasks marked as completed for display in the completed section.

Create New Task

- **\$request->validate([...]);**
 - Validates user input, ensuring title, deadline, and category meet requirements.
 - Deadline includes both date and time validation with **date_format:Y-m-d\TH:i**.
- **Task::create([...]);**
 - Stores the new task in the database with default completed set to false.

Edit and Update Task

- **Task::findOrFail(\$id);**
 - Fetches the task by its ID; throws an error if it doesn't exist.
- **\$task->update([...]);**
 - Updates the task's title, deadline, and category fields in the database.

Delete Task

- **public function destroy(\$id)**
 - Deletes the task identified by its ID from the database.

Mark Task as Complete

- **\$task->completed = true;**
 - Sets the completed status of the task to true.
 - Saves the change to the database.

3. Route Assignments

```
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  use Carbon\Carbon;
9
10 // Homepage Route
11 Route::get('/', [HomeController::class, 'index']);
12
13 // Profile Routes
14 Route::middleware('auth')->group(function () {
15     Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
16     Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
17     Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
18 });
19
20 // User Dashboard Route
21 Route::get('/dashboard', [TasksController::class, 'dashboard'])->name('dashboard');
22
23 Route::get('/user-dashboard/{userId}', [UserDashboardController::class, 'show'])
24     ->middleware(['auth'])
25     ->name('user-dashboard');
26
27 // Tasks Routes
28 Route::middleware('auth')->group(function () {
29     Route::get('/tasks', [TasksController::class, 'index'])->name('tasks.index');
30     Route::get('/tasks/create', [TasksController::class, 'create'])->name('tasks.create');
31     Route::post('/tasks/store', [TasksController::class, 'store'])->name('tasks.store');
32     Route::get('/tasks/edit/{id}', [TasksController::class, 'edit'])->name('tasks.edit');
33     Route::get('/tasks/delete/{id}', [TasksController::class, 'destroy'])->name('tasks.delete');
34     Route::get('/tasks/complete/{id}', [TasksController::class, 'complete'])->name('tasks.complete');
35 });
36
37 Route::put('/tasks/{id}', [TasksController::class, 'update'])->name('tasks.update');
38 Route::patch('/tasks/complete/{id}', [TasksController::class, 'complete'])->name('tasks.complete');
39 Route::resource('tasks', TasksController::class);
40
41 Route::get('/test-timezone', function () {
42     return now()->toDateTimeString(); // This will show the current time based on the app's timezone
43 });
44
45 require __DIR__.'/auth.php';
```

User Dashboard Route

This route listens for a GET request to `/user-dashboard/{userId}` where `{userId}` is a dynamic parameter.

This route points to the **dashboard()** method in the **TasksController**, which is responsible for handling logic related to displaying tasks or any other relevant data on the dashboard page.

The **show()** method in the **UserDashboardController** is responsible for handling the logic related to displaying the user dashboard for a specific user (identified by **userId**).

Account registration and login page

The screenshot shows the TaskMaster application interface. At the top, there's a navigation bar with 'Dashboard' and 'Tasks' links. Below this, a 'Home' section contains a welcome message: 'Welcome to TaskMaster! Your ultimate partner in staying organized and productive.' It prompts the user to 'log in' or 'register' to access more features. Below this, a large green checkmark icon is displayed above a login form. The form includes fields for 'Email' and 'Password', a 'Remember me' checkbox, and a 'LOG IN' button. A link for 'Forgot your password?' is also present.

Database Overview

The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure, including 'task_management_sys' and its tables. The main area shows the 'tasks' table with 5 rows. The table structure is as follows:

	id	title	deadline	category	completed	alert_date	created_at	updated_at
<input type="checkbox"/>	1	Web Project	2024-12-20 23:19:00	School	0	NULL	2024-12-20 13:19:33	2024-12-20 13:19:33
<input type="checkbox"/>	3	NETWORKING2 PROJECT	2024-12-17 13:00:00	School	1	NULL	2024-12-20 18:27:28	2024-12-20 18:27:39
<input type="checkbox"/>	4	CHRISTMAS PARTY	2024-12-23 09:00:00	Personal	0	NULL	2024-12-20 18:28:37	2024-12-20 18:28:37
<input type="checkbox"/>	5	PRESENTATION	2024-12-14 14:00:00	Work	1	NULL	2024-12-20 18:33:46	2024-12-20 18:33:56
<input type="checkbox"/>	6	DANCE PRACTICE	2024-12-22 13:30:00	Other	0	NULL	2024-12-20 18:35:00	2024-12-20 18:35:00

Below the table, there are options for 'Query results operations' including 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'. There is also a section for 'Bookmark this SQL query' with a label field and a checkbox to 'Let every user access this bookmark'.