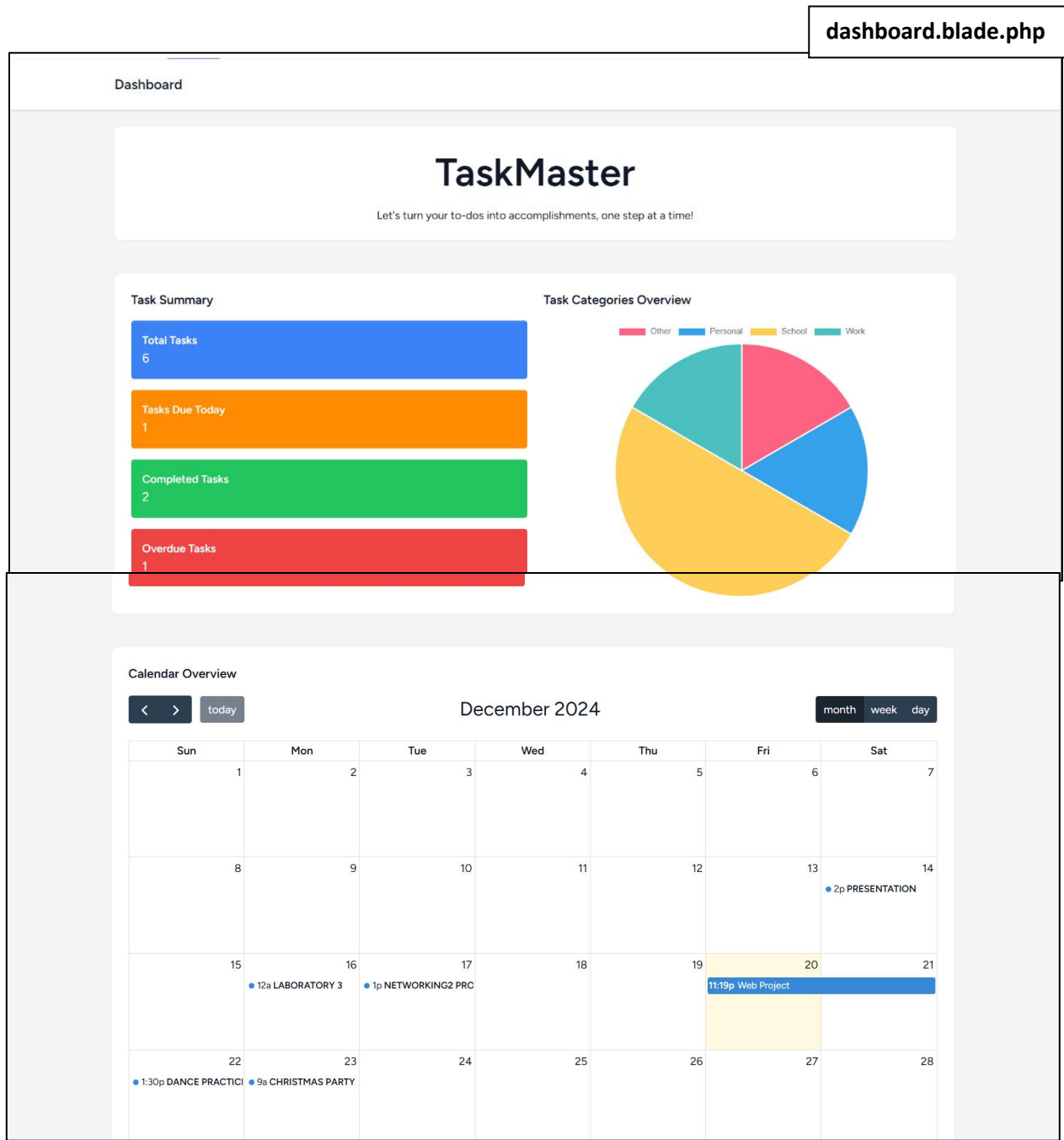


ANGELA B. ABAD | BSIT 3C
DOCUMENTATION: Laboratory 3: Populating From a Database

1. Rendered Pages and Accompanying Code



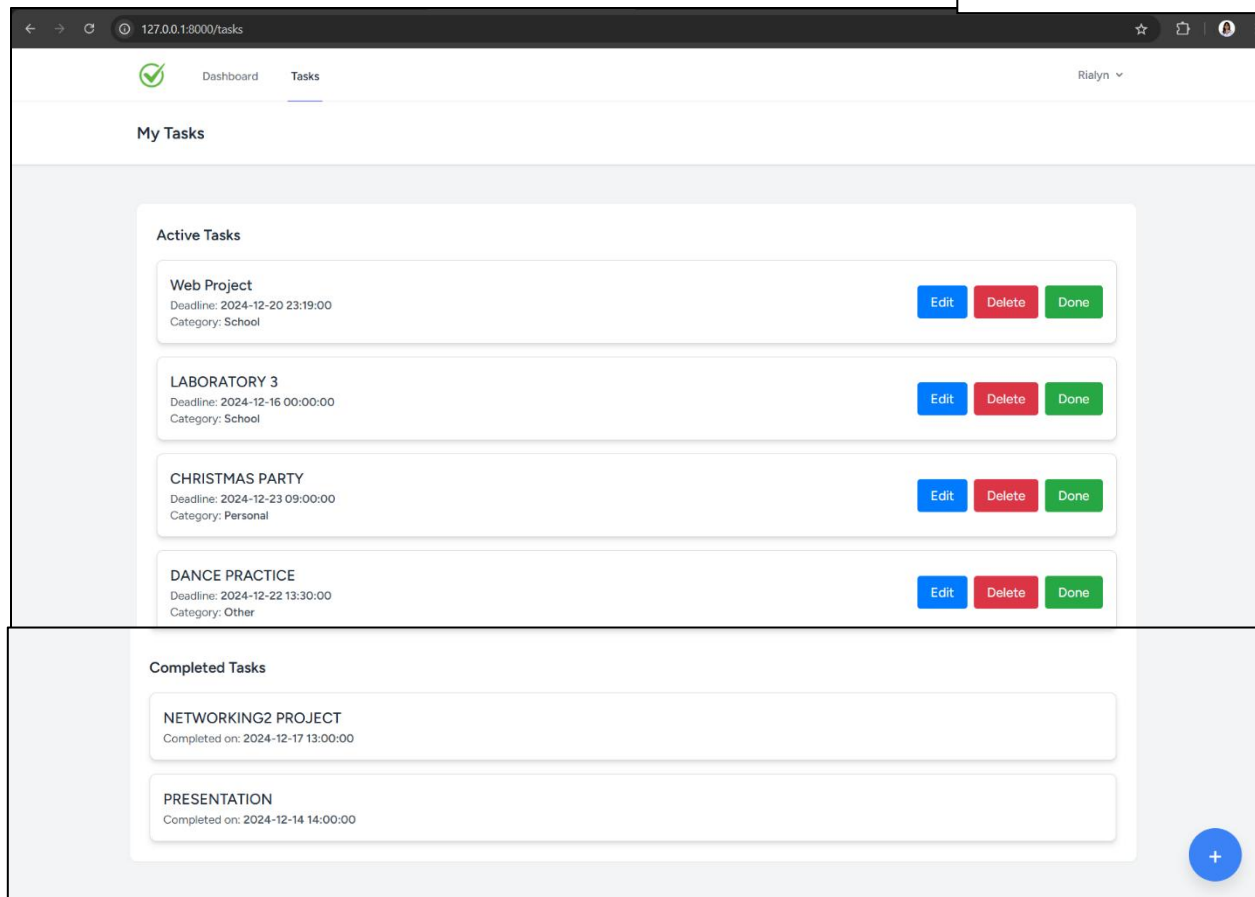
```
resources > views > dashboard.blade.php
1 @section('title', 'TaskMaster - Dashboard')
2 <x-app-layout>
3   <x-slot name="header">
4     <h2 class="font-semibold text-xl text-gray-800 leading-tight">
5       {{ __('Dashboard') }}
6     </h2>
7   </x-slot>
8
9   <div class="py-6">
10     <div class="max-w-7xl 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">{{ $overdueTasks }}</p>
45               </div>
46             </div>
47           </div>
48
49           <!-- Task Categories Overview -->
50           <div>
51             <h3 class="text-lg font-semibold mb-4">Task Categories Overview</h3>
52             <div style="width: 100%; height: 400px;">
53               <canvas id="categoryChart"></canvas>
54             </div>
55           </div>
56
57           <script>
58             const ctx = document.getElementById('categoryChart').getContext('2d');
59             const categoryChart = new Chart(ctx, {
60               type: 'pie',
61               data: {
62                 labels: @json($tasksByCategory->pluck('category')),
63                 datasets: [
64                   {
65                     label: 'Tasks by Category',
66                     data: @json($tasksByCategory->pluck('count')),
67                     backgroundColor: ['FF8C00', '36A2EB', 'FFCE56', '4BC0C0'],
68                   }
69                 ],
70               options: {
71                 responsive: true,
72                 maintainAspectRatio: false
73               }
74             });
75           </script>
76
77           <!-- Calendar Overview -->
78           <div>
79             <h3 class="text-lg font-semibold mb-4">Calendar Overview</h3>
80             <div style="width: 100%; height: 500px;">
81               <div id="calendar"></div>
82             </div>
83           </div>
84
85           <script>
86             document.addEventListener('DOMContentLoaded', function () {
87               var calendarEl = document.getElementById('calendar');
88
89               var calendar = new FullCalendar.Calendar(calendarEl, {
90                 initialView: 'dayGridMonth',
91                 headerToolbar: {
92                   left: 'prev,next today',
93                   center: 'title',
94                   right: 'dayGridMonth,dayGridWeek,timeGridDay'
95                 },
96                 events: @json($tasks->map(function ($task) {
97                   return {
98                     'title' => $task->title,
99                     'start' => $task->deadline ? $task->deadline->toIso8601String() : null
100                   };
101                 }
102               )
103             ));
104             calendar.render();
105           </script>
106
107   </x-app-layout>
```

dashboard.blade.php

Task Summary

Task Categories Overview

Calendar Overview



2. Controller Logic

2.1 Dashboard Logic

```

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 $deadline->isToday() && !$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

```
9 class TasksController extends Controller
10 {
11     // Display list of tasks
12     public function index()
13     {
14         $activeTasks = Task::where('completed', false)->get();
15         $completedTasks = Task::where('completed', true)->get();
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             'deadline' => 'nullable|date_format:Y-m-d\TH:i',
32             'category' => 'required|string',
33         ]);
34
35         Task::create([
36             'title' => $request->title,
37             'deadline' => $request->deadline,
38             'category' => $request->category,
39             'completed' => false,
40         ]);
41
42         return redirect()->route('tasks.index');
43     }
44
45     // Edit an existing task
46     public function edit($id)
47     {
48         $task = Task::findOrFail($id);
49         return view('tasks.edit', compact('task'));
50     }
51
52     // Update an existing task
53     public function update(Request $request, $id)
54     {
55         $request->validate([
56             'title' => 'required|string|max:255',
57             'deadline' => 'nullable|date',
58             'deadline' => 'nullable|date_format:Y-m-d\TH:i',
59             'category' => 'required|string',
60         ]);
61
62         $task = Task::findOrFail($id);
63         $task->update([
64             'title' => $request->title,
65             'deadline' => $request->deadline,
66             'category' => $request->category,
67         ]);
68
69         return redirect()->route('tasks.index');
70     }
71
72     // Delete a task
73     public function destroy($id)
74     {
75         $task = Task::findOrFail($id);
76         $task->delete();
77
78         return redirect()->route('tasks.index');
79     }
80
81     // Mark a task as complete
82     public function complete($id)
83     {
84         $task = Task::findOrFail($id);
85         $task->completed = true;
86         $task->save();
87
88         return redirect()->route('tasks.index');
89     }
90 }
```

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';
```

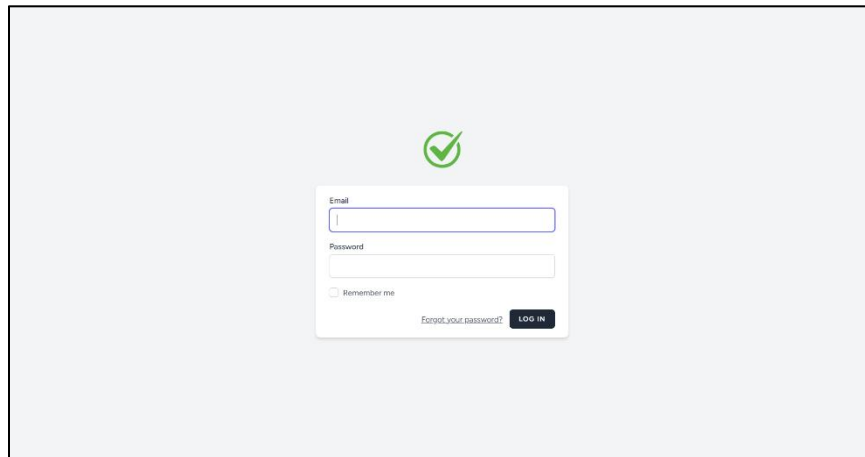
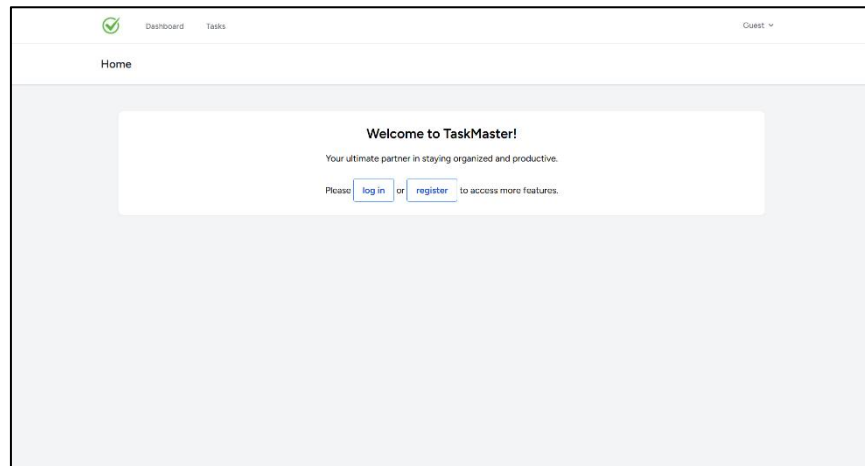
User Dashboard Route

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



phpMyAdmin

Recent Favorites

admin_role_users

admin_users

admin_user_permissions

cache

cache_locks

failed_jobs

jobs

job_batches

migrations

password_reset_tokens

sessions

tasks

test

users

dtbs

information_schema

mysql

performance_schema

phpmyadmin

test

Server: 127.0.0.1 » Database: databasee » Table: users

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Triggers

Profiling

Edit inline

Edit

Explain SQL

Create PHP code

Refresh

Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

Extra options

		id	name	email	email_verified_at	password
<input type="checkbox"/>	Edit Copy Delete	1	AIAH ARCE TA	binimaloi@gmail.com	NULL	\$2y\$12\$Bu7JqCPzw7BqZJxtSjdaV.XipxlyoL8jRSpkS9tvcnT...
<input type="checkbox"/>	Edit Copy Delete	2	Angela Abad	anglaabadjanuary@gmail.com	NULL	\$2y\$12\$C/zPwcQLFy4hZbi8z1VuetUhbcoQ36kfmJ5yUtvFb5...
<input type="checkbox"/>	Edit Copy Delete	3	Sarah	xara@gmail.com	NULL	\$2y\$12\$1/yZWycOUv7pFswzgS2dOoEM4U2fi724aH0zcJaeil...
<input type="checkbox"/>	Edit Copy Delete	4	Angela	abad@gmail.com	NULL	\$2y\$12\$XxwmpRp0M8j51j5GMLtuh4nqMNVltsDumJAq4Y5rs...
<input type="checkbox"/>	Edit Copy Delete	5	angela	asd@gmail.com	NULL	\$2y\$12\$X3XPvFxmLKGJQkzSGBoQuOWivvqMF5FMNKMbNo0cJ5b...
<input type="checkbox"/>	Edit Copy Delete	6	Angela	shesh@gmail.com	NULL	\$2v\$12\$SntiPv9TiRO0z1UTzaDlZ09iO6bol.hk55bu4VGCpwW...

Console

Press Ctrl+Enter to execute query

>SELECT * FROM `users`

>

localhost/phpmyadmin/index.php?route=/sql&pos=0&db=databasee&table=tasks

Server: 127.0.0.1 » Database: databasee » Table: tasks

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Triggers

Profiling

Edit inline

Edit

Explain SQL

Create PHP code

Refresh

Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

Extra options

		id	user_id	title	deadline	category	completed	alert_date	created_at	updated_at	status
<input type="checkbox"/>	Edit Copy Delete	5	NULL	WEB DEV FINAL PROJECT SYSTEM	2024-12-20 17:44:00	School	1	NULL	2024-12-20 08:44:04	2024-12-20 08:44:24	pending
<input type="checkbox"/>	Edit Copy Delete	6	NULL	EXERCISE 🎮❤️	2024-12-20 05:46:00	Personal	0	NULL	2024-12-20 08:45:22	2024-12-20 08:45:58	pending
<input type="checkbox"/>	Edit Copy Delete	7	NULL	BUY CHRISTMAS GIFTS 🎁🇺🇸	2024-12-22 16:48:00	Work	0	NULL	2024-12-20 08:49:01	2024-12-20 08:49:01	pending

Check all

With selected:

Edit Copy Delete Export

Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

Console

Press Ctrl+Enter to execute query

>

Database Overview