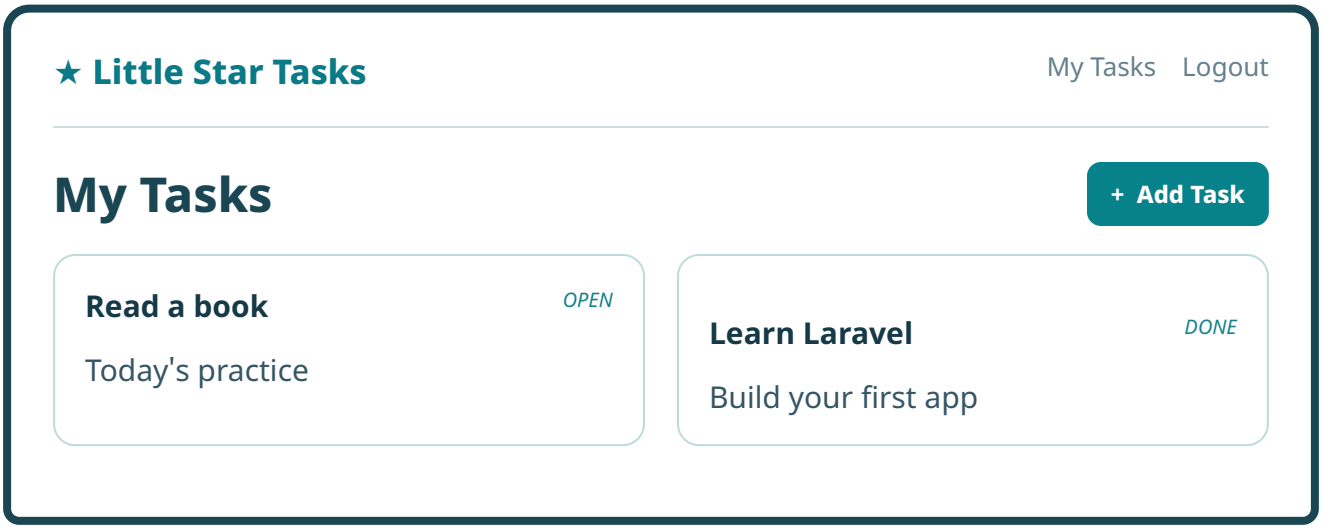
සිංහල • මුල සිට • Practical Project

Laravel
Login + CRUD

Python Django පාඩමෙන් පසු PHP හා Laravel භාවිතයෙන් ඔබේම ආරක්ෂිත Task Manager website එකක් හදමු.



ROUTES → CONTROLLER → DATABASE

Laravel 13 • PHP 8.3+ • SQLite • VS Code • Auth • CRUD • Testing • Git

මේ පොත භාවිතා කරන්නේ කොහොමද?

මෙම පොත පියවරෙන් පියවර කියවන්න. සෑම පාඩමකම සරල තේරුම → Code/Diagram → අභ්‍යාසය → Expected Result ඇත.

1. Read

සිංහල පැහැදිලි කිරීමෙන් concept එක දැනගන්න.

2. Build

VS Code හෝ CMD දී කුඩා code එක run කරන්න.

3. Test

Expected result සමඟ output එක සසඳන්න.

අපේ පොතේ නීති හතර

- අලුත් commands run කරන විට `little-star-tasks` folder එකේ සිටින්න.
- Code comments English; පැහැදිලි කිරීම් Sinhala.
- `.env`, passwords සහ database files GitHub වෙත නොයවන්න.
- සැකයක් තිබුණොත් terminal error එක කියවා test එකක් ලියන්න.

> **සියලුම පිටු සුදු background**
>
> VS Code/CMD නිරූපණය කරන code panels පමණක් කළු පැහැයෙන් ඇත. මෙහි screenshots mock-ups බව අවශ්‍ය තැන සඳහන් වේ.

අන්තර්ගතය / Roadmap

සැලසුම් කිරීමේ සිට ගිණුම් සහ Tasks කළමනාකරණය දක්වා සම්පූර්ණ ගමනක්.

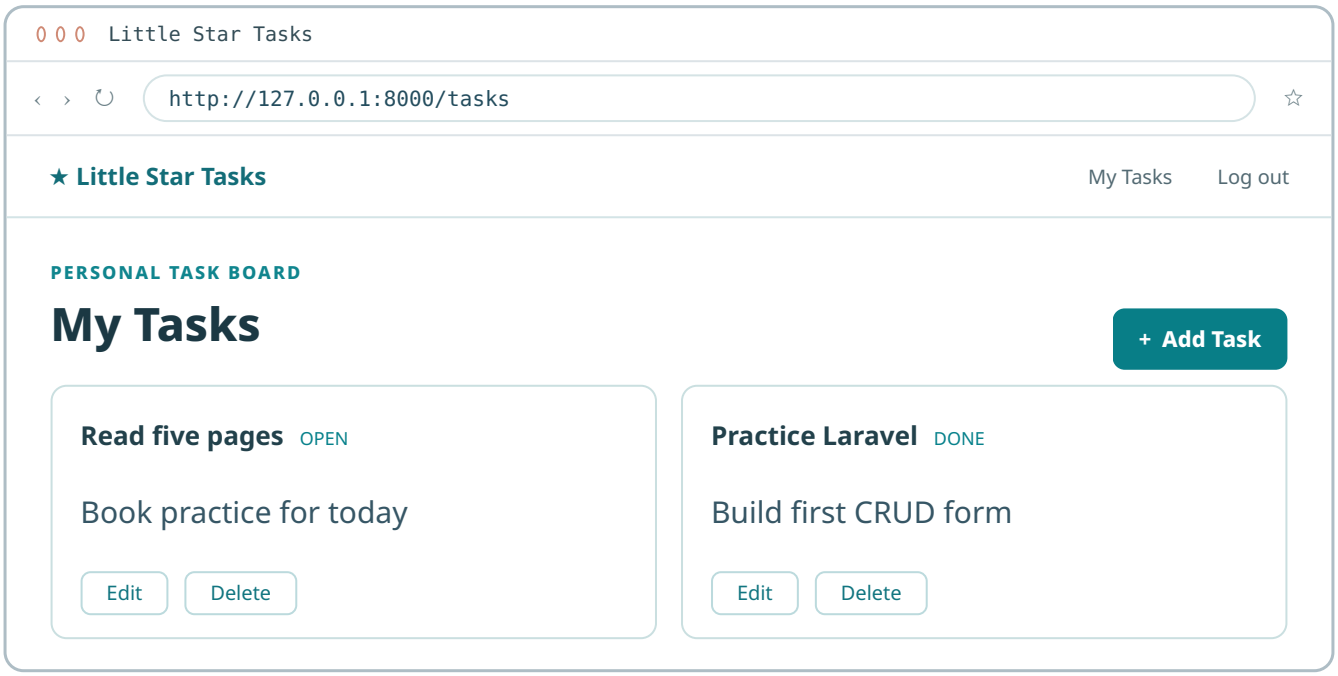
01 • හැඳින්වීම	6 Lessons
02 • පරිගණකය සූදානම්	10 Lessons
03 • PHP + MVC	6 Lessons
04 • Routes සහ Views	9 Lessons
05 • Database	7 Lessons
06 • Login සහ Security	10 Lessons
07 • CRUD Development	13 Lessons
08 • Git	2 Lessons
08 • Testing	8 Lessons
09 • Production	9 Lessons
10 • Hands-on Labs	12 Labs
11 • MCQ & Answers	12 Questions
12 • Complete Source Appendix	Source files

අපි අවසානයේ හදන app එක

Example Project

Little Star Task Manager

ඔබට tasks add, read, edit, delete කිරීමට හැකි වෙනවා. Account දෙකකට අයත් Tasks එකිනෙකට පෙනෙන්නේ නැහැ.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

අත්‍යවශ්‍ය output

Register → Login → Task list → Create → Update → Delete → Logout. මේ ක්‍රියා 7ම සාර්ථකද? ඒවාට test cases ලියාගන්න.

A. Laravel කියන්නේ මොකක්ද?

Laravel කියන්නේ PHP භාෂාවෙන් web application හදන්න උදව් කරන framework එකක්.

ගෙයක් හදනකොට දොර, ජනෙල්, බිත්ති සඳහා සැලැස්මක් අවශ්‍යයි. Laravel ද ඒ වගේ සූදානම් පද්ධතියක්: Routes, Controllers, Views, Models, Validation සහ Security කොටස් තිබෙනවා. ඔබට හැම දෙයක්ම මුල සිට ලියන්න අවශ්‍ය නැහැ.



අභ්‍යාසය: ඔබ දන්නා website එකක Login සහ Data Save කොටස් දෙකක් නම් කරන්න.

Framework එක code සංවිධානය කර දෙන උපකරණ පෙට්ටියක්.

B. ඇයි Laravel ඉගෙනගන්නේ?

නිතරම නැවත නැවත ලියන web code අඩු කිරීමයි අරමුණ.

සාමාන්‍ය PHP file ගොඩක් එකට සම්බන්ධ කිරීම කාලයක් ගනී. Laravel තුළ route එකෙන් request එක controller එකට, controller එකෙන් model හරහා database එකට යවන සැලැස්මක් ඇත. කණ්ඩායමක් එක්ක වැඩ කිරීමටත් ඉක්මනින් feature එක් කිරීමටත් ඉතා හොඳයි.



අභ්‍යාසය: MVC කියන අකුරු තුනේ තේරුම සොයා ලියන්න.

Model = data, View = screen, Controller = decisions.

A. 2026 සඳහා නිවැරදි versions

මෙම පොත Laravel 13 සහ PHP 8.3+ මත පදනම් වෙයි.

Laravel 13 නිකුත් වුණේ 2026 මාර්තු 17. PHP අවම version එක 8.3 යි. පරිගණකයේ PHP 8.4 තිබුණත් භාවිතා කළ හැකියි. Laravel 11/12 tutorial එකක instructions මෙයට සම්පූර්ණයෙන්ම ගැළපේ කියලා නොසිතන්න.

Windows PowerShell / CMD

```
php -v
composer --version
```

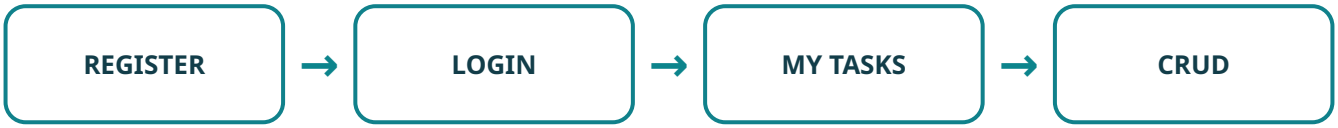
අභ්‍යාසය: CMD එකේ commands දෙකම ධාවනය කරන්න.

Versions පෙනෙන්නේ නම් setup එකේ පළමු පියවර සාර්ථකයි.

B. අපි හදන්නේ කුමක්ද?

Little Star Task Manager කියන්නේ පුද්ගලික වැඩ ලැයිස්තුවක්.

User කෙනෙක් Register වෙනවා, Login වෙනවා, Task එකක් එකතු කරනවා, List එක බලනවා, Edit/Delete කරනවා. අනිත් user කෙනෙකුගේ වැඩ නොපෙනීම අනිවාර්ය අවශ්‍යතාවක්. මේක banking app එකක් නොව ඉගෙනුම් project එකක්.



අභ්‍යාසය: ඔබේ Task Manager එකට නමක් සහ කොටස් 3ක් යෝජනා කරන්න.

මූලික හැකියාවන්: Account + Own Tasks + Logout.

A. CRUD කියන අකුරු හතර

Create, Read, Update, Delete කියන්නේ දත්ත සමඟ කරන මූලික වැඩ හතරයි.

Create = අලුත් Task එකක් දාන්න. Read = tasks බලන්න. Update = තිබෙන Task එක වෙනස් කරන්න. Delete = Task එකක් ඉවත් කරන්න. සෑම ක්‍රියාවකටම browser request එකක් සහ database ක්‍රියාවක් තිබෙනවා.

```
Windows PowerShell / CMD

POST /tasks
GET /tasks
PUT /tasks/{task}
DELETE /tasks/{task}
```

අභ්‍යාසය: ඔබගේ පාසල් වැඩ ලැයිස්තුවෙන් මේ ක්‍රියා හතරට උදාහරණ ලියන්න.

CRUD යනු database පමණක් නොව UI සහ validation ඇතුළත් workflow එකකි.

B. වෙබ් පද්ධතියේ කොටස්

Browser, PHP server, Laravel app, database යන කොටස් එකට වැඩ කරනවා.

ඔබ “Save” ක්ලික් කළ විට browser එක HTTP POST request එකක් යවයි. Laravel එය validate කරලා database එකට ලියයි. පසුව redirect කරලා අලුත් Task list එක browser එකට පෙන්වයි.



අභ්‍යාසය: “Save” button ක්ලික් කළාම data යන මඟ ඊතල 5කින් අඳින්න.

View එකෙහි වැඩ data පෙන්වීම; Model එකෙහි වැඩ data සමඟ කටයුතු කිරීම.

A. SDLC පියවරෙන් පියවර

Software Development Life Cycle (SDLC) යනු පද්ධතියක් හදන ගමනයි.

අපි Requirements ලියලා UI sketch අඳිමු. පසුව Database plan, Coding, Testing, Deployment සහ Maintenance කරමු. කුඩා project එකකටත් මේ ක්‍රමය වටිනවා; නැතිනම් password rules හෝ delete confirmation වැනි අවශ්‍යතා අමතක විය හැකියි.



අභ්‍යාසය: අද කරන්නා වූ වැඩ සඳහා SDLC checklist එකක් සකස් කරන්න. මුලින් requirements, ඉන්පසු coding.

B. SRS කියන්නේ මොකක්ද?

Software Requirements Specification එකෙන් “අපි හදන්නේ මොනවාද?” පැහැදිලි කරයි.

FR-01: User register විය යුතුයි. FR-02: Login/Logout. FR-03: Tasks CRUD. FR-04: වෙන user කෙනෙකුගේ Task එකක් edit කළ නොහැකි විය යුතුයි. NFR-01: Password hash කළ යුතුයි. NFR-02: Mobile screen එකක කියවිය හැකි විය යුතුයි.



අභ්‍යාසය: ඔබටම නව FR එකක් සහ NFR එකක් ලියන්න.

SRS නොමැතිව coding පටන්ගැනීමෙන් test කිරීම අපහසුයි.

09. User Stories

Software එක භාවිතා කරන පුද්ගලයාගේ අවශ්‍යතාව වාක්‍යයකින් ලියමු.

සරල පැහැදිලි කිරීම

“As a learner, I want to save a task so that I remember my homework.” වැනි format එකක් යොදන්න. Acceptance Criteria: title හිස් නම් save නොවිය යුතුයි; save වුණොත් list එකේ පෙනිය යුතුයි.



පුංචි අභ්‍යාසය

“As a user …” යන format එකෙන් stories දෙකක් සකස් කරන්න.

මතක තබාගන්න

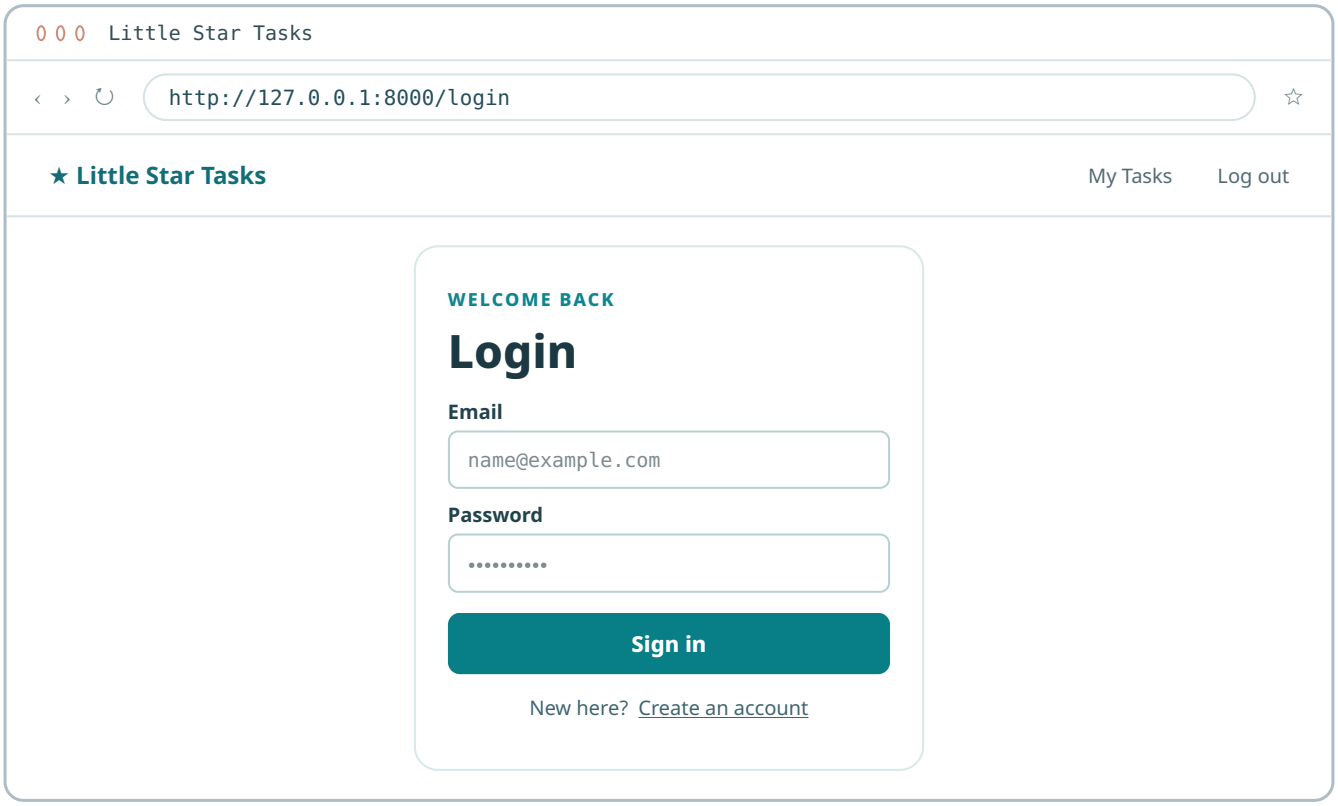
Story + Acceptance Criteria = test එකකට පදනම.

10. UI Wireframe

Coding කලින් screen එක කඩදාසියක අඳින්න.

සරල පැහැදිලි කිරීම

Login page එකට Email, Password, Sign in button සහ Register link අවශ්‍යයි. Dashboard එකට Add Task button, Filter සහ Task cards අවශ්‍යයි. Mobile screen එකක cards එක column එකක් ලෙස පෙන්වීමට සැලසුම් කරන්න.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

පුංචි අභ්‍යාසය

කඩදාසියක Login form එක අඳින්න; අත්‍යවශ්‍ය inputs රවුම් කරන්න.

මතක තබාගන්න

Sketch එකේ සිට code එකට යන විට missing buttons අඩුවේ.

A. Database ER Diagram

users table එක සහ tasks table එක අතර One-to-Many සම්බන්ධයක් තිබෙනවා.

User එකකට Task බොහෝ ගණනක් තිබිය හැකියි. නමුත් එක් Task එකකට අයිති වන්නේ එක user කෙනෙකු පමණයි. `tasks.user_id` හරහා `users.id` සම්බන්ධ වෙනවා. Database schema එකේ FK constraint ද තබනවා.



අභ්‍යාසය: Task table එකට අලුත් column එකක් යෝජනා කරන්න.

Foreign Key එක ownership එක දක්වන සබඳතාවයි.

B. Folder map එක දැනගනිමු

Laravel project එකක files වෙන වෙනම folder වල තියෙන්නේ ඇයි?

`routes/web.php` = URL මාර්ග. `app/Http/Controllers` = request decisions. `app/Models` = database models. `resources/views` = Blade HTML. `database/migrations` = database structure. `tests/Feature` = user journeys පරීක්ෂා කිරීම්.

```
Windows PowerShell / CMD

app/
├── Http/Controllers/
└── Models/
routes/web.php
resources/views/
database/migrations/
```

අභ්‍යාසය: මෙම paths පහ notes එකක ලියාගන්න.

එක file එකකට සියලු code ලියන්නේ නැහැ.

A. අවශ්‍ය software

PHP, Composer, VS Code, database සහ browser එක පළමු tools වේ.

Windows පරිගණකයක PHP 8.3+ command line සහ extensions, Composer, VS Code, Git සහ Chrome/Edge browser එක සූදානම් කරන්න. SQLite භාවිතා කරන නිසා XAMPP හෝ MySQL ස්ථාපනය මුලින්ම අත්‍යවශ්‍ය නැහැ.

```
Windows PowerShell / CMD

php -v
composer -V
git --version
```

අභ්‍යාසය: ඔබේ versions screenshot එකක් හෝ note එකක් ලෙස සුරකින්න.

Tool එකක් නැතිනම් installation නිවැරදි කරලා ඉදිරියට යන්න.

B. Composer යනු කවුද?

Composer කියන්නේ PHP libraries කළමනාකරණය කරන package manager එක.

Framework එකේ අවශ්‍ය packages අතින් download නොකර Composer මඟින් ලබාගන්න පුළුවන්. `composer.json` කියන්නේ recipe එක. `vendor/` තුළ installed packages ඇත. `vendor/` GitHub වෙත push කරන්න එපා.

```
Windows PowerShell / CMD

composer --version
composer create-project laravel/laravel demo "^13.0"
```

අභ්‍යාසය: Command එක ධාවනය කිරීමට පෙර internet තිබේද බලන්න.

Composer packages නිවැරදි folders වලට ගෙනෙයි.

A. Fresh Laravel 13 project එක

මුලින් project skeleton එක Composer හරහා හදමු.

මෙම source ZIP එක Laravel framework එකම අඩංගු package එකක් නොවේ. මුලින් official skeleton එක Composer මඟින් ලබාගෙන, tutorial ZIP එකේ `overlay` folders ඒ project එකට merge කරන්න. මෙයින් version update කිරීමට පහසු වේ.

```
Windows PowerShell / CMD

composer create-project laravel/laravel little-star-tasks "^13.0"
cd little-star-tasks
```

අභ්‍යාසය: `artisan` සහ `composer.json` files දෙක පෙනේද බලන්න.

කාලය වැඩියෙන් ගත වුණත් install එක අවසන් වනතුරු output errors පරීක්ෂා කරන්න.

B. VS Code තුළ project විවෘත කරන්න

දැන් project files ඔබේ editor එකේ පෙනෙන්න සැලසුම් කරමු.

VS Code හි File → Open Folder භාවිතයෙන් `little-star-tasks` folder එක තෝරන්න. Explorer එකෙන් folder tree එක බලන්න. Terminal → New Terminal දී CMD හෝ PowerShell එකක් විවෘත කරන්න.

```
Windows PowerShell / CMD

code .
php artisan list
```

අභ්‍යාසය: `artisan` file එක තිබෙන directory එකේම commands run කරන්න.

VS Code code editor; terminal commands run කරන තැන.

A. Application key සහ .env

`.env` file එක ඔබගේ දේශීය settings සඳහාය.

`.env.example` එකෙන් `.env` සාදන්න. APP_KEY හිස් නම් `php artisan key:generate` භාවිතා කරන්න. `.env` කිසිවිට GitHub වෙත push කරන්න එපා; passwords, keys, DB credentials රහසක් ලෙස තබන්න.

```
Windows PowerShell / CMD
```

```
Copy-Item .env.example .env
php artisan key:generate
```

අභ්‍යාසය: Git status එකෙන් `.env` stage වෙනවාද පරීක්ෂා කරන්න.

`.env.example` share කළ හැකි template එකක්; `.env` private.

B. SQLite database සූදානම්

කුඩා project එකකට SQLite ඉතා පහසුයි.

SQLite database එක තනි file එකක්. Server එකක් run කරන්න අවශ්‍ය නැහැ. මුලින්ම `database/database.sqlite` ගොනුව සාදා `.env` හි `DB_CONNECTION=sqlite` තබන්න. Windows PowerShell command එක මෙහි ඇත.

```
Windows PowerShell / CMD
```

```
New-Item -ItemType File -Force database/database.sqlite
# .env: DB_CONNECTION=sqlite
```

අභ්‍යාසය: `database/database.sqlite` file එක පෙනේද බලන්න.

Real customer data project එකකට යන්න කලින් backup policy එකක් අවශ්‍යයි.

15. Migration ධාවනය

Migration එක කියන්නේ database table සෑදීමේ version-controlled plan එකයි.

සරල පැහැදිලි කිරීම

Laravel හි default users table එකත්, අපේ tasks table එකත් migrations වලින් සාදමු. `migrate` command එකෙන් database structure සකස් වේ. Error එකක් ලැබුණොත් file path සහ DB_CONNECTION නැවත බලන්න.

Windows PowerShell / CMD

```
php artisan migrate
php artisan migrate:status
```

පුංචි අභ්‍යාසය

`tasks` migration එක `Ran` ලෙස පෙනෙනවාද පරීක්ෂා කරන්න.

මතක තබාගන්න

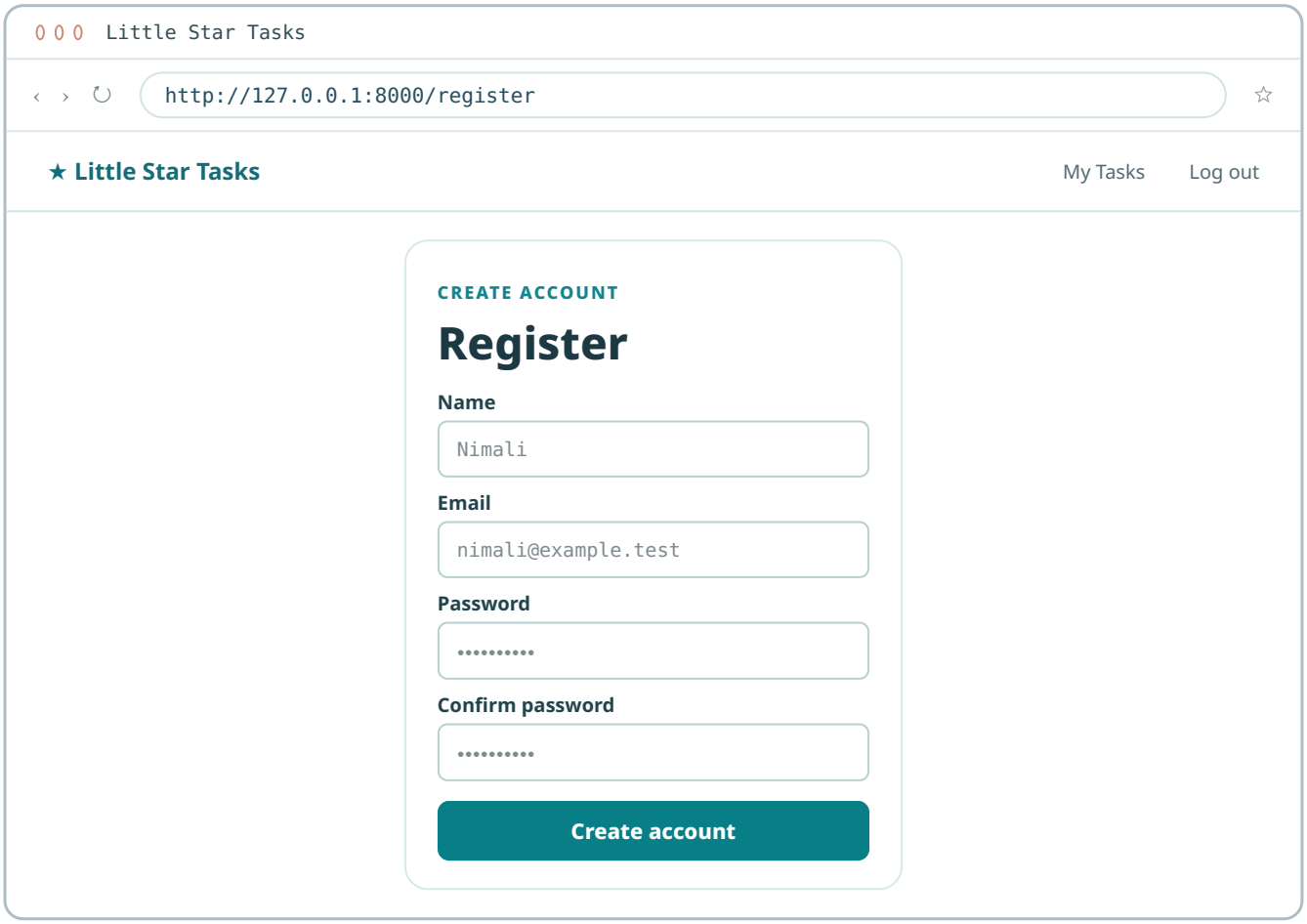
Migration code එකෙන් DB structure එක teammates ටත් නැවත ගොඩනැඟිය හැකියි.

16. Development server පටන්ගනිමු

Laravel app එක ඔබේම පරිගණකයේ browser එකෙන් බලමු.

සරල පැහැදිලි කිරීම

Terminal එකේ `php artisan serve` run කරලා `127.0.0.1:8000` වෙත යන්න. `localhost` කියන්නේ ඔබේ පරිගණකය. මෙය production server එකක් නොවේ. Terminal නවත්වන්න Ctrl+C භාවිතා කරන්න.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

පුංචි අභ්‍යාසය

Browser එකේ Register screen එක පෙනේද බලන්න.

මතක තබාගන්න

Server start නොවුණොත් terminal error එක කියවන්න.

A. Project overlay එක එකතු කරන්න

ZIP එකේ Laravel app-specific files නිසි folders වෙත copy කරන්න.

`overlay/routes/web.php` වැනි paths අලුත් project folder එකේ එම paths මත merge කළ යුතුයි. `app/Models/User.php` ද මෙම tutorial එකේ `tasks()` relationship සමඟම තිබෙනවා. Custom project එකක් නම් User model එක overwrite නොකර merge කරන්න.



අභ්‍යාසය: README_SINHALA.md checklist එකෙන් සෑම file එකක්ම හරි තැන තිබේද බලන්න.

ZIP එකේ vendor/ නැත්තේ නිවැරදි තීරණයක්.

B. Artisan commands 5ක්

Artisan කියන්නේ Laravel CLI helper එකයි.

`route:list` එක routes පෙන්වයි. `make:controller` controller skeleton එක හදයි. `make:model -m` model සහ migration එකටම scaffolding සපයයි. `test` automated tests run කරයි. `migrate:status` DB changes බලයි.

Windows PowerShell / CMD

```
php artisan route:list
php artisan make:controller SampleController
php artisan make:model Note -m
php artisan test
php artisan migrate:status
```

අභ්‍යාසය: SampleController සහ Note files lesson එක අවසානයේ practice project එකෙන් ඉවත් කරන්න.

Artisan එක coding වේගවත් කරන helper එකක්.

A. PHP variable එක

Variable කියන්නේ value එකක් තබන නමක්.

PHP හි variable නම `$` සමඟ පටන්ගන්නවා. String, integer, boolean වැනි types තියෙනවා. Laravel controllers ද PHP code මත නිර්මාණය වී තිබෙන නිසා මේ පදනම වැදගත්.

Windows PowerShell / CMD

```
<?php
$name = "Nimali";
$count = 3;
$done = false;
echo $name;
```

අභ්‍යාසය: වෙනම `practice.php` file එකක් සාදා `php practice.php` run කරන්න. Expected: Nimali

B. If condition එක

Condition එකක් අනුව වෙනස් ක්‍රියා කරන්න පුළුවන්.

Login සාර්ථක නම් dashboard එක, අසාර්ථක නම් error එක පෙන්වන්න if/else භාවිතා කරයි. `===` කියන්නේ type එකත් සමානද බලන comparison එකක්.

Windows PowerShell / CMD

```
<?php
$isLoggedIn = true;
if ($isLoggedIn === true) {
    echo "Dashboard";
} else {
    echo "Login first";
}
```

අභ්‍යාසය: `$isLoggedIn = false;` කළාම output වෙනස් කර බලන්න. Expected: Login first

A. Array සහ foreach

Task ලැයිස්තුවක් මත loop එකක් ධාවනය කරමු.

Tasks ගොඩක් පෙන්වීමට එකම HTML markup නැවත නැවත ලියන්නේ නැහැ. Blade views තුළ `@foreach` භාවිතා කරයි. PHP array එක සහ Blade loop එක අතර සම්බන්ධතාව තේරුම් ගන්න.

Windows PowerShell / CMD

```
<?php
$tasks = ["Read", "Draw", "Code"];
foreach ($tasks as $task) {
    echo $task . "\n";
}
```

අභ්‍යාසය: Tasks තුනක් වෙනස් කරලා output බලන්න.

Loop එකේ වැඩ item එකින් item එකට යාමයි.

B. Function එක

Function කියන්නේ නැවත භාවිතා කළ හැකි code කැබැල්ලක්.

Laravel Controller එකක `index`, `store`, `update` වැනි methods යනු class එක තුළ ඇති functions. Function එකකට inputs දී result එකක් ගන්න පුළුවන්.

Windows PowerShell / CMD

```
<?php
function greeting(string $name): string {
    return "Hello " . $name;
}
echo greeting("Amal");
```

අභ්‍යාසය: `greeting("Nimali")` run කර බලන්න.

Expected: Hello Nimali

A. Class සහ Object

Class යනු template එකක්; Object යනු ඒ template එකේ instance එකක්.

Laravel හි Task model සහ TaskController class ලෙස තියෙනවා. `new` keyword එකෙන් object හදන්න පුළුවන්. මේ අවස්ථාවේ OOP සියල්ල මතක තබාගන්න ඕන නැහැ; `class`, `public function`, `$this` තේරුම් ගන්න.

Windows PowerShell / CMD

```
<?php
class Book {
    public string $title = "My Notes";
}
$book = new Book();
echo $book->title;
```

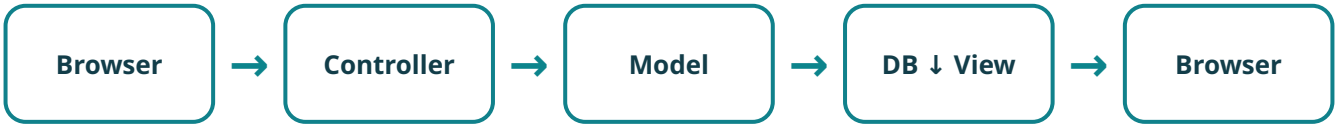
අභ්‍යාසය: Class නම Book වෙනුවට Task කරලා බලන්න.

Object එකට properties සහ methods තියෙන්න පුළුවන්.

B. MVC රූපය

Model–View–Controller එකිනෙකට වෙනස් වැඩ කරනවා.

Browser request එක Controller එකට එනවා. Controller එක Model එකෙන් data ඉල්ලනවා. Model එක DB සමඟ කටයුතු කරනවා. Controller එක View එකට data යවනවා; View එක HTML output කරයි.



අභ්‍යාසය: `TaskController.php`, `Task.php`, `index.blade.php` paths තුන හඳුනාගන්න.

MVC හොඳ structure එකක්; security එයින් ස්වයංක්‍රීයව සම්පූර්ණ වන්නේ නැහැ.

A. Route එක යනු URL පාර

`routes/web.php` තුළ URL එකකට response එකක් සම්බන්ධ කරයි.

GET request එකකින් page එකක් බලනවා. POST request එකකින් data save කරනවා. `name()` දුන්නාම code එක තුළ path hardcode නොකර `route('login')` භාවිතා කළ හැකියි.

Visual Studio Code routes/web.php

EXPLORER
app/
routes/
views/

```
Route::get("/hello", function () {
    return "Hello Laravel!";
})->name("hello");
```

අභ්‍යාසය: තම project එකේ temporary `/hello` route එකක් හදලා URL එක බලන්න. GET /hello → Hello Laravel!

B. Controller එකේ වගකීම

Controller එක request එකක් ලැබුණු පසු ඊළඟ පියවර තීරණය කරයි.

එක route එකට HTML, SQL, validation, login logic සියල්ලම ලියන්න එපා. Controller method එකේ validation, model calls සහ redirect වැනි තීරණ තබන්න. View එකේ HTML output පමණක් තබාගන්න.



අභ්‍යාසය: Route එකෙන් Controller class එකට යන ඊතලයක් අඳින්න. Business logic වෙනම තැබීම testing පහසු කරයි.

A. Blade templates

Blade යනු Laravel තුළ HTML ලිවීමට පහසු template syntax එක.

`.blade.php` file එකක `{{ $name }}` මඟින් text display කළ හැකියි. මේ output escape කරන නිසා XSS අවදානම අඩු වේ. `@foreach`, `@if`, `@extends`, `@section` ද useful.

Visual Studio Code — resources/views/tasks/index.blade.php

EXPLORER
app/
routes/
views/

```
<h1>Hello, {{ $name }}</h1>
@if($tasks->isEmpty())
  <p>No tasks yet.</p>
@endif
```

අභ්‍යාසය: `{{ }}` සහ `{!! !!}` අතර ආරක්ෂක වෙනස ලියාගන්න.

සාමාන්‍ය user data පෙන්වීමට escaped `{{ }}` භාවිතා කරන්න.

B. Layout එකක් භාවිතා කිරීම

Header, footer, styles එකට තබා පිටු කිහිපයක භාවිතා කරමු.

`layouts/app.blade.php` තුළ navbar සහ footer තබනවා. Login, Register, Tasks පිටු `@extends` මඟින් ඒ layout එක භාවිතා කරයි. `@yield('content')` කියන්නේ ඇතුළු content එක තැබෙන තැනයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
@extends("layouts.app")
@section("content")
  <h1>My Tasks</h1>
@endsection
```

අභ්‍යාසය: Login page එකත් dashboard එකත් same header භාවිතා කරනවාද බලන්න.

Reusable layouts ප්‍රතිසංස්කරණ පහසු කරයි.

A. HTTP Methods

GET, POST, PUT, DELETE අරමුණු වෙනස්ය.

GET = දත්ත කියවීම. POST = අලුත් record. PUT/PATCH = වෙනස් කිරීම. DELETE = මකාදැමීම. HTML form එක GET සහ POST පමණක් මූලිකව යවන නිසා Laravel හි `@method('PUT')` වැනි method spoofing භාවිතා කරයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
<form method="POST" action="/tasks/5">
  @csrf
  @method("DELETE")
  <button>Delete</button>
</form>
```

අභ්‍යාසය: Browser DevTools Network tab එකේ method එක සහ status බලන්න.

Method එක user permission සඳහා ප්‍රමාණවත් security check එකක් නොවේ.

B. CSRF protection

වෙන website එකකින් ඔබ වෙනුවට action එකක් යවන්න ඉඩ නොදීම වැදගත්.

POST, PUT, PATCH, DELETE HTML forms සියල්ලටම `@csrf` දාන්න. Laravel request token එක validate කරයි. CSRF ආරක්ෂාව තිබුණත් authentication සහ owner checks වෙනම අවශ්‍යයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
<form method="POST" action="{{ route('logout') }}">
  @csrf
  <button>Log out</button>
</form>
```

අභ්‍යාසය: Source project එකේ සියලු forms වල `@csrf` තිබේද ගණන් කරන්න.

CSRF token එක රහස් password එකක් නොව request authenticity token එකකි.

A. HTML forms

Browser එකෙන් typed text Laravel request එකකට යයි.

Input එකේ `name="title"` යන්න Laravel request field නමයි. `type="email"` වැනි browser validation උදව්වත් server side validate කිරීම අනිවාර්යයි. Form labels ද accessibility සඳහා වැදගත්.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
<label>Task title
  <input name="title" required maxlength="120">
</label>
```

අභ්‍යාසය: Input එකට title එකක් දී submit කළ විට server-side rule එකට ගැළපේද බලන්න.

Browser validation මඟහැරිය හැකි නිසා backend validation නොහැරන්න.

B. Validation මුල සිට

User යවන සියලු data නිවැරදි යැයි නොසිතන්න.

Title එක හිස් නම් reject. Characters 120 ඉක්මවුවොත් reject. Due date එක වලංගු date එකක් විය යුතුයි. `validate()` වැරදි inputs සඳහා errors සහ old inputs සමඟ redirect කළ හැකියි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$data = $request->validate([
  "title" => ["required", "string", "max:120"],
  "due_date" => ["nullable", "date"],
]);
```

අභ්‍යාසය: හිස් title එකක් submit කර error message එක නිරීක්ෂණය කරන්න.

Validation messages user ට ඉදිරියට යාමට උදව් කරයි.

25. Redirect සහ flash messages

Form එකක් save කළ පසු එකම POST නැවත submit වීම අඩු කරමු.

සරල පැහැදිලි කිරීම

`redirect()->route('tasks.index')` යවලා dashboard එකට නැවත යන්න. `with('success',...)` මඟින් එක් request එකකට පණිවිඩයක් පෙන්වන්න පුළුවන්. මෙයට Post/Redirect/Get pattern කියනවා.

Visual Studio Code — Laravel / PHP

EXPLORER

little-star-tasks/
├ app/
├ routes/
├ resources/
├ database/
└ tests/

```
return redirect()->route("tasks.index")
    ->with("success", "Task created.");
```

පුංචි අභ්‍යාසය

Save කර browser refresh කළ විට duplicate record එකක් නොවන බව බලන්න.

මතක තබාගන්න

PRG pattern නැවත form POST වීම අඩු කරයි.

A. Model එකක් හඳුනාගනිමු

Laravel Eloquent Model එක DB table එකක් නිරූපණය කරයි.

`Task` model එකට tasks table එකට කියවීම් සහ වෙනස්කම් කළ හැකියි. `fillable` තුළ අවසර ලැබෙන fields තබයි. `user_id` user form එකෙන් විශ්වාස නොකර signed-in user relationship එකෙන් set කරනවා.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
class Task extends Model {
  protected $fillable = ["title", "details", "due_date", "is_done"];
}
```

අභ්‍යාසය: `user_id` fillable list එකට නොදැම්මේ ඇයි කියලා සිතන්න.

Ownership server side එකෙන් තීරණය කළ යුතුයි.

B. Migration එකේ කොටස්

Table එකේ columns සහ foreign keys migration එකකින් නිර්මාණය වේ.

`id` primary key. `user_id` foreign key. `title` varchar(120). `details` optional. `due_date` nullable. `is_done` boolean. timestamps මඟින් created_at/updated_at. Database schema එක code එකේ version control යටතේ තබයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$table->id();
$table->foreignId("user_id")->constrained()->cascadeOnDelete();
$table->string("title", 120);
$table->boolean("is_done")->default(false);
```

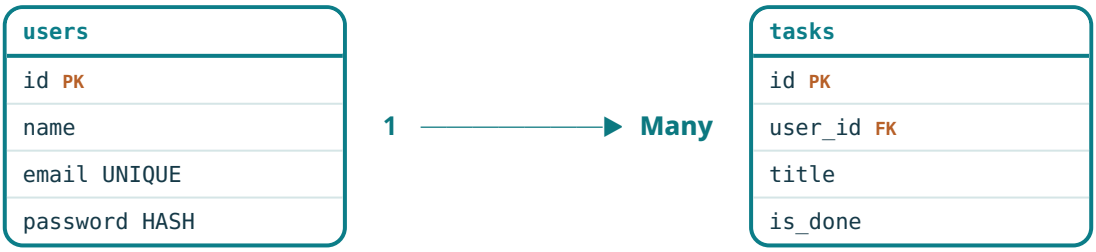
අභ්‍යාසය: Migration සහ model fields එකිනෙකට ගැළපේද සසඳන්න.

Foreign key data integrity එක ශක්තිමත් කරයි.

A. One user has many tasks

`users.id` → `tasks.user_id` මඟින් සම්බන්ධ කරමු.

User model එකේ `tasks(): HasMany` method එක තිබේ. Task model එකේ `user(): BelongsTo` method එක තිබේ. මෙයින් `$request->user()->tasks()` කියලා තමන්ගේ data පමණක් පහසුවෙන් කියවිය හැකියි.



අභ්‍යාසය: User A සහ User B වෙන වෙනම tasks ගබඩා වීම ඇඳ පෙන්වන්න.

Model relation යනු DB foreign-key relationship එකට Laravel interface එකක්.

B. SQLite තුළ data යන හැටි

DB file එකක් ඇතුළේ table rows ගබඩා වෙනවා.

New task එකක් save කළාම `tasks` table එකට අලුත් row එකක් යයි. `user_id` user account එකට සම්බන්ධ කරයි. Table එක අතපසු කරලා editor එකකින් SQLite binary එක සෘජුව වෙනස් කරන්න එපා.

DATA EXAMPLE

```
id | user_id | title          | is_done
1  | 2       | Read a book    | 0
2  | 2       | Practice PHP   | 1
```

අභ්‍යාසය: ඉහත rows දෙක User 2 ට අයත් කියලා හඳුනාගන්න.

Database rows එකිනෙක වෙනස්ව පවතී.

A. Eloquent create

PHP array data එක row එකක් බවට පත් කරමු.

Controller එක request validate කරලා `$request->user()->tasks()->create($data)` හාවිතා කරයි. මේ relationship එක user_id ඝඇයෙන් set කරයි. Form එකේ hidden user_id දී ownership තීරණය කරන්න එපා.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$data = $request->validate([...]);
$request->user()->tasks()->create($data);
```

අභ්‍යාසය: Data save කිරීමට පෙර validation කරන තැන පෙන්වන්න.

Always trust the server-side signed-in user ID.

B. Pagination සහ filters

Records 1000ක් තිබුණත් සියල්ල එක පිටුවක පෙන්වන්න එපා.

`paginate(8)` මඟින් එක් page එකකට 8 tasks පෙන්වයි. `where('is_done', true)` වැනි filter එකක් යෙදිය හැකියි. `withQueryString()` filter එක pagination links වලට තබාගන්න උදව් කරයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$tasks = $request->user()->tasks()
    ->latest()->paginate(8)->withQueryString();
```

අභ්‍යාසය: Tasks 9ක් සාදා pages දෙකක් හටගන්නේද බලන්න.

Pagination UI වේගවත් සහ පැහැදිලි කරයි.

29. MySQL වෙත මාරු වීම

SQLite ඉගෙනගැනීමට හොඳයි; පසුව MySQL භාවිතා කළ හැකියි.

සරල පැහැදිලි කිරීම

`.env` හි DB_CONNECTION=mysql, DB_HOST, DB_PORT, DB_DATABASE, DB_USERNAME, DB_PASSWORD සකස් කරලා MySQL database එක කලින් හදන්න. SQLite data ස්වයංක්‍රීයව MySQL වෙත යන්නේ නැහැ; වෙනම migration/import සැලසුමක් අවශ්‍යයි.

Windows PowerShell / CMD

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=little_star
DB_USERNAME=your_local_user
```

පුංචි අභ්‍යාසය

Demo data backup එකක් ගෙන පසුව පමණක් database switch කරන්න.

මතක තබාගන්න

`.env` password screen share නොකරන්න.

A. Login workflow එක

Login යනු user කවුදැයි හඳුනාගැනීමයි.

Registration → Password hashing → Login credentials check → Session cookie → Protected pages → Logout යන ගමන මතක තබාගන්න. `auth` middleware හරහා ආරක්ෂිත pages සම්බන්ධ කරයි.



අභ්‍යාසය: Session cookie එක password එක නොවන බව පැහැදිලි කරන්න.

Login තිබීම පමණක් owner authorization සඳහා ප්‍රමාණවත් නැහැ.

B. Registration controller

නව user account එකක් save කිරීමට පෙර data validate කළ යුතුයි.

Email unique විය යුතුයි. Password confirmation එක සමාන විය යුතුයි. Password අවම අකුරු 8කින් තබනවා (නිෂ්පාදන පද්ධතියකට වැඩි policy අවශ්‍ය විය හැක). User record එක save වූ පසු login session එක ආරම්භ කරයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$data = $request->validate([
 "email" => ["required", "email", "unique:users,email"],
 "password" => ["required", "confirmed", Password::min(8)],
]);
```

අභ්‍යාසය: එකම email දෙවරක් register කරන විට system එකේ හැසිරීම බලන්න.

Validation නැති account creation අනතුරුදායකයි.

A. Passwords hash කරන්නේ ඇයි?

Password එක database එකේ plain text ලෙස නොතබන්න.

`Hash::make()` password එක ආරක්ෂිත hash එකකට පරිවර්තනය කරයි. මෙය encryption එකක් නොව one-way process එකක්. Laravel `Auth::attempt` එම hash එකට password ගැළපේද පරීක්ෂා කරයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
"password" => Hash::make($data["password"]),
```

අභ්‍යාසය: Database එකේ `password` column එකේ raw password නොතිබෙන බව පරීක්ෂා කරන්න.

Hash එකක් නැවත original password එකට හැරවීමට අදහස් නොකරයි.

B. Auth::attempt()

Login credentials නිවැරදිද කියලා Laravel පරීක්ෂා කරයි.

`Auth::attempt` success නම් `session()->regenerate()` කරන්න. මෙය session fixation වැනි අවදානම් අඩු කරයි. Fail නම් password එක browser වෙත නැවත නොයවන්න; email පමණක් old input එකක් ලෙස ගබඩා කළ හැකියි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
if (Auth::attempt($credentials)) {
    $request->session()->regenerate();
    return redirect()->route("tasks.index");
}
```

අභ්‍යාසය: වැරදි password එකකින් login උත්සාහ කර message එක බලන්න.

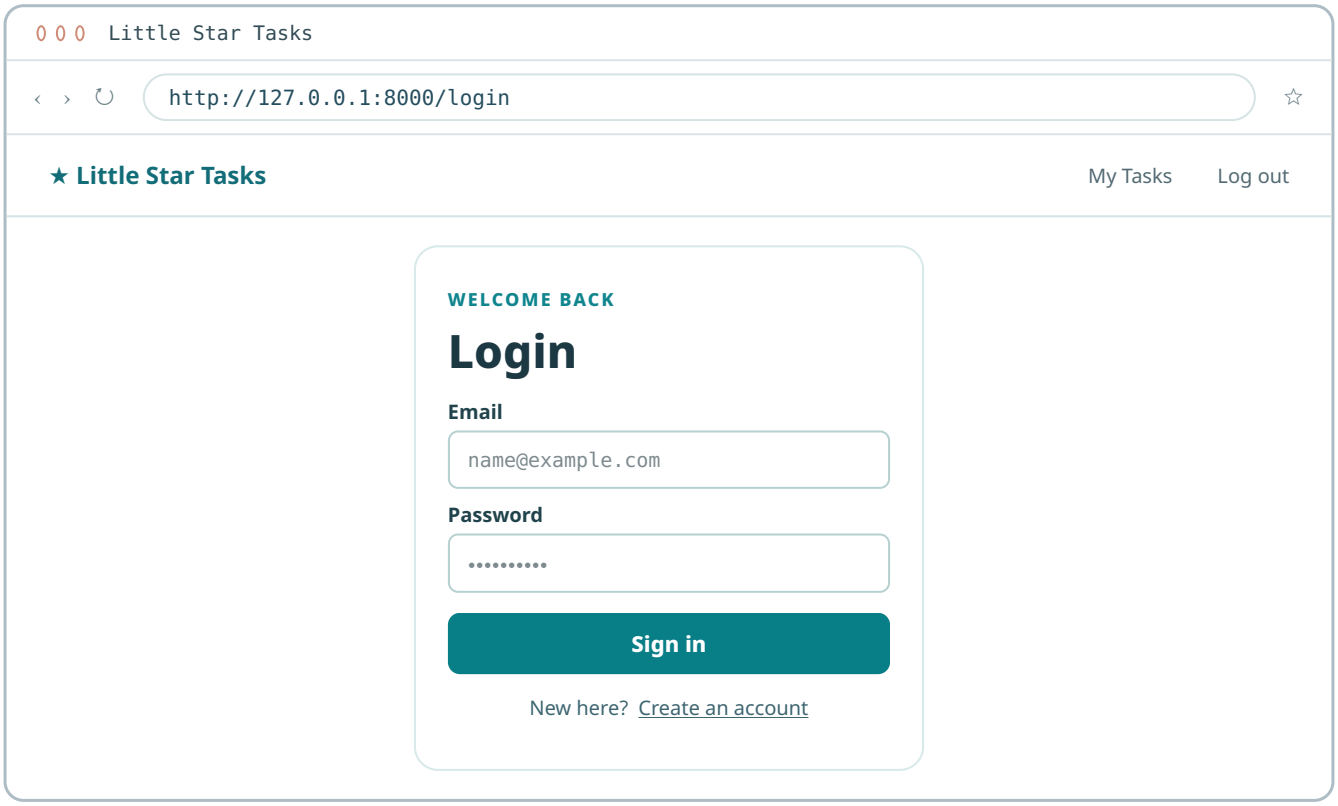
Login success යනවා කියලා audit/security සම්පූර්ණ වෙන්නේ නැහැ.

32. Login screen preview

ඇතුළත් කරන්නේ email සහ password පමණයි.

සරල පැහැදිලි කිරීම

Form එකේ `@csrf`, `type="email"`, `type="password"`, labels සහ submit button තියෙනවා. Register link එක තබා user ට හොඳ navigation එකක් සපයනවා. Error message එක කෙටි සහ ආරක්ෂිත විය යුතුයි.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

පුංචි අභ්‍යාසය

Login screen එකේ keyboard එකෙන් Tab භාවිතා කර fields මාරු කළ හැකිද බලන්න.

මතක තබාගන්න

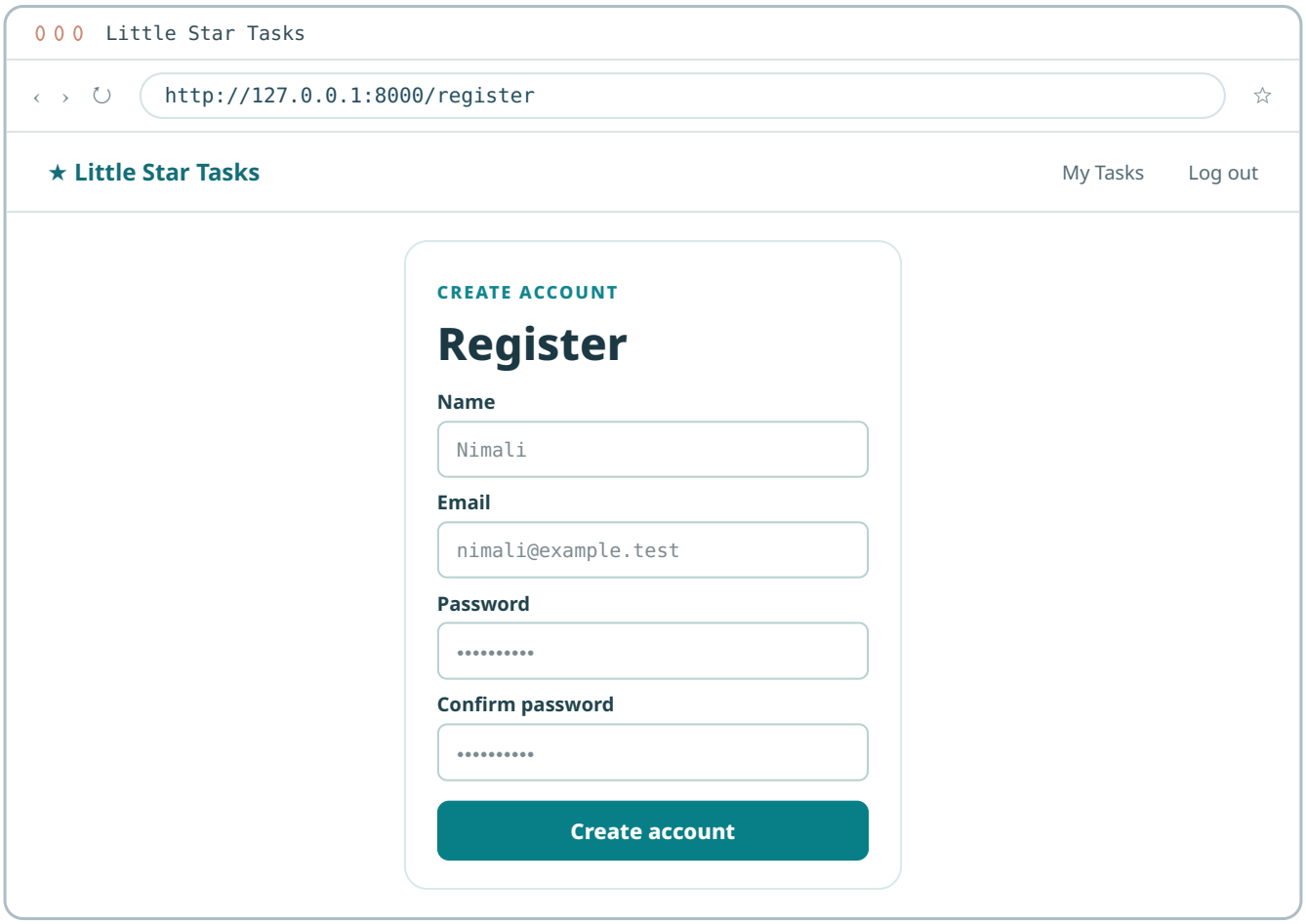
Password එක browser address bar එකේ නොයවන්න.

33. Registration screen preview

Password confirmation එකේ අරමුණ typing mistakes අඩු කිරීමයි.

සරල පැහැදිලි කිරීම

Name, Email, Password සහ Confirm Password fields තබන්න. Form validation failures user ට පැහැදිලිව පෙන්වන්න. Real users සම්බන්ධ පද්ධතියකට privacy policy, terms, email verification සහ account recovery අවශ්‍ය විය හැකියි.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

පුංචි අභ්‍යාසය

Name හිස්ව submit කරලා expected validation error එක ලියාගන්න.

මතක තබාගන්න

Tutorial එක production authentication සේවාවක් නොවේ.

A. Logout ආරක්ෂිතව

Logout එකේදී session එක අවලංගු කර token එක අලුත් කරන්න.

GET link එකකින් logout කරනවාට වඩා POST form එකක් භාවිතා කරන්න. Logout කළ පසු protected Task page නැවත බලද්දී login page එකට redirect විය යුතුයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
Auth::logout();
$request->session()->invalidate();
$request->session()->regenerateToken();
```

අභ්‍යාසය: Logout → browser back → refresh කිරීමෙන් Task data පෙනෙනවාද පරීක්ෂා කරන්න.

Caching/security policies නිසා shared devices වල අමතර සැලකිල්ලක් අවශ්‍යයි.

B. auth middleware

Login නැති user කෙනෙකුට `/tasks` වෙත ඇතුළු වීමට ඉඩ නොදෙමු.

Route group එකට `middleware('auth')` එකතු කරමු. මෙය login කර නැති අය redirect කරයි. නමුත් Login වූ User B ට User A ගේ Task edit කළ හැකිද කියලා වෙනම authorization check අවශ්‍යයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
Route::middleware("auth")->group(function () {
    Route::resource("tasks", TaskController::class);
});
```

අභ්‍යාසය: Incognito browser එකකින් `/tasks` URL එක විවෘත කරන්න.

Expected: Login page වෙත redirect.

A. guest middleware

Login වූ user කෙනෙක් නැවත Register form එකට නොයවමු.

Guest middleware එකෙන් register/login pages guests සඳහා පමණක් යොදයි. Login වූ user කෙනෙකුට dashboard වෙත redirect විය හැකියි. User experience සහ route clarity දෙකටම මෙය වැදගත්.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
Route::middleware("guest")->group(function () {
    Route::get("/login", [LoginController::class, "create"]);
});
```

අභ්‍යාසය: Already logged-in user කෙනෙක් `/login` URL එක බලන විට කුමක් සිදුවේද බලන්න.

auth = logged-in required; guest = logged-out required.

B. Rate limiting

Password guesses බොහොමයක් අවම කරගැනීමට request limit යොදමු.

Tutorial එකේ login/register POST endpoints මත `throttle:6,1` middleware යොදා තිබෙනවා. මෙය අමතර ආරක්ෂක ස්ථරයක් පමණයි; monitoring, MFA, bot protection, account recovery තවදුරටත් අවශ්‍ය විය හැකියි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
Route::post("/login", [LoginController::class, "store"])
    ->middleware("throttle:6,1");
```

අභ්‍යාසය: Repeated invalid login requests සඳහා test case එකක් ලියන්න.

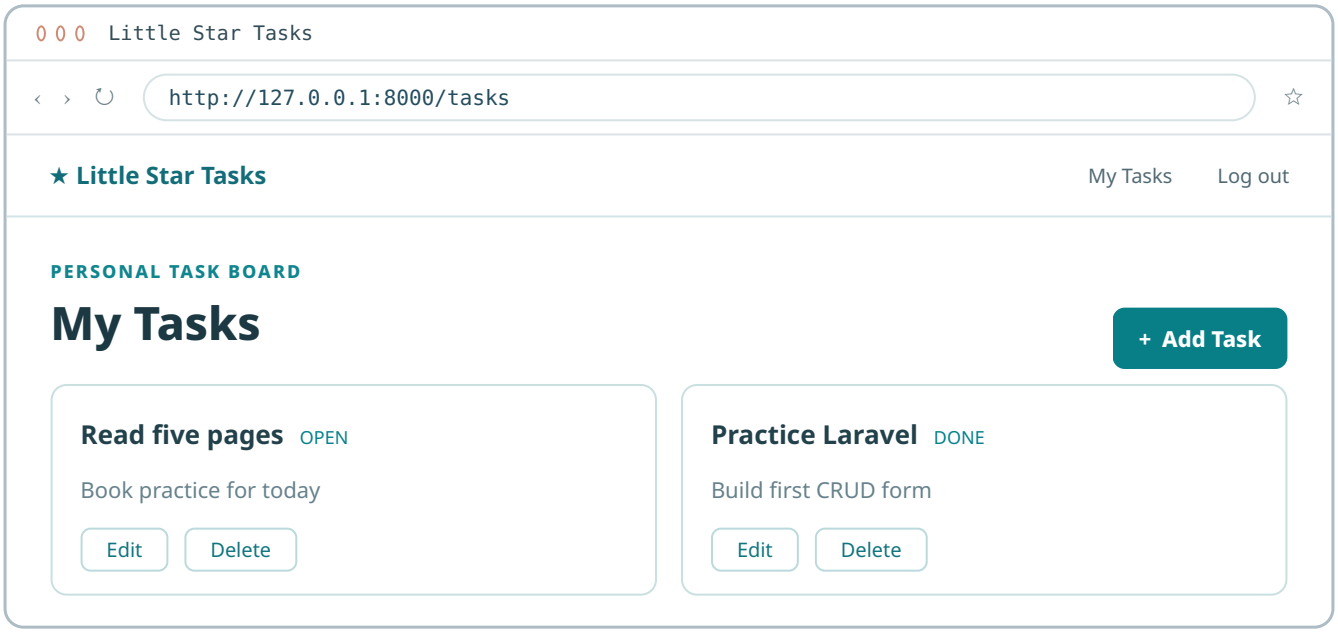
Rate limits හැම ප්‍රහාරයක්ම නවත්වන්නේ නැහැ.

36. Dashboard preview

Login වුණාට පසු “My Tasks” page එක පෙන්වමු.

සරල පැහැදිලි කිරීම

Task title, details, due date, status, Edit සහ Delete buttons, Add Task button, Filter එකක් තිබිය යුතුයි. System එක responsive වීම ඉගෙනගන්න mobile browser width එක අඩු කර බලන්න.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

පුංචි අභ්‍යාසය

Dashboard sketch එකේ buttons එක් එක් අරමුණ labels සමඟ ලියන්න.

මතක තබාගන්න

UI එක හොඳයි කියලා database logic හරි කියලා තීරණය කළ නොහැකියි.

A. Create: form එක

අලුත් Task එකක් browser එකෙන් submit කරමු.

Create page එකේ title required, details optional, date optional සහ progress status තිබෙනවා. Form method POST වේ. `@csrf` අවශ්‍යයි. Save button එක ඔබ data එකක් save කිරීමට යන බව පැහැදිලි කළ යුතුයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
<form method="POST" action="{{ route('tasks.store') }}">
  @csrf
  <input name="title" required>
  <input name="is_done" value="0" type="hidden">
</form>
```

අභ්‍යාසය: Title එකක් නැතිව save කිරීමේ test එක බලන්න.

Form fields සහ backend validation rules එකිනෙකට ගැළපිය යුතුයි.

B. Create: controller store()

සුරැකීමට පෙර server data validate කරයි.

Store action එක request validate කරලා `$request->user()->tasks()->create()` භාවිතා කරයි. User ID client request එකෙන් නොගනී. පසුව dashboard වෙත redirect කර success message එක පෙන්වයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
public function store(Request $request) {
    $data = $this->validated($request);
    $request->user()->tasks()->create($data);
    return redirect()->route("tasks.index");
}
```

අභ්‍යාසය: User A එකතු කළ task එක User B account එකෙන් නොපෙනෙන බව බලන්න.

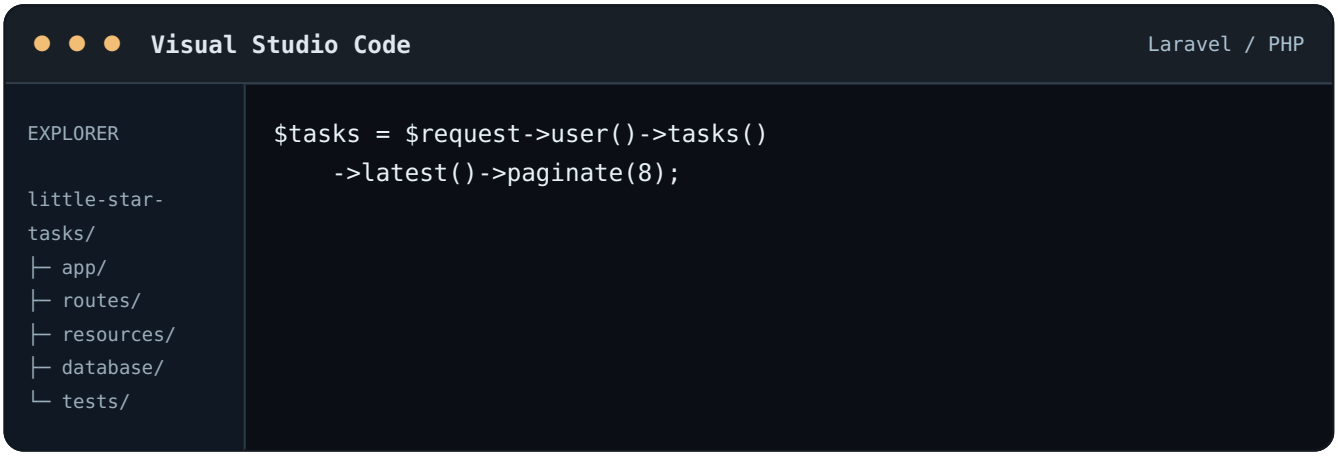
Server-set ownership යනු core security rule එකයි.

38. Read: own records list

Dashboard එකට tasks ගන්නා විට owner scope පාවිච්චි කරන්න.

සරල පැහැදිලි කිරීම

`Task::all()` ලෙස සියලු tasks query කිරීම multi-user app එකකට වැරදියි. ඒ වෙනුවට `$request->user()->tasks()` මඟින් logged-in person ගේ tasks පමණක් retrieve කරන්න. Pagination සහ filters එකතු කරන්න.



පුංචි අභ්‍යාසය

Controller එකේ query එක user_id අනුව බැඳී තිබෙන බව පෙන්වන්න.

මතක තබාගන්න

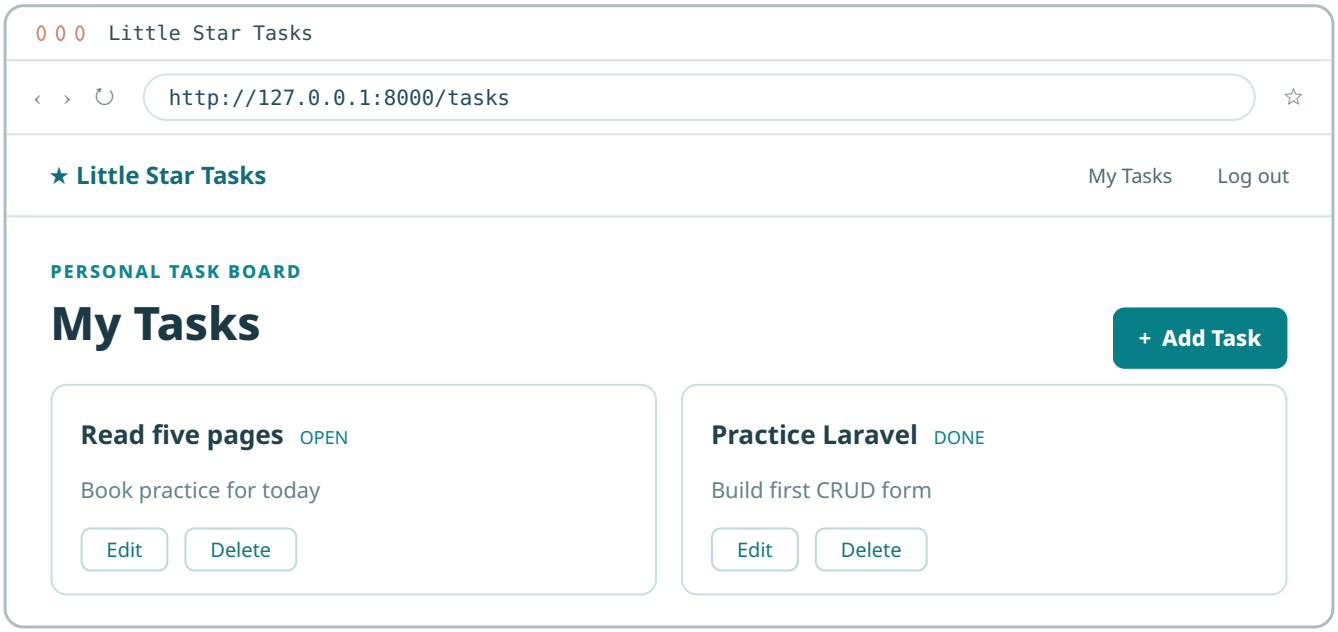
Read security fail නම් private data leak විය හැකියි.

39. Task card එකේ UI

Record එකක් card එකක් ලෙස පෙන්වමු.

සරල පැහැදිලි කිරීම

Title එක විශාලයි. Description එක කෙටිව පෙන්වයි. Status එක Done/Open ලෙස දක්වයි. Edit සහ Delete buttons පැහැදිලිව පෙනේ. `{{ $task->title }}` escape කරන නිසා script content execute වීමට උදව් නොකරයි.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

පුංචි අභ්‍යාසය

HTML code ලෙස title එකක් දී output safe ද බලන්න.

මතක තබාගන්න

XSS පාලනයට escaped Blade output වැදගත්.

A. Filter: Open/Done

Tasks එකක් Done හෝ Open අනුව වෙන් කර බලමු.

URL query එකේ `status=done` හෝ `status=open` ගෙන Laravel query එකට condition එකක් යොදනවා. Unknown status එකක් ආවොත් safe default එක list all own tasks යි. Database filter එකට කලින් user scope යොදනවා.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
if ($request->query("status") === "done") {
    $query->where("is_done", true);
}
```

අභ්‍යාසය: Tasks 3ක් හදා status අනුව filter results සටහන් කරන්න. Pagination links වල status preserve කළ හැකියි.

B. Update: edit page

පැරණි values form fields වලට පුරවා දෙමු.

`edit` method එක task එක අයිති කාටද පරීක්ෂා කරලා පමණක් view එකට යවයි. Blade `old('title', $task->title)` මඟින් validation fail වුණත් typed value තබාගන්න පුළුවන්.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
<input name="title" value="{{ old('title', $task->title) }}">
```

අභ්‍යාසය: Edit page එකේ පැරණි title එක ස්වයංක්‍රීයව පෙනෙනවාද බලන්න. Owner නොවන user ට 404 පෙන්වයි.

A. Update: save changes

Task එක වෙනස් කිරීමට PUT request එකක් යවමු.

`@method('PUT')` මඟින් HTML form එක Laravel PUT route එකට map කරයි. Update කිරීමට පෙර owner check, පසුව validation, පසුව model update. වෙන User ගේ task එකක් ID වෙනස් කර update නොවිය යුතුයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
@csrf
@method("PUT")
<button type="submit">Update task</button>
```

අභ්‍යාසය: URL එකේ task ID එක වෙනස් කර අනෙක් user task එක edit කිරීමට උත්සාහ කරන්න.

Expected: 404; අනෙක් user data නොවෙනස් වේ.

B. Delete: confirmation

Delete action එක ඉතා පැහැදිලි විය යුතුයි.

User ට accidental clicks වැළැක්වීම සඳහා confirmation dialog එකක් පෙන්වන්න. එය security control එකක් නොවේ: actual server owner check සහ CSRF validation අත්‍යවශ්‍යයි. Delete response එකෙන් dashboard එකට යවා message එකක් පෙන්වන්න.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
<form method="POST" action="{{ route('tasks.destroy', $task) }}">
  @csrf @method("DELETE")
  <button>Delete</button>
</form>
```

අභ්‍යාසය: Delete Cancel සහ Confirm අත්හදා බලන්න.

Production systems හි audit/undo/soft delete අවශ්‍ය විය හැකියි.

42. Authorization: own Task පමණයි

Login වුණා කියලා හැම record එකක්ම edit කළ හැකි නොවේ.

සරල පැහැදිලි කිරීම

`ownedTask()` helper එක `$request->user()->tasks()->findOrFail($task->id)` ලෙස query කරයි. ID එක exists වුවත් වෙන user කෙනෙකුට අයිති නම් 404 දක්වයි. Browser URL එක වෙනස් කිරීමෙන් data access නොලැබිය යුතුයි.



පුංචි අභ්‍යාසය

User A සහ B හදා User B සිට User A task ID එකට යන්න.

මතක තබාගන්න

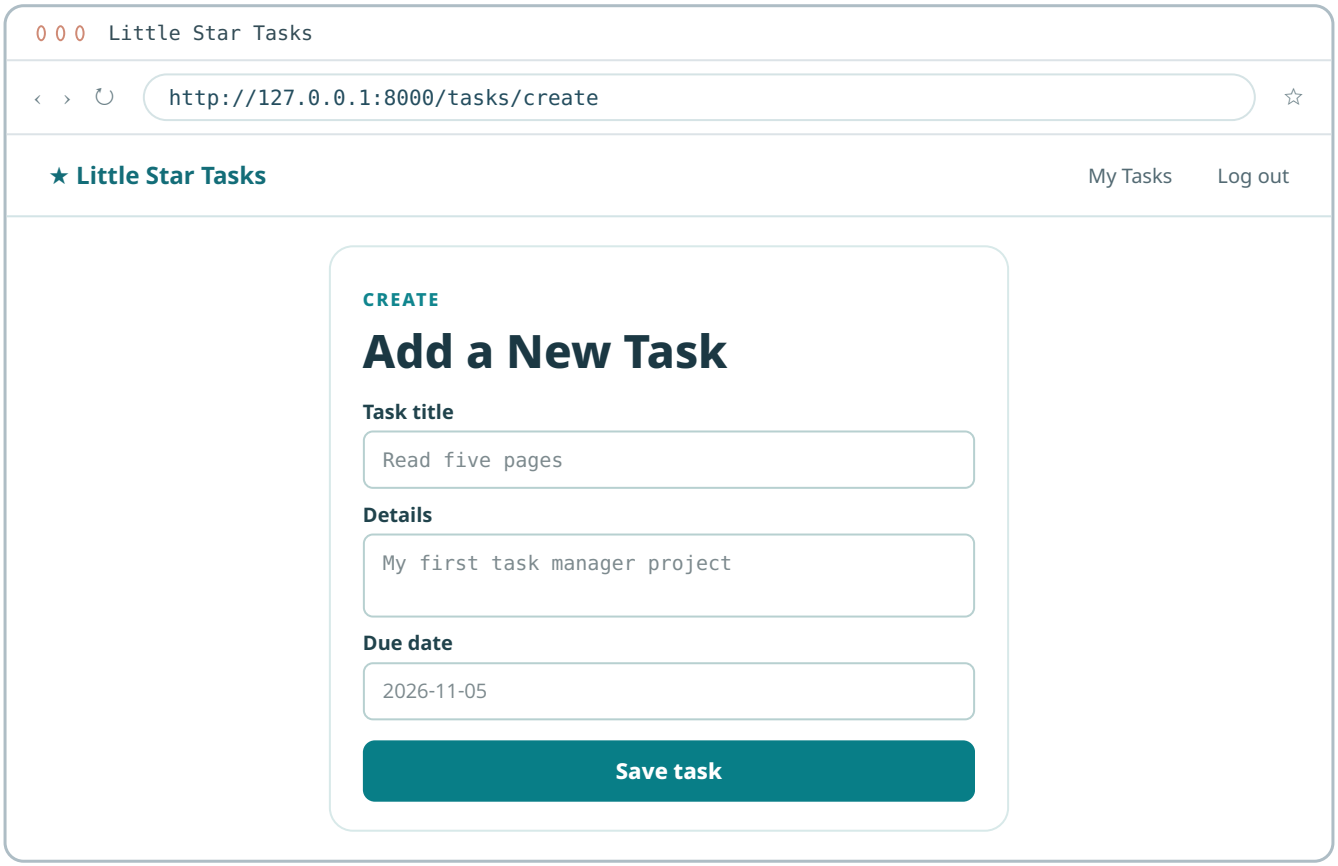
Authentication ≠ Authorization.

43. Task Form preview

Add/Edit screens දෙකටම එකම form partial එක පාවිච්චි කරමු.

සරල පැහැදිලි කිරීම

`tasks/_form.blade.php` එකෙන් title, details, due date, status fields දෙපිටුවකම reuse කරයි. Form action සහ button label පමණක් වෙනස් වේ. Duplicate markup අඩු වීමෙන් future changes පහසු වෙයි.



ඉගෙනීම සඳහා සකස් කළ browser preview එකකි; සැබෑ deployed screenshot එකක් නොවේ.

පුංචි අභ්‍යාසය

“Create” සහ “Edit” අතර වෙනස් field එක කුමක්ද යන්න ලියන්න.

මතක තබාගන්න

Reusable form partial එක design consistency වැඩි කරයි.

A. Empty state

Task එකක් නැති විට හිස් screen එකක් නොතබන්න.

"No tasks yet" සහ "Add your first task" වැනි message එකක් පෙන්වන්න. එයින් user ට next action එක පැහැදිලි වේ. Status filter එකක් නිසා records නැති විටත් same message එක යෝග්‍යයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
@if($tasks->isEmpty())
  <p>No tasks in this view.</p>
@endif
```

අභ්‍යාසය: අලුත් account එකක dashboard එක පරීක්ෂා කරන්න.

Empty state යනු UX design එකේ වැදගත් කොටසක්.

B. Mobile design

Phone එකක screen එක කුඩා නිසා UI එක හැඩගැසිය යුතුයි.

CSS `grid-template-columns: repeat(auto-fill, minmax(...))` භාවිතා කර tasks cards screen width අනුව වෙනස් වෙයි. Buttons අතර spacing තබන්න. Keyboard focus ලකුණු පැහැදිලිව පෙන්වන්න.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
@media (max-width: 560px) {
  .nav { flex-direction: column; }
}
```

අභ්‍යාසය: Chrome DevTools mobile viewport එකක login සහ dashboard බලන්න.

Responsive design = ඕනෑම device එකක usable.

A. Git init සහ status

Git යනු code වෙනස්කම්වල ඉතිහාසය තබන system එක.

Fresh Laravel project එකක Git already තිබෙන්න පුළුවන්. `git status` බලන්න. `.env` සහ `vendor/` නොයවන ලෙස `.gitignore` පරීක්ෂා කරන්න. Source code commit කරලා version history එක ආරක්ෂා කරමු.

```
Windows PowerShell / CMD
```

```
git status
git add .
git commit -m "Build login and task CRUD"
```

අභ්‍යාසය: Commit එකට `.env` එක්වී නැති බව තහවුරු කරන්න.

Git commits source changes සඳහා; DB backup සඳහා නොවේ.

B. GitHub push

Repository එකක් online තබා team එකක් සමඟ වැඩ කළ හැකියි.

GitHub හි හිස් repository එකක් සාදා `git remote add origin ...`, `git branch -M main`, `git push -u origin main` භාවිතා කරමු. URL එක ඔබේම repository URL එක විය යුතුයි. Public repo එකට keys, passwords හෝ private data යවන්න එපා.

```
Windows PowerShell / CMD
```

```
git remote add origin https://github.com/YOU/little-star-tasks.git
git branch -M main
git push -u origin main
```

අභ්‍යාසය: Push කිරීමට පෙර GitHub repository එක private තබනවාද public තබනවාද තීරණය කරන්න.

GitHub Pages එකෙන් PHP/Laravel server එකක් host කළ නොහැකියි.

A. Testing ඇයි?

Application එකේ buttons වැඩ කළාම පමණක් සාර්ථක කියන්න බැහැ.

Validation rules, login protection, ownership, logout සහ database operations වෙන වෙනම test කළ යුතුයි. Manual browser tests සහ automated Laravel HTTP tests දෙකම වැදගත්. Test case එකකට input, steps, expected result සහ actual result තිබිය යුතුයි.

DATA EXAMPLE

```
Test Case: Guest visits /tasks
Expected: redirect to /login
Actual: (run and write here)
```

අභ්‍යාසය: ඔබේම test cases තුනක් ලියන්න.

QA යනු bugs හොයාගැනීම පමණක් නොව තත්ත්වය පවත්වාගැනීමයි.

B. RefreshDatabase

Testing සඳහා clean database state එකක් අවශ්‍යයි.

Laravel test class එක `use RefreshDatabase;` භාවිතා කරයි. Test එකකට user සහ tasks ගොඩනැඟීම සහ පසුව database state reset කිරීමෙන් tests එකිනෙක බලපෑම් අවම වේ.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```php
use Illuminate\Foundation\Testing\RefreshDatabase;

class TaskCrudTest extends TestCase {
    use RefreshDatabase;
}
```

අභ්‍යාසය: Tests එකිනෙක independent විය යුත්තේ ඇයි ලියන්න.

Test data සහ සැබෑ users data කිසිවිට එකට නොතබන්න.

A. Guest test

Login නොමැති කෙනෙකුට tasks නොපෙන්විය යුතුයි.

Test එක `/tasks` වෙත GET request එකක් යවා `/login` redirect බලාපොරොත්තු වේ. මෙය middleware වැරදී ඉවත් වී නැති බව අවම මට්ටමකින් පරීක්ෂා කරයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
public function test_guest_needs_login(): void {
    $this->get("/tasks")->assertRedirect("/login");
}
```

අභ්‍යාසය: `php artisan test --filter=TaskCrudTest` ධාවනය කරන්න.

Protected routes browser සහ test දෙකේම check කරන්න.

B. Login සහ Register tests

Account lifecycle එක සාර්ථකද පරීක්ෂා කරන්න.

Registration test එක user DB row සෑදීමත් login session සාර්ථක වීමත් බලයි. Login test එක valid password ගනී; logout කළ පසු `assertGuest()` බලාපොරොත්තු වේ. Duplicate email test එකත් එකතු කළ හැකියි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$this->post("/register", [ /* valid data */ ])
    ->assertRedirect("/tasks");
$this->assertAuthenticated();
```

අභ්‍යාසය: Password mismatch test එකක් ඔබ විසින් එකතු කරන්න.

Automated tests වල test emails පමණක් යොදන්න.

A. CRUD tests

Create → Update → Delete ගමන සම්පූර්ණව පරීක්ෂා කරමු.

`actingAs($user)` මඟින් test user login වී සිටින ලෙස සකස් කරමු. POST එකෙන් task එකක් save කර, PUT එකෙන් status වෙනස් කර, DELETE එකෙන් record එක ඉවත් කරන්න. Database assertions අවශ්‍යයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$this->actingAs($user)->post("/tasks", [
    "title" => "Read", "is_done" => "0"
]);
```

අභ්‍යාසය: Project ZIP එකේ `TaskCrudTest.php` කියවන්න.

Controller methods වැඩ කළත් data isolation tests අනිවාර්යයි.

B. Security isolation test

User B ට User A ගේ records නොපෙනීම නීතියක් ලෙස පරීක්ෂා කරමු.

ඉහත Task ID වෙනස් කර GET/PUT/DELETE requests යවන test එකක් සාදා `assertNotFound()` බලාපොරොත්තු වෙමු. User A ගේ record එක database එකේ නොවෙනස්ව තිබෙන බවත් assert කළ යුතුයි.

Visual Studio Code — Laravel / PHP

EXPLORER
app/
routes/
views/

```
$this->actingAs($bob)
    ->delete("/tasks/".$alicesTask->id)
    ->assertNotFound();
```

අභ්‍යාසය: Authorization test එක fail වුණොත් release නතර කරන්න.

Login තිබුණත් owner checks අත්‍යවශ්‍යය.

A. Manual QA checklist

Automated tests නොදකින UX ගැටලුත් සොයමු.

Keyboard navigation, empty states, error text, mobile responsive, wrong email, wrong password, two users, logout/back button, long title, special characters සහ missing dates පරීක්ෂා කරන්න. Evidence screenshot සහ steps සුරකින්න.

DATA EXAMPLE

```
TC-01 Login OK?
TC-02 Wrong password?
TC-03 Only my tasks?
TC-04 Mobile view OK?
```

අභ්‍යාසය: QA checklist එක completion mark සමඟ පුරවන්න.

Manual QA සැබෑ user journey එක අනුකරණය කරයි.

B. Bug Report එක ලියමු

Bug එකක් හමු වුණොත් තවත් අයෙකුට නැවත සෑදිය හැකිව ලියන්න.

Title: Other user task editable. Steps: login B, open A task URL, submit PUT. Expected: 404. Actual: updated. Severity: Critical. Screenshot, app version, test date සහ fix verification එකතු කරන්න.

DATA EXAMPLE

```
BUG-07 | Severity: Critical
Steps: Login as B → try A task
Expected: 404
Actual: ...
Status: Open / Fixed / Verified
```

අභ්‍යාසය: ඔබ හමුවූ UI bug එකකට report එකක් සකස් කරන්න.

Clear bug reports නැතිනම් fix verify කිරීම අපහසුයි.

A. Backup basics

Database backup නැතිව production යාම අනතුරුදායකයි.

Local SQLite demo එකේ `.sqlite` file එක application ලියමින් නොසිටින අවස්ථාවක හෝ SQLite backup API/tool භාවිතයෙන් backup කළ යුතුයි. MySQL භාවිතා කරන විට consistent backups සහ restore test සැලසුම් කරන්න. `.env` secrets වෙනම ආරක්ෂා කරන්න.



අභ්‍යාසය: Sample data backup එකක් ගෙන වෙනම test copy එකක restore කරන්න.

Backup schedule, retention සහ encryption real projects සඳහා අවශ්‍යයි.

B. Debug mode සහ HTTPS

`APP_DEBUG=true` developer machine එකට පමණයි.

Production site එකක stack traces, SQL details හෝ configuration info public වෙන්න ඉඩ නොදෙන්න. HTTPS අනිවාර්යයි. Hosting web root එක Laravel `public/` වෙත point විය යුතුයි; project root හෝ `.env` public folder එකට expose නොකරන්න.

Windows PowerShell / CMD

```
APP_ENV=production
APP_DEBUG=false
APP_URL=https://your-domain.example
```

අභ්‍යාසය: Public server එකේ `.env` URL එක browser එකෙන් විවෘත කළ නොහැකිද verify කරන්න.

Secure deployment එක code එකට වඩා වැඩි වගකීමක්.

A. Database deploy කිරීම

Production database එකක් migration වලින් හදන විට ගොඩක් සැලකිලිමත් වන්න.

`php artisan migrate --force` අවශ්‍ය විට පමණක් සහ reviewed backup එකකින් පසුව run කරන්න. Migration code හා actual schema එක සසඳන්න. Rollback strategy එකක් නැතිව data delete කරන changes නොකරන්න.

```
Windows PowerShell / CMD

composer install --no-dev --optimize-autoloader
php artisan migrate --force
```

අභ්‍යාසය: Release checklist එකේ database backup checkpoint එකක් දාන්න.

Migration rollback code තිබුණත් deleted data අනිවාර්යයෙන් නැවත එන්නේ නැහැ.

B. Email verification සහ password reset

මෙම tutorial app එකේ login/register minimal implementation එකකි.

සැබෑ users සඳහා forgot-password, email verification, MFA, audit logging, password reset throttling, privacy controls සහ session management යොදන්න. Laravel official starter kits Fortify සමඟ authentication routes සපයන නිසා production මට්ටමට ඒවා වඩා සුදුසුයි.



අභ්‍යාසය: මෙම demo එක public login portal ලෙස ඒ ආකාරයෙන්ම deploy නොකළ යුත්තේ ඇයි පැහැදිලි කරන්න.

Production auth වැඩ සඳහා official starter kit සලකා බලන්න.

A. Common errors - 404

URL එක නොගැළපුණාම හෝ route නැතිවුණාම 404 එයි.

`php artisan route:list` run කරලා path සහ method පරීක්ෂා කරන්න. Route name එක typo ද? Form action වැරදිද? User B ට User A task එකට 404 එන එක නම් security design එකේ නිවැරදි හැසිරීමකි.

Windows PowerShell / CMD

```
php artisan route:list
php artisan route:clear
```

අභ්‍යාසය: Task edit URL එකේ අයිතිය නැති ID එකකට 404 එන බව වෙනම වාර්තා කරන්න.

සියලු 404 bugs නොවේ.

B. Common errors - 419/500

CSRF හෝ server error එකක් ආවොත් හේතුව දැනගනිමු.

419 බොහෝවිට CSRF/session mismatch විය හැකියි. Form එකේ `@csrf` තිබේද සහ session config බලන්න. 500 error එකක් නම් local logs බලන්න; public site එකේ DEBUG ON කරන්න එපා.

Windows PowerShell / CMD

```
php artisan optimize:clear
# Inspect storage/logs/laravel.log locally
```

අභ්‍යාසය: Error එක නැවත සිදුකිරීමට පියවර ලියන්න.

Error message එක sensitive data නොපෙන්වන ලෙස සකස් කළ යුතුයි.

A. Git workflow සහ release

Code එක අතුරුදන් නොවන ලෙස changes version-control කරමු.

New feature එකක් branch එකක ලියලා tests run කර commit කරන්න. Pull request review එකක් කරන්න. `.env`, credentials, database files සහ secrets commit නොකරන්න. Release notes සහ rollback plan එක සෑම deploy එකකටම සකස් කරන්න.

Windows PowerShell / CMD

```
git checkout -b feature/task-filter
git add .
git commit -m "Add task filter"
```

අභ්‍යාසය: Git history එකේ කිසිදු password එකක් නැති බව scan කරන්න. Git යනු database backup replacement එකක් නොවේ.

B. 2026 Laravel Ecosystem

Laravel 13 තුළ නව ප්‍රවණතා තිබුණත් මූලික වෙබ් ආරක්ෂාව වෙනස් නොවේ.

2026 දී Laravel 13 හි AI SDK, JSON:API සහ vector/semantic search වැනි දියුණු හැකියාවන් ගැන උනන්දුවක් තියෙනවා. නමුත් ආරම්භකයෙකුට පළමුව routing, validation, auth, database, testing හොඳින් අවබෝධ කරගැනීම වැදගත්.



අභ්‍යාසය: AI tool එකෙන් code ලැබුණත් review/test කළ යුතු කරුණු තුනක් ලියන්න.

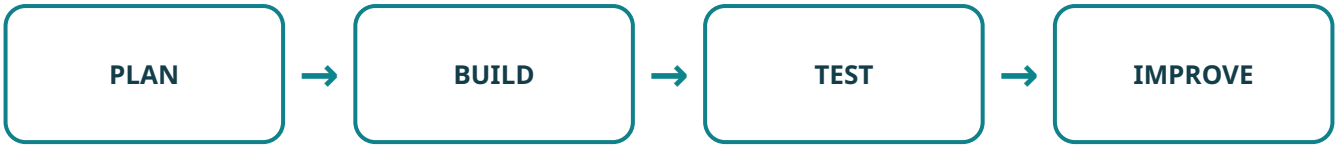
AI-generated code වල auth/ownership bugs තිබිය හැකි නිසා blindly paste නොකරන්න.

54. අවසන් skill map

ඔබට දැන් login සහ CRUD project එකක සම්පූර්ණ workflow පෙනෙනවා.

සරල පැහැදිලි කිරීම

Requirements → UI → migrations → models → controllers → views → auth → tests → Git → deployment check. මේ පියවර එකිනෙක සම්බන්ධයි. ඊළඟ project එකක් ලෙස Student Management, Inventory Tracker හෝ Simple Notes App එකක් නිර්මාණය කළ හැකියි.



පුංචි අභ්‍යාසය

පොත නොබලා අලුත් “Notes CRUD” feature එකක් සැලසුම් කරන්න.

මතක තබාගන්න

Code ලිවීම පමණක් නොව software development process එකම ඉගෙනගන්න.

01 • Route challenge: Hello Laravel route

Hello Laravel route

ඔබ කළ යුතු දේ

/hello route එකක් සාදා `Hello Sri Lanka` text එක return කරන්න. Route name `hello` ලෙස දෙන්න.

කොහේ වැඩ කරන්නේ?

`routes/web.php`

Expected Result

GET /hello → Hello Sri Lanka

01

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

02 • Blade practice: My profile page

My profile page

ඔබ කළ යුතු දේ

`resources/views/profile.blade.php` view එකක් සාදා name සහ favorite subject පෙන්වන්න. User data `{{ }}` මඟින් escape කරන්න.

කොහේ වැඩ කරන්නේ?

`resources/views`

Expected Result

HTML එකේ දත්ත පෙනේ; raw script execute නොවේ.

02

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

03 • Validation practice: Empty title blocker

Empty title blocker

ඔබ කළ යුතු දේ

Title හිස්ව POST /tasks යවන්න. Database එකේ row එකක් නොසාදිය යුතුයි. Error message එක පැහැදිලිව පෙන්වන්න.

කොහේ වැඩ කරන්නේ?

`TaskController`

Expected Result

Validation errors displayed, no row created.

03

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

04 • Account practice: Register + login

Register + login

ඔබ කළ යුතු දේ

අලුත් emails දෙකක් භාවිතා කර accounts දෙකක් සාදා වෙන වෙනම login කරන්න. Wrong password එකක් ද පරීක්ෂා කරන්න.

කොහේ වැඩ කරන්නේ?

`Auth controllers`

Expected Result

Accounts separate, wrong password fails.

04

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

05 • CRUD practice: First task

First task

ඔබ කළ යුතු දේ

Task එකක් create, edit, done mark, delete කරලා අවසානයේ record count එක check කරන්න.

කොහේ වැඩ කරන්නේ?

`TaskController`

Expected Result

CRUD all four operations work.

05

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

06 • Privacy practice: Two-person security

Two-person security

ඔබ කළ යුතු දේ

User A ගේ task ID එක User B භාවිතයෙන් edit/delete කිරීමට උත්සාහ කරන්න.

කොහේ වැඩ කරන්නේ?

`Task ownership`

Expected Result

404; User A task unchanged.

06

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

07 • UI practice: Mobile dashboard

Mobile dashboard

ඔබ කළ යුතු දේ

Chrome DevTools responsive view තුළ 375px width එකක dashboard බලන්න.

කොහේ වැඩ කරන්නේ?

CSS

Expected Result

Buttons readable and cards one column.

07

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

08 • Git practice: Clean commit

Clean commit

ඔබ කළ යුතු දේ

Local git commit එකක් සාදා `.env` file එක staged නොවී තිබෙන බව පරීක්ෂා කරන්න.

කොහේ වැඩ කරන්නේ?

`Git`

Expected Result

Source committed, no secrets.

08

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

09 • QA practice: Manual bug report

Manual bug report

ඔබ කළ යුතු දේ

Intentional typo හෝ UI issue එකක් සොයා bug report එකක් ලියන්න.

කොහේ වැඩ කරන්නේ?

QA

Expected Result

Steps, expected, actual, screenshot.

09

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

10 • Database practice: SQLite backup

SQLite backup

ඔබ කළ යුතු දේ

Practice database එක backup කර වෙනම local test copy එකකින් restore කළ හැකිද බලන්න.

කොහේ වැඩ කරන්නේ?

Backup

Expected Result

Verified recovery of demo data.

10

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

11 • Feature request: Priority field design

Priority field design

ඔබ කළ යුතු දේ

Task table එකට priority column එකක් එක් කරන සැලැස්ම ලියන්න. Migration + form + validation + tests ගැන සිතන්න.

කොහේ වැඩ කරන්නේ?

SDLC

Expected Result

End-to-end change plan with tests.

11

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

12 • Final challenge: New Notes CRUD

New Notes CRUD

ඔබ කළ යුතු දේ

Login සහ personal notes CRUD application එකක් සැලසුම් කර authorization test එකක් යෝජනා කරන්න.

කොහේ වැඩ කරන්නේ?

`Final`

Expected Result

SRS, ER diagram, route map, QA plan.

12

LAB TASK

Try → Observe → Explain → Test

- ☐ Code ready
- ☐ Result checked
- ☐ Screenshot saved
- ☐ Bug fixed

Mini QA Report

Check	Expected	Actual	Pass?
Start	Feature opens	_____	☐
Valid input	Correct result	_____	☐
Wrong input	Safe error	_____	☐

MCQ • ප්‍රශ්න 1 - 4

සෑම ප්‍රශ්නයකටම හොඳම පිළිතුර තෝරන්න. මෙහි පිළිතුරු පහළ වෙනම පිටුවක ඇත.

1. Laravel framework එක භාවිතා කරන්නේ කුමක් සඳහාද?

(A) Web app සංවිධානය කර ඉදි කිරීමට

(B) Image pixels වෙනස් කිරීමට

(C) Windows install කිරීමට

2. CRUD හි U අකුරෙන් අදහස් වන්නේ?

(A) Upload

(B) Update

(C) User

3. Password database එකේ තබන නිවැරදි ක්‍රමය?

(A) Plain text

(B) Hash

(C) URL query

4. Login නැති user ට tasks නවත්වන middleware එක?

(A) guest

(B) auth

(C) cache

MCQ • ප්‍රශ්න 5 - 8

සෑම ප්‍රශ්නයකටම හොඳම පිළිතුර තෝරන්න. මෙහි පිළිතුරු පහළ වෙනම පිටුවක ඇත.

5. User B, User A ගේ Task එක edit නොවීමට වඩා වැදගත්?

(A) Owner check
(B) Blue button
(C) Bigger logo

6. SQLite සඳහා database එක සාමාන්‍යයෙන්?

(A) තනි file එකක්
(B) Images folder එකක්
(C) API key එකක්

7. POST form එකක @csrf භාවිතා කරන්නේ?

(A) HTML heading
(B) CSRF protection
(C) DB backup

8. Route list පෙන්වන command එක?

(A) php artisan route:list
(B) php -v
(C) git push

MCQ • ප්‍රශ්න 9 - 12

සෑම ප්‍රශ්නයකටම හොඳම පිළිතුර තෝරන්න. මෙහි පිළිතුරු පහළ වෙනම පිටුවක ඇත.

9. Production හි APP_DEBUG විය යුත්තේ?

A true

B false

C any

10. Laravel PHP 13 minimum PHP version?

A PHP 7.4

B PHP 8.3

C PHP 5.6

11. Laravel model එකේ fillable කුමකටද?

A Mass-assignable fields

B Page title

C CSS color

12. API හෝ web change එකකට QA checklist අවශ්‍ය ඇයි?

A Bug risk අඩු කර ගැනීමට

B Fonts download කිරීමට

C Screen අඳුරු කිරීමට

MCQ පිළිතුරු

පිළිතුරු බැලීමට පෙර ඔබේ තේරීම් නැවත කියවන්න.

01 A • Web app සංවිධානය කර ඉදි කිරීමට

02 B • Update

03 B • Hash

04 B • auth

05 A • Owner check

06 A • තනි file එකක්

07 B • CSRF protection

08 A • php artisan route:list

09 B • false

10 B • PHP 8.3

11 A • Mass-assignable fields

12 A • Bug risk අඩු කර ගැනීමට

ඔබට තව පුහුණුවක් අවශ්‍යද?

වැරදි වූ ප්‍රශ්න සඳහා අදාළ lesson එක නැවත කියවා, code run කරලා ප්‍රතිඵලය ලියන්න.

නිල මූලාශ්‍ර සහ Version Note

2026 • නිල ලේඛන / Official references

Laravel 13 release notes

https://laravel.com/docs/13.x/releases

Installation

https://laravel.com/docs/13.x/installation

Authentication & starter kits

https://laravel.com/docs/13.x/authentication
https://laravel.com/docs/13.x/starter-kits

Validation and Database

https://laravel.com/docs/13.x/validation
https://laravel.com/docs/13.x/migrations

Testing

https://laravel.com/docs/13.x/http-tests

Software version note

2026 ඔක්තෝබර් සඳහා Laravel 13 සහ PHP 8.3+ භාවිතා කරන ලදි. Framework update සමඟ screen labels සහ commands ටිකක් වෙනස් විය හැකියි.

Source Appendix • overlay/routes/web.php

මෙම code එක source ZIP එකේ `overlay/routes/web.php` ගොනුවේ ඇත. පිටුව 1/1. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/routes/web.php

```
<?php

use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\Auth\RegisterController;
use App\Http\Controllers\TaskController;
use Illuminate\Support\Facades\Route;

// Send visitors to the right starting page.
Route::get('/', function () {
    return redirect()->route(auth()->check() ? 'tasks.index' : 'login');
});

// Only visitors who have not logged in can see these forms.
Route::middleware('guest')->group(function () {
    Route::get('/register', [RegisterController::class, 'create'])->name('register');
    Route::post('/register', [RegisterController::class, 'store'])-
>name('register.store')->middleware('throttle:6,1');
    Route::get('/login', [LoginController::class, 'create'])->name('login');
    Route::post('/login', [LoginController::class, 'store'])->name('login.store')-
>middleware('throttle:6,1');
});

// A user must log in before viewing or changing tasks.
Route::middleware('auth')->group(function () {
    Route::post('/logout', [LoginController::class, 'destroy'])->name('logout');
    Route::resource('tasks', TaskController::class)->except(['show']);
});
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Models/User.php

මෙම code එක source ZIP එකේ `overlay/app/Models/User.php` ගොනුවේ ඇත. පිටුව 1/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Models/User.php

```php
<?php

namespace App\Models;

use Database\Factories\UserFactory;
use Illuminate\Database\Eloquent\Attributes\Fillable;
use Illuminate\Database\Eloquent\Attributes\Hidden;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

#[Fillable(['name', 'email', 'password'])]
#[Hidden(['password', 'remember_token'])]
class User extends Authenticatable
{
    /** @use HasFactory<UserFactory> */
    use HasFactory, Notifiable;

    // Each person owns many separate tasks.
    public function tasks(): HasMany
    {
        return $this->hasMany(Task::class);
    }

    protected function casts(): array
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Models/User.php

මෙම code එක source ZIP එකේ `overlay/app/Models/User.php` ගොනුවේ ඇත. පිටුව 2/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Models/User.php

```
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Models/Task.php

මෙම code එක source ZIP එකේ `overlay/app/Models/Task.php` ගොනුවේ ඇත. පිටුව 1/1. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Models/Task.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Task extends Model
{
    protected $fillable = ['title', 'details', 'due_date', 'is_done'];

    protected function casts(): array
    {
        return ['due_date' => 'date', 'is_done' => 'boolean'];
    }

    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }
}
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/database/migrations/2026_10_31_000001_create_tasks_table.php

මෙම code එක source ZIP එකේ `overlay/database/migrations/2026_10_31_000001_create_tasks_table.php` ගොනුවේ ඇත. පිටුව 1/1. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code overlay/database/migrations/2026 10 31 000001 create tasks table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    public function up(): void
    {
        Schema::create('tasks', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->cascadeOnDelete();
            $table->string('title', 120);
            $table->text('details')->nullable();
            $table->date('due_date')->nullable();
            $table->boolean('is_done')->default(false);
            $table->timestamps();
            $table->index(['user_id', 'is_done']);
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('tasks');
    }
};
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/Auth/RegisterController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/Auth/RegisterController.php` ගොනුවේ ඇත. පිටුව 1/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/Auth/RegisterController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;
use Illuminate\View\View;

class RegisterController extends Controller
{
    public function create(): View
    {
        return view('auth.register');
    }

    public function store(Request $request): RedirectResponse
    {
        // Validate the values before adding a new account.
        $data = $request->validate([
            'name' => ['required', 'string', 'max:100'],
            'email' => ['required', 'email', 'max:255', 'unique:users,email'],
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/Auth/RegisterController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/Auth/RegisterController.php` ගොනුවේ ඇත. පිටුව 2/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/Auth/RegisterController.php

```
            'password' => ['required', 'confirmed', Password::min(8)],
        ]);

        // Hashing protects passwords if a database is exposed.
        $user = User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);

        Auth::login($user);
        $request->session()->regenerate();

        return redirect()->route('tasks.index')->with('success', 'Welcome! Your account is
ready.');
    }
}
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/Auth/LoginController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/Auth/LoginController.php` ගොනුවේ ඇත. පිටුව 1/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/Auth/LoginController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\View\View;

class LoginController extends Controller
{
    public function create(): View
   {
        return view('auth.login');
   }

    public function store(Request $request): RedirectResponse
   {
        // Compare the supplied password with Laravel's saved hash.
        $credentials = $request->validate([
            'email' => ['required', 'email'],
            'password' => ['required', 'string'],
       ]);

        if (! Auth::attempt($credentials)) {
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/Auth/LoginController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/Auth/LoginController.php` ගොනුවේ ඇත. පිටුව 2/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/Auth/LoginController.php

```
            return back()->withErrors(['email' => 'These details do not match.'])
                ->onlyInput('email');
        }

        // Change the session ID after successful authentication.
        $request->session()->regenerate();

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

     public function destroy(Request $request): RedirectResponse
    {
        Auth::logout();
        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return redirect()->route('login');
    }
}
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/TaskController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/TaskController.php` ගොනුවේ ඇත. පිටුව 1/4. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/TaskController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\Task;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Validation\Rule;
use Illuminate\View\View;

class TaskController extends Controller
{
    public function index(Request $request): View
   {
        // Always scope database reads to the signed-in account.
        $query = $request->user()->tasks()->latest();
        if ($request->query('status') === 'done') {
            $query->where('is_done', true);
        } elseif ($request->query('status') === 'open') {
            $query->where('is_done', false);
       }
        $tasks = $query->paginate(8)->withQueryString();

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

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/TaskController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/TaskController.php` ගොනුවේ ඇත. පිටුව 2/4. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/TaskController.php

```
    public function create(): View
    {
        return view('tasks.create');
    }

    public function store(Request $request): RedirectResponse
    {
        $data = $this->validated($request);
        // The relationship automatically sets this user's user_id.
        $request->user()->tasks()->create($data);

        return redirect()->route('tasks.index')->with('success', 'Task created.');
    }

    public function edit(Request $request, Task $task): View
    {
        $task = $this->ownedTask($request, $task);

        return view('tasks.edit', compact('task'));
    }

    public function update(Request $request, Task $task): RedirectResponse
    {
        $task = $this->ownedTask($request, $task);
        $task->update($this->validated($request));
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/TaskController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/TaskController.php` ගොනුවේ ඇත. පිටුව 3/4. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/TaskController.php

```
        return redirect()->route('tasks.index')->with('success', 'Task updated.');
    }

    public function destroy(Request $request, Task $task): RedirectResponse
    {
        $this->ownedTask($request, $task)->delete();

        return redirect()->route('tasks.index')->with('success', 'Task deleted.');
    }

    private function ownedTask(Request $request, Task $task): Task
    {
        // A task belonging to somebody else is treated as not found.
        return $request->user()->tasks()->findOrFail($task->id);
    }

    private function validated(Request $request): array
    {
        $data = $request->validate([
            'title' => ['required', 'string', 'max:120'],
            'details' => ['nullable', 'string', 'max:1000'],
            'due_date' => ['nullable', 'date'],
            'is_done' => ['required', Rule::in(['0', '1'])],
       ]);
        $data['is_done'] = $data['is_done'] === '1';
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/app/Http/Controllers/TaskController.php

මෙම code එක source ZIP එකේ `overlay/app/Http/Controllers/TaskController.php` ගොනුවේ ඇත. පිටුව 4/4. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/app/Http/Controllers/TaskController.php

```
        return $data;
    }
}
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/layouts/app.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/layouts/app.blade.php` ගොනුවේ ඇත. පිටුව 1/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/layouts/app.blade.php

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>@yield('title', 'Little Star Tasks')</title>
    <link rel="stylesheet" href="{{ asset('css/little-star.css') }}">
</head>
<body>
<header class="topbar">
    <div class="shell nav">
        <a class="brand" href="{{ route('tasks.index') }}">★ Little Star Tasks</a>
        <div class="nav-right">
            @auth
                <span>Hello, {{ auth()->user()->name }}</span>
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/layouts/app.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/layouts/app.blade.php` ගොනුවේ ඇත. පිටුව 2/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/layouts/app.blade.php

```
                <form method="POST" action="{{ route('logout') }}">
                    @csrf
                    <button class="nav-link" type="submit">Log out</button>
                </form>
            @else
                <a href="{{ route('login') }}">Login</a>
                <a href="{{ route('register') }}">Register</a>
            @endauth
        </div>
    </div>
</header>
<main class="shell main">
    @if (session('success'))
        <div class="alert success" role="status">{{ session('success') }}</div>
    @endif
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/layouts/app.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/layouts/app.blade.php` ගොනුවේ ඇත. පිටුව 3/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code overlay/resources/views/layouts/app.blade.php

```
    @if ($errors->any())
        <div class="alert error" role="alert">
            <strong>Please fix the following:</strong>
            <ul>@foreach ($errors->all() as $error)<li>{{ $error }}</li>@endforeach</ul>
        </div>
    @endif
    @yield('content')
</main>
<footer class="footer">Practice project · egotechworld.com · 2026</footer>
</body>
</html>
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/auth/login.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/auth/login.blade.php` ගොනුවේ ඇත. පිටුව 1/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/auth/login.blade.php

```
@extends('layouts.app')
@section('title', 'Login - Little Star Tasks')
@section('content')
<section class="auth-panel">
    <p class="eyebrow">WELCOME BACK</p>
    <h1>Login</h1>
    <p>Enter your email and password to open your task list.</p>
    <form method="POST" action="{{ route('login.store') }}" class="stack">
        @csrf
        <label>Email<input type="email" name="email" value="{{ old('email') }}" required
autocomplete="email" maxlength="255"></label>
        <label>Password<input type="password" name="password" required
autocomplete="current-password"></label>
        <button class="btn" type="submit">Sign in</button>
    </form>
    <p class="hint">New here? <a href="{{ route('register') }}">Create an account</a>.</p>
</section>
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/auth/login.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/auth/login.blade.php` ගොනුවේ ඇත. පිටුව 2/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/auth/login.blade.php

```
@endsection
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/auth/register.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/auth/register.blade.php` ගොනුවේ ඇත. පිටුව 1/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/auth/register.blade.php

```
@extends('layouts.app')
@section('title', 'Register - Little Star Tasks')
@section('content')
<section class="auth-panel">
    <p class="eyebrow">CREATE A FREE PRACTICE ACCOUNT</p>
    <h1>Register</h1>
    <form method="POST" action="{{ route('register.store') }}" class="stack">
        @csrf
        <label>Name<input name="name" value="{{ old('name') }}" maxlength="100" required
autocomplete="name"></label>
        <label>Email<input name="email" type="email" value="{{ old('email') }}"
maxlength="255" required autocomplete="email"></label>
        <label>Password<input name="password" type="password" minlength="8" required
autocomplete="new-password"></label>
        <label>Confirm password<input name="password_confirmation" type="password"
minlength="8" required autocomplete="new-password"></label>
        <button class="btn" type="submit">Create account</button>
    </form>
    <p class="hint">Have an account? <a href="{{ route('login') }}">Login</a>.</p>
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/auth/register.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/auth/register.blade.php` ගොනුවේ ඇත. පිටුව 2/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/auth/register.blade.php

```
</section>
@endsection
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/tasks/index.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/tasks/index.blade.php` ගොනුවේ ඇත. පිටුව 1/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/tasks/index.blade.php

```
@extends('layouts.app')
@section('title', 'My Tasks - Little Star Tasks')
@section('content')
<div class="page-head">
    <div><p class="eyebrow">PERSONAL TASK BOARD</p><h1>My Tasks</h1><p>Only you can see
and change your own tasks.</p></div>
    <a class="btn" href="{{ route('tasks.create') }}">+ Add Task</a>
</div>
<form class="filters" method="GET" action="{{ route('tasks.index') }}">
    <label>Status
        <select name="status">
            <option value="" @selected(request('status') === null || request('status') ===
'')>All</option>
            <option value="open" @selected(request('status') === 'open')>Open</option>
            <option value="done" @selected(request('status') === 'done')>Done</option>
        </select>
    </label>
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/tasks/index.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/tasks/index.blade.php` ගොනුවේ ඇත. පිටුව 2/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/tasks/index.blade.php

```
    <button class="btn secondary" type="submit">Filter</button>
</form>
@if($tasks->isEmpty())
    <div class="empty">No tasks in this view. Create your first task.</div>
@else
    <div class="task-grid">
        @foreach($tasks as $task)
            <article class="task-card">
                <div class="task-top"><strong>{{ $task->title }}</strong><span
class="chip">{{ $task->is_done ? 'Done' : 'Open' }}</span></div>
                <p>{{ $task->details ?: 'No details yet.' }}</p>
                <small>Due: {{ $task->due_date ? $task->due_date->format('Y-m-d') : 'Any
day' }}</small>
                <div class="actions">
                    <a class="btn secondary" href="{{ route('tasks.edit',
$task) }}">Edit</a>
                    <form method="POST" action="{{ route('tasks.destroy', $task) }}"
onsubmit="return confirm('Delete this task?')">
                        @csrf @method('DELETE')
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/tasks/index.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/tasks/index.blade.php` ගොනුවේ ඇත. පිටුව 3/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/tasks/index.blade.php

```
                        <button type="submit" class="btn danger">Delete</button>
                    </form>
                </div>
            </article>
        @endforeach
    </div>
    <div class="pagination">{{ $tasks->links() }}</div>
@endif
@endsection
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/tasks/create.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/tasks/create.blade.php` ගොනුවේ ඇත. පිටුව 1/1. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/tasks/create.blade.php

```
@extends('layouts.app')
@section('title', 'New Task - Little Star Tasks')
@section('content')
<section class="form-panel">
    <p class="eyebrow">CREATE</p><h1>Add a New Task</h1>
    <form method="POST" action="{{ route('tasks.store') }}" class="stack">
        @include('tasks._form', ['button' => 'Save task'])
    </form>
</section>
@endsection
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/tasks/edit.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/tasks/edit.blade.php` ගොනුවේ ඇත. පිටුව 1/1. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/tasks/edit.blade.php

```
@extends('layouts.app')
@section('title', 'Edit Task - Little Star Tasks')
@section('content')
<section class="form-panel">
    <p class="eyebrow">UPDATE</p><h1>Edit Task</h1>
    <form method="POST" action="{{ route('tasks.update', $task) }}" class="stack">
        @include('tasks._form', ['button' => 'Update task'])
    </form>
</section>
@endsection
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/tasks/_form.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/tasks/_form.blade.php` ගොනුවේ ඇත. පිටුව 1/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/tasks/ form.blade.php

```
@csrf
@if(isset($task)) @method('PUT') @endif
<label>Task title
    <input name="title" value="{{ old('title', $task->title ?? '') }}" maxlength="120"
required>
</label>
<label>Details
    <textarea name="details" rows="4" maxlength="1000">{{ old('details', $task->details ??
'') }}</textarea>
</label>
<label>Due date (optional)
    <input type="date" name="due_date" value="{{ old('due_date', isset($task) && $task-
>due_date ? $task->due_date->format('Y-m-d') : '') }}">
</label>
<label>Progress
    <select name="is_done">
        <option value="0" @selected((string)old('is_done', isset($task) && $task-
>is_done ? '1' : '0') === '0')>Open</option>
        <option value="1" @selected((string)old('is_done', isset($task) && $task-
>is_done ? '1' : '0') === '1')>Done</option>
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/resources/views/tasks/_form.blade.php

මෙම code එක source ZIP එකේ `overlay/resources/views/tasks/_form.blade.php` ගොනුවේ ඇත. පිටුව 2/2. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/resources/views/tasks/ form.blade.php

```
    </select>
</label>
<div class="actions"><button type="submit" class="btn">{{ $button }}</button><a class="btn secondary" href="{{ route('tasks.index') }}">Cancel</a></div>
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/public/css/little-star.css

මෙම code එක source ZIP එකේ `overlay/public/css/little-star.css` ගොනුවේ ඇත. පිටුව 1/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/public/css/little-star.css

```
/* Simple, accessible CSS for a beginner-friendly Laravel project. */
:root { color-scheme: light; font-family: system-ui, sans-serif; color: #163542;
background: #fff; }
* { box-sizing: border-box; }
body { margin: 0; background: #fff; line-height: 1.5; }
a { color: #097783; } button, input, select, textarea { font: inherit; }
.topbar { border-bottom: 2px solid #e1ecec; background: #fff; }
.shell { max-width: 980px; margin: auto; padding: 0 22px; }
.nav { padding: 16px 22px; display: flex; align-items: center; justify-content: space-
between; gap: 20px; }
.brand { font-weight: 800; text-decoration: none; font-size: 20px; color: #126774; }
.nav-right { display: flex; align-items: center; gap: 16px; flex-wrap: wrap; }
.nav-right form, .actions form { margin: 0; }
.nav-link { border: 0; background: transparent; color: #126774; cursor: pointer; }
.main { min-height: calc(100vh - 160px); padding-top: 40px; padding-bottom: 48px; }
h1 { font-size: clamp(28px, 4vw, 38px); margin: 6px 0 12px; }
p { color: #435c67; }
.eyebrow { color: #087b84; font-size: 12px; letter-spacing: 1.4px; font-weight: 800; }
.auth-panel, .form-panel { max-width: 560px; border: 1px solid #d4e3e5; border-radius:
18px; padding: 26px; background: white; }
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/public/css/little-star.css

මෙම code එක source ZIP එකේ `overlay/public/css/little-star.css` ගොනුවේ ඇත. පිටුව 2/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/public/css/little-star.css

```
.stack, label { display: grid; gap: 8px; }
.stack { gap: 17px; margin-top: 22px; }
label { color: #204452; font-weight: 700; font-size: 14px; }
input, textarea, select { width: 100%; min-width: 0; border: 1px solid #aebfc3; border-
radius: 10px; padding: 11px 12px; background: white; color: #163542; }
input:focus, textarea:focus, select:focus { outline: 2px solid #2bbfc4; outline-offset:
2px; }
.btn { display: inline-flex; align-items: center; justify-content: center; border: 1px
solid #097783; background: #087b84; border-radius: 9px; padding: 9px 15px; color: white;
font-weight: 750; text-decoration: none; cursor: pointer; }
.btn.secondary { background: #fff; color: #087b84; }
.btn.danger { background: #a83040; border-color: #a83040; }
.hint { font-size: 14px; margin-top: 20px; }
.alert { padding: 14px; border: 1px solid; border-radius: 8px; margin-bottom: 24px; }
.alert.success { border-color: #48a684; color: #1b644e; background: #fff; }
.alert.error { border-color: #c86d75; color: #8e2330; background: #fff; }
.page-head { display: flex; align-items: center; justify-content: space-between; gap:
15px; flex-wrap: wrap; }
.filters { margin: 24px 0; display: flex; align-items: end; gap: 12px; flex-wrap: wrap; }
.filters select { min-width: 170px; }
.task-grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(245px, 1fr));
gap: 18px; }
.task-card { border: 1px solid #d3e4e4; border-radius: 15px; padding: 20px; background:
#fff; }
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/public/css/little-star.css

මෙම code එක source ZIP එකේ `overlay/public/css/little-star.css` ගොනුවේ ඇත. පිටුව 3/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/public/css/little-star.css

```
.task-top, .actions { display: flex; gap: 10px; align-items: center; justify-content:
space-between; flex-wrap: wrap; }
.task-top strong { font-size: 18px; overflow-wrap: anywhere; }
.chip { color: #087b84; border: 1px solid #c2dadb; border-radius: 20px; padding: 3px 9px;
font-size: 12px; }
.task-card p { min-height: 48px; overflow-wrap: anywhere; }
.task-card small { color: #607a81; }
.actions { justify-content: flex-start; margin-top: 16px; }
.empty { border: 1px dashed #abc3c3; padding: 30px; border-radius: 14px; }
.footer { text-align: center; padding: 20px; border-top: 1px solid #d9e8ea; color:
#6c838a; }
.pagination { margin-top: 24px; }
@media(max-width: 560px) { .nav { align-items: start; flex-direction: column; } .main
{ padding-top: 24px; } .auth-panel, .form-panel { padding: 18px; } }
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/tests/Feature/TaskCrudTest.php

මෙම code එක source ZIP එකේ `overlay/tests/Feature/TaskCrudTest.php` ගොනුවේ ඇත. පිටුව 1/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/tests/Feature/TaskCrudTest.php

```php
<?php

namespace Tests\Feature;

use App\Models\Task;
use App\Models\User;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;

class TaskCrudTest extends TestCase
{
    use RefreshDatabase;

    public function test_a_guest_is_redirected_to_login(): void
    {
        $this->get('/tasks')->assertRedirect('/login');
    }

    public function test_registration_creates_and_logs_in_a_user(): void
    {
        $this->post('/register', [
            'name' => 'Kavi', 'email' => 'kavi@example.test',
            'password' => 'safePassword123',
            'password_confirmation' => 'safePassword123',
        ])->assertRedirect('/tasks');
        $this->assertAuthenticated();
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/tests/Feature/TaskCrudTest.php

මෙම code එක source ZIP එකේ `overlay/tests/Feature/TaskCrudTest.php` ගොනුවේ ඇත. පිටුව 2/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/tests/Feature/TaskCrudTest.php

```
        $this->assertDatabaseHas('users', ['email' => 'kavi@example.test']);
    }

    public function test_login_and_logout_work(): void
    {
        $user = User::factory()->create(['password' => bcrypt('Password123')]);
        $this->post('/login', ['email' => $user->email, 'password' => 'Password123'])
            ->assertRedirect('/tasks');
        $this->assertAuthenticatedAs($user);
        $this->post('/logout')->assertRedirect('/login');
        $this->assertGuest();
    }

    public function test_user_can_create_update_and_delete_own_task(): void
    {
        $user = User::factory()->create();
        $this->actingAs($user)->post('/tasks', [
            'title' => 'Read a book', 'details' => 'Two pages', 'due_date' => '',
'is_done' => '0',
        ])->assertRedirect('/tasks');
        $task = $user->tasks()->firstOrFail();
        $this->assertSame('Read a book', $task->title);
        $this->actingAs($user)->put('/tasks/'.$task->id, [
            'title' => 'Read ten pages', 'details' => '', 'due_date' => '', 'is_done' =>
'1',
        ])->assertRedirect('/tasks');
        $this->assertTrue($task->fresh()->is_done);
        $this->actingAs($user)->delete('/tasks/'.$task->id)->assertRedirect('/tasks');
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

Source Appendix • overlay/tests/Feature/TaskCrudTest.php

මෙම code එක source ZIP එකේ `overlay/tests/Feature/TaskCrudTest.php` ගොනුවේ ඇත. පිටුව 3/3. සම්පූර්ණ file එක සංස්කරණය කිරීමට ZIP file එක භාවිතා කරන්න.

Visual Studio Code — overlay/tests/Feature/TaskCrudTest.php

```
        $this->assertDatabaseMissing('tasks', ['id' => $task->id]);
    }

     public function test_user_cannot_read_edit_or_delete_another_users_task(): void
    {
        $alice = User::factory()->create();
        $bob = User::factory()->create();
        $task = $alice->tasks()->create(['title' => 'Alice secret', 'is_done' => false]);
        $this->actingAs($bob)->get('/tasks')->assertDontSee('Alice secret');
        $this->actingAs($bob)->get('/tasks/'.$task->id.'/edit')->assertNotFound();
        $this->actingAs($bob)->put('/tasks/'.$task->id, [
            'title' => 'Bad edit', 'details' => '', 'due_date' => '', 'is_done' => '1',
        ])->assertNotFound();
        $this->actingAs($bob)->delete('/tasks/'.$task->id)->assertNotFound();
        $this->assertDatabaseHas('tasks', ['id' => $task->id, 'title' => 'Alice secret']);
    }

     public function test_invalid_task_is_not_saved(): void
    {
        $user = User::factory()->create();
        $this->actingAs($user)->post('/tasks', [
            'title' => '', 'is_done' => 'unexpected',
        ])->assertSessionHasErrors(['title', 'is_done']);
        $this->assertDatabaseCount('tasks', 0);
    }
}
```

සටහන: Code lines සංක්ෂිප්තව දක්වා ඇත. File එකේ English comments පියවරෙන් පියවර තේරුම් ගන්න උදව් වේ.

මගේ Laravel ගමන



ඔබට හැකි වුණේ මොනවාද?

Laravel route, Controller, Blade view, Eloquent model, migration, login, registration, CRUD, authorization සහ QA එකට සම්බන්ධ කරගෙන අධ්‍යාපනික Task Manager එකක් සැලසුම් කළා.

මගේ project නම: _____

මම අවසන් කළ දිනය: _____

ඊළඟට හදන feature එක: _____

egotechworld.com