**Project Report**

**CLUSY OS  
Clustered Operating System**

**Submitted By: Firnas Ahamed**

**Date of Submission: 14/03/2024**

# ****Abstract****

Table of Contents

[Abstract 1](#_Toc149676847)

[1. Introduction: 4](#_Toc149676848)

[2. Device Information: 5](#_Toc149676849)

[2.1 Device Specifications 5](#_Toc149676850)

[2.2 Common Usage Scenarios: 5](#_Toc149676851)

[2.2.1 Multimedia Applications 5](#_Toc149676852)

[2.2.2 Gaming 5](#_Toc149676853)

[2.2.3 Productivity Tasks 5](#_Toc149676854)

[2.3 Device Special Features 6](#_Toc149676855)

[2.3.1 Camera Capabilities 6](#_Toc149676856)

[2.3.2 Sensors 6](#_Toc149676857)

[2.4 Compatibility and Hardware Resources 6](#_Toc149676858)

[3. Requirements 7](#_Toc149676859)

[3.1 High Availability 7](#_Toc149676860)

[3.2 Load Balancing 7](#_Toc149676861)

[3.3 Fault Tolerance 7](#_Toc149676862)

[3.4 Scalability 8](#_Toc149676863)

[3.5 Resource Management: 8](#_Toc149676864)

[3.6 Security and Authentication: 8](#_Toc149676865)

[3.7 Data Synchronization: 8](#_Toc149676866)

[3.8 Performance Optimization: 9](#_Toc149676867)

[3.9 User Experience: 9](#_Toc149676868)

[4. Architecture Choices: 10](#_Toc149676869)

[4.1 Shared-Nothing Architecture 10](#_Toc149676870)

[4.1.1 What Is Shared Nothing Architecture? 10](#_Toc149676871)

[4.2 Master-Slave Model 12](#_Toc149676872)

[4.2.2 What is Master-Slave Model? 12](#_Toc149676873)

[5. Memory, Device, and File Management Policies 15](#_Toc149676874)

[5.1 Memory Management 15](#_Toc149676875)

[5.2 Device Management 15](#_Toc149676876)

[5.3 File Management 16](#_Toc149676877)

[6. Reliability, Performance, and Portability Strategies 17](#_Toc149676878)

[6.1 Reliability 17](#_Toc149676879)

[6.2 Performance 17](#_Toc149676880)

[6.3 Portability 17](#_Toc149676881)

[7. Assumptions 18](#_Toc149676882)

[7.1 Hardware Reliability 18](#_Toc149676883)

[7.2 User Behavior 18](#_Toc149676884)

[7.3 Device Selection 18](#_Toc149676885)

[7.4 Device Usage 18](#_Toc149676886)

[7.5 Operating System Requirements 18](#_Toc149676887)

[7.6 System Architecture 18](#_Toc149676888)

[7.7 Critical Evaluation of Architecture 18](#_Toc149676889)

[8. Critical Evaluation 19](#_Toc149676890)

[8.1 Advantages of the Chosen Architecture 19](#_Toc149676891)

[8.2 Potential Challenges of the Chosen Architecture 19](#_Toc149676892)

[8.3 Complexity of Distributed Memory Management 19](#_Toc149676893)

[8.4 Trade-offs in Memory Management 20](#_Toc149676894)

[9. Existing Research 20](#_Toc149676895)

[9.1 Influential Research Papers 20](#_Toc149676896)

[9.2 Insights Gained from Research 21](#_Toc149676897)

[11. References 22](#_Toc149676898)

# 1. Introduction:

# 2. A step by step progress how

## 2.1 Environment Setup

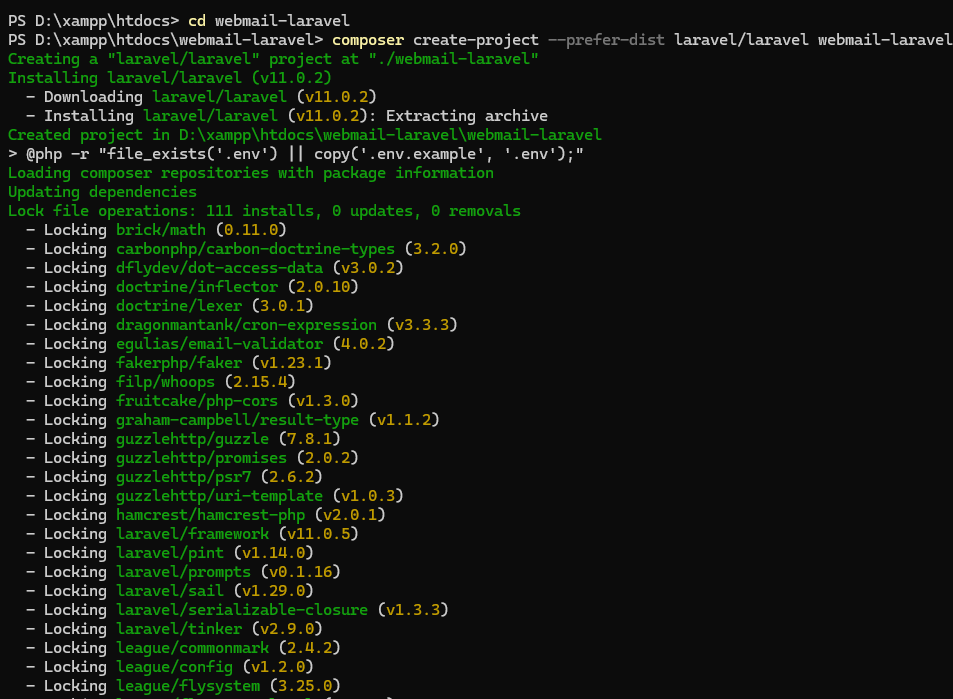
1. Github Repo created

A computer screen with white text

Description automatically generated

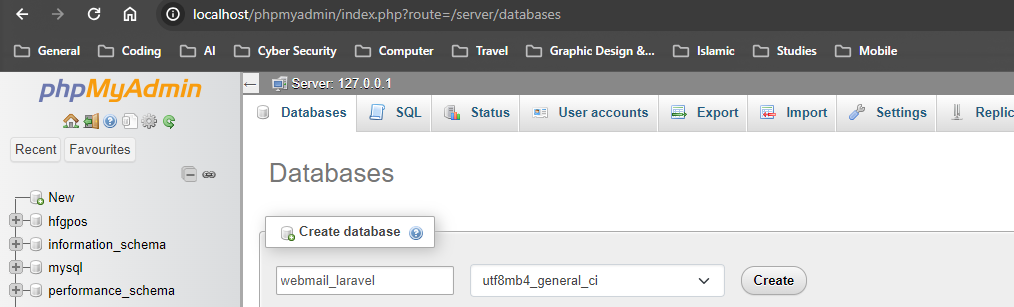
2. Install Laravel: First, you need to install Laravel. You can do this via Composer, assuming you have Composer installed on your system. Run the following command in your terminal:

composer create-project --prefer-dist laravel/laravel webmail-laravel



A screenshot of a computer program

Description automatically generated

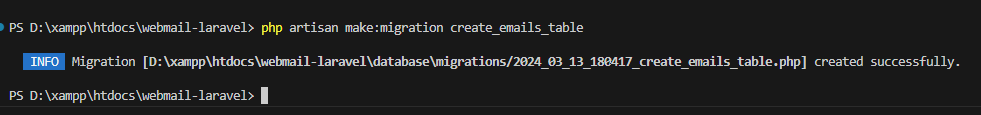
3. DB Creation in MYSQL PHPmyadmin

4. Database Configuration: Make sure your database connection details are correctly configured in the .env file.

A screenshot of a computer program

Description automatically generated

5. Migration: Create a migration for the email table by running the following command:

php artisan make:migration create\_emails\_table

6. edited the Migration File with these commands additionally

Schema::create('emails', function (Blueprint $table) {

            $table->id();

            $table->string('recipient');

            $table->string('subject');

            $table->text('message');

            $table->enum('status', ['sent', 'draft'])->default('draft');

            $table->timestamp('sent\_at')->nullable();

            $table->timestamps();

        });

    }

A black background with many small dots

Description automatically generated with medium confidence7. Migration Command

8. Implement Laravel Default User Authentication

Install Laravel Breeze:

let's install Laravel Breeze for authentication:

composer require laravel/breeze --dev

A screenshot of a computer

Description automatically generated9.asas

10.

php artisan serv

and register a user

A blue and black rectangular object

Description automatically generated with medium confidenceA screenshot of a login form

Description automatically generated

A screenshot of a computer

Description automatically generatedChanged the welcome page

Now I am going to add a button on the dashboard for send new emails

To do that initially I added in the routes/web.php

use App\Http\Controllers\EmailController;

Route::get('/send-email', [EmailController::class, 'create'])->name('send-email.create');

Route::middleware('auth')->group(function () {

    Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');

    Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');

    Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');

    Route::get('/send-email', [EmailController::class, 'create'])->name('send-email.create');

});

Create the Email Controller

php artisan make:controller EmailController

Edited the EmailController

A screen shot of a computer program

Description automatically generated

A computer screen with text

Description automatically generatedCreated a new Blade file for New Email Form Creation

Now Going to add a table in Dashboard after the button created with laraval default paginations to view all the emails from DB

Added a function in EmailController

public function dashboard(){

        $emails = Email::paginate(10);

        return view('dashboard', compact('emails'));

    }

To make visible now added a model to get from emails table

PS D:\xampp\htdocs\webmail-laravel> php artisan make:model Email

   INFO  Model [D:\xampp\htdocs\webmail-laravel\app\Models\Email.php] created successfully.

Edited the email model file

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Email extends Model

{

    use HasFactory;

    protected $table = 'emails';

    protected $fillable = [

        'recipient',

        'subject',

        'message',

        'status',

        'created\_at'

    ];

}

To provide fake data to your database using factories in Laravel, you'll need to follow these steps:

Define Factory

Define a factory for your Email model. If you haven't already created a factory, you can generate one using the artisan command:

php artisan make:factory EmailFactory --model=Email

**Define Factory Attributes**

In your **EmailFactory.php** file, define the attributes you want to generate for your emails. You can use Faker to generate realistic data. For example:

php  
return [

            'recipient' => $this->faker->email,

            'subject' => $this->faker->sentence,

            'message' => $this->faker->paragraph,

            'status' => $this->faker->randomElement(['sent', 'draft']),

            'created\_at' => $this->faker->dateTimeBetween('-1 year', 'now'),

        ];

    }

**Seeder**

Create a seeder file using the following artisan command:

PS D:\xampp\htdocs\webmail-laravel> php artisan make:seeder EmailSeeder

   INFO  Seeder [D:\xampp\htdocs\webmail-laravel\database\seeders\EmailSeeder.php] created successfully.

public function run(): void

    {

        //For 15 fake emails

        Email::factory()->count(15)->create();

    }

**Run Seeder**

Finally, run the seeder to populate your database with fake data:

php artisan db:seed --class=EmailSeeder