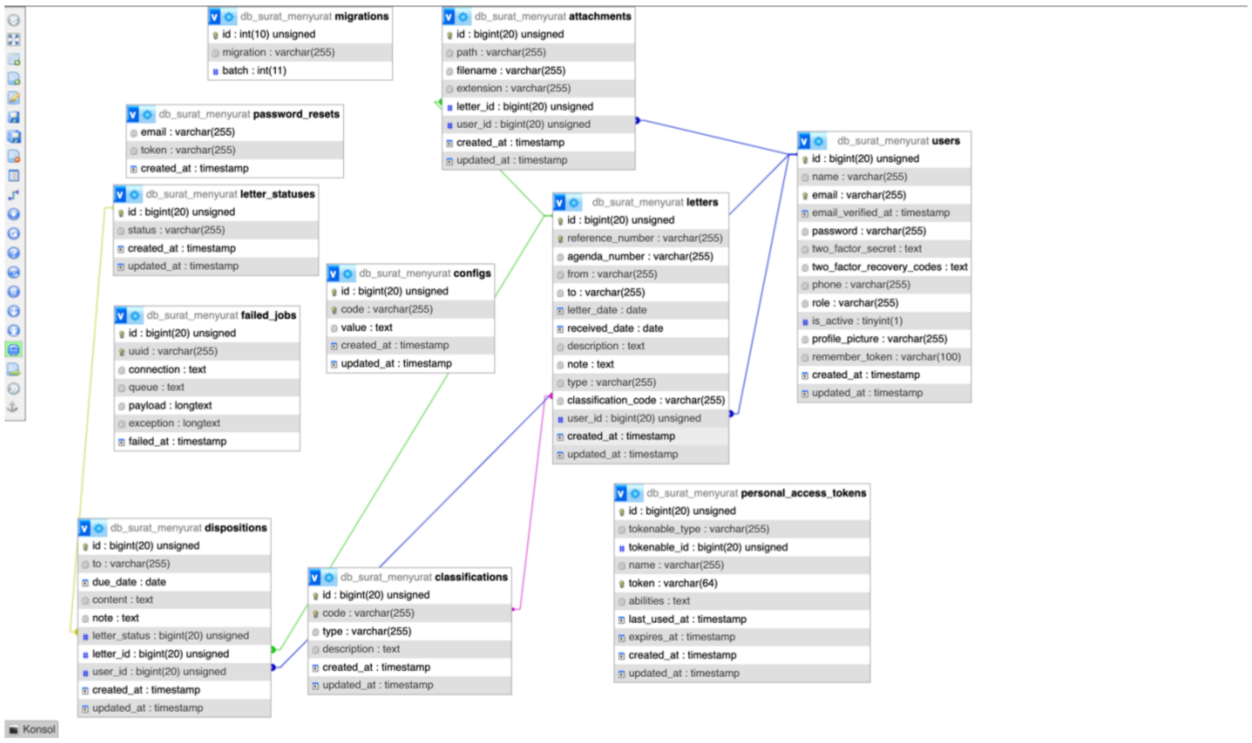
NAMA : AHMAD RIAN SYAIFULLAH RITONGA
NIM : H1D022010
SOAL : SISTEM SURAT MENYURAT (NIM AKHIR 0)

LAPORAN UAS PEMROGRAMAN WEB II

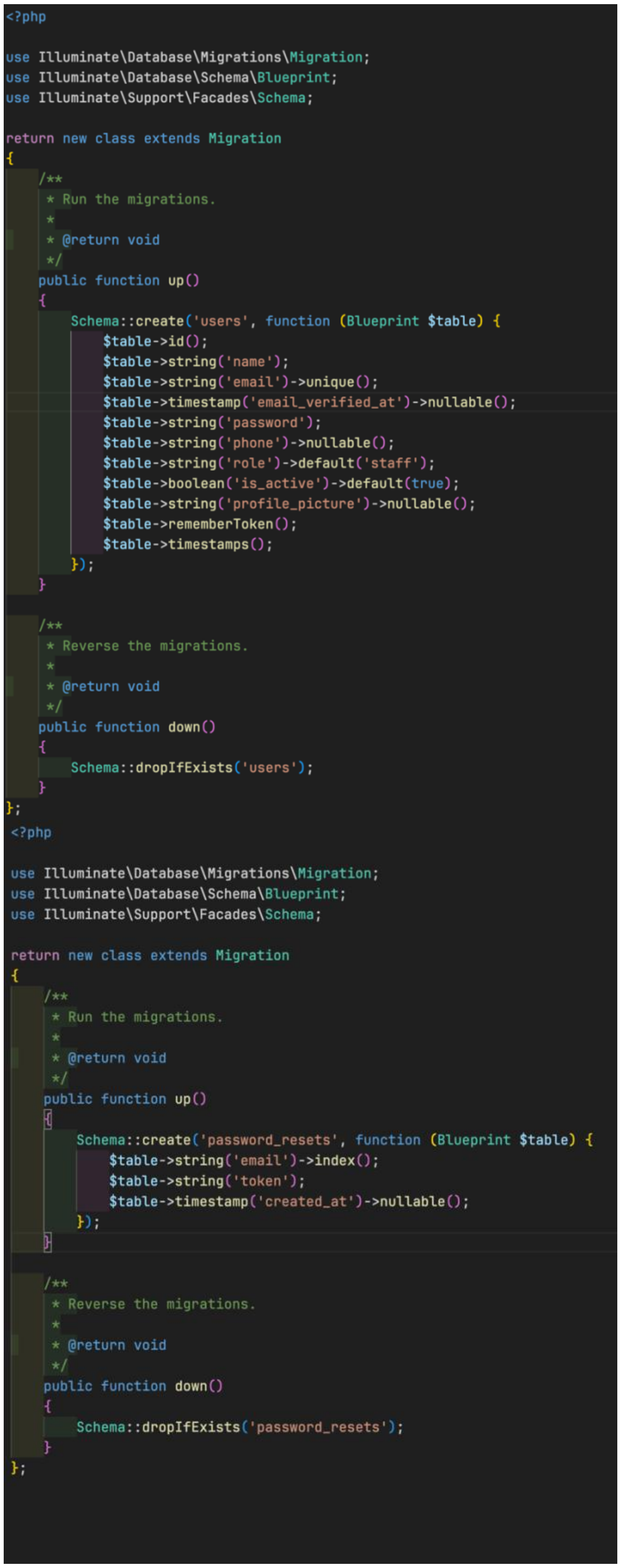
1. Link github : https://github.com/ahmad-rian/Uas_Pemweb_Ahmad-Rian_H1D022010.git

2. Desainer :



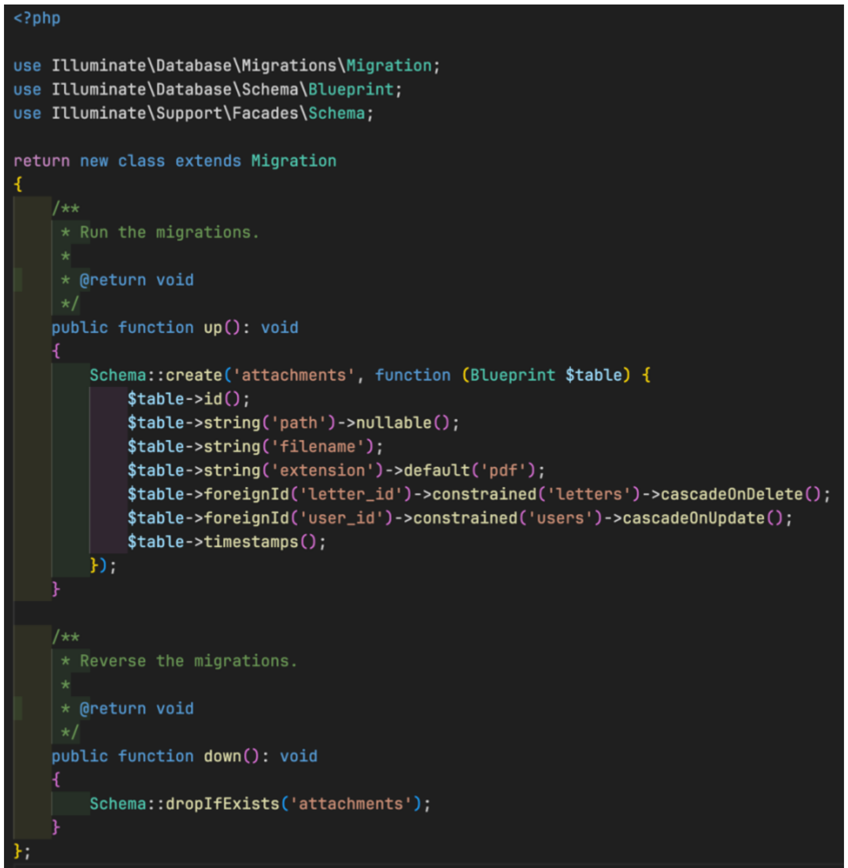
```
db_surat_menyurat migrations
id : int(10) unsigned
migration : varchar(255)
batch : int(11)

db_surat_menyurat attachments
id : bigint(20) unsigned
path : varchar(255)
filename : varchar(255)
extension : varchar(255)
letter_id : bigint(20) unsigned
user_id : bigint(20) unsigned
created_at : timestamp
updated_at : timestamp

db_surat_menyurat password_resets
email : varchar(255)
token : varchar(255)
created_at : timestamp

db_surat_menyurat users
id : bigint(20) unsigned
name : varchar(255)
email : varchar(255)
email_verified_at : timestamp
password : varchar(255)
two_factor_secret : text
two_factor_recovery_codes : text
phone : varchar(255)
role : varchar(255)
is_active : tinyint(1)
profile_picture : varchar(255)
remember_token : varchar(100)
created_at : timestamp
updated_at : timestamp

db_surat_menyurat letter_statuses
id : bigint(20) unsigned
status : varchar(255)
created_at : timestamp
updated_at : timestamp

db_surat_menyurat letters
id : bigint(20) unsigned
reference_number : varchar(255)
agenda_number : varchar(255)
from : varchar(255)
to : varchar(255)
letter_date : date
received_date : date
description : text
note : text
type : varchar(255)
classification_code : varchar(255)
user_id : bigint(20) unsigned
created_at : timestamp
updated_at : timestamp

db_surat_menyurat configs
id : bigint(20) unsigned
code : varchar(255)
value : text
created_at : timestamp
updated_at : timestamp

db_surat_menyurat failed_jobs
id : bigint(20) unsigned
uuid : varchar(255)
connection : text
queue : text
payload : longtext
exception : longtext
failed_at : timestamp

db_surat_menyurat personal_access_tokens
id : bigint(20) unsigned
tokenable_type : varchar(255)
tokenable_id : bigint(20) unsigned
name : varchar(255)
token : varchar(64)
abilities : text
last_used_at : timestamp
expires_at : timestamp
created_at : timestamp
updated_at : timestamp

db_surat_menyurat dispositions
id : bigint(20) unsigned
to : varchar(255)
due_date : date
content : text
note : text
letter_status : bigint(20) unsigned
letter_id : bigint(20) unsigned
user_id : bigint(20) unsigned
created_at : timestamp
updated_at : timestamp

db_surat_menyurat classifications
id : bigint(20) unsigned
code : varchar(255)
type : varchar(255)
description : text
created_at : timestamp
updated_at : timestamp
```

3. FILE MIGRATIONS :

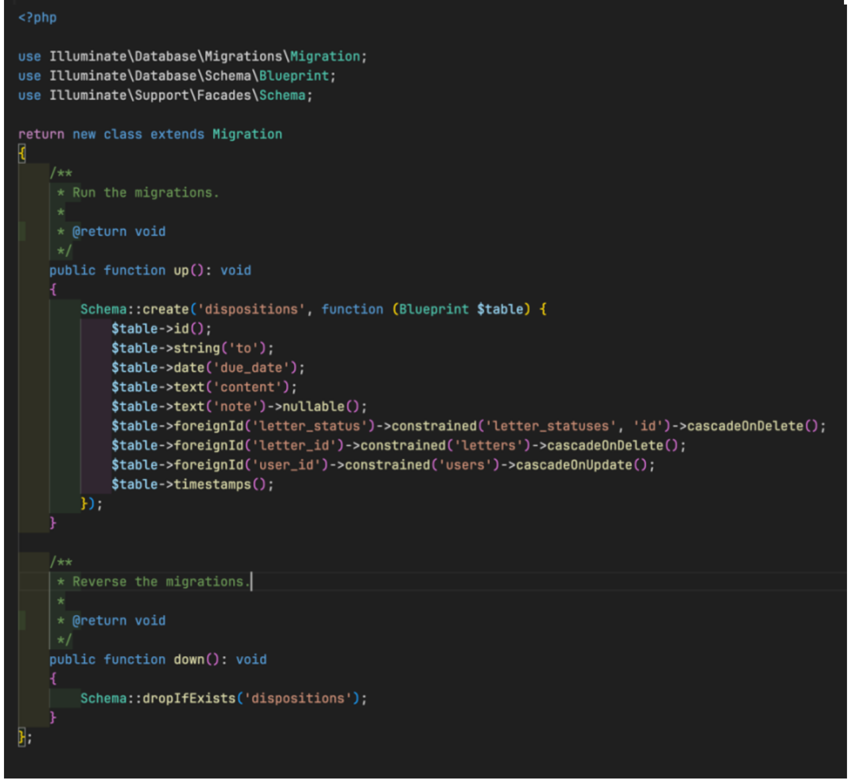


```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->string('phone')->nullable();
            $table->string('role')->default('staff');
            $table->boolean('is_active')->default(true);
            $table->string('profile_picture')->nullable();
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
};
```

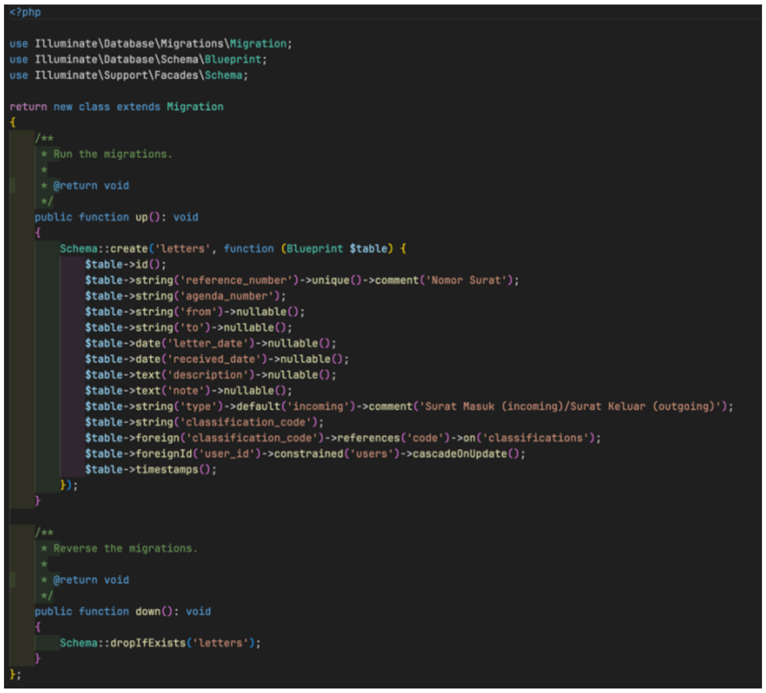
```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
};
```



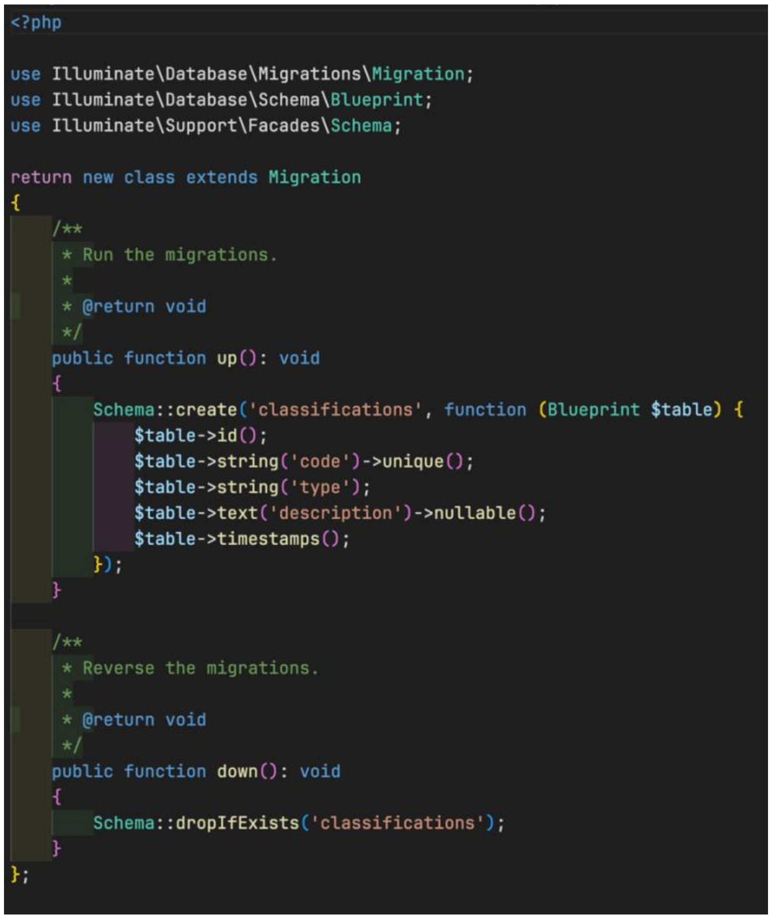
```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(): void
    {
        Schema::create('attachments', function (Blueprint $table) {
            $table->id();
            $table->string('path')->nullable();
            $table->string('filename');
            $table->string('extension')->default('pdf');
            $table->foreignId('letter_id')->constrained('letters')->cascadeOnDelete();
            $table->foreignId('user_id')->constrained('users')->cascadeOnUpdate();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down(): void
    {
        Schema::dropIfExists('attachments');
    }
};
```



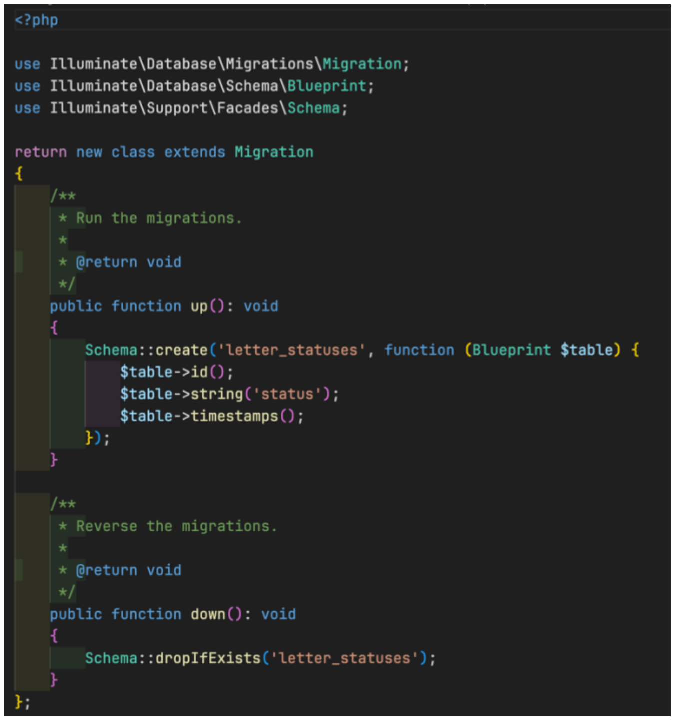
```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(): void
    {
        Schema::create('dispositions', function (Blueprint $table) {
            $table->id();
            $table->string('to');
            $table->date('due_date');
            $table->text('content');
            $table->text('note')->nullable();
            $table->foreignId('letter_status')->constrained('letter_statuses', 'id')->cascadeOnDelete();
            $table->foreignId('letter_id')->constrained('letters')->cascadeOnDelete();
            $table->foreignId('user_id')->constrained('users')->cascadeOnUpdate();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down(): void
    {
        Schema::dropIfExists('dispositions');
    }
};
```



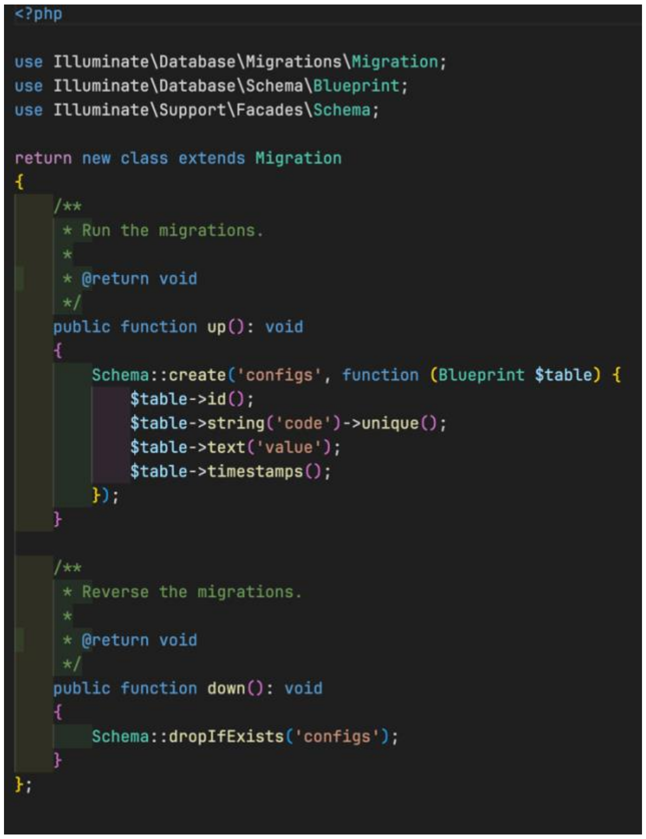
```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(): void
    {
        Schema::create('letters', function (Blueprint $table) {
            $table->id();
            $table->string('reference_number')->unique()->comment('Nomor Surat');
            $table->string('agenda_number');
            $table->string('from')->nullable();
            $table->string('to')->nullable();
            $table->date('letter_date')->nullable();
            $table->date('received_date')->nullable();
            $table->text('description')->nullable();
            $table->text('note')->nullable();
            $table->string('type')->default('incoming')->comment('Surat Masuk (incoming)/Surat Keluar (outgoing)');
            $table->string('classification_code');
            $table->foreign('classification_code')->references('code')->on('classifications');
            $table->foreignId('user_id')->constrained('users')->cascadeOnUpdate();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down(): void
    {
        Schema::dropIfExists('letters');
    }
};
```



```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(): void
    {
        Schema::create('classifications', function (Blueprint $table) {
            $table->id();
            $table->string('code')->unique();
            $table->string('type');
            $table->text('description')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down(): void
    {
        Schema::dropIfExists('classifications');
    }
};
```

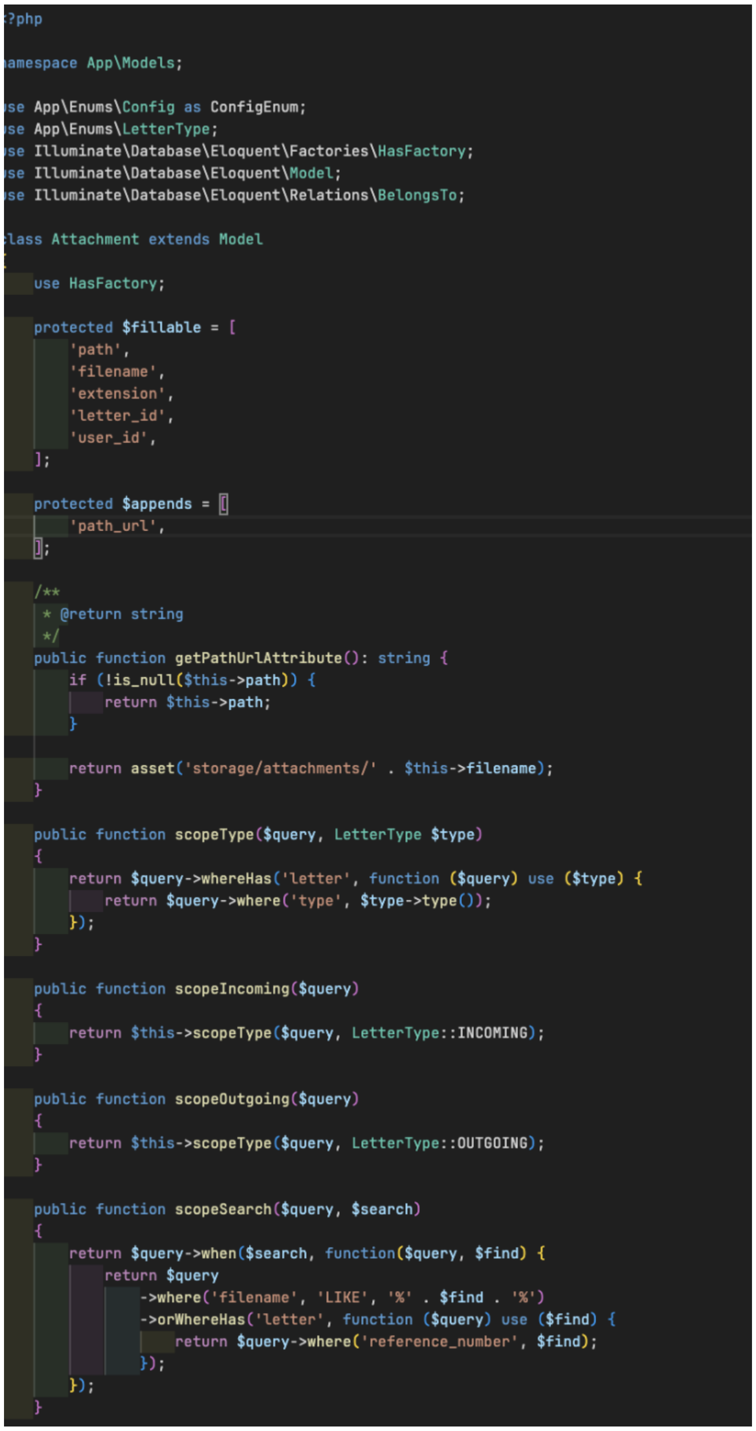


```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(): void
    {
        Schema::create('letter_statuses', function (Blueprint $table) {
            $table->id();
            $table->string('status');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down(): void
    {
        Schema::dropIfExists('letter_statuses');
    }
};
```

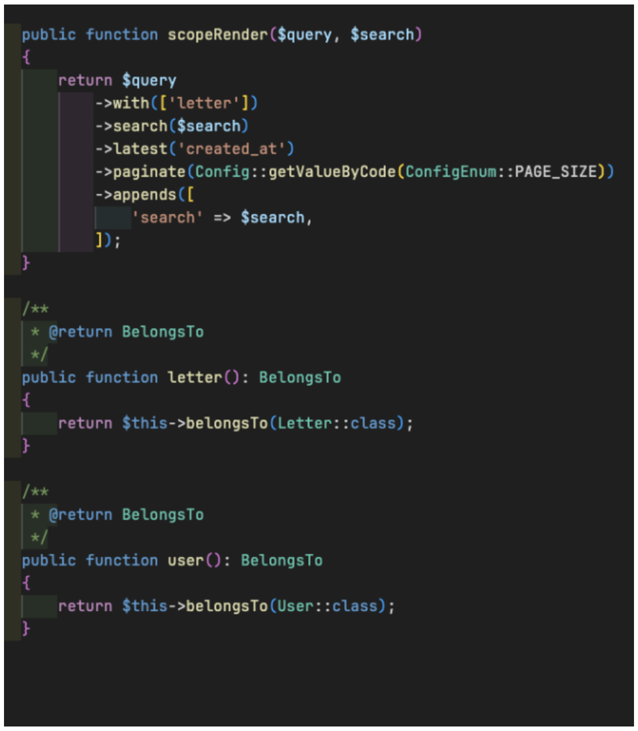


```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(): void
    {
        Schema::create('configs', function (Blueprint $table) {
            $table->id();
            $table->string('code')->unique();
            $table->text('value');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down(): void
    {
        Schema::dropIfExists('configs');
    }
};
```

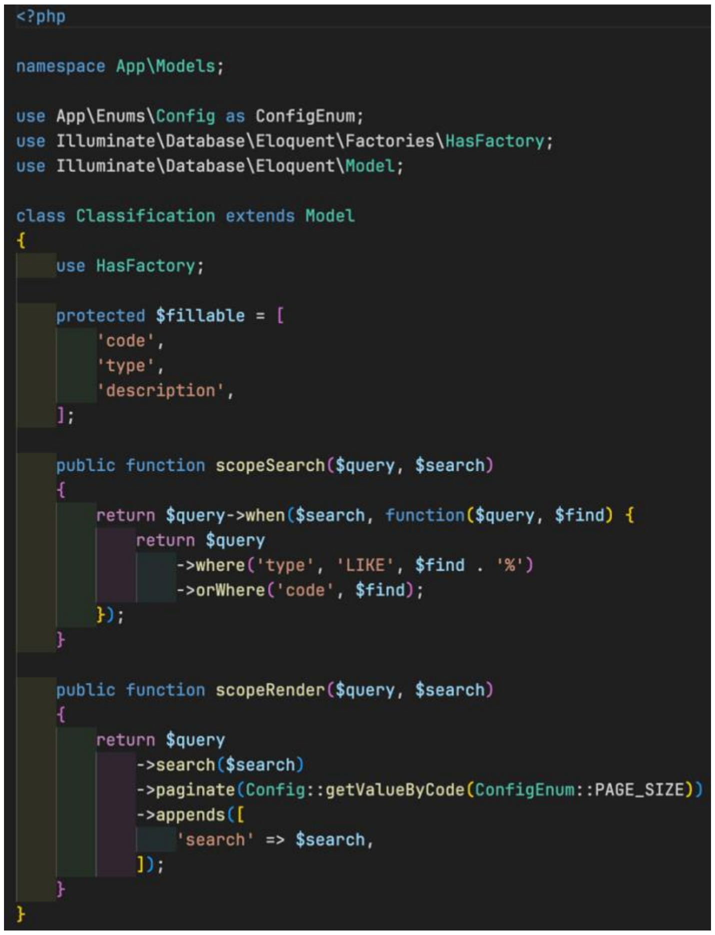
4. File Model :



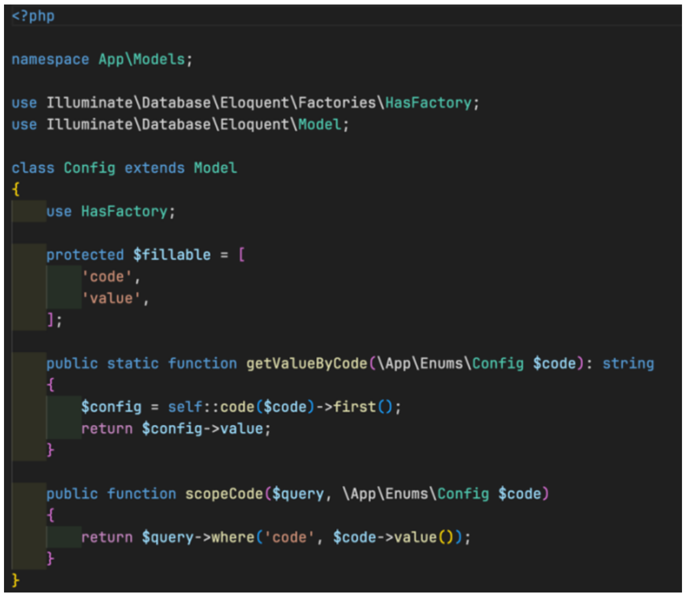
```
?php

amespace App\Models;

se App\Enums\Config as ConfigEnum;
se App\Enums\LetterType;
se Illuminate\Database\Eloquent\Factories\HasFactory;
se Illuminate\Database\Eloquent\Model;
se Illuminate\Database\Eloquent\Relations\BelongsTo;

lass Attachment extends Model

   use HasFactory;

   protected $fillable = [
       'path',
       'filename',
       'extension',
       'letter_id',
       'user_id',
   ];

   protected $appends = [
       'path_url',
   ];

   /**
    * @return string
    */
   public function getPathUrlAttribute(): string {
       if (!is_null($this->path)) {
           return $this->path;
       }

       return asset('storage/attachments/' . $this->filename);
   }

   public function scopeType($query, LetterType $type)
   {
       return $query->whereHas('letter', function ($query) use ($type) {
           return $query->where('type', $type->type());
       });
   }

   public function scopeIncoming($query)
   {
       return $this->scopeType($query, LetterType::INCOMING);
   }

   public function scopeOutgoing($query)
   {
       return $this->scopeType($query, LetterType::OUTGOING);
   }

   public function scopeSearch($query, $search)
   {
       return $query->when($search, function($query, $find) {
           return $query
               ->where('filename', 'LIKE', '%' . $find . '%')
               ->orWhereHas('letter', function ($query) use ($find) {
                   return $query->where('reference_number', $find);
               });
       });
   }
```



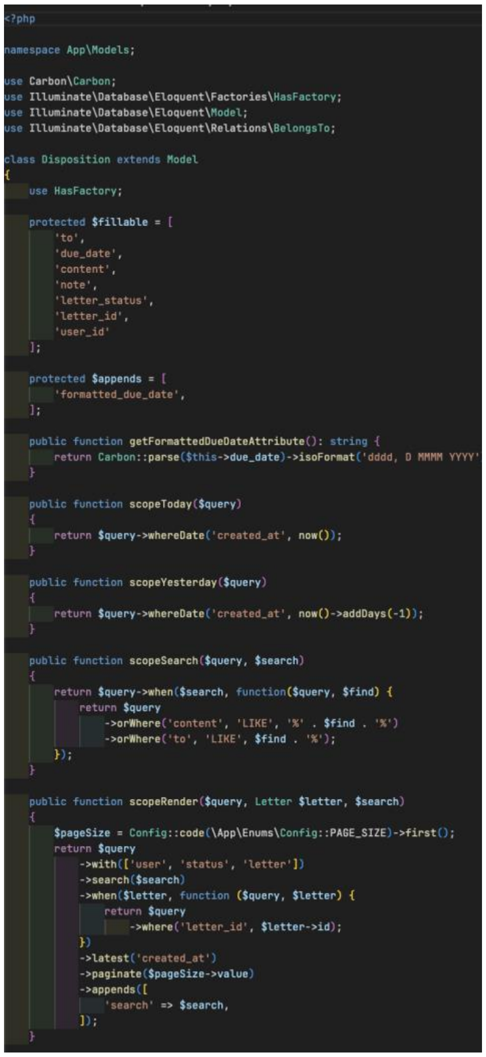
```php
public function scopeRender($query, $search)
{
    return $query
        ->with(['letter'])
        ->search($search)
        ->latest('created_at')
        ->paginate(Config::getValueByCode(ConfigEnum::PAGE_SIZE))
        ->appends([
            'search' => $search,
        ]);
}

/**
 * @return BelongsTo
 */
public function letter(): BelongsTo
{
    return $this->belongsTo(Letter::class);
}

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



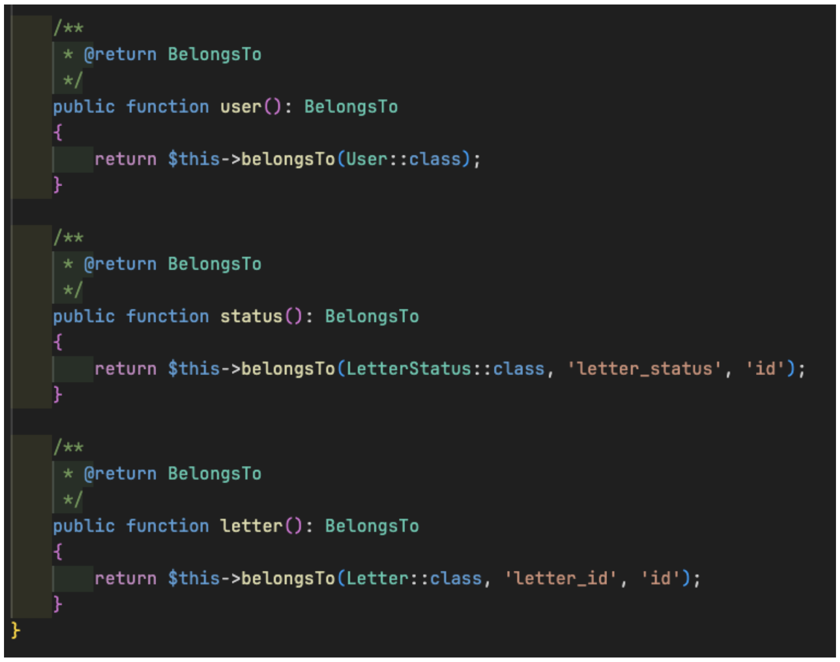
```php
<?php

namespace App\Models;

use App\Enums\Config as ConfigEnum;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Classification extends Model
{
    use HasFactory;

    protected $fillable = [
        'code',
        'type',
        'description',
    ];

    public function scopeSearch($query, $search)
    {
        return $query->when($search, function($query, $find) {
            return $query
                ->where('type', 'LIKE', $find . '%')
                ->orWhere('code', $find);
        });
    }

    public function scopeRender($query, $search)
    {
        return $query
            ->search($search)
            ->paginate(Config::getValueByCode(ConfigEnum::PAGE_SIZE))
            ->appends([
                'search' => $search,
            ]);
    }
}
```



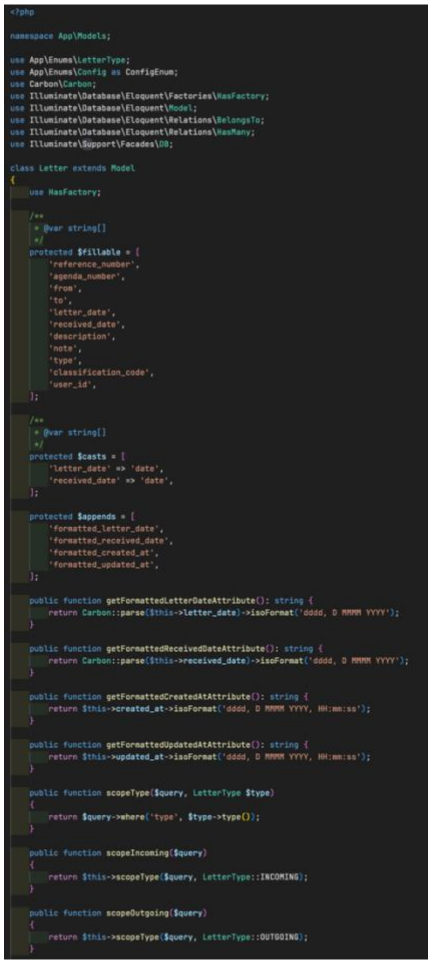
```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Config extends Model
{
    use HasFactory;

    protected $fillable = [
        'code',
        'value',
    ];

    public static function getValueByCode(\App\Enums\Config $code): string
    {
        $config = self::code($code)->first();
        return $config->value;
    }

    public function scopeCode($query, \App\Enums\Config $code)
    {
        return $query->where('code', $code->value());
    }
}
```



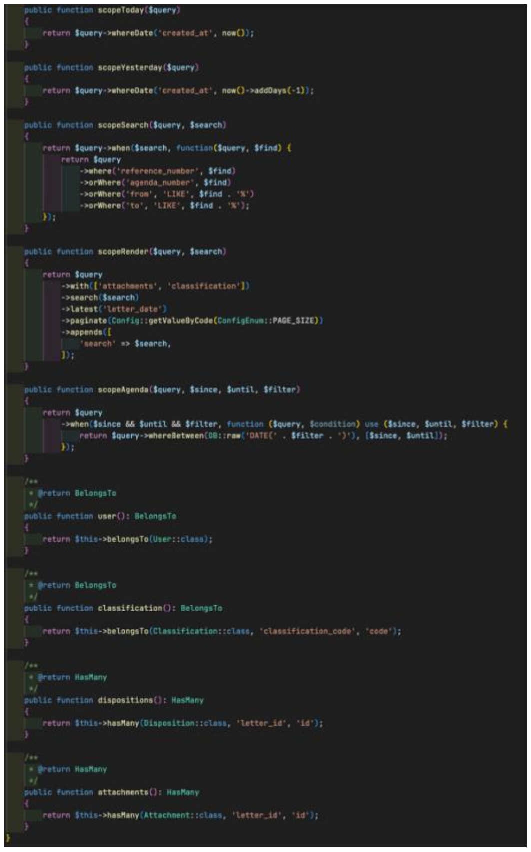
```php
<?php

namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Disposition extends Model
{
    use HasFactory;

    protected $fillable = [
        'to',
        'due_date',
        'content',
        'note',
        'letter_status',
        'letter_id',
        'user_id'
    ];

    protected $appends = [
        'formatted_due_date',
    ];

    public function getFormattedDueDateAttribute(): string {
        return Carbon::parse($this->due_date)->isFormat('dddd, D MMMM YYYY'
    }

    public function scopeToday($query)
    {
        return $query->whereDate('created_at', now());
    }

    public function scopeYesterday($query)
    {
        return $query->whereDate('created_at', now()->addDays(-1));
    }

    public function scopeSearch($query, $search)
    {
        return $query->when($search, function($query, $find) {
            return $query
                ->orWhere('content', 'LIKE', '%' . $find . '%')
                ->orWhere('to', 'LIKE', $find . '%');
        });
    }

    public function scopeRender($query, Letter $letter, $search)
    {
        $pageSize = Config::code(\App\Enums\Config::PAGE_SIZE)->first();
        return $query
            ->with(['user', 'status', 'letter'])
            ->search($search)
            ->when($letter, function ($query, $letter) {
                return $query
                    ->where('letter_id', $letter->id);
            })
            ->latest('created_at')
            ->paginate($pageSize->value)
            ->appends([
                'search' => $search,
            ]);
    }
```



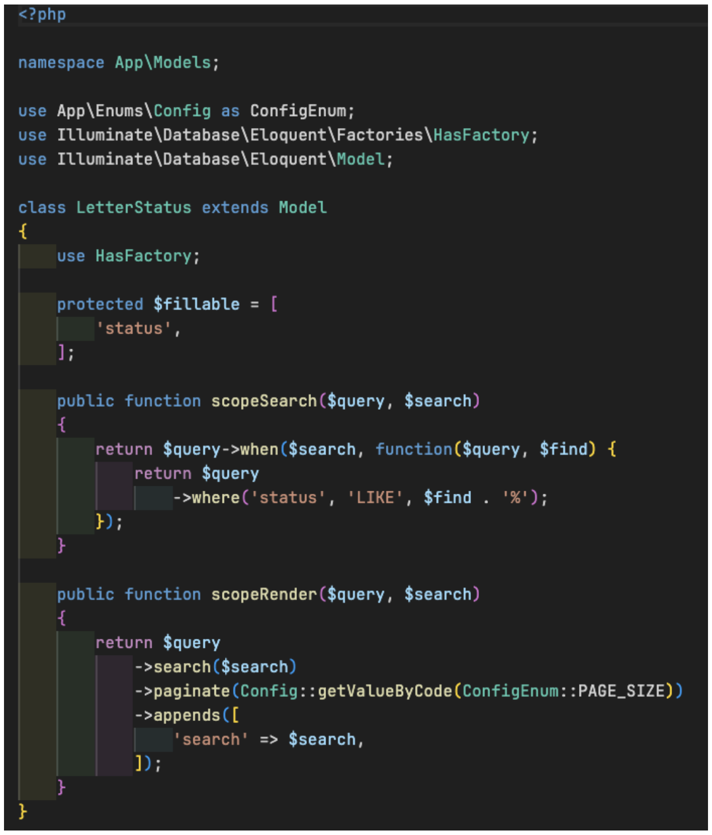
```php
    /**
     * @return BelongsTo
     */
    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }

    /**
     * @return BelongsTo
     */
    public function status(): BelongsTo
    {
        return $this->belongsTo(LetterStatus::class, 'letter_status', 'id');
    }

    /**
     * @return BelongsTo
     */
    public function letter(): BelongsTo
    {
        return $this->belongsTo(Letter::class, 'letter_id', 'id');
    }
}
```



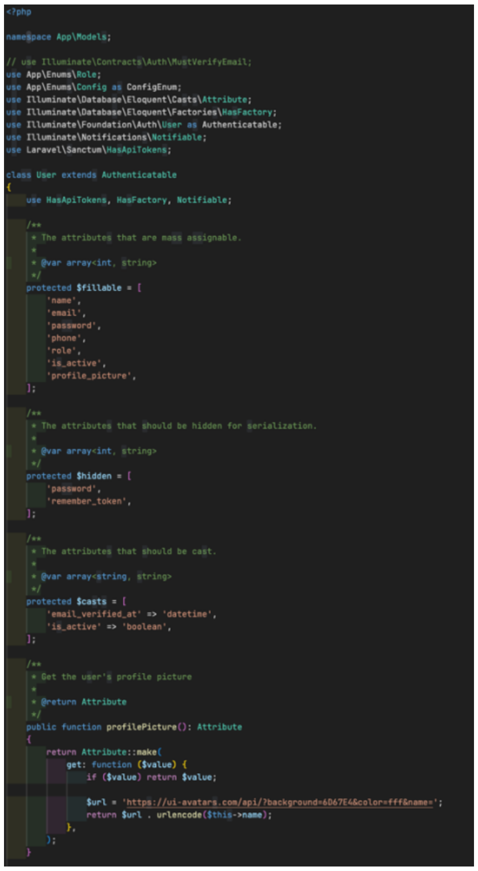
```php
<?php

namespace App\Models;

use App\Enums\LetterType;
use App\Enums\Config as ConfigEnum;
use Carbon\Carbon;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Support\Facades\DB;

class Letter extends Model
{
    use HasFactory;

    /**
     * @var string[]
     */
    protected $fillable = [
        'reference_number',
        'agenda_number',
        'from',
        'to',
        'letter_date',
        'received_date',
        'description',
        'note',
        'type',
        'classification_code',
        'user_id',
    ];

    /**
     * @var string[]
     */
    protected $casts = [
        'letter_date' => 'date',
        'received_date' => 'date',
    ];

    protected $appends = [
        'formatted_letter_date',
        'formatted_received_date',
        'formatted_created_at',
        'formatted_updated_at',
    ];

    public function getFormattedLetterDateAttribute(): string {
        return Carbon::parse($this->letter_date)->isoFormat('dddd, D MMMM YYYY');
    }

    public function getFormattedReceivedDateAttribute(): string {
        return Carbon::parse($this->received_date)->isoFormat('dddd, D MMMM YYYY');
    }

    public function getFormattedCreatedAtAttribute(): string {
        return $this->created_at->isoFormat('dddd, D MMMM YYYY, HH:mm:ss');
    }

    public function getFormattedUpdatedAtAttribute(): string {
        return $this->updated_at->isoFormat('dddd, D MMMM YYYY, HH:mm:ss');
    }

    public function scopeType($query, LetterType $type)
    {
        return $query->where('type', $type->type());
    }

    public function scopeIncoming($query)
    {
        return $this->scopeType($query, LetterType::INCOMING);
    }

    public function scopeOutgoing($query)
    {
        return $this->scopeType($query, LetterType::OUTGOING);
    }
```



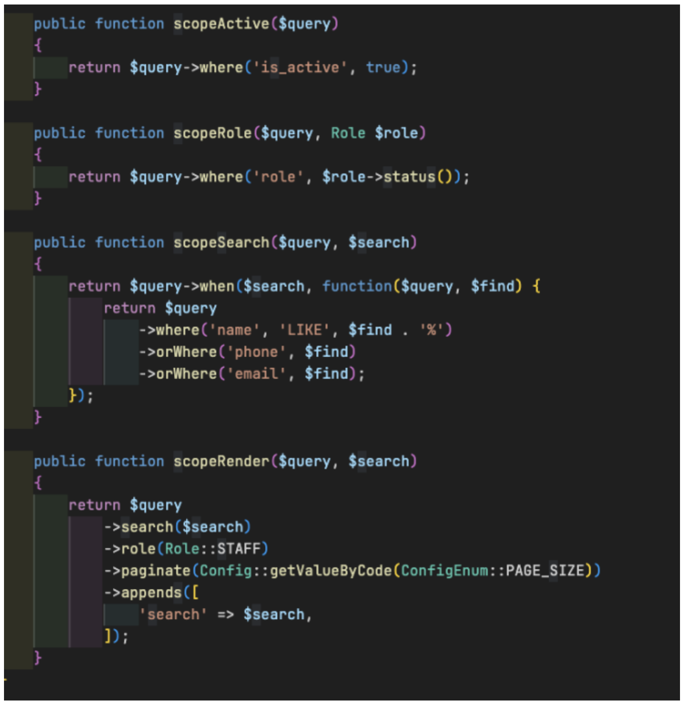
```php
    public function scopeToday($query)
    {
        return $query->whereDate('created_at', now());
    }

    public function scopeYesterday($query)
    {
        return $query->whereDate('created_at', now()->addDays(-1));
    }

    public function scopeSearch($query, $search)
    {
        return $query->when($search, function($query, $find) {
            return $query
                ->where('reference_number', $find)
                ->orWhere('agenda_number', $find)
                ->orWhere('from', 'LIKE', $find . '%')
                ->orWhere('to', 'LIKE', $find . '%');
        });
    }

    public function scopeRender($query, $search)
    {
        return $query
            ->with(['attachments', 'classification'])
            ->search($search)
            ->latest('letter_date')
            ->paginate(Config::getValueByCode(ConfigEnum::PAGE_SIZE))
            ->appends([
                'search' => $search,
            ]);
    }

    public function scopeAgenda($query, $since, $until, $filter)
    {
        return $query
            ->when($since && $until && $filter, function ($query, $condition) use ($since, $until, $filter) {
                return $query->whereBetween(DB::raw('DATE(' . $filter . ')'), [$since, $until]);
            });
    }

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

    /**
     * @return BelongsTo
     */
    public function classification(): BelongsTo
    {
        return $this->belongsTo(Classification::class, 'classification_code', 'code');
    }

    /**
     * @return HasMany
     */
    public function dispositions(): HasMany
    {
        return $this->hasMany(Disposition::class, 'letter_id', 'id');
    }

    /**
     * @return HasMany
     */
    public function attachments(): HasMany
    {
        return $this->hasMany(Attachment::class, 'letter_id', 'id');
    }
}
```



```php
<?php

namespace App\Models;

use App\Enums\Config as ConfigEnum;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class LetterStatus extends Model
{
    use HasFactory;

    protected $fillable = [
        'status',
    ];

    public function scopeSearch($query, $search)
    {
        return $query->when($search, function($query, $find) {
            return $query
                ->where('status', 'LIKE', $find . '%');
        });
    }

    public function scopeRender($query, $search)
    {
        return $query
            ->search($search)
            ->paginate(Config::getValueByCode(ConfigEnum::PAGE_SIZE))
            ->appends([
                'search' => $search,
            ]);
    }
}
```

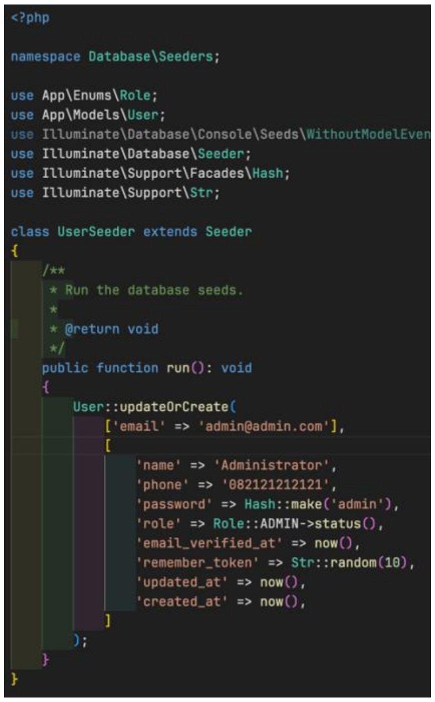


```php
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use App\Enums\Role;
use App\Enums\Config as ConfigEnum;
use Illuminate\Database\Eloquent\Casts\Attribute;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
        'phone',
        'role',
        'is_active',
        'profile_picture',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
        'is_active' => 'boolean',
    ];

    /**
     * Get the user's profile picture
     *
     * @return Attribute
     */
    public function profilePicture(): Attribute
    {
        return Attribute::make(
            get: function ($value) {
                if ($value) return $value;

                $url = 'https://ui-avatars.com/api/?background=6D67E4&color=fff&name=';
                return $url . urlencode($this->name);
            },
        );
    }
```

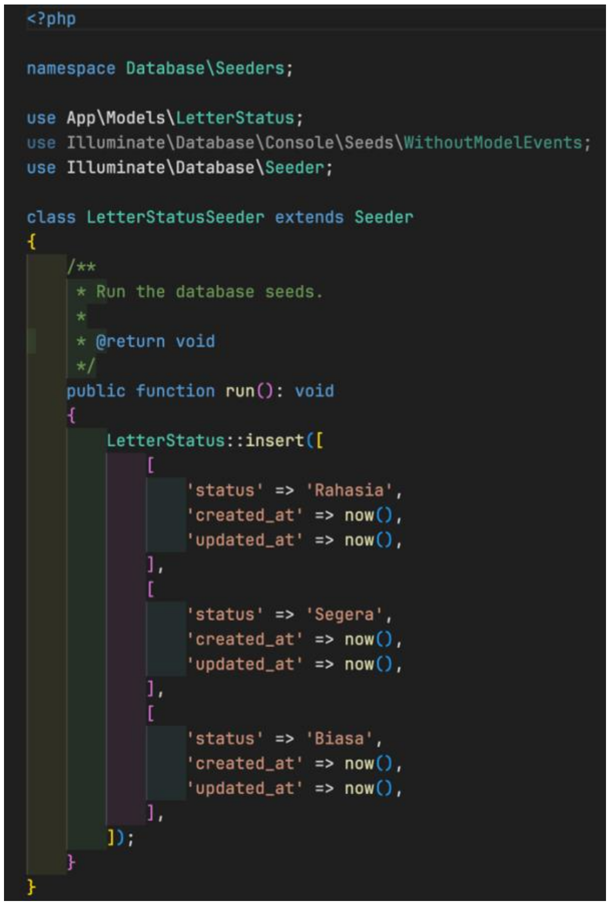


```php
    public function scopeActive($query)
    {
        return $query->where('is_active', true);
    }

    public function scopeRole($query, Role $role)
    {
        return $query->where('role', $role->status());
    }

    public function scopeSearch($query, $search)
    {
        return $query->when($search, function($query, $find) {
            return $query
                ->where('name', 'LIKE', $find . '%')
                ->orWhere('phone', $find)
                ->orWhere('email', $find);
        });
    }

    public function scopeRender($query, $search)
    {
        return $query
            ->search($search)
            ->role(Role::STAFF)
            ->paginate(Config::getValueByCode(ConfigEnum::PAGE_SIZE))
            ->appends([
                'search' => $search,
            ]);
    }
```

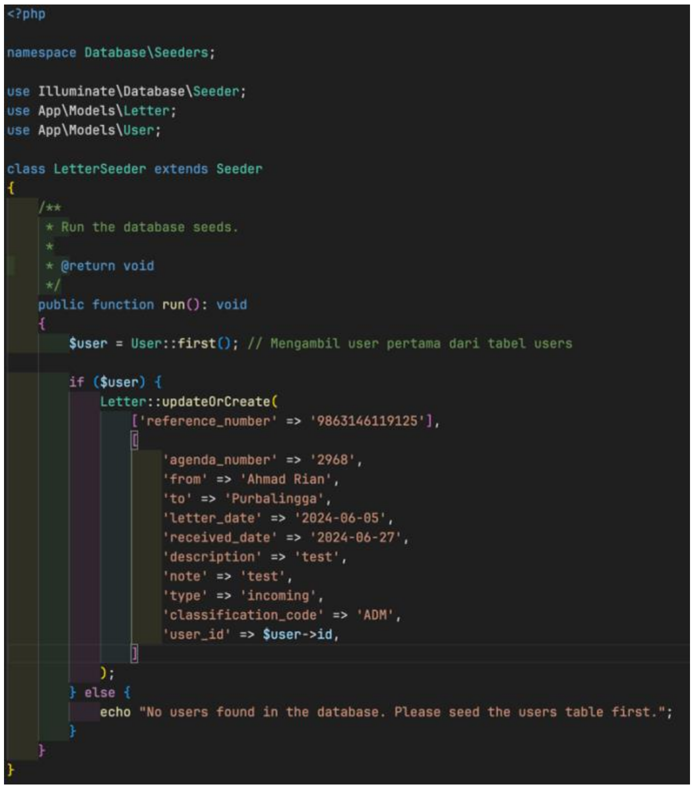
5. File Seeder :



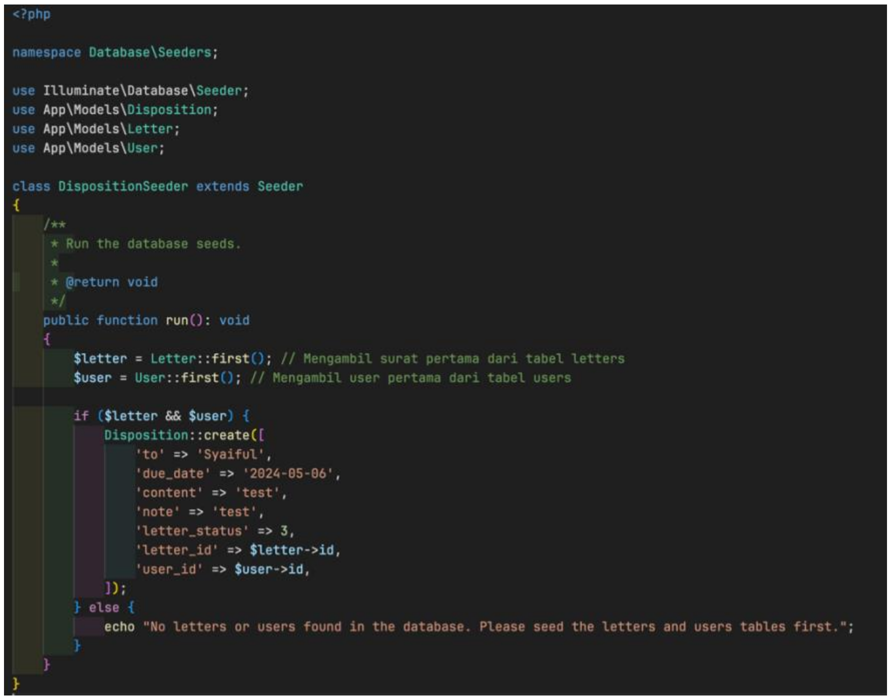
```php
<?php

namespace Database\Seeders;

use App\Enums\Role;
use App\Models\User;
use Illuminate\Database\Console\Seeds\WithoutModelEven
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run(): void
    {
        User::updateOrCreate(
            ['email' => 'admin@admin.com'],
            [
                'name' => 'Administrator',
                'phone' => '082121212121',
                'password' => Hash::make('admin'),
                'role' => Role::ADMIN->status(),
                'email_verified_at' => now(),
                'remember_token' => Str::random(10),
                'updated_at' => now(),
                'created_at' => now(),
            ]
        );
    }
}
```



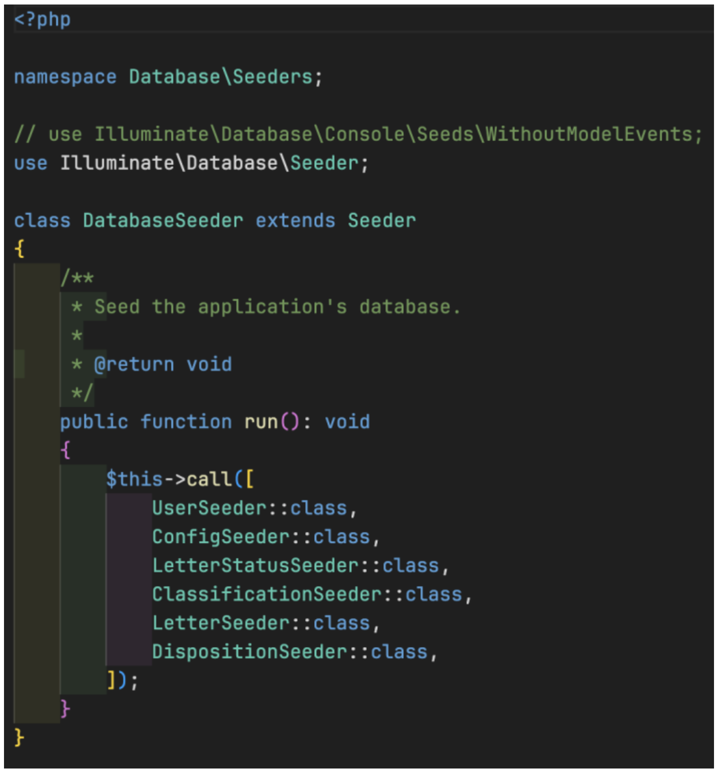
```php
<?php

namespace Database\Seeders;

use App\Models\LetterStatus;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class LetterStatusSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run(): void
    {
        LetterStatus::insert([
            [
                'status' => 'Rahasia',
                'created_at' => now(),
                'updated_at' => now(),
            ],
            [
                'status' => 'Segera',
                'created_at' => now(),
                'updated_at' => now(),
            ],
            [
                'status' => 'Biasa',
                'created_at' => now(),
                'updated_at' => now(),
            ],
        ]);
    }
}
```



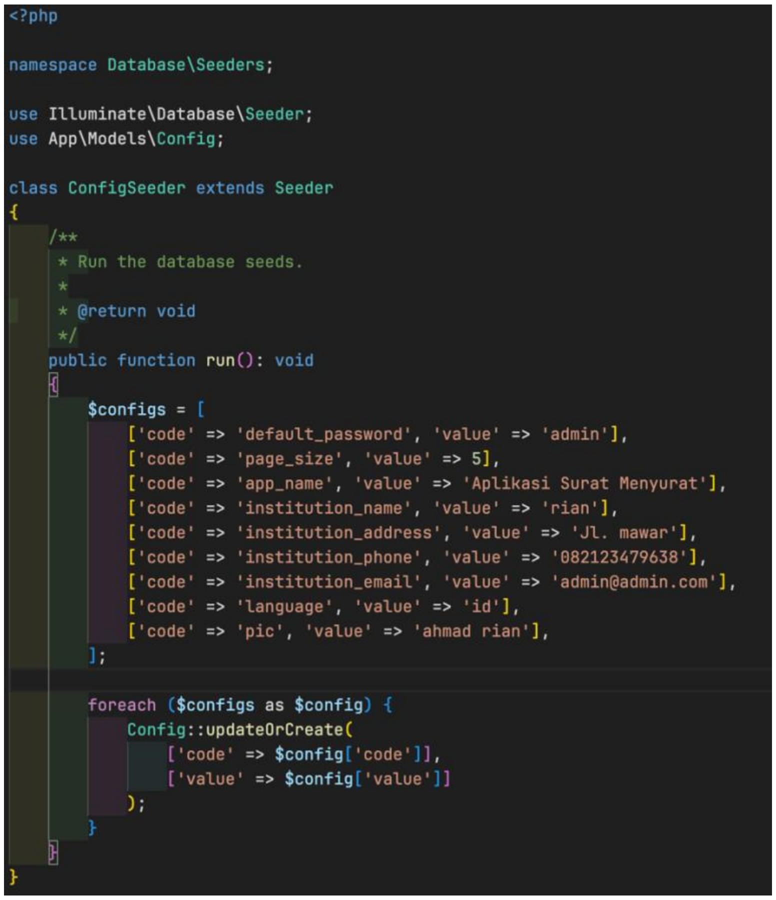
```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Letter;
use App\Models\User;

class LetterSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run(): void
    {
        $user = User::first(); // Mengambil user pertama dari tabel users

        if ($user) {
            Letter::updateOrCreate(
                ['reference_number' => '9863146119125'],
                [
                    'agenda_number' => '2968',
                    'from' => 'Ahmad Rian',
                    'to' => 'Purbalingga',
                    'letter_date' => '2024-06-05',
                    'received_date' => '2024-06-27',
                    'description' => 'test',
                    'note' => 'test',
                    'type' => 'incoming',
                    'classification_code' => 'ADM',
                    'user_id' => $user->id,
                ]
            );
        } else {
            echo "No users found in the database. Please seed the users table first.";
        }
    }
}
```



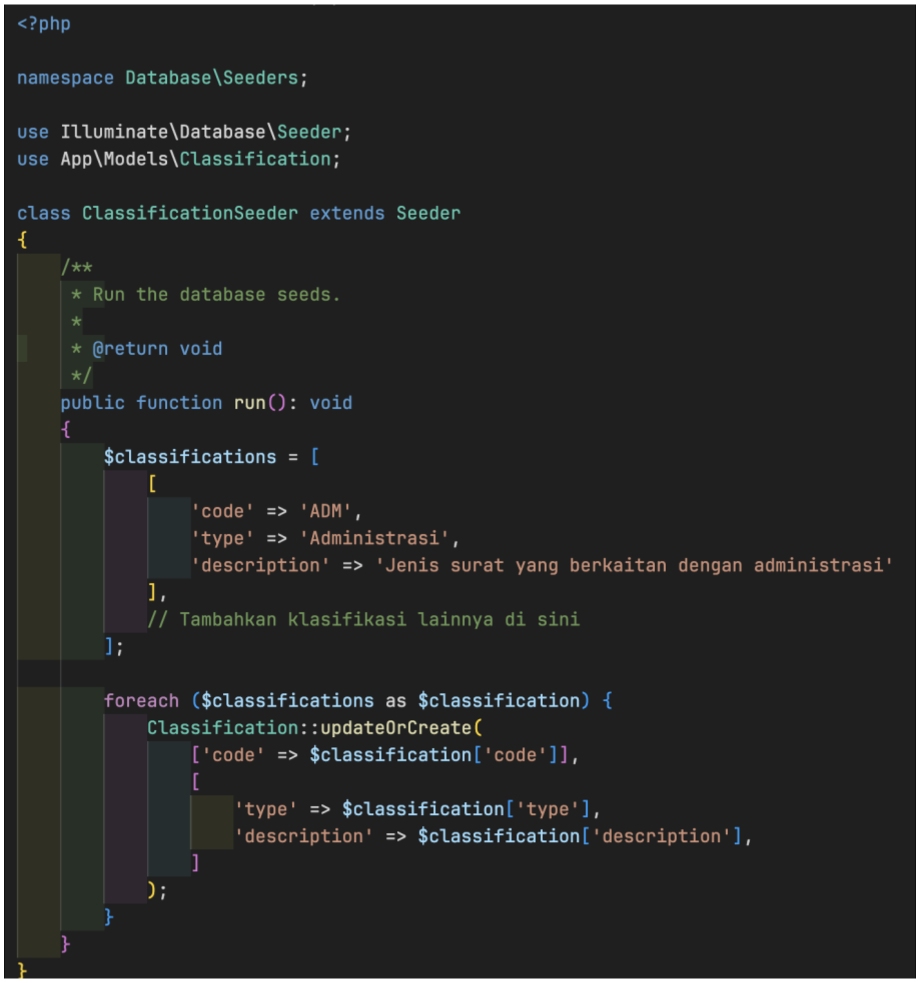
```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Disposition;
use App\Models\Letter;
use App\Models\User;

class DispositionSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run(): void
    {
        $letter = Letter::first(); // Mengambil surat pertama dari tabel letters
        $user = User::first(); // Mengambil user pertama dari tabel users

        if ($letter && $user) {
            Disposition::create([
                'to' => 'Syaiful',
                'due_date' => '2024-05-06',
                'content' => 'test',
                'note' => 'test',
                'letter_status' => 3,
                'letter_id' => $letter->id,
                'user_id' => $user->id,
            ]);
        } else {
            echo "No letters or users found in the database. Please seed the letters and users tables first.";
        }
    }
}
```



```php
<?php

namespace Database\Seeders;

// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run(): void
    {
        $this->call([
            UserSeeder::class,
            ConfigSeeder::class,
            LetterStatusSeeder::class,
            ClassificationSeeder::class,
            LetterSeeder::class,
            DispositionSeeder::class,
        ]);
    }
}
```

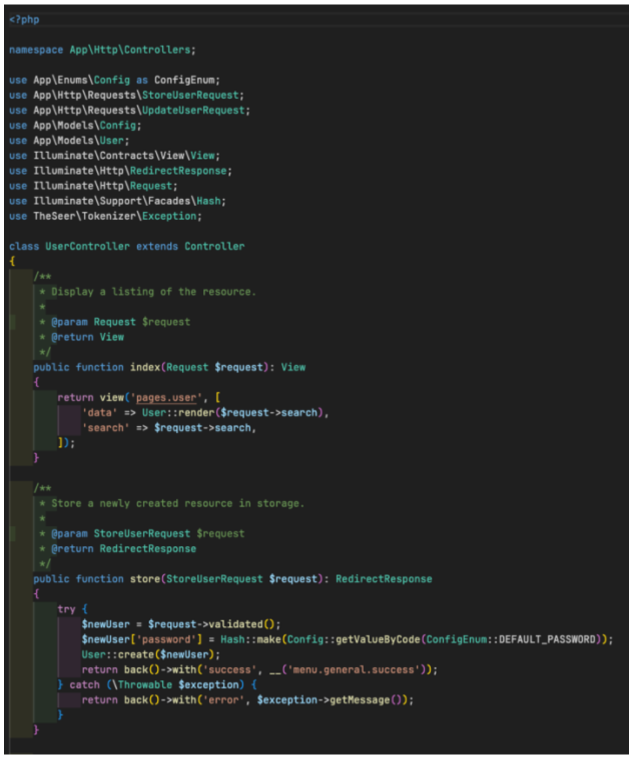


```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Config;

class ConfigSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run(): void
    {
        $configs = [
            ['code' => 'default_password', 'value' => 'admin'],
            ['code' => 'page_size', 'value' => 5],
            ['code' => 'app_name', 'value' => 'Aplikasi Surat Menyurat'],
            ['code' => 'institution_name', 'value' => 'rian'],
            ['code' => 'institution_address', 'value' => 'Jl. mawar'],
            ['code' => 'institution_phone', 'value' => '082123479638'],
            ['code' => 'institution_email', 'value' => 'admin@admin.com'],
            ['code' => 'language', 'value' => 'id'],
            ['code' => 'pic', 'value' => 'ahmad rian'],
        ];

        foreach ($configs as $config) {
            Config::updateOrCreate(
                ['code' => $config['code']],
                ['value' => $config['value']]
            );
        }
    }
}
```

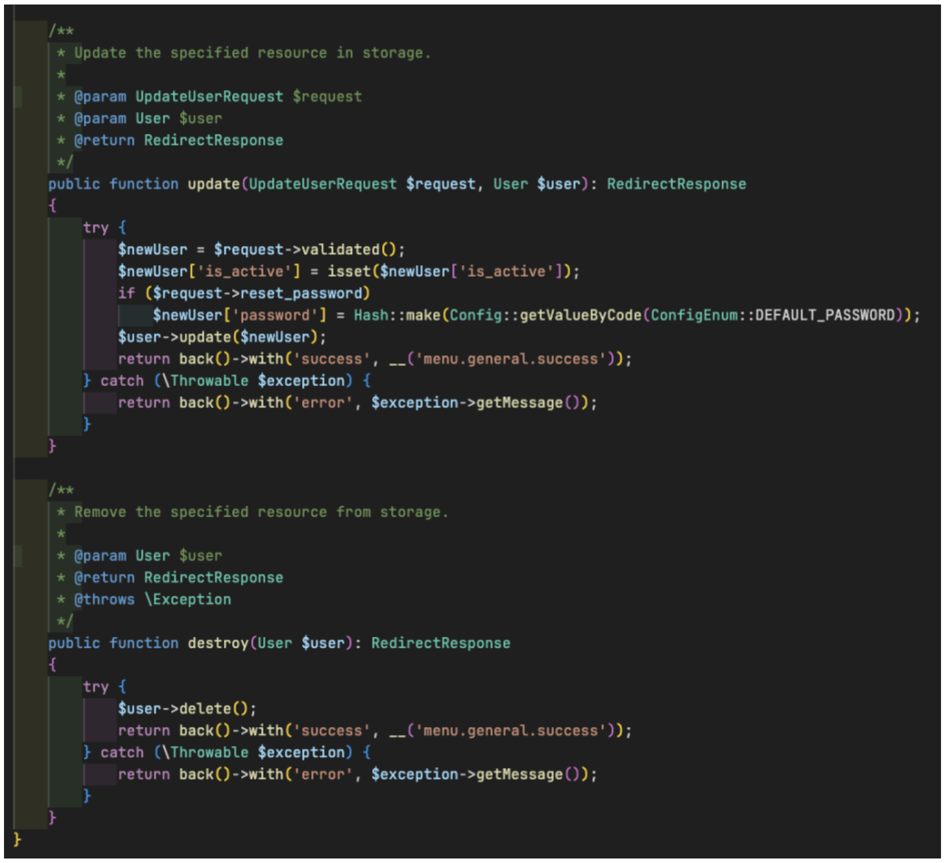


```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Classification;

class ClassificationSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run(): void
    {
        $classifications = [
            [
                'code' => 'ADM',
                'type' => 'Administrasi',
                'description' => 'Jenis surat yang berkaitan dengan administrasi'
            ],
            // Tambahkan klasifikasi lainnya di sini
        ];

        foreach ($classifications as $classification) {
            Classification::updateOrCreate(
                ['code' => $classification['code']],
                [
                    'type' => $classification['type'],
                    'description' => $classification['description'],
                ]
            );
        }
    }
}
```

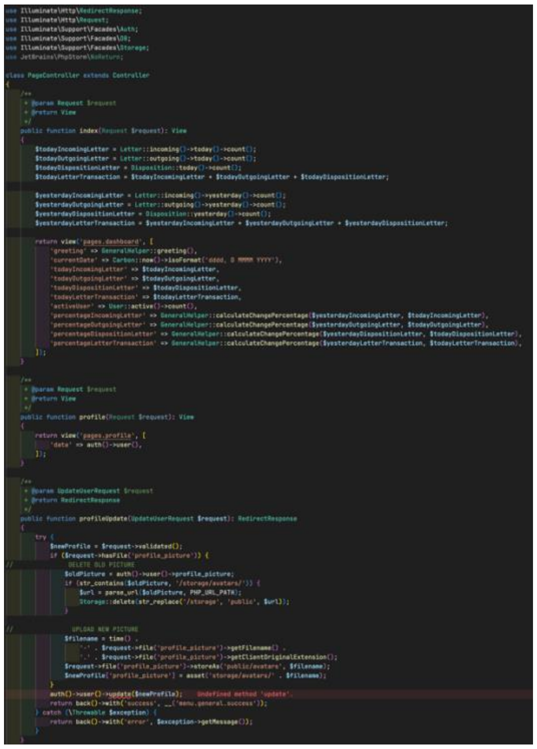
6. File Controller :



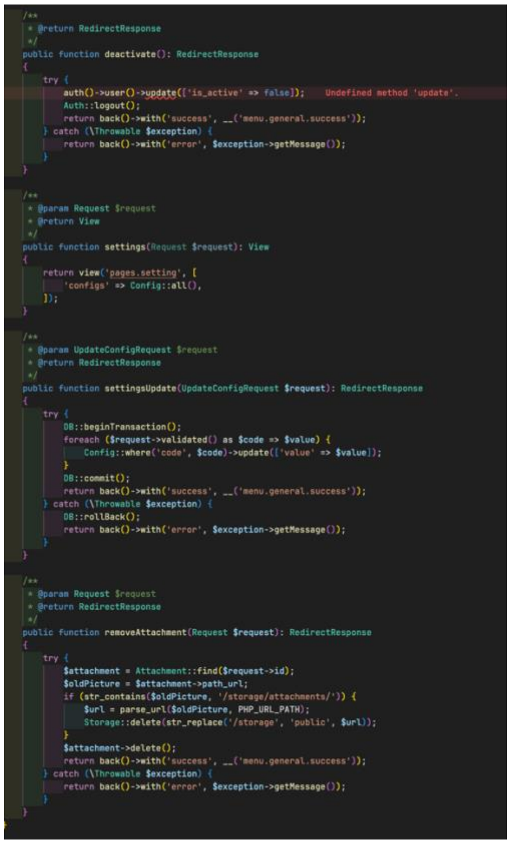
```php
<?php

namespace App\Http\Controllers;

use App\Enums\Config as ConfigEnum;
use App\Http\Requests\StoreUserRequest;
use App\Http\Requests\UpdateUserRequest;
use App\Models\Config;
use App\Models\User;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use TheSeer\Tokenizer\Exception;

class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @param Request $request
     * @return View
     */
    public function index(Request $request): View
    {
        return view('pages.user', [
            'data' => User::render($request->search),
            'search' => $request->search,
        ]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param StoreUserRequest $request
     * @return RedirectResponse
     */
    public function store(StoreUserRequest $request): RedirectResponse
    {
        try {
            $newUser = $request->validated();
            $newUser['password'] = Hash::make(Config::getValueByCode(ConfigEnum::DEFAULT_PASSWORD));
            User::create($newUser);
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
```



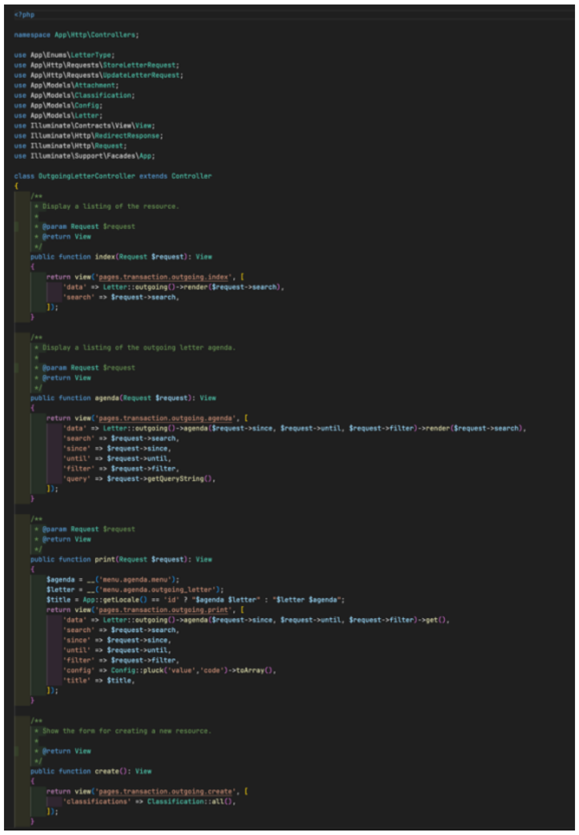
```php
    /**
     * Update the specified resource in storage.
     *
     * @param UpdateUserRequest $request
     * @param User $user
     * @return RedirectResponse
     */
    public function update(UpdateUserRequest $request, User $user): RedirectResponse
    {
        try {
            $newUser = $request->validated();
            $newUser['is_active'] = isset($newUser['is_active']);
            if ($request->reset_password)
                $newUser['password'] = Hash::make(Config::getValueByCode(ConfigEnum::DEFAULT_PASSWORD));
            $user->update($newUser);
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param User $user
     * @return RedirectResponse
     * @throws \Exception
     */
    public function destroy(User $user): RedirectResponse
    {
        try {
            $user->delete();
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
}
```



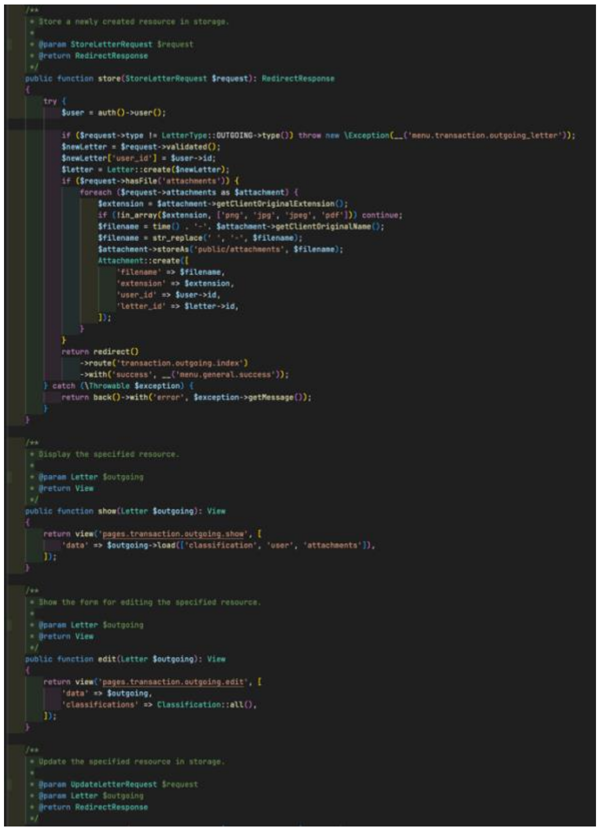
```php
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Storage;
use JetBrains\PhpStorm\NoReturn;

class PageController extends Controller
{
    /**
     * @param Request $request
     * @return View
     */
    public function index(Request $request): View
    {
        $todayIncomingLetter = Letter::incoming()->today()->count();
        $todayOutgoingLetter = Letter::outgoing()->today()->count();
        $todayDispositionLetter = Disposition::today()->count();
        $todayLetterTransaction = $todayIncomingLetter + $todayOutgoingLetter + $todayDispositionLetter;

        $yesterdayIncomingLetter = Letter::incoming()->yesterday()->count();
        $yesterdayOutgoingLetter = Letter::outgoing()->yesterday()->count();
        $yesterdayDispositionLetter = Disposition::yesterday()->count();
        $yesterdayLetterTransaction = $yesterdayIncomingLetter + $yesterdayOutgoingLetter + $yesterdayDispositionLetter;

        return view('pages.dashboard', [
            'greeting' => GeneralHelper::greeting(),
            'currentDate' => Carbon::now()->isoFormat('dddd, D MMMM YYYY'),
            'todayIncomingLetter' => $todayIncomingLetter,
            'todayOutgoingLetter' => $todayOutgoingLetter,
            'todayDispositionLetter' => $todayDispositionLetter,
            'todayLetterTransaction' => $todayLetterTransaction,
            'activeUser' => User::active()->count(),
            'percentageIncomingLetter' => GeneralHelper::calculateChangePercentage($yesterdayIncomingLetter, $todayIncomingLetter),
            'percentageOutgoingLetter' => GeneralHelper::calculateChangePercentage($yesterdayOutgoingLetter, $todayOutgoingLetter),
            'percentageDispositionLetter' => GeneralHelper::calculateChangePercentage($yesterdayDispositionLetter, $todayDispositionLetter),
            'percentageLetterTransaction' => GeneralHelper::calculateChangePercentage($yesterdayLetterTransaction, $todayLetterTransaction),
        ]);
    }

    /**
     * @param Request $request
     * @return View
     */
    public function profile(Request $request): View
    {
        return view('pages.profile', [
            'data' => auth()->user(),
        ]);
    }

    /**
     * @param UpdateUserRequest $request
     * @return RedirectResponse
     */
    public function profileUpdate(UpdateUserRequest $request): RedirectResponse
    {
        try {
            $newProfile = $request->validated();
            if ($request->hasFile('profile_picture')) {
//               DELETE OLD PICTURE
                $oldPicture = auth()->user()->profile_picture;
                if (str_contains($oldPicture, '/storage/avatars/')) {
                    $url = parse_url($oldPicture, PHP_URL_PATH);
                    Storage::delete(str_replace('/storage', 'public', $url));
                }

//               UPLOAD NEW PICTURE
                $filename = time() .
                    '-' . $request->file('profile_picture')->getFilename() .
                    '.' . $request->file('profile_picture')->getClientOriginalExtension();
                $request->file('profile_picture')->storeAs('public/avatars', $filename);
                $newProfile['profile_picture'] = asset('storage/avatars/' . $filename);
            }
            auth()->user()->update($newProfile);    Undefined method 'update'.
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
```



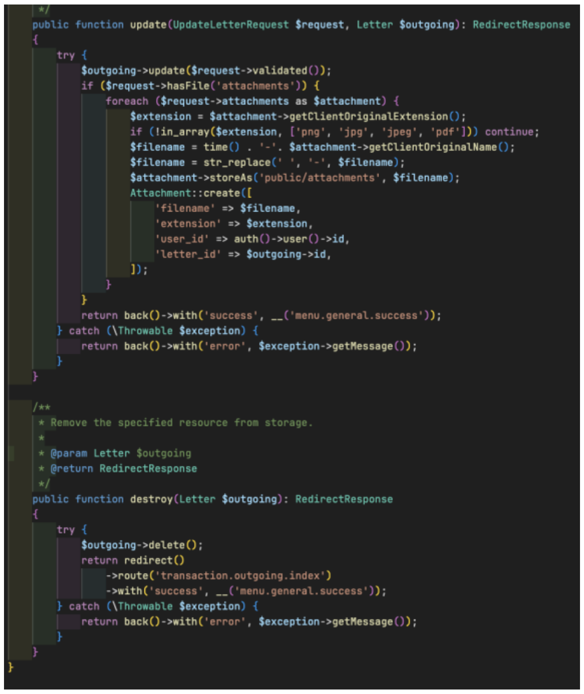
```php
    /**
     * @return RedirectResponse
     */
    public function deactivate(): RedirectResponse
    {
        try {
            auth()->user()->update(['is_active' => false]);    Undefined method 'update'.
            Auth::logout();
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * @param Request $request
     * @return View
     */
    public function settings(Request $request): View
    {
        return view('pages.setting', [
            'configs' => Config::all(),
        ]);
    }

    /**
     * @param UpdateConfigRequest $request
     * @return RedirectResponse
     */
    public function settingsUpdate(UpdateConfigRequest $request): RedirectResponse
    {
        try {
            DB::beginTransaction();
            foreach ($request->validated() as $code => $value) {
                Config::where('code', $code)->update(['value' => $value]);
            }
            DB::commit();
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            DB::rollBack();
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * @param Request $request
     * @return RedirectResponse
     */
    public function removeAttachment(Request $request): RedirectResponse
    {
        try {
            $attachment = Attachment::find($request->id);
            $oldPicture = $attachment->path_url;
            if (str_contains($oldPicture, '/storage/attachments/')) {
                $url = parse_url($oldPicture, PHP_URL_PATH);
                Storage::delete(str_replace('/storage', 'public', $url));
            }
            $attachment->delete();
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
}
```



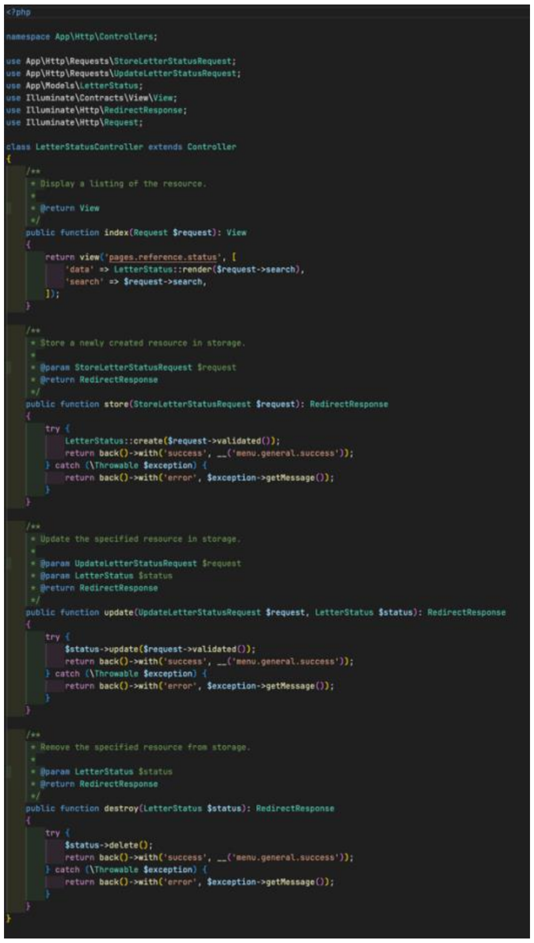
```php
<?php

namespace App\Http\Controllers;

use App\Enums\LetterType;
use App\Http\Requests\StoreLetterRequest;
use App\Http\Requests\UpdateLetterRequest;
use App\Models\Attachment;
use App\Models\Classification;
use App\Models\Config;
use App\Models\Letter;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\App;

class OutgoingLetterController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @param Request $request
     * @return View
     */
    public function index(Request $request): View
    {
        return view('pages.transaction.outgoing.index', [
            'data' => Letter::outgoing()->render($request->search),
            'search' => $request->search,
        ]);
    }

    /**
     * Display a listing of the outgoing letter agenda.
     *
     * @param Request $request
     * @return View
     */
    public function agenda(Request $request): View
    {
        return view('pages.transaction.outgoing.agenda', [
            'data' => Letter::outgoing()->agenda($request->since, $request->until, $request->filter)->render($request->search),
            'search' => $request->search,
            'since' => $request->since,
            'until' => $request->until,
            'filter' => $request->filter,
            'query' => $request->getQueryString(),
        ]);
    }

    /**
     * @param Request $request
     * @return View
     */
    public function print(Request $request): View
    {
        $agenda = __('menu.agenda.menu');
        $letter = __('menu.agenda.outgoing_letter');
        $title = App::getLocale() == 'id' ? "$agenda $letter" : "$letter $agenda";
        return view('pages.transaction.outgoing.print', [
            'data' => Letter::outgoing()->agenda($request->since, $request->until, $request->filter)->get(),
            'search' => $request->search,
            'since' => $request->since,
            'until' => $request->until,
            'filter' => $request->filter,
            'config' => Config::pluck('value','code')->toArray(),
            'title' => $title,
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return View
     */
    public function create(): View
    {
        return view('pages.transaction.outgoing.create', [
            'classifications' => Classification::all(),
        ]);
    }
```



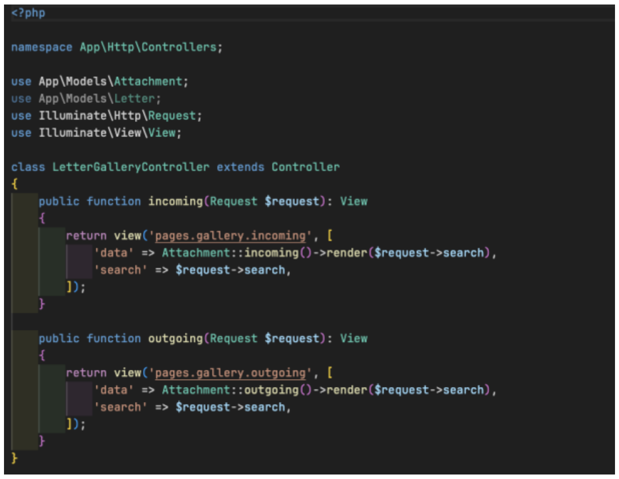
```php
    /**
     * Store a newly created resource in storage.
     *
     * @param StoreLetterRequest $request
     * @return RedirectResponse
     */
    public function store(StoreLetterRequest $request): RedirectResponse
    {
        try {
            $user = auth()->user();

            if ($request->type != LetterType::OUTGOING->type()) throw new \Exception(__('menu.transaction.outgoing_letter'));
            $newLetter = $request->validated();
            $newLetter['user_id'] = $user->id;
            $letter = Letter::create($newLetter);
            if ($request->hasFile('attachments')) {
                foreach ($request->attachments as $attachment) {
                    $extension = $attachment->getClientOriginalExtension();
                    if (!in_array($extension, ['png', 'jpg', 'jpeg', 'pdf'])) continue;
                    $filename = time() . '-'. $attachment->getClientOriginalName();
                    $filename = str_replace(' ', '-', $filename);
                    $attachment->storeAs('public/attachments', $filename);
                    Attachment::create([
                        'filename' => $filename,
                        'extension' => $extension,
                        'user_id' => $user->id,
                        'letter_id' => $letter->id,
                    ]);
                }
            }
            return redirect()
                ->route('transaction.outgoing.index')
                ->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Display the specified resource.
     *
     * @param Letter $outgoing
     * @return View
     */
    public function show(Letter $outgoing): View
    {
        return view('pages.transaction.outgoing.show', [
            'data' => $outgoing->load(['classification', 'user', 'attachments']),
        ]);
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param Letter $outgoing
     * @return View
     */
    public function edit(Letter $outgoing): View
    {
        return view('pages.transaction.outgoing.edit', [
            'data' => $outgoing,
            'classifications' => Classification::all(),
        ]);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param UpdateLetterRequest $request
     * @param Letter $outgoing
     * @return RedirectResponse
     */
```



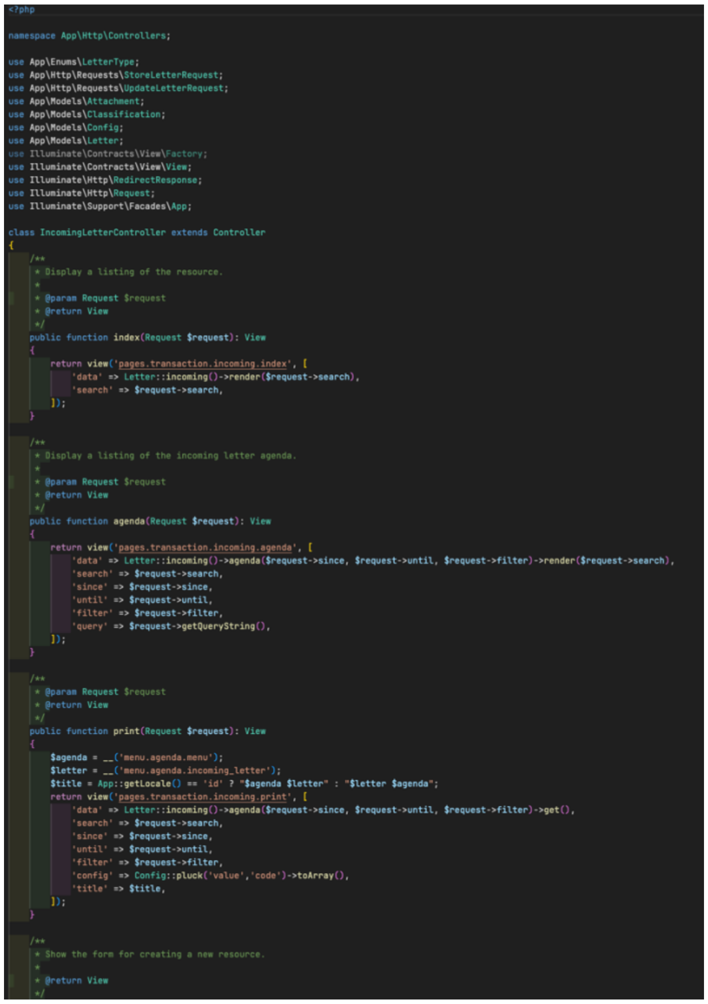
```php
     */
    public function update(UpdateLetterRequest $request, Letter $outgoing): RedirectResponse
    {
        try {
            $outgoing->update($request->validated());
            if ($request->hasFile('attachments')) {
                foreach ($request->attachments as $attachment) {
                    $extension = $attachment->getClientOriginalExtension();
                    if (!in_array($extension, ['png', 'jpg', 'jpeg', 'pdf'])) continue;
                    $filename = time() . '-'. $attachment->getClientOriginalName();
                    $filename = str_replace(' ', '-', $filename);
                    $attachment->storeAs('public/attachments', $filename);
                    Attachment::create([
                        'filename' => $filename,
                        'extension' => $extension,
                        'user_id' => auth()->user()->id,
                        'letter_id' => $outgoing->id,
                    ]);
                }
            }
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param Letter $outgoing
     * @return RedirectResponse
     */
    public function destroy(Letter $outgoing): RedirectResponse
    {
        try {
            $outgoing->delete();
            return redirect()
                ->route('transaction.outgoing.index')
                ->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
}
```



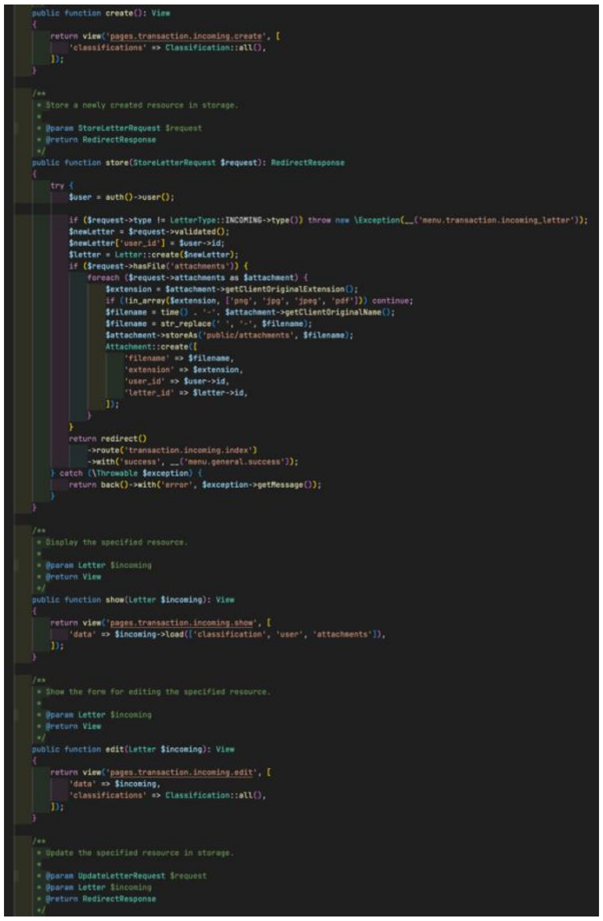
```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\StoreLetterStatusRequest;
use App\Http\Requests\UpdateLetterStatusRequest;
use App\Models\LetterStatus;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

class LetterStatusController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return View
     */
    public function index(Request $request): View
    {
        return view('pages.reference.status', [
            'data' => LetterStatus::render($request->search),
            'search' => $request->search,
        ]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param StoreLetterStatusRequest $request
     * @return RedirectResponse
     */
    public function store(StoreLetterStatusRequest $request): RedirectResponse
    {
        try {
            LetterStatus::create($request->validated());
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Update the specified resource in storage.
     *
     * @param UpdateLetterStatusRequest $request
     * @param LetterStatus $status
     * @return RedirectResponse
     */
    public function update(UpdateLetterStatusRequest $request, LetterStatus $status): RedirectResponse
    {
        try {
            $status->update($request->validated());
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param LetterStatus $status
     * @return RedirectResponse
     */
    public function destroy(LetterStatus $status): RedirectResponse
    {
        try {
            $status->delete();
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
}
```



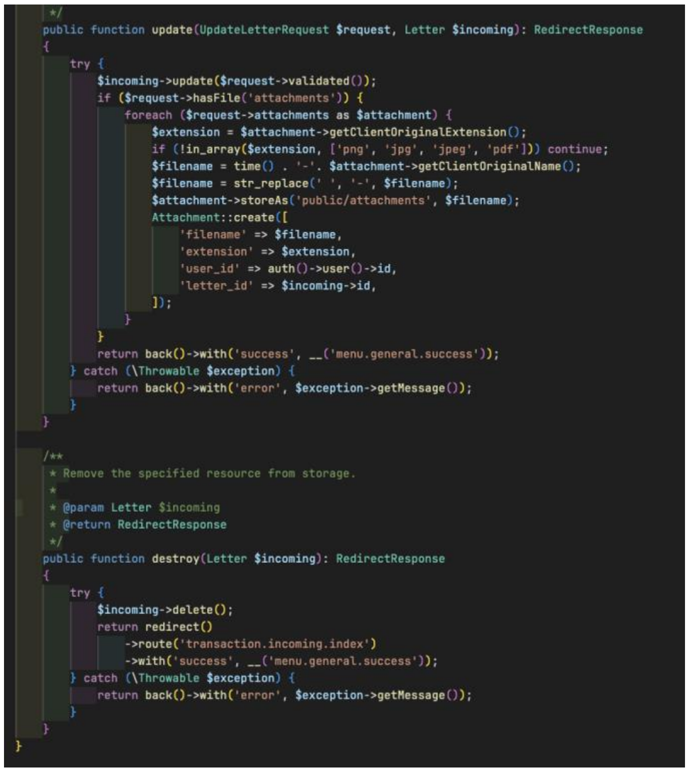
```php
<?php

namespace App\Http\Controllers;

use App\Models\Attachment;
use App\Models\Letter;
use Illuminate\Http\Request;
use Illuminate\View\View;

class LetterGalleryController extends Controller
{
    public function incoming(Request $request): View
    {
        return view('pages.gallery.incoming', [
            'data' => Attachment::incoming()->render($request->search),
            'search' => $request->search,
        ]);
    }

    public function outgoing(Request $request): View
    {
        return view('pages.gallery.outgoing', [
            'data' => Attachment::outgoing()->render($request->search),
            'search' => $request->search,
        ]);
    }
}
```



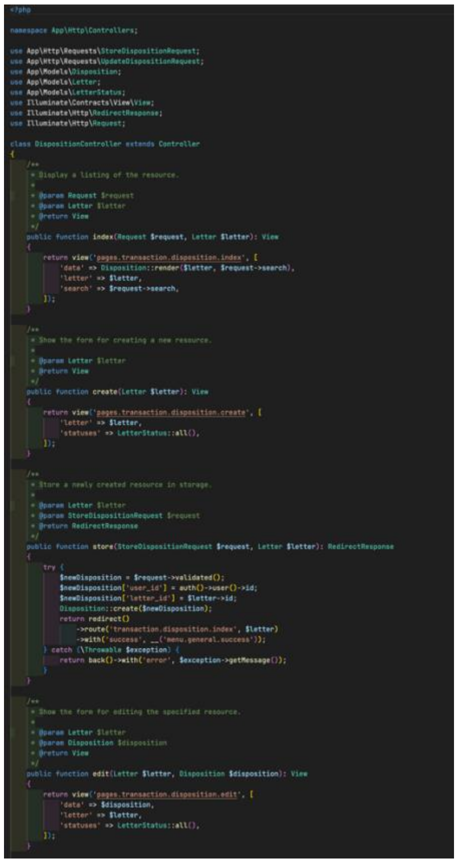
```php
<?php

namespace App\Http\Controllers;

use App\Enums\LetterType;
use App\Http\Requests\StoreLetterRequest;
use App\Http\Requests\UpdateLetterRequest;
use App\Models\Attachment;
use App\Models\Classification;
use App\Models\Config;
use App\Models\Letter;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\App;

class IncomingLetterController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @param Request $request
     * @return View
     */
    public function index(Request $request): View
    {
        return view('pages.transaction.incoming.index', [
            'data' => Letter::incoming()->render($request->search),
            'search' => $request->search,
        ]);
    }

    /**
     * Display a listing of the incoming letter agenda.
     *
     * @param Request $request
     * @return View
     */
    public function agenda(Request $request): View
    {
        return view('pages.transaction.incoming.agenda', [
            'data' => Letter::incoming()->agenda($request->since, $request->until, $request->filter)->render($request->search),
            'search' => $request->search,
            'since' => $request->since,
            'until' => $request->until,
            'filter' => $request->filter,
            'query' => $request->getQueryString(),
        ]);
    }

    /**
     * @param Request $request
     * @return View
     */
    public function print(Request $request): View
    {
        $agenda = __('menu.agenda.menu');
        $letter = __('menu.agenda.incoming_letter');
        $title = App::getLocale() == 'id' ? "$agenda $letter" : "$letter $agenda";
        return view('pages.transaction.incoming.print', [
            'data' => Letter::incoming()->agenda($request->since, $request->until, $request->filter)->get(),
            'search' => $request->search,
            'since' => $request->since,
            'until' => $request->until,
            'filter' => $request->filter,
            'config' => Config::pluck('value','code')->toArray(),
            'title' => $title,
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return View
     */
```



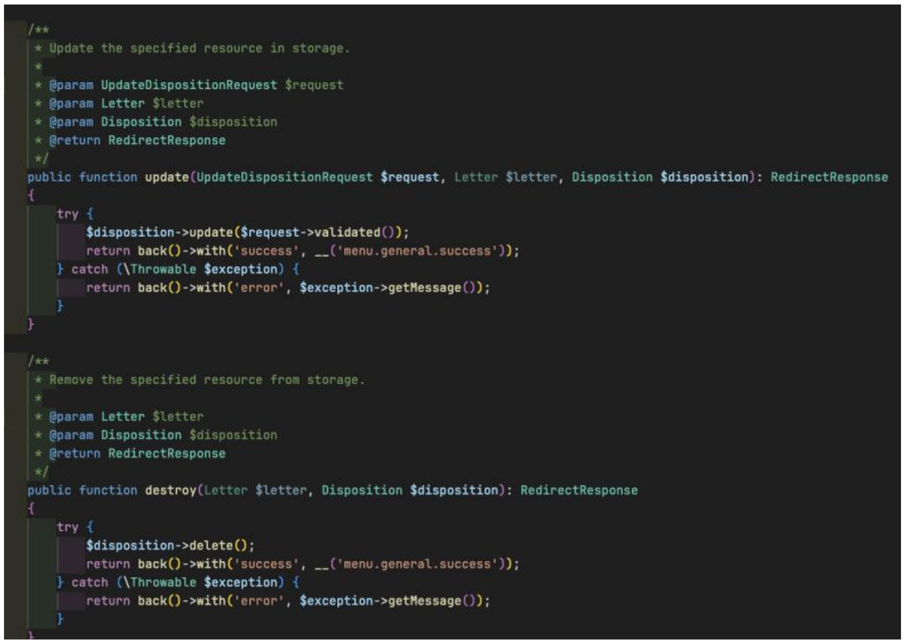
```php
    public function create(): View
    {
        return view('pages.transaction.incoming.create', [
            'classifications' => Classification::all(),
        ]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param StoreLetterRequest $request
     * @return RedirectResponse
     */
    public function store(StoreLetterRequest $request): RedirectResponse
    {
        try {
            $user = auth()->user();

            if ($request->type != LetterType::INCOMING->type()) throw new \Exception(__('menu.transaction.incoming_letter'));
            $newLetter = $request->validated();
            $newLetter['user_id'] = $user->id;
            $letter = Letter::create($newLetter);
            if ($request->hasFile('attachments')) {
                foreach ($request->attachments as $attachment) {
                    $extension = $attachment->getClientOriginalExtension();
                    if (!in_array($extension, ['png', 'jpg', 'jpeg', 'pdf'])) continue;
                    $filename = time() . '-'. $attachment->getClientOriginalName();
                    $filename = str_replace(' ', '-', $filename);
                    $attachment->storeAs('public/attachments', $filename);
                    Attachment::create([
                        'filename' => $filename,
                        'extension' => $extension,
                        'user_id' => $user->id,
                        'letter_id' => $letter->id,
                    ]);
                }
            }
            return redirect()
                ->route('transaction.incoming.index')
                ->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Display the specified resource.
     *
     * @param Letter $incoming
     * @return View
     */
    public function show(Letter $incoming): View
    {
        return view('pages.transaction.incoming.show', [
            'data' => $incoming->load(['classification', 'user', 'attachments']),
        ]);
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param Letter $incoming
     * @return View
     */
    public function edit(Letter $incoming): View
    {
        return view('pages.transaction.incoming.edit', [
            'data' => $incoming,
            'classifications' => Classification::all(),
        ]);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param UpdateLetterRequest $request
     * @param Letter $incoming
     * @return RedirectResponse
     */
```



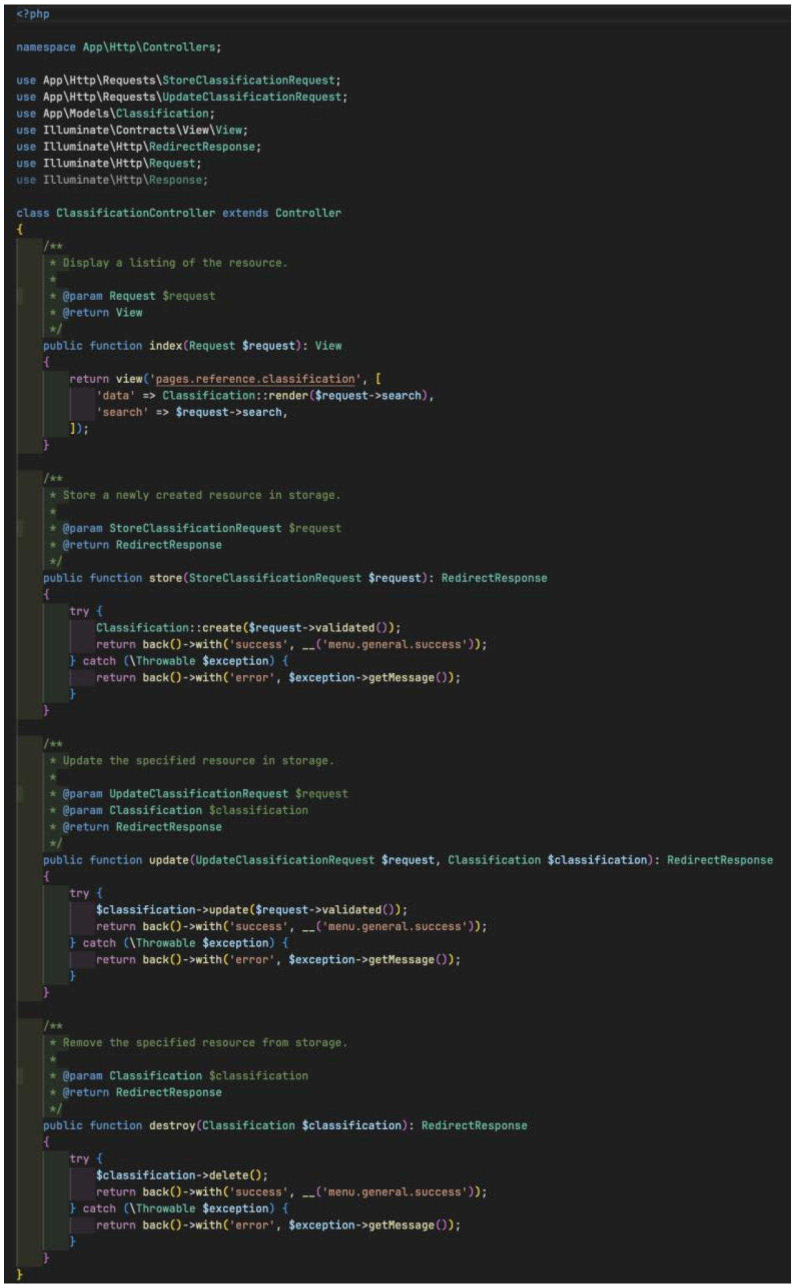
```php
     */
    public function update(UpdateLetterRequest $request, Letter $incoming): RedirectResponse
    {
        try {
            $incoming->update($request->validated());
            if ($request->hasFile('attachments')) {
                foreach ($request->attachments as $attachment) {
                    $extension = $attachment->getClientOriginalExtension();
                    if (!in_array($extension, ['png', 'jpg', 'jpeg', 'pdf'])) continue;
                    $filename = time() . '-'. $attachment->getClientOriginalName();
                    $filename = str_replace(' ', '-', $filename);
                    $attachment->storeAs('public/attachments', $filename);
                    Attachment::create([
                        'filename' => $filename,
                        'extension' => $extension,
                        'user_id' => auth()->user()->id,
                        'letter_id' => $incoming->id,
                    ]);
                }
            }
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param Letter $incoming
     * @return RedirectResponse
     */
    public function destroy(Letter $incoming): RedirectResponse
    {
        try {
            $incoming->delete();
            return redirect()
                ->route('transaction.incoming.index')
                ->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
}
```



```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\StoreDispositionRequest;
use App\Http\Requests\UpdateDispositionRequest;
use App\Models\Disposition;
use App\Models\Letter;
use App\Models\LetterStatus;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

class DispositionController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @param Request $request
     * @param Letter $letter
     * @return View
     */
    public function index(Request $request, Letter $letter): View
    {
        return view('pages.transaction.disposition.index', [
            'data' => Disposition::render($letter, $request->search),
            'letter' => $letter,
            'search' => $request->search,
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @param Letter $letter
     * @return View
     */
    public function create(Letter $letter): View
    {
        return view('pages.transaction.disposition.create', [
            'letter' => $letter,
            'statuses' => LetterStatus::all(),
        ]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param Letter $letter
     * @param StoreDispositionRequest $request
     * @return RedirectResponse
     */
    public function store(StoreDispositionRequest $request, Letter $letter): RedirectResponse
    {
        try {
            $newDisposition = $request->validated();
            $newDisposition['user_id'] = auth()->user()->id;
            $newDisposition['letter_id'] = $letter->id;
            Disposition::create($newDisposition);
            return redirect()
                ->route('transaction.disposition.index', $letter)
                ->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param Letter $letter
     * @param Disposition $disposition
     * @return View
     */
    public function edit(Letter $letter, Disposition $disposition): View
    {
        return view('pages.transaction.disposition.edit', [
            'data' => $disposition,
            'letter' => $letter,
            'statuses' => LetterStatus::all(),
        ]);
    }
```



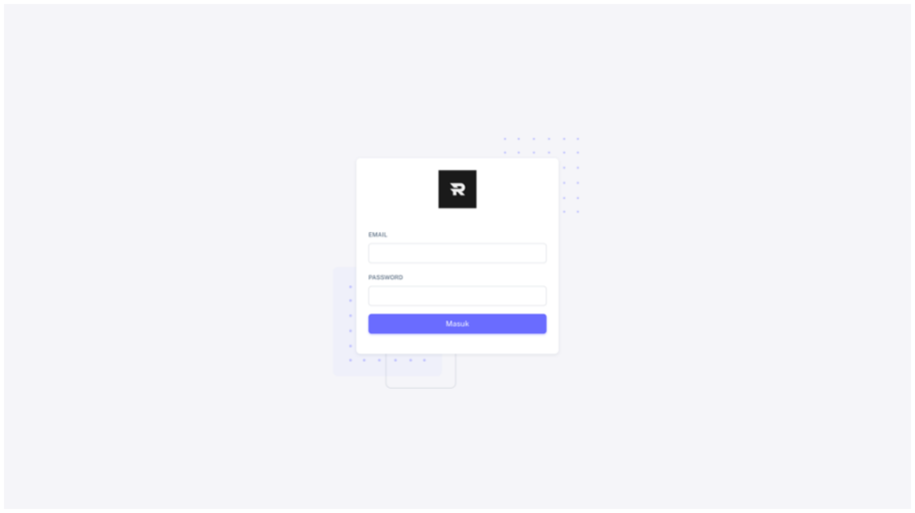
```php
    /**
     * Update the specified resource in storage.
     *
     * @param UpdateDispositionRequest $request
     * @param Letter $letter
     * @param Disposition $disposition
     * @return RedirectResponse
     */
    public function update(UpdateDispositionRequest $request, Letter $letter, Disposition $disposition): RedirectResponse
    {
        try {
            $disposition->update($request->validated());
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param Letter $letter
     * @param Disposition $disposition
     * @return RedirectResponse
     */
    public function destroy(Letter $letter, Disposition $disposition): RedirectResponse
    {
        try {
            $disposition->delete();
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
```



```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\StoreClassificationRequest;
use App\Http\Requests\UpdateClassificationRequest;
use App\Models\Classification;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;

class ClassificationController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @param Request $request
     * @return View
     */
    public function index(Request $request): View
    {
        return view('pages.reference.classification', [
            'data' => Classification::render($request->search),
            'search' => $request->search,
        ]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param StoreClassificationRequest $request
     * @return RedirectResponse
     */
    public function store(StoreClassificationRequest $request): RedirectResponse
    {
        try {
            Classification::create($request->validated());
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Update the specified resource in storage.
     *
     * @param UpdateClassificationRequest $request
     * @param Classification $classification
     * @return RedirectResponse
     */
    public function update(UpdateClassificationRequest $request, Classification $classification): RedirectResponse
    {
        try {
            $classification->update($request->validated());
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param Classification $classification
     * @return RedirectResponse
     */
    public function destroy(Classification $classification): RedirectResponse
    {
        try {
            $classification->delete();
            return back()->with('success', __('menu.general.success'));
        } catch (\Throwable $exception) {
            return back()->with('error', $exception->getMessage());
        }
    }
}
```

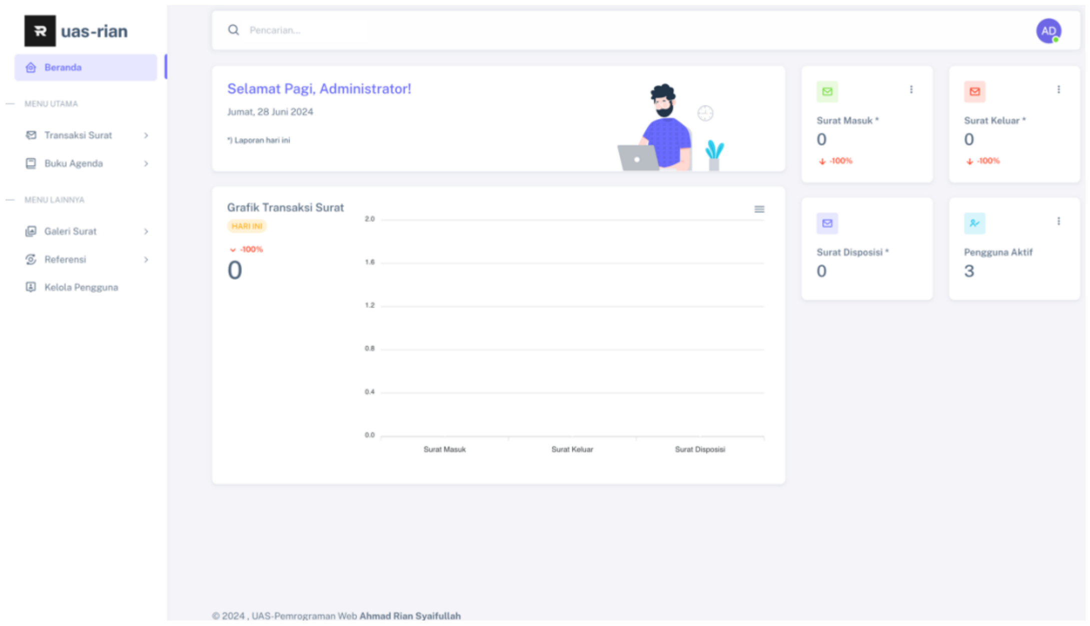
7. ScreenShot Aplikasi :

- Halaman Login

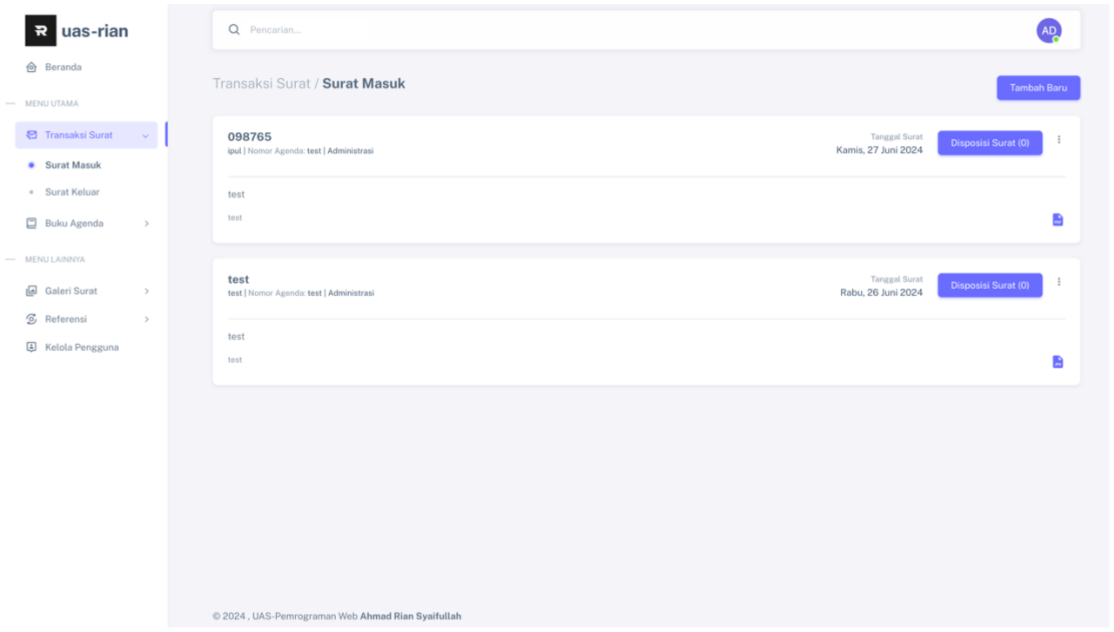


Email : admin@admin.com password : admin

- Halaman Dashboard



- Halaman Transaksi Surat Penerima (Surat Masuk)



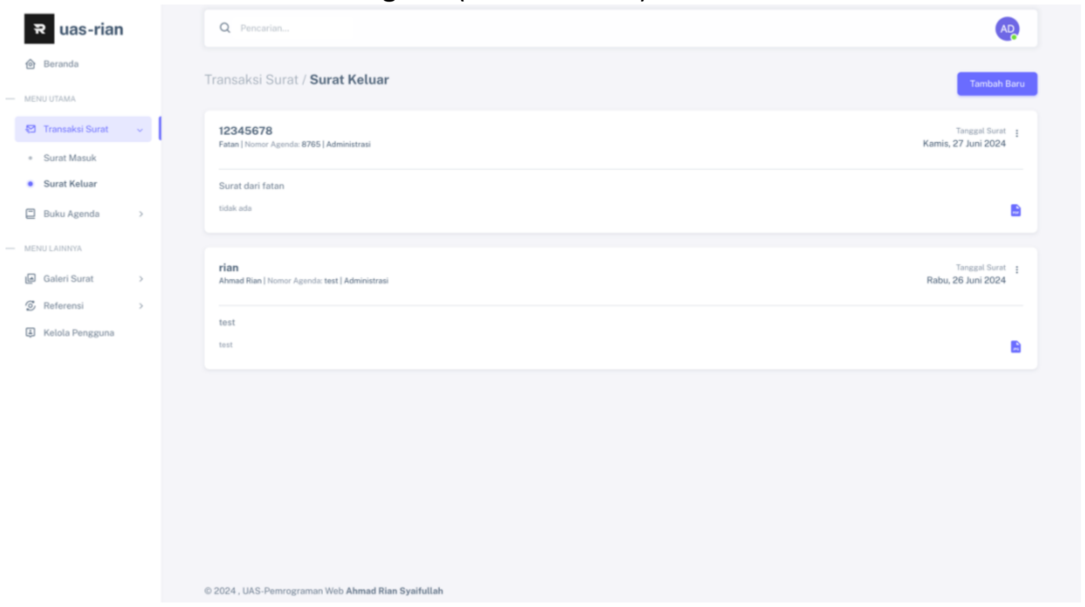
Transaksi Surat / **Surat Masuk**

098765
ipul | Nomor Agenda: test | Administrasi
Tanggal Surat
Kamis, 27 Juni 2024
test
test

test
test | Nomor Agenda: test | Administrasi
Tanggal Surat
Rabu, 26 Juni 2024
test
test

© 2024 , UAS-Pemrograman Web **Ahmad Rian Syaifullah**

- Halaman Transaksi Surat Pengirim (Surat Keluar)



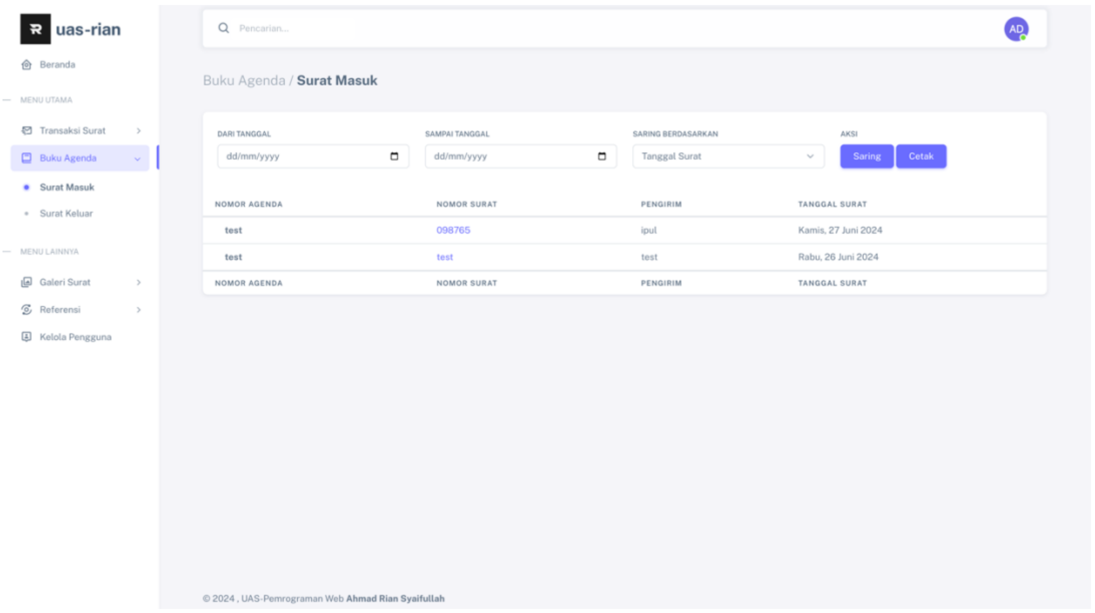
Transaksi Surat / **Surat Keluar**

12345678
Fatan | Nomor Agenda: 8765 | Administrasi
Tanggal Surat
Kamis, 27 Juni 2024
Surat dari fatan
tidak ada

rian
Ahmad Rian | Nomor Agenda: test | Administrasi
Tanggal Surat
Rabu, 26 Juni 2024
test
test

© 2024 , UAS-Pemrograman Web **Ahmad Rian Syaifullah**

- Halaman Arsip Surat



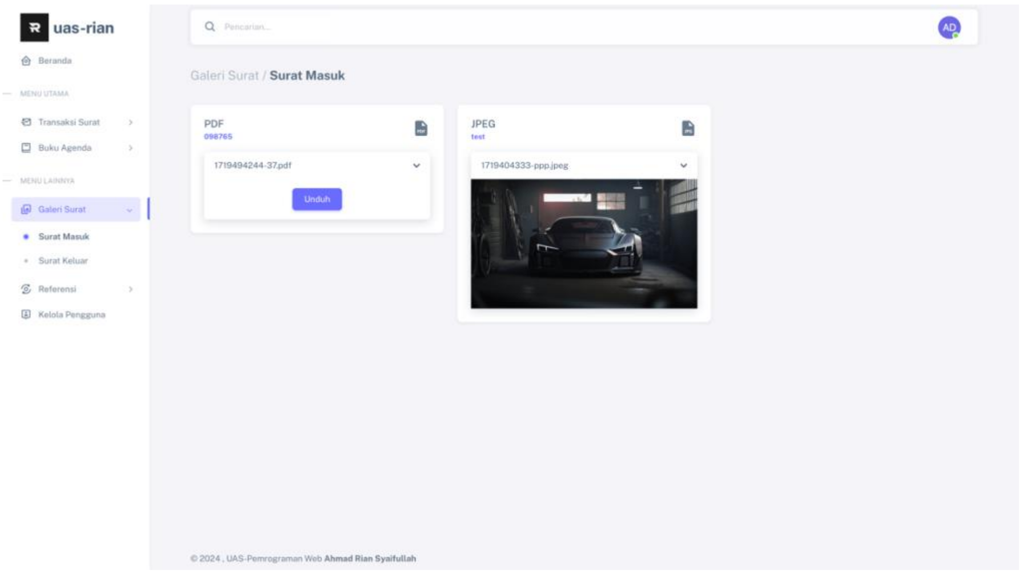
Buku Agenda / **Surat Masuk**

DARI TANGGAL	SAMPAI TANGGAL	SARING BERDASARKAN	AKSI
dd/mm/yyyy	dd/mm/yyyy	Tanggal Surat	

NOMOR AGENDA	NOMOR SURAT	PENGIRIM	TANGGAL SURAT
test	098765	ipul	Kamis, 27 Juni 2024
test	test	test	Rabu, 26 Juni 2024
NOMOR AGENDA	NOMOR SURAT	PENGIRIM	TANGGAL SURAT

© 2024 , UAS-Pemrograman Web **Ahmad Rian Syaifullah**

- Halaman Galeri Surat



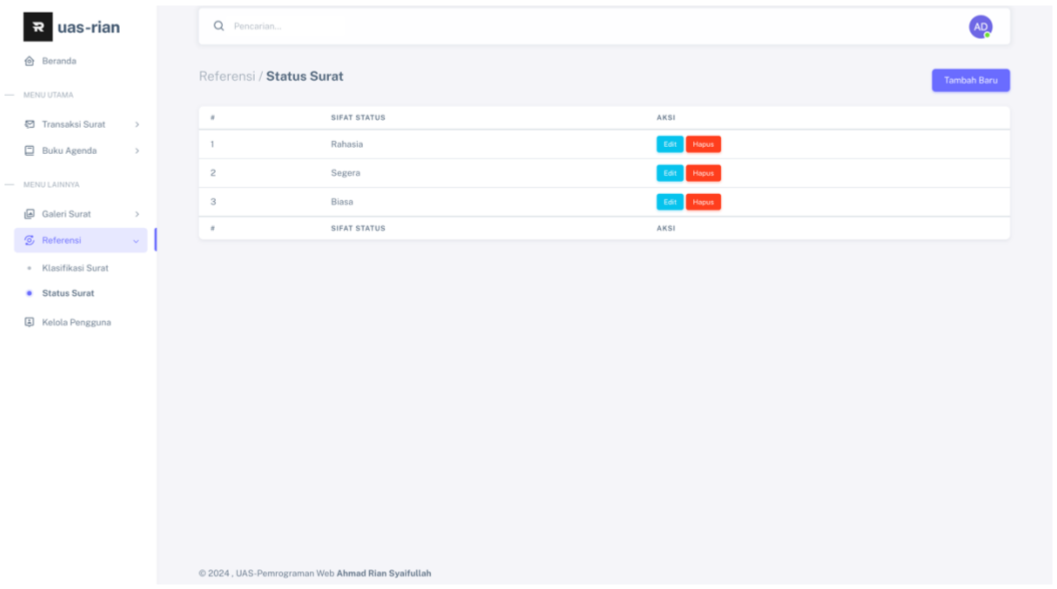
Galeri Surat / **Surat Masuk**

PDF
098765
1719494244-37.pdf

JPEG
test
1719404333-ppp.jpeg

© 2024 , UAS-Pemrograman Web **Ahmad Rian Syaifullah**

- Halaman Kategori Surat

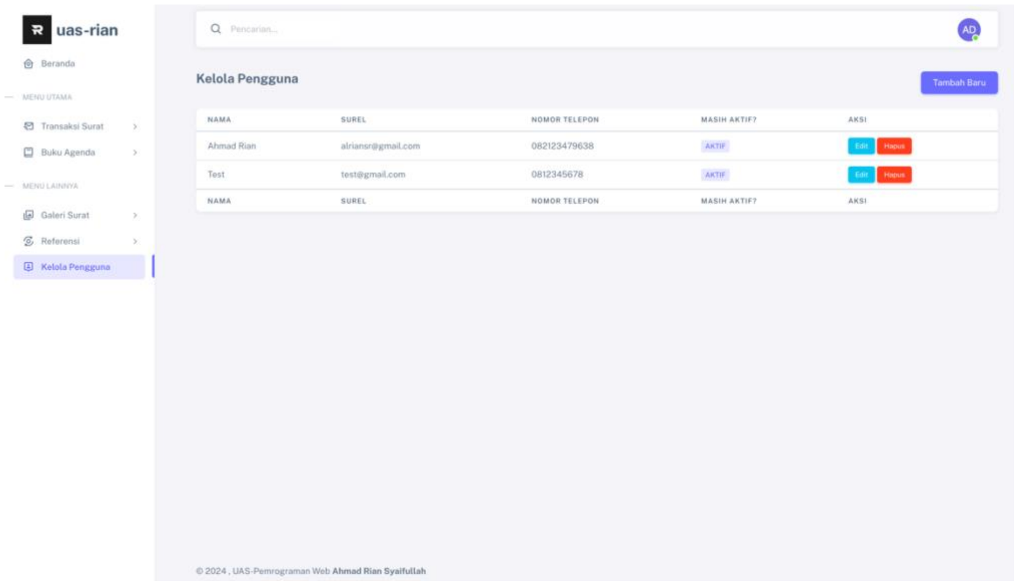


Referensi / Status Surat

#	SIFAT STATUS	AKSI
1	Rahasia	Edit Hapus
2	Segera	Edit Hapus
3	Biasa	Edit Hapus
#	SIFAT STATUS	AKSI

© 2024 , UAS-Pemrograman Web **Ahmad Rian Syaifullah**

- Halaman Kelola User

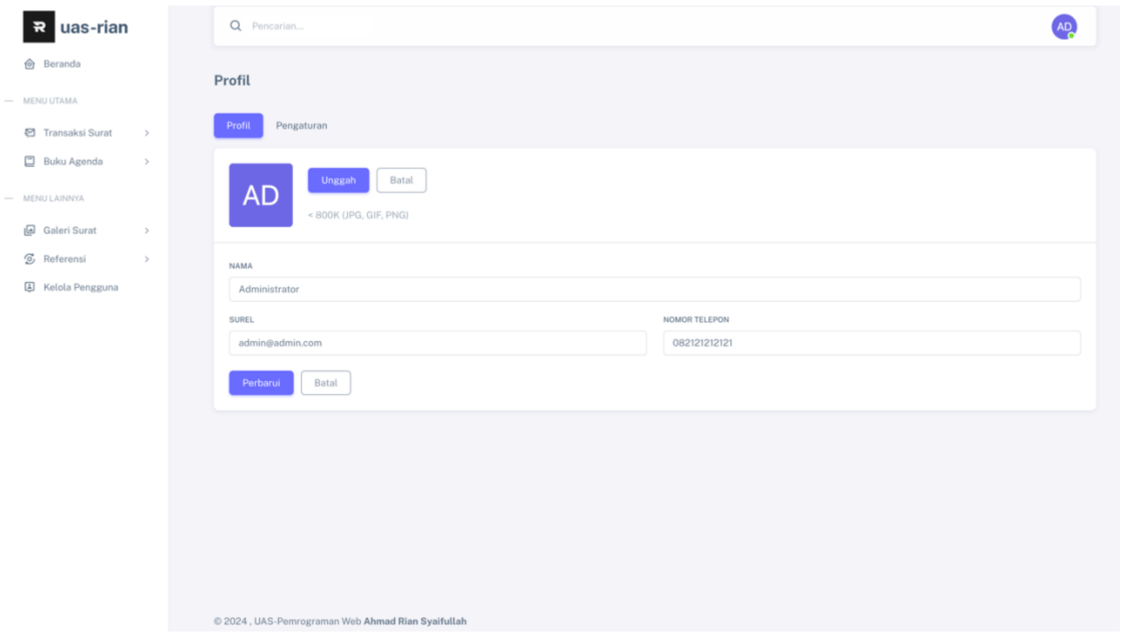


Kelola Pengguna

NAMA	SUREL	NOMOR TELEPON	MASIH AKTIF?	AKSI
Ahmad Rian	alriansr@gmail.com	082123479638	AKTIF	Edit Hapus
Test	test@gmail.com	0812345678	AKTIF	Edit Hapus
NAMA	SUREL	NOMOR TELEPON	MASIH AKTIF?	AKSI

© 2024 , UAS-Pemrograman Web **Ahmad Rian Syaifullah**

- Halaman Edit User :



Profil

NAMA
Administrator

SUREL
admin@admin.com

NOMOR TELEPON
082121212121

© 2024 , UAS-Pemrograman Web **Ahmad Rian Syaifullah**