# L A M P I R A N

**Lampiran -1** Sertifikat Laboratorium Praktikum V



SERTIFIKAT

Nomor : 505/PRK/LB-TI/II.3.AU/D/2023

Diberikan Kepada :

Ahmad

NIM. 219280127

Telah Menyelesaikan

PRAKTIKUM V

Laboratorium Teknik Informatika

Program Studi Teknik Informatika Fakultas Teknik

Universitas Muhammadiyah Parepare

Pada tanggal 16 Januari 2023 - 29 Maret 2023

Parepare, 15 Juni 2023

KETUA PRODI TEKNIK INFORMATIKA

Mughaffir Yunus, ST.,MT.

NBM. 1252 809

KEPALA LABORATORIUM

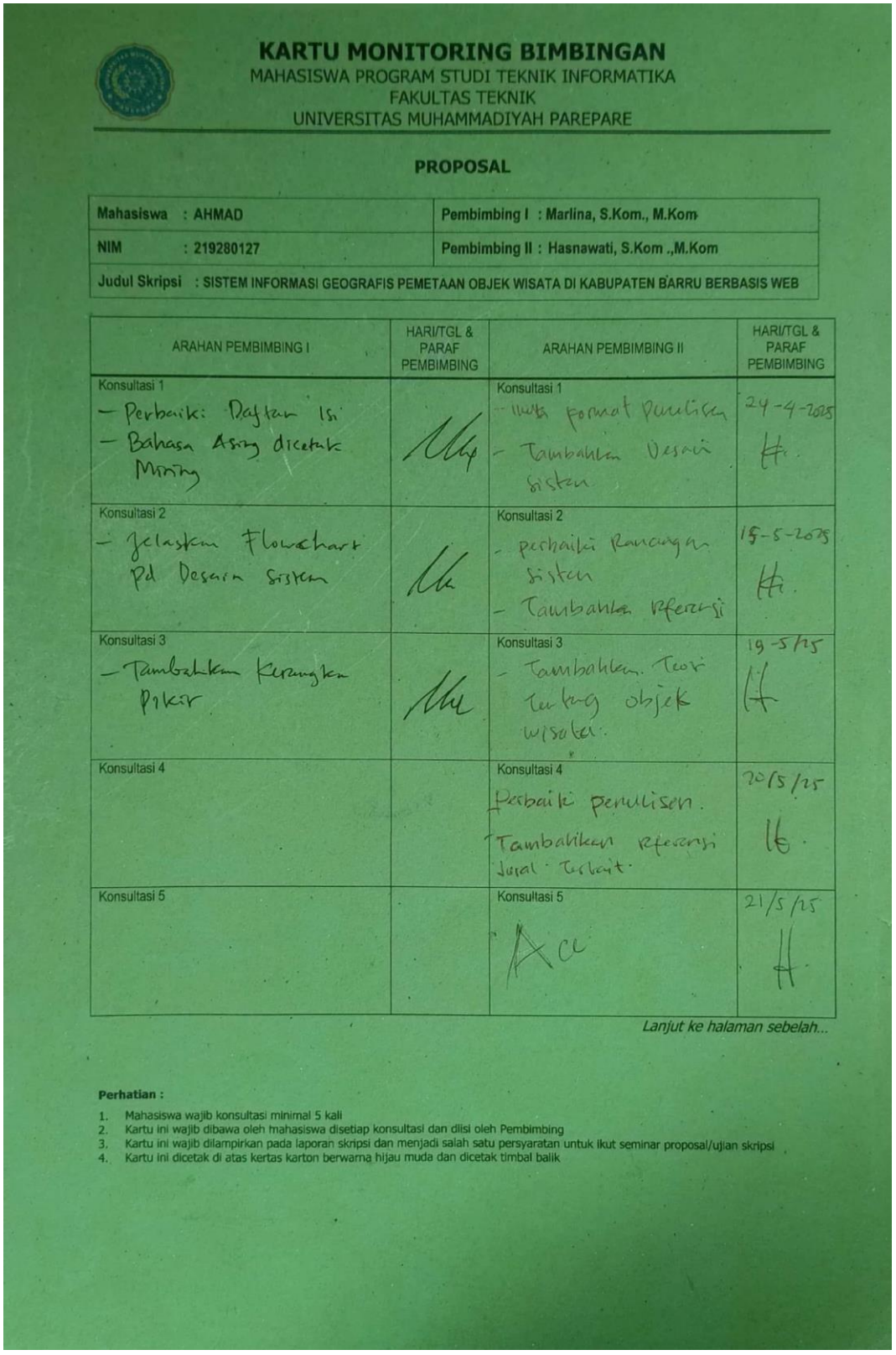
Ir. Untung Suwardoyo, S.Kom., MT., IPP.

NBM. 1288 973

**Nilai Praktikum laboratorium Teknik Informatika**
**Fakultas Teknik Universitas Muhammadiyah Parepare Tahun 2023**

| MATERI | NILAI ANGKA |
|---|---|
| PENGOLAHAN CITRA DIGITAL | 88 |
| AUGMENTED DAN VIRTUAL REALITY | 40 |
| MIKROKONTROLER | 45 |
| PEMROGRAMAN MOBILE | 45 |
| RATA-RATA | 54 |
| NILAI HURUF | C |

**Lampiran – 2** Kartu Monitoring Bimbingan Proposal



**KARTU MONITORING BIMBINGAN**
MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

**PROPOSAL**

| | |
|---|---|
| **Mahasiswa : AHMAD** | **Pembimbing I : Marlina, S.Kom., M.Kom** |
| **NIM : 219280127** | **Pembimbing II : Hasnawati, S.Kom .,M.Kom** |
| **Judul Skripsi : SISTEM INFORMASI GEOGRAFIS PEMETAAN OBJEK WISATA DI KABUPATEN BARRU BERBASIS WEB** | |

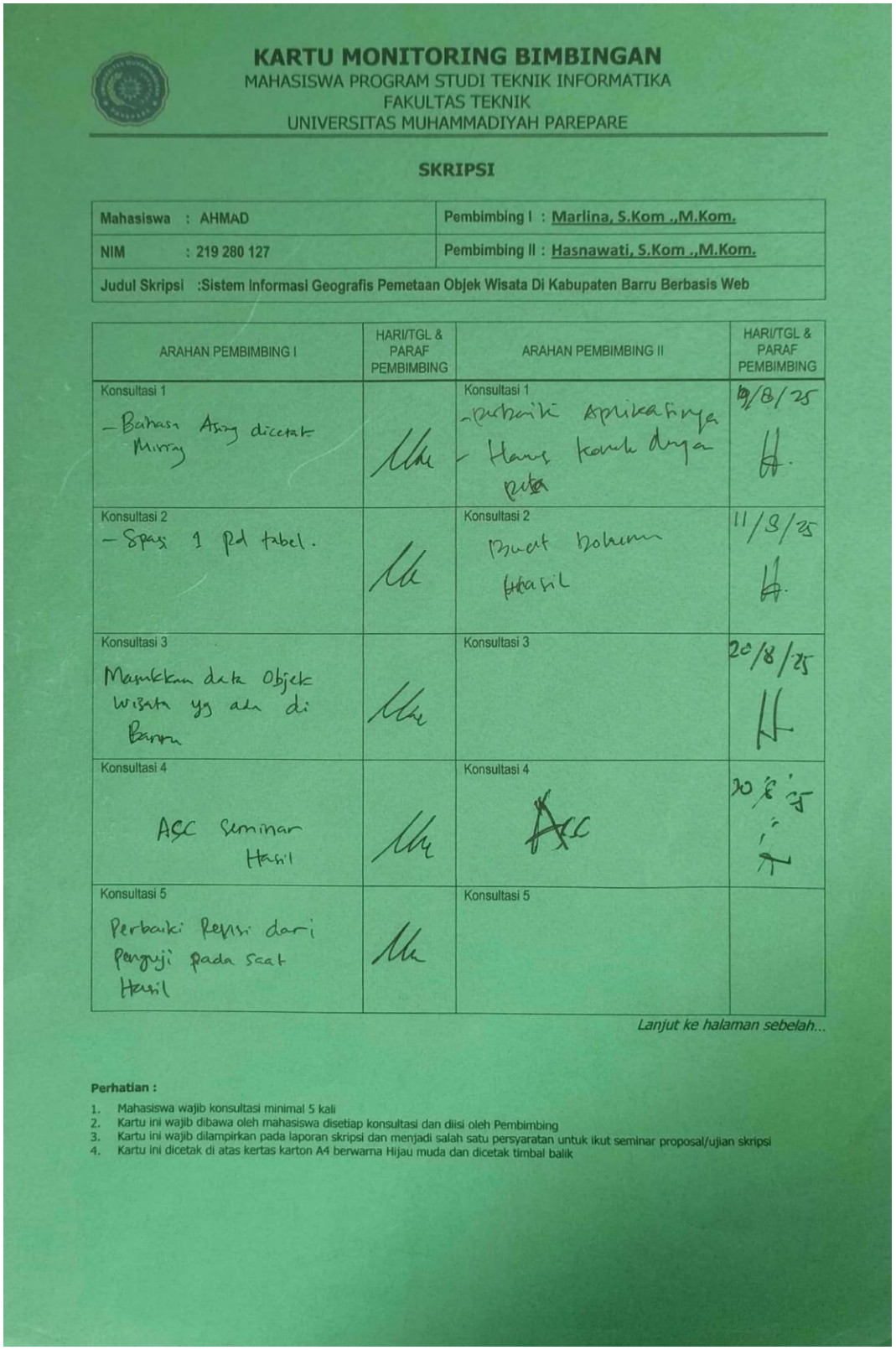
| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 1<br>– Perbaiki: Daftar Isi<br>– Bahasa Asing dicetak Miring | | Konsultasi 1<br>- Ikuti format penulisan<br>- Tambahkan Desain sistem | 24-4-2025 |
| Konsultasi 2<br>– Jelaskan Flowchart pd Desain Sistem | | Konsultasi 2<br>- perbaiki Rancangan sistem<br>- Tambahkan Referensi | 15-5-2025 |
| Konsultasi 3<br>– Tambahkan Kerangka Pikir | | Konsultasi 3<br>- Tambahkan. Teori tentang objek wisata. | 19-5/25 |
| Konsultasi 4 | | Konsultasi 4<br>Perbaiki penulisan.<br>Tambahkan Referensi Jurnal. Terkait. | 20/5/25 |
| Konsultasi 5 | | Konsultasi 5<br>Acc | 21/5/25 |

*Lanjut ke halaman sebelah...*

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

**Lampiran – 3** Kartu Monitoring Bimbingan Skripsi



**KARTU MONITORING BIMBINGAN**
MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

**SKRIPSI**

| | |
|---|---|
| Mahasiswa : AHMAD | Pembimbing I : Marlina, S.Kom .,M.Kom. |
| NIM : 219 280 127 | Pembimbing II : Hasnawati, S.Kom .,M.Kom. |
| Judul Skripsi :Sistem Informasi Geografis Pemetaan Objek Wisata Di Kabupaten Barru Berbasis Web | |

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 1<br>- Bahasa Asing dicetak Miring | | Konsultasi 1<br>- Perbaiki Aplikasinya<br>- Harus kontribusi Peta | 9/8/25 |
| Konsultasi 2<br>- Spasi 1 Pd tabel. | | Konsultasi 2<br>Buat Bahasan Hasil | 11/8/25 |
| Konsultasi 3<br>Masukkan data Objek wisata yg ada di Barru | | Konsultasi 3 | 20/8/25 |
| Konsultasi 4<br>ACC Seminar Hasil | | Konsultasi 4<br>ACC | 20/8/25 |
| Konsultasi 5<br>Perbaiki Revisi dari Penguji pada saat Hasil | | Konsultasi 5 | |

*Lanjut ke halaman sebelah...*

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton A4 berwarna Hijau muda dan dicetak timbal balik

*Lanjutan...*

| ARAHAN PEMBIMBING I | HARI/TGL & PARAF PEMBIMBING | ARAHAN PEMBIMBING II | HARI/TGL & PARAF PEMBIMBING |
|---|---|---|---|
| Konsultasi 6 | | Konsultasi 6 | |
| Konsultasi 7 | | Konsultasi 7 | |
| Konsultasi 8<br>Acc Ujian Tutup | | Konsultasi 8 | |
| Konsultasi 9 | | Konsultasi 9 | |
| Konsultasi 10 | | Konsultasi 10 | |

Parepare, Agustus 2025

Mengetahui
Ketua Program Studi

**Mughaffir Yunus, ST., MT.**
NBM. 1252 809

Mahasiswa

Ahmad
NIM. *219 280 127*

**Perhatian :**

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton A4 berwarna Hijau muda dan dicetak timbal balik

**Lampiran – 4** Listing Program

```
<?php

namespace App\Http\Controllers\Admin;


use App\Http\Controllers\Controller;

use App\Models\Destination;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Hash;


class AdminController extends Controller

{

    /**

     * Show admin login form

     */

    public function showLoginForm()

    {

        // If already logged in, redirect to dashboard

        if (session('is_admin')) {

            return redirect()->route('admin.dashboard');

        }


        return view('admin.login');

    }


    /**

     * Handle admin login

     */

    public function login(Request $request)

    {

        $request->validate([

            'password' => 'required|string'

        ]);


        // Simple password check (you can make this more secure)

        if ($request->password === 'admin123') {

            session(['is_admin' => true, 'admin_user' => 'admin']);


            return redirect()->route('admin.dashboard')

                ->with('success', 'Selamat datang di Admin Panel!');

        }


        return back()

            ->withInput()

            ->with('error', 'Password admin salah!');

    }


    /**

     * Handle admin logout

     */

    public function logout(Request $request)

    {
```

```
        session()->forget(['is_admin', 'admin_user']);
        session()->flush();

        return redirect()->route('admin.login')
            ->with('success', 'Anda telah logout dari admin panel.');
    }

    /**
     * Show admin dashboard
     */
    public function dashboard()
    {
        try {
            // Get statistics
            $totalDestinations = Destination::count();
            $activeDestinations = Destination::where('is_active', true)->count();
            $inactiveDestinations = Destination::where('is_active', false)->count();
            $trashedDestinations = Destination::onlyTrashed()->count();

            // Get recent destinations
            $recentDestinations = Destination::latest()
                ->take(5)
                ->get();

            // Get category statistics
            $categoryStats = [];
            $categories = Destination::getCategories();

            foreach ($categories as $key => $category) {
                $categoryStats[] = [
                    'name' => $category['name'],
                    'icon' => $category['icon'],
                    'color' => $category['color'],
                    'count' => Destination::where('category', $key)->count()
                ];
            }

            return view('admin.dashboard', compact(
                'totalDestinations',
                'activeDestinations',
                'inactiveDestinations',
                'trashedDestinations',
                'recentDestinations',
                'categoryStats'
            ));

        } catch (\Exception $e) {
            \Log::error('Dashboard error: ' . $e->getMessage());
```

```
            return view('admin.dashboard', [
                'totalDestinations' => 0,
                'activeDestinations' => 0,
                'inactiveDestinations' => 0,
                'trashedDestinations' => 0,
                'recentDestinations' =>
collect(),
                'categoryStats' => []
            ]);
        }
    }
}
<?php

namespace
App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Destination;
use Illuminate\Http\Request;
use Illuminate\Support\Str;
use Illuminate\Support\Facades\File;
use Illuminate\Validation\Rule;

class AdminDestinationController
extends Controller
{
    /**
     * Display a listing of destinations
with filtering and search
     */
    public function index(Request
$request)
    {
        $query = Destination::query();

        // Search functionality
        if ($request->has('search') &&
!empty($request->search)) {
            $search = $request->search;
            $query->where(function($q) use
($search) {
                $q->where('name', 'LIKE',
"%{$search}%")
                  ->orWhere('description',
'LIKE', "%{$search}%")
                  ->orWhere('address', 'LIKE',
"%{$search}%");
            });
        }

        // Category filter
        if ($request->has('category') &&
$request->category !== 'all') {
            $query->where('category',
$request->category);
        }

        // Status filter
        if ($request->has('status') &&
$request->status !== 'all') {
            $isActive = $request->status ===
'active';
```

```
        $query->where('is_active', $isActive);
    }

    // Sorting
    $sortBy = $request->get('sort', 'created_at');
    $sortDir = $request->get('direction', 'desc');

    $allowedSorts = ['name', 'category', 'created_at', 'updated_at', 'is_active'];
    if (in_array($sortBy, $allowedSorts)) {
        $query->orderBy($sortBy, $sortDir);
    }

    $destinations = $query->paginate(10)->appends($request->all());

    // Statistics
    $stats = [
        'total' => Destination::count(),
        'active' => Destination::where('is_active', true)->count(),
        'inactive' => Destination::where('is_active', false)->count(),
        'categories' => Destination::select('category')
            ->groupBy('category')
            ->selectRaw('count(*) as count, category')
            ->pluck('count', 'category')
            ->toArray()
    ];

    $categories = Destination::getCategories();

    return view('admin.destinations.index', compact(
        'destinations',
        'categories',
        'stats',
        'request'
    ));
}

/**
 * Show the form for creating a new destination
 */
public function create()
{
    $categories = Destination::getCategories();
    return view('admin.destinations.create', compact('categories'));
}

/**
```

```
    * Store a newly created destination
    */
   public function store(Request $request)
   {
       $request->validate([
           'name' => 'required|string|max:255',
           'category' => 'required|in:' . implode(',', array_keys(Destination::getCategories())),
           'description' => 'required|string',
           'latitude' => 'required|numeric|between:-90,90',
           'longitude' => 'required|numeric|between:-180,180',
           'main_image' => 'required|image|mimes:jpeg,jpg,png|max:2048',
           'address' => 'nullable|string',
           'contact' => 'nullable|string',
           'is_active' => 'boolean'
       ]);

       // Handle file upload
       $mainImagePath = null;
       if ($request->hasFile('main_image')) {
           $mainImage = $request->file('main_image');
           $mainImageName = Str::slug($request->name) . '-main-' . time() . '.' . $mainImage->getClientOriginalExtension();
           $uploadPath = public_path('images/destinations');
           if (!file_exists($uploadPath)) {
               mkdir($uploadPath, 0755, true);
           }

           $mainImage->move($uploadPath, $mainImageName);
           $mainImagePath = $mainImageName;
       }

       Destination::create([
           'name' => $request->name,
           'category' => $request->category,
           'description' => $request->description,
           'latitude' => $request->latitude,
           'longitude' => $request->longitude,
           'address' => $request->address,
           'contact' => $request->contact,
           'main_image' => $mainImagePath,
           'is_active' => $request->has('is_active'),
       ]);

       return redirect()->route('admin.destinations.index')
```

```
            ->with('success', 'Destinasi
wisata berhasil ditambahkan!');
    }


    /**
     * Display the specified destination
     */
    public function show(Destination
$destination)
    {
        $relatedDestinations =
Destination::where('category',
$destination->category)
            ->where('id', '!=', $destination-
>id)
            ->where('is_active', true)
            ->limit(3)
            ->get();


        $stats = [
            'category_count' =>
Destination::where('category',
$destination->category)->count(),
            'same_area_count' =>
Destination::whereBetween('latitude', [
                $destination->latitude - 0.1,
                $destination->latitude + 0.1
            ])->whereBetween('longitude', [
                $destination->longitude - 0.1,
                $destination->longitude + 0.1
            ])->where('id', '!=', $destination-
>id)->count(),
        ];

        return
view('admin.destinations.show',
compact(
            'destination',
            'relatedDestinations',
            'stats'
        ));
    }


    /**
     * Show the form for editing the
specified destination
     */
    public function edit(Destination
$destination)
    {
        $categories =
Destination::getCategories();


        // Check if main image exists
        $imageExists =
File::exists(public_path('images/destinat
ions/' . $destination->main_image));


        // Get existing gallery images
        $galleryImages = [];
        if ($destination->gallery) {
            foreach ($destination->gallery as
$image) {
                $galleryImages[] = [
                    'filename' => $image,
```

```
                'exists' =>
File::exists(public_path('images/destinat
ions/' . $image)),
                'url' =>
asset('images/destinations/' . $image)
            ];
        }
    }

    return
view('admin.destinations.edit', compact(
        'destination',
        'categories',
        'imageExists',
        'galleryImages'
    ));
  }

  /**
   * Update the specified destination
   */
  public function update(Request
$request, Destination $destination)
  {
    // Validation with conditional rules
    $rules = [
      'name' => [
        'required',
        'string',
        'max:255',
        Rule::unique('destinations')-
>ignore($destination->id)
      ],
      'category' => 'required|in:' .
implode(',',
array_keys(Destination::getCategories())
),
      'description' => 'required|string',
      'address' =>
'nullable|string|max:500',
      'contact' =>
'nullable|string|max:100',
      'latitude' =>
'required|numeric|between:-90,90',
      'longitude' =>
'required|numeric|between:-180,180',
      'main_image' =>
'nullable|image|mimes:jpeg,jpg,png|max:
2048',
      'gallery.*' =>
'nullable|image|mimes:jpeg,jpg,png|max:
2048',
      'entry_fees' => 'nullable|array',
      'entry_fees.*.vehicle' =>
'required_with:entry_fees|string',
      'entry_fees.*.fee' =>
'required_with:entry_fees|numeric|min:0
',
      'activities' => 'nullable|array',
      'activities.*' => 'string|max:100',
      'operating_hours' =>
'nullable|string|max:100',
      'visitor_info' => 'nullable|string',
      'is_active' => 'boolean',
      'remove_gallery' =>
'nullable|array',
      'remove_gallery.*' => 'string'
    ];
```

```
        $request->validate($rules);


        // Prepare update data
        $updateData = [
            'name' => $request->name,
            'category' => $request->category,
            'description' => $request->description,
            'address' => $request->address,
            'contact' => $request->contact,
            'latitude' => $request->latitude,
            'longitude' => $request->longitude,
            'operating_hours' => $request->operating_hours ?? '24 jam',
            'visitor_info' => $request->visitor_info,
            'is_active' => $request->has('is_active'),
        ];


        // Handle entry fees
        if ($request->has('entry_fees') && is_array($request->entry_fees)) {
            $entryFees = [];
            foreach ($request->entry_fees as $fee) {
                if (!empty($fee['vehicle']) && !empty($fee['fee'])) {
                    $entryFees[$fee['vehicle']] = (int) $fee['fee'];
                }
            }
            $updateData['entry_fees'] = !empty($entryFees) ? $entryFees : null;
        } else {
            $updateData['entry_fees'] = null;
        }


        // Handle activities
        if ($request->has('activities')) {
            $activities = array_filter($request->activities, function($activity) {
                return !empty(trim($activity));
            });
            $updateData['activities'] = !empty($activities) ? array_values($activities) : null;
        } else {
            $updateData['activities'] = null;
        }


        // Handle main image update
        if ($request->hasFile('main_image')) {
            // Delete old main image if exists
            if ($destination->main_image) {
                $oldImagePath = public_path('images/destinations/' . $destination->main_image);
                if (File::exists($oldImagePath)) {
```

```
File::delete($oldImagePath);
            }
        }

        // Upload new main image
        $mainImage = $request->file('main_image');
        $mainImageName = Str::slug($request->name) . '-main-' . time() . '.' . $mainImage->getClientOriginalExtension();

        $uploadPath = public_path('images/destinations');
        if (!file_exists($uploadPath)) {
            mkdir($uploadPath, 0755, true);
        }

        $mainImage->move($uploadPath, $mainImageName);
        $updateData['main_image'] = $mainImageName;
    }

    // Handle gallery images
    $currentGallery = $destination->gallery ?? [];

    // Remove selected images from gallery
    if ($request->has('remove_gallery') && is_array($request->remove_gallery)) {
        foreach ($request->remove_gallery as $imageToRemove) {
            // Remove from filesystem
            $imagePath = public_path('images/destinations/' . $imageToRemove);
            if (File::exists($imagePath)) {
                File::delete($imagePath);
            }

            // Remove from gallery array
            $currentGallery = array_values(array_filter($currentGallery, function($img) use ($imageToRemove) {
                return $img !== $imageToRemove;
            }));
        }
    }

    // Add new gallery images
    if ($request->hasFile('gallery')) {
        foreach ($request->file('gallery') as $index => $galleryImage) {
            if ($galleryImage->isValid()) {
                $galleryImageName = Str::slug($request->name) . '-gallery-' . time() . '-' . $index . '.' . $galleryImage->getClientOriginalExtension();
```

```
                $uploadPath =
public_path('images/destinations');
                $galleryImage-
>move($uploadPath,
$galleryImageName);


                $currentGallery[] =
$galleryImageName;
            }
        }
    }


    $updateData['gallery'] =
!empty($currentGallery) ?
$currentGallery : null;


    // Update slug if name changed
    if ($destination->name !==
$request->name) {
        $updateData['slug'] =
Str::slug($request->name);
    }


    // Update the destination
    $destination-
>update($updateData);


    return redirect()-
>route('admin.destinations.show',
$destination)
        ->with('success', 'Destinasi
wisata berhasil diperbarui!');
  }

  /**
   * Delete single destination with soft
delete support
   */
  public function destroy(Request
$request, Destination $destination)
  {
    try {
      // Log the deletion attempt
      \Log::info('Admin attempting to
delete destination', [
        'destination_id' =>
$destination->id,
        'destination_name' =>
$destination->name,
        'admin_session' =>
session('admin_user', 'unknown')
      ]);


      // Store destination data for
potential restore
      $destinationData = $destination-
>toArray();


      // Handle image cleanup
      $this-
>cleanupDestinationImages($destination
);


      // Perform soft delete (or hard
delete based on request)
      if ($request->has('force_delete')
&& $request->force_delete == 'true') {
```

```
            // Hard delete - permanently
remove
            $destination->forceDelete();
            $message = 'Destinasi berhasil
dihapus secara permanen!';
        } else {
            // Soft delete - can be restored
            $destination->delete();
            $message = 'Destinasi berhasil
Dihapus.';
        }

        if ($request->expectsJson()) {
            return response()->json([
                'success' => true,
                'message' => $message
            ]);
        }

        return redirect()-
>route('admin.destinations.index')
            ->with('success', $message)
            ->with('deleted_destination',
$destinationData); // For undo
functionality

    } catch (\Exception $e) {
        \Log::error('Failed to delete
destination', [
            'destination_id' =>
$destination->id,
            'error' => $e->getMessage()
        ]);

        if ($request->expectsJson()) {
            return response()->json([
                'success' => false,
                'message' => 'Gagal
menghapus destinasi: ' . $e-
>getMessage()
            ], 500);
        }

        return redirect()->back()
            ->with('error', 'Gagal
menghapus destinasi: ' . $e-
>getMessage());
    }
}

/**
 * Get destination preview data for
delete confirmation
 */
public function
getDeletePreview(Destination
$destination)
{
    try {
        $categoryInfo = $destination-
>getCategoryInfo();

        return response()->json([
            'success' => true,
            'data' => [
                'id' => $destination->id,
```

```
                'name' => $destination->name,
                'category' => [
                    'name' => $categoryInfo['name'],
                    'icon' => $categoryInfo['icon'],
                    'color' => $categoryInfo['color']
                ],
                'main_image_url' => $destination->main_image ?
asset('images/destinations/' . $destination->main_image) :
asset('images/placeholder.jpg'),
                'created_at' => $destination->created_at->format('d M Y'),
                'gallery_count' => $destination->gallery ? count($destination->gallery) : 0,
                'is_active' => $destination->is_active,
                'address' => $destination->address,
                'description' => \Str::limit($destination->description, 100)
            ]
        ]);

    } catch (\Exception $e) {
        \Log::error('Failed to get destination preview', [
            'destination_id' => $destination->id,
            'error' => $e->getMessage()
        ]);

        return response()->json([
            'success' => false,
            'message' => 'Gagal memuat preview destinasi: ' . $e->getMessage()
        ], 500);
    }
}

/**
 * Get bulk preview data for bulk operations
 */
public function getBulkPreview(Request $request)
{
    try {
        $request->validate([
            'destination_ids' => 'required|array|min:1',
            'destination_ids.*' => 'exists:destinations,id'
        ]);

        $destinations = Destination::whereIn('id', $request->destination_ids)
            ->select('id', 'name', 'category', 'main_image', 'is_active')
```

```
            ->get()
            ->map(function($dest) {
                $categoryInfo = $dest->getCategoryInfo();

                return [
                    'id' => $dest->id,
                    'name' => $dest->name,
                    'category' => [
                        'name' => $categoryInfo['name'],
                        'icon' => $categoryInfo['icon'],
                        'color' => $categoryInfo['color']
                    ],
                    'main_image_url' => $dest->main_image ?
asset('images/destinations/' . $dest->main_image) :
asset('images/placeholder.jpg'),
                    'is_active' => $dest->is_active
                ];
            });

        return response()->json([
            'success' => true,
            'data' => [
                'destinations' => $destinations,
                'count' => $destinations->count()
            ]
        ]);

    } catch (\Illuminate\Validation\ValidationException $e) {
        return response()->json([
            'success' => false,
            'message' => 'Data tidak valid',
            'errors' => $e->errors()
        ], 422);

    } catch (\Exception $e) {
        \Log::error('Failed to get bulk preview', [
            'error' => $e->getMessage(),
            'destination_ids' => $request->destination_ids ?? []
        ]);

        return response()->json([
            'success' => false,
            'message' => 'Gagal memuat preview bulk operation: ' . $e->getMessage()
        ], 500);
    }
}

/**
```

```
    * Preview image before upload
    */
   public function
previewImage(Request $request)
   {
      $request->validate([
         'image' =>
'required|image|mimes:jpeg,jpg,png|max
:2048'
      ]);

      if ($request->hasFile('image')) {
         $image = $request->file('image');
         $imageName = 'preview-' .
time() . '.' . $image-
>getClientOriginalExtension();

         $uploadPath =
public_path('images/temp');
         if (!file_exists($uploadPath)) {
            mkdir($uploadPath, 0755,
true);
         }

         $image->move($uploadPath,
$imageName);

         return response()->json([
            'success' => true,
            'image_url' =>
asset('images/temp/' . $imageName),
            'image_name' =>
$imageName
         ]);
      }

      return response()->json([
         'success' => false,
         'message' => 'Gagal mengupload
gambar'
      ]);
   }

   /**
    * Delete temporary image
    */
   public function
deleteTempImage(Request $request)
   {
      $imageName = $request-
>input('image_name');
      $imagePath =
public_path('images/temp/' .
$imageName);

      if (File::exists($imagePath)) {
         File::delete($imagePath);
         return response()->json(['success'
=> true]);
      }

      return response()->json(['success'
=> false]);
   }
```

```
    /**
     * Validate coordinates in real-time
     */
    public function validateCoordinates(Request $request)
    {
        $lat = $request->input('latitude');
        $lng = $request->input('longitude');

        // Basic validation
        if (!is_numeric($lat) || !is_numeric($lng)) {
            return response()->json([
                'valid' => false,
                'message' => 'Koordinat harus berupa angka'
            ]);
        }

        if ($lat < -90 || $lat > 90) {
            return response()->json([
                'valid' => false,
                'message' => 'Latitude harus antara -90 dan 90'
            ]);
        }

        if ($lng < -180 || $lng > 180) {
            return response()->json([
                'valid' => false,
                'message' => 'Longitude harus antara -180 dan 180'
            ]);
        }

        // Check if coordinates are in TTU area (approximate bounds)
        $ttuBounds = [
            'lat_min' => -10.0,
            'lat_max' => -8.5,
            'lng_min' => 123.5,
            'lng_max' => 125.5
        ];

        $inTTU = $lat >= $ttuBounds['lat_min'] && $lat <= $ttuBounds['lat_max'] &&
                 $lng >= $ttuBounds['lng_min'] && $lng <= $ttuBounds['lng_max'];

        return response()->json([
            'valid' => true,
            'in_ttu' => $inTTU,
            'message' => $inTTU ? 'Koordinat berada di wilayah Barru' : 'Koordinat di luar wilayah Barru',
            'google_maps_url' => "https://www.google.com/maps?q={$lat},{$lng}"
        ]);
    }

                \Log::info('Deleted image file', ['path' => $imagePath]);
```

```
            }
        }

        return true;
    } catch (\Exception $e) {
        \Log::warning('Failed to cleanup some images', [
            'destination_id' => $destination->id,
            'error' => $e->getMessage()
        ]);
        return false;
    }
  }
}
<?php

namespace App\Http\Controllers;

use App\Models\Destination;
use Illuminate\Http\Request;

class DestinationController extends Controller
{
  public function index(Request $request)
  {
    $query = Destination::active(); // Hanya destinasi aktif

    // Search functionality
    if ($request->has('search') && $request->search) {
        $query->search($request->search);
    }

    // Filter by category
    if ($request->has('category') && $request->category) {
        $query->byCategory($request->category);
    }

    $destinations = $query->orderBy('name', 'asc')->paginate(9);
    $categories = Destination::getCategories();

    return view('destinations.index', compact('destinations', 'categories'));
  }

  public function show(Destination $destination)
  {
    return view('destinations.show', compact('destination'));
  }
}
<?php

namespace App\Http\Controllers;
```

```
use App\Models\Destination;
use Illuminate\Http\Request;

class MapController extends Controller
{
    /**
     * Display map view with all destinations and categories.
     */
    public function index()
    {
        $destinations = Destination::where('is_active', true)->get();
        $categories = Destination::getCategories();

        return view('map.index', compact('destinations', 'categories'));
    }

    /**
     * Get filtered destinations as JSON (based on category or search).
     */
    public function getDestinationsJson(Request $request)
    {
        $query = Destination::where('is_active', true);

        // Filter by category
        if ($request->has('category') && $request->category !== 'all') {
            $query->where('category', $request->category);
        }

        // Search by name, description, or address
        if ($request->has('search') && !empty($request->search)) {
            $search = $request->search;
            $query->where(function($q) use ($search) {
                $q->where('name', 'LIKE', "%{$search}%")
                  ->orWhere('description', 'LIKE', "%{$search}%")
                  ->orWhere('address', 'LIKE', "%{$search}%");
            });
        }

        $destinations = $query->select('id', 'name', 'slug', 'category', 'description', 'address', 'latitude', 'longitude', 'main_image')
            ->get()
            ->map(function ($destination) {
                $categoryInfo = $destination->getCategoryInfo();

                return [
                    'id' => $destination->id,
```

```
                'name' => $destination->name,
                'slug' => $destination->slug,
                'category' => $destination->category,
                'category_name' => $categoryInfo['name'],
                'category_icon' => $categoryInfo['icon'],
                'category_color' => $categoryInfo['color'],
                'description' => \Str::limit($destination->description, 100),
                'address' => $destination->address,
                'latitude' => (float) $destination->latitude,
                'longitude' => (float) $destination->longitude,
                'image' => asset('images/destinations/' . $destination->main_image),
                'url' => route('destinations.show', $destination->slug)
            ];
        });

    return response()->json($destinations);
  }

  /**
   * Get nearest destinations from user location.
   */
  public function getNearestDestinations(Request $request)
  {
    $request->validate([
      'latitude' => 'required|numeric|between:-90,90',
      'longitude' => 'required|numeric|between:-180,180',
      'radius' => 'integer|min:1|max:100'
    ]);

    $userLat = $request->latitude;
    $userLng = $request->longitude;
    $radius = $request->radius ?? 50; // Default 50km

    $destinations = Destination::where('is_active', true)
      ->get()
      ->map(function ($destination) use ($userLat, $userLng) {
        $distance = $this->calculateDistance(
          $userLat, $userLng,
          $destination->latitude, $destination->longitude
        );

        $categoryInfo = $destination->getCategoryInfo();
```

```
            return [
                'id' => $destination->id,
                'name' => $destination->name,
                'slug' => $destination->slug,
                'category' => $destination->category,
                'category_name' => $categoryInfo['name'],
                'category_icon' => $categoryInfo['icon'],
                'category_color' => $categoryInfo['color'],
                'description' => \Str::limit($destination->description, 100),
                'address' => $destination->address,
                'latitude' => (float) $destination->latitude,
                'longitude' => (float) $destination->longitude,
                'distance' => round($distance, 2),
                'travel_time' => $this->estimateTravelTime($distance),
                'image' => asset('images/destinations/' . $destination->main_image),
                'url' => route('destinations.show', $destination->slug)
            ];
        })
        ->filter(function ($destination) use ($radius) {
            return $destination['distance'] <= $radius;
        })
        ->sortBy('distance')
        ->take(10)
        ->values();

    return response()->json([
        'success' => true,
        'data' => $destinations,
        'total' => $destinations->count(),
        'radius' => $radius,
        'user_location' => [
            'latitude' => $userLat,
            'longitude' => $userLng
        ]
    ]);
}

/**
 * Calculate distance between two coordinates using Haversine formula.
 */
private function calculateDistance($lat1, $lng1, $lat2, $lng2)
{
    $earthRadius = 6371; // km

    $dLat = deg2rad($lat2 - $lat1);
    $dLng = deg2rad($lng2 - $lng1);
```

```
        $a = sin($dLat / 2) * sin($dLat / 2)
+
            cos(deg2rad($lat1)) *
cos(deg2rad($lat2)) *
            sin($dLng / 2) * sin($dLng / 2);


        $c = 2 * atan2(sqrt($a), sqrt(1 -
$a));


        return $earthRadius * $c;
    }


    /**
     * Estimate travel time based on
distance (in km).
     */
    private function
estimateTravelTime($distance)
    {
        $averageSpeed = 30; // km/h
        $timeInHours = $distance /
$averageSpeed;
        $timeInMinutes =
round($timeInHours * 60);


        if ($timeInMinutes < 60) {
            return $timeInMinutes . ' menit';
        } else {
            $hours = floor($timeInMinutes /
60);
            $minutes = $timeInMinutes %
60;
            return $minutes > 0
                ? $hours . ' jam ' . $minutes . '
menit'
                : $hours . ' jam';
        }
    }
}
<?php


namespace App\Http\Middleware;


use Closure;
use Illuminate\Http\Request;


class AdminAuth
{
    public function handle(Request
$request, Closure $next)
    {
        if (!session('admin_logged_in')) {
            return redirect()-
>route('admin.login')->withErrors([
                'access' => 'Silakan login
sebagai admin terlebih dahulu!'
            ]);
        }


        return $next($request);
    }
}
<?php
```

```
namespace App\Http\Middleware;


use Closure;

use Illuminate\Http\Request;

use
Symfony\Component\HttpFoundation\R
esponse;


class AdminMiddleware

{

   /**

    * Handle an incoming request.

    */

   public function handle(Request
$request, Closure $next): Response

   {

      // Check if user is logged in as
admin

      if (!session('is_admin')) {

         // For AJAX requests, return
JSON response

         if ($request->expectsJson() ||
$request->ajax()) {

            return response()->json([

               'success' => false,

               'message' => 'Unauthorized
access. Please login as admin.',

               'redirect' =>
route('admin.login')

            ], 401);

         }

         // For regular requests, redirect to
admin login

         return redirect()-
>route('admin.login')

            ->with('error', 'Silakan login
terlebih dahulu untuk mengakses admin
panel.');

      }


      return $next($request);

   }

}

<?php


namespace App\Models;


use
Illuminate\Database\Eloquent\Model;

use
Illuminate\Database\Eloquent\SoftDelet
es;

use Illuminate\Support\Str;


class Destination extends Model

{

   use SoftDeletes; // Enable soft delete
functionality


   protected $fillable = [

      'name', 'slug', 'category',
'description', 'address', 'contact',

      'latitude', 'longitude', 'main_image',
'gallery', 'entry_fees',
```

```
        'operating_hours', 'visitor_info',
'activities', 'is_active'
    ];

    protected $casts = [
        'gallery' => 'array',
        'entry_fees' => 'array',
        'activities' => 'array',
        'latitude' => 'decimal:8',
        'longitude' => 'decimal:8',
        'is_active' => 'boolean',
        'deleted_at' => 'datetime',
    ];

    protected $dates = ['deleted_at']; //
Specify soft delete date field

    // Kategori yang tersedia
    public static function getCategories()
    {
        return [
            'wisata-alam' => [
                'name' => 'Wisata Alam',
                'icon' => '',
                'color' => 'green'
            ],
            'pantai' => [
                'name' => 'Pantai',
                'icon' => '',
                'color' => 'blue'
            ],
            'sejarah' => [
                'name' => 'Bersejarah',
                'icon' => '',
                'color' => 'yellow'
            ],
            'budaya' => [
                'name' => 'Budaya',
                'icon' => '',
                'color' => 'purple'
            ],
            'religi' => [
                'name' => 'Religi',
                'icon' => '',
                'color' => 'indigo'
            ]
        ];
    }

    public function getCategoryInfo()
    {
        $categories = self::getCategories();
        return $categories[$this->category]
?? $categories['wisata-alam'];
    }

    public function getRouteKeyName()
    {
        return 'slug';
    }
```

```
    /**
     * Get the main image URL
     */
    public function getMainImageUrlAttribute()
    {
        if ($this->main_image) {
            return asset('images/destinations/' . $this->main_image);
        }
        return asset('images/placeholder.jpg');
    }

    /**
     * Get gallery URLs
     */
    public function getGalleryUrlsAttribute()
    {
        if (!$this->gallery || !is_array($this->gallery)) {
            return [];
        }

        return array_map(function($image) {
            return asset('images/destinations/' . $image);
        }, $this->gallery);
    }

    /**
     * Scope for active destinations
     */
    public function scopeActive($query)
    {
        return $query->where('is_active', true);
    }

    /**
     * Scope for filtering by category
     */
    public function scopeByCategory($query, $category)
    {
        if ($category && $category !== 'all') {
            return $query->where('category', $category);
        }
        return $query;
    }

    /**
     * Scope for search functionality
     */
    public function scopeSearch($query, $search)
    {
        if ($search) {
```

```
            return $query->where(function($q) use ($search) {
                $q->where('name', 'LIKE', "%{$search}%")
                  ->orWhere('description', 'LIKE', "%{$search}%")
                  ->orWhere('address', 'LIKE', "%{$search}%");
            });
        }
        return $query;
    }

    /**
     * Check if destination has entry fees
     */
    public function hasEntryFees()
    {
        return $this->entry_fees && is_array($this->entry_fees) && count($this->entry_fees) > 0;
    }

    /**
     * Get formatted entry fees
     */
    public function getFormattedEntryFees()
    {
        if (!$this->hasEntryFees()) {
            return 'Gratis';
        }

        $fees = [];
        foreach ($this->entry_fees as $type => $fee) {
            $fees[] = ucfirst($type) . ': Rp ' . number_format($fee, 0, ',', '.');
        }

        return implode(', ', $fees);
    }

    protected static function boot()
    {
        parent::boot();

        static::creating(function ($destination) {
            $destination->slug = Str::slug($destination->name);
        });

        static::updating(function ($destination) {
            if ($destination->isDirty('name')) {
                $destination->slug = Str::slug($destination->name);
            }
        });

        // Log soft delete events
        static::deleted(function ($destination) {
```

```
        \Log::info('Destination soft deleted', [
            'id' => $destination->id,
            'name' => $destination->name,
            'deleted_at' => now()
        ]);
    });

    static::restored(function ($destination) {
        \Log::info('Destination restored', [
            'id' => $destination->id,
            'name' => $destination->name,
            'restored_at' => now()
        ]);
    });
  }
}
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
  /** @use HasFactory<\Database\Factories\UserFactory> */
  use HasFactory, Notifiable;

  /**
   * The attributes that are mass assignable.
   *
   * @var list<string>
   */
  protected $fillable = [
    'name',
    'email',
    'password',
  ];

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

  /**
```

```
    * Get the attributes that should be cast.
    *
    * @return array<string, string>
    */
   protected function casts(): array
   {
      return [
         'email_verified_at' => 'datetime',
         'password' => 'hashed',
      ];
   }
}
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
use App\Http\Middleware\AdminMiddleware;

return Application::configure(basePath: dirname(__DIR__))
   ->withRouting(
      web: __DIR__.'/../routes/web.php',
      commands: __DIR__.'/../routes/console.php',
      health: '/up',
   )
   ->withMiddleware(function (Middleware $middleware) {
      $middleware->alias([
         'admin' => AdminMiddleware::class,
      ]);
   })
   ->withExceptions(function (Exceptions $exceptions) {
      //
   })->create();
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
   public function run()
   {
      $this->call([
         DestinationSeeder::class,
      ]);
   }
}
<?php

use Illuminate\Support\Facades\Route;
```

```
use
App\Http\Controllers\MapController;

use
App\Http\Controllers\DestinationContro
ller;

use
App\Http\Controllers\Admin\AdminCon
troller;

use
App\Http\Controllers\Admin\AdminDes
tinationController;


// Home page

Route::get('/', function () {

    return view('home');

})->name('home');


// Map routes

Route::get('/peta', [MapController::class,
'index'])->name('map.index');

Route::get('/api/destinations',
[MapController::class,
'getDestinationsJson'])-
>name('map.destinations.json');

Route::post('/api/nearest',
[MapController::class,
'getNearestDestinations'])-
>name('map.nearest');


// Destination routes (public)

Route::get('/destinasi',
[DestinationController::class, 'index'])-
>name('destinations.index');

Route::get('/destinasi/{destination}',
[DestinationController::class, 'show'])-
>name('destinations.show');


Route::prefix('admin')->name('admin.')-
>group(function () {


    // === AUTHENTICATION
ROUTES (No middleware required)
===

    Route::get('login',
[AdminController::class,
'showLoginForm'])->name('login');

    Route::post('login',
[AdminController::class, 'login'])-
>name('login.post');

    Route::post('logout',
[AdminController::class, 'logout'])-
>name('logout');


    // === PROTECTED ADMIN
ROUTES ===

    Route::middleware(['admin'])-
>group(function () {


        // Dashboard

        Route::get('dashboard',
[AdminController::class, 'dashboard'])-
>name('dashboard');

        Route::get('/', function () {

            return redirect()-
>route('admin.dashboard');

        });


        // === DESTINATIONS CRUD -
FULL RESOURCE ROUTES ===

        Route::resource('destinations',
AdminDestinationController::class);
```

```
Route::patch('destinations/{destination}/
toggle-status',
[AdminDestinationController::class,
'toggleStatus'])
        ->name('destinations.toggle-
status');


    // Export destinations to CSV
    Route::get('destinations-export',
[AdminDestinationController::class,
'exportCsv'])
        ->name('destinations.export');


    Route::post('destinations/bulk-
delete',
[AdminDestinationController::class,
'bulkDelete'])
        ->name('destinations.bulk-
delete');


    // Trash management routes

Route::get('destinations/trash/index',
[AdminDestinationController::class,
'trashed'])
        ->name('destinations.trashed');


    // Restore single destination from
trash

Route::post('destinations/{id}/restore',
[AdminDestinationController::class,
'restore'])
        ->name('destinations.restore');


    // Bulk restore destinations from
trash
    Route::post('destinations/bulk-
restore',
[AdminDestinationController::class,
'bulkRestore'])
        ->name('destinations.bulk-
restore');


    // Clean up orphaned images
    Route::post('destinations/cleanup-
orphaned-images',
[AdminDestinationController::class,
'cleanupOrphanedImages'])
        ->name('destinations.cleanup-
images');


    // Force delete (permanent delete
from trash)

Route::delete('destinations/{destination}
/force-delete',
function(\App\Models\Destination
$destination) {
        return
app(AdminDestinationController::class)
->destroy(
            request()-
>merge(['force_delete' => 'true']),
            $destination
        );
    })->name('destinations.force-
delete');


Route::get('destinations/{destination}/de
lete-preview',
```

```
[AdminDestinationController::class,
'getDeletePreview'])
        ->name('destinations.delete-
preview');


    // Quick stats for bulk operations
    Route::post('destinations/bulk-
preview',
[AdminDestinationController::class,
'getBulkPreview'])
        ->name('destinations.bulk-
preview');


    // API routes for enhanced
UPDATE features
    Route::prefix('api')->name('api.')-
>group(function () {
        // Image preview and
management
        Route::post('/preview-image',
[AdminDestinationController::class,
'previewImage'])
            ->name('preview-image');


        Route::delete('/delete-temp-
image',
[AdminDestinationController::class,
'deleteTempImage'])
            ->name('delete-temp-image');


        // Coordinate validation
        Route::post('/validate-
coordinates',
[AdminDestinationController::class,
'validateCoordinates'])
            ->name('validate-
coordinates');
    });
  });
});
<?php


namespace Database\Seeders;


use Illuminate\Database\Seeder;


class DatabaseSeeder extends Seeder
{
  public function run()
  {
    $this->call([
      DestinationSeeder::class,
    ]);
  }
}
```