LAMPIRAN

Lampiran 1 *Source Code*

Model Kualitas Udara

```
<?php

namespace App\Models;

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

class AirIndex extends Model
{
    use HasFactory;

    protected $guarded = [];
    protected $table  = "air_indices";

    protected $casts = [
        'created_at' => 'datetime:Y-m-d H:i:s',
    ];
}
```

Model Suhu dan Kelembaban

```
<?php

namespace App\Models;

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

class TemperatureHumidityIndex extends Model
{
    use HasFactory;

    protected $guarded = [];
    protected $table = "temperature_humidity_indices";

    protected $casts = [
        'created_at' => 'datetime:Y-m-d H:i:s',
    ];
}
```

View Halaman Monitoring

```
@extends('layouts.front')

@section('content')
   <div class="container">
      <div class="row">
         <div class="col-md-12 text-center mt-5">
               <img src="{{ asset('images/logo-dishub-dki.png') }}" alt="logo" class="img-fluid"
width="200">
           <h1 class="display-3 text-capitalize">Monitoring kualitas udara unit pengelola pengujian
kendaraan bermotor pulogadung</h1>
         </div>

         <div class="col-md-12 text-center mt-5">
            <div class="card bg-white">
               <div class="card-header bg-warning text-bg-warning">
                  <h1 class="display-5 text-capitalize">Data kualitas udara saat ini terintegrasi LHK
Jakarta</h1>
               </div>
               <div class="card-body">
                  {!! $jakartaAirIndexHtml !!}
               </div>
            </div>
         </div>
      </div>
   </div>

   <section class="bg-white p-5 mb-2 shadow-sm">
      <div class="container">
         <div class="row">
            <div class="col-md-12">
               <h1 class="text-primary text-uppercase mb-4">Filter</h1>
               <div class="card">
                  <div class="card-body">
                     <div class="form-group">
                        <h4>Filter Rentang Tanggal</h4>
                        <input id="rentang-tanggal" type="text" class="form-control" name="daterange"
value="" />
                     </div>

                  </div>
               </div>
            </div>
         </div>
      </div>
   </section>

   <section class="bg-white p-5">
      <div class="container">
         <h1 class="text-primary text-uppercase font-monospace mb-4">Data Kualitas Udara</h1>
         <div class="row">
            <div class="col-md-12 mb-3">
               <div class="card">
                  <div class="card-body">
                     <div class="form-group">
                        <h4>Pilih Status</h4>
                        <select name="status" id="status" class="form-control">
```

```
                <option value="">Semua</option>
                <option value="Baik">Baik</option>
                <option value="Sedang">Sedang</option>
                <option value="Tidak Sehat">Tidak Sehat</option>
                <option value="Sangat Tidak Sehat">Sangat Tidak Sehat</option>
                <option value="Berbahaya">Berbahaya</option>
              </select>
            </div>

        </div>
    </div>
</div>
<div class="col-md-12 mb-3">
    <div class="card bg-white">
        <div class="card-body position-relative">
            <h1 class="display-5 text-capitalize">Grafik Kualitas Udara</h1>
            <div class="position-relative" style="height: 500px">
                {!! $airIndexChart->container() !!}
            </div>
        </div>
    </div>
</div>
<div class="col-md-4">
    <div class="card text-center">
        <div class="card-header bg-primary-subtle">
            Rata-rata Harian
        </div>
        <div class="card-body">
            <h1 id="average-air-index-daily" class="card-title">0</h1>
        </div>
    </div>
</div>

<div class="col-md-4">
    <div class="card text-center">
        <div class="card-header bg-secondary-subtle">
            Rata-rata Bulanan
        </div>
        <div class="card-body">
            <h1 id="average-air-index-monthly" class="card-title">0</h1>
        </div>
    </div>
</div>

<div class="col-md-4">
    <div class="card text-center">
        <div class="card-header bg-success-subtle">
            Rata-rata Tahunan
        </div>
        <div class="card-body">
            <h1 id="average-air-index-yearly" class="card-title">0</h1>
        </div>
    </div>
</div>

<div class="col-md-12 mt-3">
    <div class="card bg-white">
        <div class="card-body">
```

```
                    <h5 class="text-capitalize">Tabel Kualitas Udara</h5>
                    <div class="table-responsive">
                      <table id="airIndexTable" class="table table-striped">
                        <thead>
                        <tr>
                          <th>No</th>
                          <th>Nilai Udara</th>
                          <th>Status</th>
                          <th>Tanggal</th>
                        </tr>
                        </thead>
                        <tbody>
                        </tbody>
                      </table>
                    </div>
                  </div>
                </div>
              </div>

            </div>
          </section>

          <section class="bg-white p-5 mt-2 shadow-sm">
            <div class="container">
                    <h1 class="text-primary text-uppercase  font-monospace  mb-4">Data  Kelembaban
Suhu</h1>
              <div class="row">
                <div class="col-md-12 mb-3">
                  <div class="card bg-white">
                    <div class="card-body position-relative">
                      <h1 class="display-5 text-capitalize">Grafik Kelembaban Suhu</h1>
                      <div class="position-relative" style="height: 500px">
                        {!! $temperatureHumidityIndexChart->container() !!}
                      </div>
                    </div>
                  </div>
                </div>
                <div class="col-md-4">
                  <div class="card text-center">
                    <div class="card-header bg-primary-subtle">
                      Rata-rata Harian
                    </div>
                    <div class="card-body">
                      <h1 id="average-temperature-humidity-index-daily" class="card-title">0</h1>
                    </div>
                  </div>
                </div>

                <div class="col-md-4">
                  <div class="card text-center">
                    <div class="card-header bg-secondary-subtle">
                      Rata-rata Bulanan
                    </div>
                    <div class="card-body">
                      <h1 id="average-temperature-humidity-index-monthly" class="card-title">0</h1>
                    </div>
                  </div>
```

```
            </div>

            <div class="col-md-4">
               <div class="card text-center">
                  <div class="card-header bg-success-subtle">
                     Rata-rata Tahunan
                  </div>
                  <div class="card-body">
                     <h1 id="average-temperature-humidity-index-yearly" class="card-title">0</h1>
                  </div>
               </div>
            </div>

            <div class="col-md-12 mt-3">
               <div class="card bg-white">
                  <div class="card-body">
                     <h5 class="text-capitalize">Tabel Kelembaban Suhu</h5>
                     <div class="table-responsive">
                        <table id="temperatureHumidityIndexTable" class="table table-striped">
                           <thead>
                              <tr>
                                 <th>No</th>
                                 <th>Nilai Suhu</th>
                                 <th>Nilai Kelembaban</th>
                                 <th>Tanggal</th>
                              </tr>
                           </thead>
                           <tbody>
                           </tbody>
                        </table>
                     </div>
                  </div>
               </div>
            </div>

         </div>
      </div>
   </section>
@endsection

@push('scripts')
{!! $airIndexChart->script() !!}
{!! $temperatureHumidityIndexChart->script() !!}

<script type="text/javascript">
   const airIndexTable = $('#airIndexTable').DataTable({
      processing: true,
      serverSide: true,
      ajax: '{{ route('air-index') }}',
      columns: [
         { data: 'DT_RowIndex', name: 'DT_RowIndex', orderable: false, searchable: false },
         { data: 'nilai_udara', name: 'nilai_udara' },
         { data: 'status', name: 'status' },
         { data: 'created_at', name: 'created_at' },
      ]
   });

   const temperatureHumidityIndexTable = $('#temperatureHumidityIndexTable').DataTable({
```

```
    processing: true,
    serverSide: true,
    ajax: '{{ route('temperature-humidity-index') }}',
    columns: [
        { data: 'DT_RowIndex', name: 'DT_RowIndex', orderable: false, searchable: false },
        { data: 'nilai_suhu', name: 'nilai_suhu' },
        { data: 'nilai_lembab', name: 'nilai_lembab' },
        { data: 'created_at', name: 'created_at' },
    ]
});

window.{{ $airIndexChart->id }}
window.{{ $temperatureHumidityIndexChart->id }}

$(function() {
    $('input[name="daterange"]').daterangepicker({
        autoUpdateInput: false,
    });
});
    function airIndexYearly(){
        $.ajax({
            url: '{{ route('air-index-statistic') }}',
            type: 'GET',
            dataType: 'json',
            data: {
                start_date: '{{ now()->startOfYear()->format('Y-m-d') }}',
                end_date: '{{ now()->endOfYear()->format('Y-m-d') }}'
            },
            success: function (data) {
                $('#average-air-index-yearly').html(data.average);
            }
        })
    }

    function airIndexMonthly(){
        $.ajax({
            url: '{{ route('air-index-statistic') }}',
            type: 'GET',
            dataType: 'json',
            data: {
                start_date: '{{ now()->startOfMonth()->format('Y-m-d') }}',
                end_date: '{{ now()->endOfMonth()->format('Y-m-d') }}'
            },
            success: function (data) {
                $('#average-air-index-monthly').html(data.average);
            }
        })
    }

    function airIndexDaily(){
        $.ajax({
            url: '{{ route('air-index-statistic') }}',
            type: 'GET',
            dataType: 'json',
            data: {
                start_date: '{{ now()->startOfDay()->format('Y-m-d') }}',
                end_date: '{{ now()->endOfDay()->format('Y-m-d') }}'
            },
```

```
        success: function (data) {
            $('#average-air-index-daily').html(data.average);
        }
    })
}

function airIndexStatistic(){
    airIndexYearly();
    airIndexMonthly();
    airIndexDaily();
}

airIndexStatistic();

function temperatureHumidityIndexYearly(){
    $.ajax({
        url: '{{ route('temperature-humidity-index-statistic') }}',
        type: 'GET',
        dataType: 'json',
        data: {
            start_date: '{{ now()->startOfYear()->format('Y-m-d') }}',
            end_date: '{{ now()->endOfYear()->format('Y-m-d') }}'
        },
        success: function (data) {
            $('#average-temperature-humidity-index-yearly').html(data.average);
        }
    })
}

function temperatureHumidityIndexMonthly(){
    $.ajax({
        url: '{{ route('temperature-humidity-index-statistic') }}',
        type: 'GET',
        dataType: 'json',
        data: {
            start_date: '{{ now()->startOfMonth()->format('Y-m-d') }}',
            end_date: '{{ now()->endOfMonth()->format('Y-m-d') }}'
        },
        success: function (data) {
            $('#average-temperature-humidity-index-monthly').html(data.average);
        }
    })
}

function temperatureHumidityIndexDaily(){
    $.ajax({
        url: '{{ route('temperature-humidity-index-statistic') }}',
        type: 'GET',
        dataType: 'json',
        data: {
            start_date: '{{ now()->startOfDay()->format('Y-m-d') }}',
            end_date: '{{ now()->endOfDay()->format('Y-m-d') }}'
        },
        success: function (data) {
            $('#average-temperature-humidity-index-daily').html(data.average);
        }
    })
}
```

```
    function temperatureHumidityIndexStatistic(){
        temperatureHumidityIndexYearly();
        temperatureHumidityIndexMonthly();
        temperatureHumidityIndexDaily();
    }

    temperatureHumidityIndexStatistic();

    $('input[name="daterange"]').on('apply.daterangepicker', function(ev, picker) {
                    $(this).val(picker.startDate.format('MM/DD/YYYY') + ' - ' +
picker.endDate.format('MM/DD/YYYY'));
        const startDate = picker.startDate.format('YYYY-MM-DD');
        const endDate = picker.endDate.format('YYYY-MM-DD');
                                        airIndexTable.ajax.url(`{{ route('air-index')
}}?start_date=${startDate}&end_date=${endDate}`).load();
            temperatureHumidityIndexTable.ajax.url(`{{ route('temperature-humidity-index')
}}?start_date=${startDate}&end_date=${endDate}`).load();

        var air_index_chart_original_api_url = {{ $airIndexChart->id }}_api_url;
        if(air_index_chart_original_api_url.includes('?')){
            air_index_chart_original_api_url = air_index_chart_original_api_url.split('?')[0];
        }
                var air_index_chart_new_api_url = air_index_chart_original_api_url +
`?start_date=${startDate}&end_date=${endDate}`;
        {{ $airIndexChart->id }}_refresh(air_index_chart_new_api_url);

                        var temperature_humidity_index_chart_original_api_url = {{
$temperatureHumidityIndexChart->id }}_api_url;
        if(temperature_humidity_index_chart_original_api_url.includes('?')){
                                temperature_humidity_index_chart_original_api_url =
temperature_humidity_index_chart_original_api_url.split('?')[0];
        }
                                var temperature_humidity_index_chart_new_api_url =
temperature_humidity_index_chart_original_api_url +
`?start_date=${startDate}&end_date=${endDate}`;
                                        {{ $temperatureHumidityIndexChart->id
}}_refresh(temperature_humidity_index_chart_new_api_url);

    });

    $('input[name="daterange"]').on('cancel.daterangepicker', function(ev, picker) {
        $(this).val('');

        airIndexTable.ajax.url(`{{ route('air-index') }}`).load();
            temperatureHumidityIndexTable.ajax.url(`{{ route('temperature-humidity-index')
}}`).load();

        var air_index_chart_original_api_url = {{ $airIndexChart->id }}_api_url;
        if(air_index_chart_original_api_url.includes('?')){
            air_index_chart_original_api_url = air_index_chart_original_api_url.split('?')[0];
        }
        var air_index_chart_new_api_url = air_index_chart_original_api_url;
        {{ $airIndexChart->id }}_refresh(air_index_chart_new_api_url);

                        var temperature_humidity_index_chart_original_api_url = {{
$temperatureHumidityIndexChart->id }}_api_url;
        if(temperature_humidity_index_chart_original_api_url.includes('?')){
```

```
                temperature_humidity_index_chart_original_api_url = temperature_humidity_index_chart_original_api_url.split('?')[0];
            }
            var temperature_humidity_index_chart_new_api_url = temperature_humidity_index_chart_original_api_url;
            {{ $temperatureHumidityIndexChart->id }}_refresh(temperature_humidity_index_chart_new_api_url);
        });

        $('#status').on('change', function(){
            const status = $(this).val();
            airIndexTable.ajax.url(`{{ route('air-index') }}?status=${status}`).load();

            var air_index_chart_original_api_url = {{ $airIndexChart->id }}_api_url;
            if(air_index_chart_original_api_url.includes('?')){
                air_index_chart_original_api_url = air_index_chart_original_api_url.split('?')[0];
            }
            var air_index_chart_new_api_url = air_index_chart_original_api_url + `?status=${status}`;
            {{ $airIndexChart->id }}_refresh(air_index_chart_new_api_url);

        });

        // polling for update

        setInterval(function(){
            airIndexStatistic();
            airIndexTable.ajax.reload();

            temperatureHumidityIndexStatistic();
            temperatureHumidityIndexTable.ajax.reload();
        }, 10000);

        async function getLatestAirIndexData(oldData){
            $.ajax({
                url: '{{ route('latest-air-index-data') }}',
                type: 'GET',
                dataType: 'json',
                success: function (data) {
                    console.log(data);
                    oldData.push(data?.nilai_udara);
                }
            })
        }

        async function getLatestTemperatureHumidityIndexData(oldSuhuData, oldLembabData){
            $.ajax({
                url: '{{ route('latest-temperature-humidity-index-data') }}',
                type: 'GET',
                dataType: 'json',
                success: function (data) {
                    // console.log(data.nilai_lembab, data.nilai_suhu);
                    oldSuhuData.push(data?.nilai_suhu);
                    oldLembabData.push(data?.nilai_lembab);
                }
            })
        }

        // reload chart per second and chart like running from right to left
```

```
    setInterval(async function(){
        if($('input[name="daterange"]').val() !== '' || $('#status').val() !== ''){
            return;
        }
        {{--console.log(window.{{ $airIndexChart->id }}.data);--}}
        {{--console.log(window.{{ $airIndexChart->id }}.data);--}}
        const labels = window.{{ $airIndexChart->id }}.data.labels;
        const oldData = window.{{ $airIndexChart->id }}.data.datasets[0].data;

        // oldData.push(Math.floor(Math.random() * 100));

        await getLatestAirIndexData(oldData);

        // update labels to time
        const now = new Date();
        const time = now.getHours() + ":" + now.getMinutes() + ":" + now.getSeconds();
        labels.push(time);

        oldData.shift();
        labels.shift();

        window.{{ $airIndexChart->id }}.data.datasets[0].data = oldData
        window.{{ $airIndexChart->id }}.data.labels = labels;

        window.{{ $airIndexChart->id }}.update();
    }, 1000);

    setInterval(async function(){
        {{--console.log(window.{{ $temperatureHumidityIndexChart->id }}.data);--}}
        // check if daterange is not empty
        if($('input[name="daterange"]').val() !== ''){
            return;
        }
        const labels = window.{{ $temperatureHumidityIndexChart->id }}.data.labels;
                    const oldSuhuData = window.{{ $temperatureHumidityIndexChart->id
}}.data.datasets[0].data;
                    const oldLembabData = window.{{ $temperatureHumidityIndexChart->id
}}.data.datasets[1].data;

        // oldData.push(Math.floor(Math.random() * 100));

        await getLatestTemperatureHumidityIndexData(oldSuhuData, oldLembabData);

        // update labels to time
        const now = new Date();
        const time = now.getHours() + ":" + now.getMinutes() + ":" + now.getSeconds();
        labels.push(time);

        oldSuhuData.shift();
        oldLembabData.shift();
        labels.shift();

        window.{{ $temperatureHumidityIndexChart->id }}.data.datasets[0].data = oldSuhuData
        window.{{ $temperatureHumidityIndexChart->id }}.data.datasets[1].data = oldLembabData
        window.{{ $temperatureHumidityIndexChart->id }}.data.labels = labels;

        window.{{ $temperatureHumidityIndexChart->id }}.update();
    }, 1000);
```

```
    </script>
@endpush
```

API*Controller*

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use App\Models\User;
use App\Models\AirIndex;
use App\Models\TemperatureHumidityIndex;
use DB;

class ApiController extends Controller
{


    public function addData(Request $request)
    {

        if($request->ppm <= 50){
            $ket = 'Baik';
        }elseif($request->ppm >= 51 and $request->ppm <= 100){
            $ket = 'Sedang';
        }elseif($request->ppm >= 101 and $request->ppm <= 199){
            $ket = 'Tidak Sehat';
        }elseif($request->ppm >= 200 and $request->ppm <= 299){
            $ket = 'Sangat Tidak Sehat';
        }else{
            $ket = 'Berbahaya';
        }

        $us = new TemperatureHumidityIndex;
        $us->nilai_suhu      = $request->temperature;
        $us->nilai_lembab  = $request->humidity;
        $us->save();

        if ($us->save()) {

            $as = new AirIndex;
            $as->nilai_udara = $request->ppm;
            $as->status = $ket;
            $as->save();

            return returnAPI(200, 'Success');
        }
        else {
            return returnAPI(200, 'Failed');
        }

    }
}
```

FrontController

```
<?php

namespace App\Http\Controllers;

use App\Charts\AirIndexChart;
use App\Charts\TemperatureHumidityIndexChart;
use App\Models\AirIndex;
use App\Models\TemperatureHumidityIndex;
use Carbon\Carbon;
use Flowframe\Trend\Trend;
use Flowframe\Trend\TrendValue;
use http\Client;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use Seblhaire\DateRangePickerHelper\DateRangePickerHelper;
use Yajra\DataTables\Facades\DataTables;

class FrontController extends Controller
{
    /**
     * Handle the incoming request.
     */
    public function __invoke(Request $request)
    {
        $jakartaAirIndexUrl = 'https://lingkunganhidup.jakarta.go.id/getudara';
        // ajax request
        $jakartaAirIndexResponse = Http::get($jakartaAirIndexUrl);
        $jakartaAirIndexHtml = $jakartaAirIndexResponse->body();

        $trendAirIndex = Trend::model(AirIndex::class)
            ->between(
                start: $request->input('start_date') ?? now()->subMonth()->startOfMonth(),
                end: $request->input('end_date') ?? now()->subMonth()->endOfMonth()
            )
            ->perDay()
            ->average('nilai_udara');

        $airIndexChart = new AirIndexChart();
        $airIndexChart->labels($trendAirIndex->map(fn(TrendValue $value) => $value->date));
        $airIndexChart->dataset('Nilai Udara', 'line', $trendAirIndex->map(fn(TrendValue $value) =>
$value->aggregate));
        $airIndexChart->load(route('air-index-chart'));

        $trendTemperatureHumidityIndex = Trend::model(TemperatureHumidityIndex::class)
            ->between(
                    start: $request->input('start_date') ?  \Illuminate\Support\Carbon::parse($request-
>input('start_date')) : now()->subMonth()->startOfMonth(),
                     end: $request->input('end_date') ?  \Illuminate\Support\Carbon::parse($request-
>input('end_date')) : now()->subMonth()->endOfMonth()
            )
            ->perDay()
            ->average('nilai_suhu');

        $trendTemperatureHumidityIndex2 = Trend::model(TemperatureHumidityIndex::class)
            ->between(
```

```
                    start: $request->input('start_date') ? \Illuminate\Support\Carbon::parse($request->input('start_date')) : now()->subMonth()->startOfMonth(),
                    end: $request->input('end_date') ? \Illuminate\Support\Carbon::parse($request->input('end_date')) : now()->subMonth()->endOfMonth()
                )
                ->perDay()
                ->average('nilai_lembab');

        $temperatureHumidityIndexChart = new TemperatureHumidityIndexChart();
                    $temperatureHumidityIndexChart->labels($trendTemperatureHumidityIndex->map(fn(TrendValue $value) => $value->date));
                    $temperatureHumidityIndexChart->dataset('Nilai Suhu', 'line', $trendTemperatureHumidityIndex->map(fn(TrendValue $value) => $value->aggregate));
                    $temperatureHumidityIndexChart->dataset('Nilai Lembab', 'line', $trendTemperatureHumidityIndex2->map(fn(TrendValue $value) => $value->aggregate));
        $temperatureHumidityIndexChart->load(route('temperature-humidity-index-chart'));
                    return view('landing', compact('jakartaAirIndexHtml', 'airIndexChart', 'temperatureHumidityIndexChart'));

    }

    /**
     * @throws \Exception
     */
    public function airIndex(Request $request)
    {
        $airIndices = AirIndex::query()->latest();

        if($request->has('start_date')) {
            $airIndices->whereDate('created_at', '>=', $request->start_date);
        }

        if($request->has('end_date')) {
            $airIndices->whereDate('created_at', '<=', $request->end_date);
        }

        if($request->has('status')) {
            $airIndices->where('status', $request->status);
        }

        return DataTables::of($airIndices)
            ->addIndexColumn()
            ->make(true);

    }

    public function airIndexStatistic(Request $request)
    {
        $airIndices = AirIndex::query();

        if($request->has('start_date')) {
            $airIndices->whereDate('created_at', '>=', Carbon::parse($request->start_date));
        }

        if($request->has('end_date')) {
            $airIndices->whereDate('created_at', '<=', Carbon::parse($request->end_date));
        }
```

```
        if($request->has('status')) {
            $airIndices->where('status', $request->status);
        }

        $airIndexStatistic = [
            'average' => round($airIndices->clone()->avg('nilai_udara'), 2),
            'max' => round($airIndices->clone()->max('nilai_udara'), 2),
            'min' => round($airIndices->clone()->min('nilai_udara'),2),
        ];

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

    public function airIndexChart(Request $request)
    {
        $trend = Trend::model(AirIndex::class)
            ->between(
                        start: $request->input('start_date') ?\Illuminate\Support\Carbon::parse($request-
>input('start_date')) : now()->subMonth()->startOfMonth(),
                        end: $request->input('end_date') ?  \Illuminate\Support\Carbon::parse($request-
>input('end_date')) : now()->subMonth()->endOfMonth()
            )
            ->perDay()
            ->average('nilai_udara');

        $airIndexChart = new AirIndexChart();
//        $airIndexChart->labels($trend->map(fn(TrendValue $value) => $value->date));
        $airIndexChart->dataset('Nilai Udara', 'line', $trend->map(fn(TrendValue $value) => $value-
>aggregate));

        return $airIndexChart->api();
    }

    public function latestAirIndexData(Request $request)
    {
        $airIndex = AirIndex::query()->latest()->first();

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

    public function temperatureHumidityIndex(Request $request)
    {
        $temperatureHumidityIndices = TemperatureHumidityIndex::query()->latest();

        if($request->has('start_date')) {
            $temperatureHumidityIndices->whereDate('created_at', '>=', $request->start_date);
        }

        if($request->has('end_date')) {
            $temperatureHumidityIndices->whereDate('created_at', '<=', $request->end_date);
        }

        return DataTables::of($temperatureHumidityIndices)
            ->addIndexColumn()
            ->make(true);

    }
```

```
    public function temperatureHumidityIndexStatistic(Request $request)
    {
        $temperatureHumidityIndices = TemperatureHumidityIndex::query();

        if($request->has('start_date')) {
                $temperatureHumidityIndices->whereDate('created_at', '>=', Carbon::parse($request->start_date));
        }

        if($request->has('end_date')) {
                $temperatureHumidityIndices->whereDate('created_at', '<=', Carbon::parse($request->end_date));
        }

        $temperatureHumidityIndexStatistic = [
            'average_suhu' => round($temperatureHumidityIndices->clone()->avg('nilai_suhu'), 2),
          'max_suhu' => round($temperatureHumidityIndices->clone()->max('nilai_suhu'), 2),
          'min_suhu' => round($temperatureHumidityIndices->clone()->min('nilai_suhu'),2),
          'average_lembab' => round($temperatureHumidityIndices->clone()->avg('nilai_lembab'), 2),
          'max_lembab' => round($temperatureHumidityIndices->clone()->max('nilai_lembab'), 2),
          'min_lembab' => round($temperatureHumidityIndices->clone()->min('nilai_lembab'),2),
        ];

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

    public function temperatureHumidityIndexChart(Request $request)
    {
        $trend = Trend::model(TemperatureHumidityIndex::class)
            ->between(
                    start: $request->input('start_date') ?  \Illuminate\Support\Carbon::parse($request->input('start_date')) : now()->subMonth()->startOfMonth(),
                     end: $request->input('end_date') ?  \Illuminate\Support\Carbon::parse($request->input('end_date')) : now()->subMonth()->endOfMonth()
            )
            ->perDay()
            ->average('nilai_suhu');

        $trend2 = Trend::model(TemperatureHumidityIndex::class)
            ->between(
                    start: $request->input('start_date') ?  \Illuminate\Support\Carbon::parse($request->input('start_date')) : now()->subMonth()->startOfMonth(),
                     end: $request->input('end_date') ?  \Illuminate\Support\Carbon::parse($request->input('end_date')) : now()->subMonth()->endOfMonth()
            )
            ->perDay()
            ->average('nilai_lembab');

        $temperatureHumidityIndexChart = new AirIndexChart();

          $temperatureHumidityIndexChart->dataset('Nilai Suhu', 'line', $trend->map(fn(TrendValue $value) => $value->aggregate))->options([
            'color' => '#ff0000',
            'backgroundColor' => 'rgba(255, 99, 132, 0.2)',
        ]);
                    $temperatureHumidityIndexChart->dataset('Nilai Lembab', 'line', $trend2->map(fn(TrendValue $value) => $value->aggregate))->options([
            'color' => '#0000ff',
```

```
            'backgroundColor' => 'rgba(255, 206, 86, 0.2)',
        ]);

        // set color

        return $temperatureHumidityIndexChart->api();
    }

    public function latestTemperatureHumidityIndexData(Request $request)
    {
        $temperatureHumidityIndex = TemperatureHumidityIndex::query()->latest()->first();

        return response()->json($temperatureHumidityIndex);
    }
}
```

UNIVERSITAS DARMA PERSADA

Jl. Taman Malaka Selatan, Pondok Kelapa, Jakarta Timur, Indonesia 13450
Telp. (021) 8649051, 8649053, 8649057 Fax. (021) 8649052
E-mail : humas@unsada.ac.id Home page : http//www.unsada.ac.id

LEMBAR REVISI - SIDANG SKRIPSI

NIM/Nama : 2018230091 Bambang Gunarto
Fakultas/Prodi : Teknik / Teknologi Informasi

No.	Keterangan Revisi	Dosen
1.	Skema/peletakan alat dirangka / urutan ditulis di bab 4.	By Adam.
	Buat/tampilkan uji di lapangan selama 3 hari 24 jam, hasilnya tampilkan dalam tabel. Perbaiki laporan, jenis font	Yan Sofyan

Mengetahui,

Ka Prodi Teknologi Informasi

Herianto, S.Pd., MT.

UNIVERSITAS DARMA PERSADA
UPT PERPUSTAKAAN
Gedung Rektorat Lantai 3,
Jl.Taman Malaka Selatan, Pondok Kelapa – Jakarta Timur 13450

SURAT KETERANGAN
HASIL PENGECEKAN TURNITIN

UPT Perpustakaan Universitas Darma Persada menerangkan telah selesai melakukan pemeriksaan duplikasi/*similarity* menggunakan perangkat lunak Turnitin terhadap hasil karya sebagai berikut:

Judul : Perancangan Monitoring Kualitas Udara Berbasis Internet Of Things Pada Unit Pengelolaan Pengujian Kendaraan Bermotor Pulogadung

Penulis : Bambang Gunarto

NIM : 2018230091

Tgl pemeriksaan : 29 Januari 2024

Dengan hasil Tingkat Kesamaan (*similarity index*) **21%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 29 Januari 2024

Ka.UPT Perpustakaan Unsada

Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi

Batas maksimal similarity 25% untuk Fakultas Teknik, Kelautan dan Pasca Sarjana