

LAMPIRAN
Lampiran 1 Surat Izin Penelitian



UNUGHA CILACAP
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER (FMIKOM)

Keputusan Kemendikbud RI Nomor : 264/E/O/2014 Tanggal 23 Juli 2014

Nomor : B/242/Ybk.041.10/PT/2025	Cilacap, 03 Juni 2025
Lampiran : -	
Hal : Permohonan Observasi Skripsi	

Yth.
Kepala BMKG Cilacap
di -
Tempat

Assalamu'alaikum Wr.Wb.

Salam silaturahmi dan sejahtera kami sampaikan semoga kita senantiasa mendapatkan ridlo dan pertolongan dari Allah SWT dalam menjalankan aktivitas sehari-hari. Amin.

Sehubungan dengan tuntutan kebutuhan mahasiswa untuk mendapatkan pengalaman nyata di lapangan, maka Fakultas Matematika dan Ilmu Komputer Universitas Nahdlatul Ulama Al Ghazali Cilacap menugaskan kepada mahasiswa kami:

Nama	: Achmad Ulul Azmi Wafiqi
NIM	: 212221004
Prodi	: Informatika

Untuk mengadakan riset terkait dengan skripsi yang sedang di kerjakan dengan Judul "ANALISIS PERBANDINGAN TEKNIK WEB SCRAPING BEATIFULSOUP DAN LXML PADA APLIKASI CONVEXVIEW BERBASIS WEB".

Berkenaan dengan hal tersebut, maka kami mengajukan permohonan kepada Kepala BMKG Cilacap untuk mengizinkan mahasiswa kami melaksanakan kegiatan tersebut.

Demikian surat ini kami sampaikan, atas bimbingan dan kerjasamanya disampaikan terimakasih.

Wassalamu'alaikum Wr.Wb.



Mochamad T.A. Aziz Zein, M.Kom
NIK. 41 230714 020

UNIVERSITAS NAHDLATUL 'ULAMA AL GHAZALI CILACAP

Jln. Kemerdekaan Barat No.17 Kesugihan Cilacap Jawa Tengah K.Pos 53274, [http:// unugha.ac.id](http://unugha.ac.id),
Email : fmikom@unugha.ac.id Telp. : (0282) 695415, 695407, Fax : (0282) 695407

Lampiran 2 Link Design Figma

<https://www.figma.com/design/n8HTOiwRfOmsK5qf1j7Lib/Untitled?node-id=0-1&p=f&t=IT4d0QqWFG9CsxW-0>

Lampiran 3 Link Github

<https://github.com/azmiwafiqi12/ConvexView>

Lampiran 4 Kode Pengujian *BeautifulSoup*

```
import schedule
import time
from datetime import datetime
import requests
import os
from bs4 import BeautifulSoup
import re
from PIL import Image
from io import BytesIO
import psutil
import speedtest

url = 'https://www.bmkg.go.id/cuaca/satelit/himawari-ir-enhanced/25'
output_dir = r'E:\CloudIdentification\backend\assets\hasilbs4'

percobaan_ke = 1
maksimal_percobaan = 20

def sanitize_filename(filename):
    return re.sub(r'[\<>:"/\|?*]', '_', filename)

def ukur_kecepatan_internet():
    print("Mengukur kecepatan internet...")
    st = speedtest.Speedtest()
    st.get_best_server()

    download_speed = st.download() / 1_000_000 # Mbps
    upload_speed = st.upload() / 1_000_000 # Mbps
    ping = st.results.ping # ms

    print(f"Ping: {ping:.2f} ms")
    print(f"Download: {download_speed:.2f} Mbps")
    print(f"Upload: {upload_speed:.2f} Mbps\n")

    return ping, download_speed, upload_speed

def scrape_images():
    global percobaan_ke

    if percobaan_ke > maksimal_percobaan:
        print("Seluruh percobaan selesai.")
        exit()

    start_time = time.time()
    process = psutil.Process(os.getpid())

    try:
        print(f"[{datetime.now()}] Memulai Scraping BeautifulSoup percobaan {percobaan_ke}...")
```

```

ping, download_speed, upload_speed =
ukur_kecepatan_internet()

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124
Safari/537.36"
    }
    req = requests.get(url, headers=headers)
    soup = BeautifulSoup(req.content, 'HTML.parser')

    img_links = soup.find_all('a', href=lambda href: href and
'IMAGE/HIMA' in href)
    base_url = 'https://www.bmkg.go.id'

    if not img_links:
        print("Tidak ditemukan gambar dengan pola IMAGE/HIMA")
        return

    os.makedirs(output_dir, exist_ok=True)

    for img in img_links:
        link = img['href']
        if not link.startswith('http'):
            link = base_url + link

        original_filename = sanitize_filename(link.split('/')[1])

        base_name = original_filename.rsplit('.', 1)[0]
        renamed_filename =
f"percobaan_Bs4{percobaan_ke}_{base_name}.png"

        response = requests.get(link, stream=True)
        if response.status_code == 200:
            image = Image.open(BytesIO(response.content))
            image = image.convert("RGB")
            save_path = os.path.join(output_dir, renamed_filename)
            image.save(save_path, format='PNG')
            print(f"Gambar disimpan: {save_path}")

        else:
            print(f"Gagal mengunduh: {link}")

    except Exception as e:
        print(f"Terjadi kesalahan: {e}")
    finally:
        end_time = time.time()
        duration = end_time - start_time
        mem_usage = process.memory_info().rss / (1024 * 1024)

        print(f"Durasi Scraping BeautifulSoup percobaan
{percobaan_ke}: {duration:.2f} detik")
        print(f"Penggunaan memori: {mem_usage:.2f} MB\n")

        log_path = os.path.join(output_dir, "log_percobaanbs.txt")
        is_new_file = not os.path.exists(log_path)

        with open(log_path, "a") as f:
            if is_new_file:
                f.write("Percobaan | Durasi (s) | Memori (MB) | Ping
(ms) | Download (Mbps) | Upload (Mbps)\n")
            f.write(
                f"{percobaan_ke} | {duration:.2f} | {mem_usage:.2f}
| "

```

```

f"{ping:.2f} | {download_speed:.2f} |
{upload_speed:.2f}\n"
    )

    percobaan_ke += 1

# Jadwal Scraping setiap menit ke-7, 17, 27, 37, 47, 57
for minute in [7, 17, 27, 37, 47, 57]:
    schedule.every().hour.at(f"{minute:02}").do(scrape_images)

while True:
    schedule.run_pending()
    time.sleep(1)

```

Lampiran 5 Kode Pengujian *lxml*

```

import schedule
import time
from datetime import datetime
import requests
import os
from lxml import HTML
import re
from PIL import Image
from io import BytesIO
import psutil
import speedtest

url = 'https://www.bmkg.go.id/cuaca/satelit/himawari-ir-
enhanced/25'
output_base_dir = r'E:\CloudIdentification\backend\assets\hasilxml'

percobaan_ke = 1
maksimal_percobaan = 10

def sanitize_filename(filename):
    return re.sub(r'[\<>:"/\|?*]', '_', filename)

def ukur_kecepatan_internet():
    print("Mengukur kecepatan internet...")
    st = speedtest.Speedtest()
    st.get_best_server()

    download_speed = st.download() / 1_000_000 # Mbps
    upload_speed = st.upload() / 1_000_000 # Mbps
    ping = st.results.ping # ms

    print(f"Ping: {ping:.2f} ms")
    print(f"Download: {download_speed:.2f} Mbps")
    print(f"Upload: {upload_speed:.2f} Mbps\n")

    return ping, download_speed, upload_speed

def scrape_images():
    global percobaan_ke

    if percobaan_ke > maksimal_percobaan:
        print("Seluruh percobaan selesai.")
        exit()

    start_time = time.time()

```

```

        process = psutil.Process(os.getpid())
        try:
            print(f"[{datetime.now()}] Memulai Scraping percobaan {percobaan_ke}...")

            ping, download_speed, upload_speed =
ukur_kecepatan_internet()

            headers = {
                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124
Safari/537.36"
            }
            req = requests.get(url, headers=headers)
            if req.status_code != 200:
                print("Gagal mengakses halaman")
                return

            tree = HTML.fromstring(req.content)
            img_links = tree.XPath("//a[contains(@href,
'IMAGE/HIMA')]/@href")
            base_url = 'https://www.bmkg.go.id'

            if not img_links:
                print("Tidak ditemukan link gambar")
                return

            os.makedirs(output_base_dir, exist_ok=True)

            for link in img_links:
                if not link.startswith('http'):
                    link = base_url + link

                # Ambil nama file asli dan ubah
                original_filename = sanitize_filename(link.split('/')[1])

                base_name = original_filename.rsplit('.', 1)[0]
                renamed_filename =
f"percobaan{percobaan_ke}_{base_name}.png"

                response = requests.get(link, stream=True)
                if response.status_code == 200:
                    image = Image.open(BytesIO(response.content))
                    image = image.convert("RGB")
                    save_path = os.path.join(output_base_dir,
renamed_filename)
                    image.save(save_path, format='PNG')
                    print(f"Gambar disimpan: {save_path}")
                else:
                    print(f"Gagal mengunduh gambar: {link}")

            except Exception as e:
                print(f"Terjadi kesalahan: {e}")
            finally:
                end_time = time.time()
                duration = end_time - start_time
                mem_usage = process.memory_info().rss / (1024 * 1024)

                print(f"Durasi Scraping percobaan {percobaan_ke}:
{duration:.2f} detik")
                print(f"Penggunaan memori: {mem_usage:.2f} MB\n")

                # Simpan log ke file
                log_path = os.path.join(output_base_dir,
"log_percobaanXML.txt")

```

```

        is_new_file = not os.path.exists(log_path)

        with open(log_path, "a") as f:
            if is_new_file:
                f.write("Percobaan | Durasi (s) | Memori (MB) | Ping (ms) | Download (Mbps) | Upload (Mbps)\n")
            f.write(
                f"{percobaan_ke} | {duration:.2f} | {mem_usage:.2f} | "
                f"{ping:.2f} | {download_speed:.2f} | {upload_speed:.2f}\n"
            )

            percobaan_ke += 1

# Jadwal Scraping tiap menit ke-7, 17, 27, 37, 47, 57
for minute in [7, 17, 27, 37, 47, 57]:
    schedule.every().hour.at(f":{minute:02}").do(scrape_images)

while True:
    schedule.run_pending()
    time.sleep(1)

```

Lampiran 6 Kode main.py

```

import os
import threading
import time
import schedule
import logging
from FastAPI import FastAPI
from FastAPI.middleware.cors import CORSMiddleware
from FastAPI.responses import FileResponse
from FastAPI.responses import JSONResponse
from FastAPI.staticfiles import StaticFiles
import glob
from . import image_scrap
from . import identifikasinocrop
from . import Kabupaten_crop
import importlib
import sys

sys.path.append(os.path.dirname(__file__))
# FastAPI Init
app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

app.mount("/static", StaticFiles(directory="backend/assets/image"),
name="static")

# Log
logging.basicConfig(
    level=logging.INFO,
    format='[% (asctime)s] %(message)s',
    handlers=[

```

```

        logging.FileHandler("log_pipeline.txt"),
        logging.StreamHandler()
    ]
)

# Pipeline modul kecamatan
kecamatan_modules = [
    "adipala_crop",      "bantarsari_crop",      "binangun_crop",
    "cilacapselatan_crop",
    "cilacaptengah_crop", "cilacaputara_crop", "cimanggu_crop",
    "cipari_crop",
    "dayeuhluhur_crop",   "gandrungmangu_crop", "jeruklegi_crop",
    "kampunglaut_crop",
    "karangpucung_crop",  "kawunganten_crop",  "kedungreja_crop",
    "kesugihan_crop",
    "kroya_crop", "majenang_crop", "maos_crop", "nusawungu_crop",
    "patimuan_crop",
    "sampang_crop", "sidareja_crop", "wanareja_crop"
]

def run_pipeline():
    logging.info("Memulai seluruh proses pipeline...")

    try:
        logging.info(" [1] Scraping citra dari website...")
        image_scrap.scrape_images()

        logging.info(" [2] Identifikasi awan dari gambar hasil
Scraping...")

        file_path =
os.path.abspath("backend/assets/image/H08_EH_Jateng.png")
        if os.path.exists(file_path):
            identifikasinocrop.proses_gambar(file_path)
        else:
            logging.warning(f"File citra tidak ditemukan:
{file_path}")

        logging.info(" [3] Crop wilayah Kabupaten Cilacap...")
        Kabupaten_crop.crop_kabupaten()

        # Looping modul kecamatan
        for i, module_name in enumerate(kecamatan_modules, start=1):
            nama_kecamatan = module_name.replace("_crop",
            "").cAPITalize()
            try:
                mod = importlib.import_module(module_name)
                func_name = f"crop_{module_name.replace('_',
                ''})"

                crop_func = getattr(mod, func_name)
                logging.info(f"[{i:02d}] Crop Kecamatan
{nama_kecamatan}...")
                crop_func()
            except Exception as e:
                logging.error(f"Terjadi kesalahan saat crop
{nama_kecamatan}: {e}")

            except Exception as e:
                logging.error(f"Terjadi kesalahan saat menjalankan pipeline
utama: {e}")

    def scheduler_thread():
        for minute in [7, 17, 27, 37, 47, 57]:
            schedule.every().hour.at(f":{minute:02d}").do(run_pipeline)

```

```

        logging.info("Scheduler aktif... Menunggu waktu eksekusi pertama.")
        while True:
            schedule.run_pending()
            time.sleep(1)

    # Run scheduler
    @app.on_event("startup")
    def start_scheduler():
        threading.Thread(target=scheduler_thread, daemon=True).start()

    # Endpoint FastAPI
    @app.get("/result/image")
    def get_processed_image():
        image_path = os.path.abspath(r"E:\CloudIdentification\backend\assets\image\Kabupaten_final.png")
        if os.path.exists(image_path):
            return FileResponse(image_path, media_type="image/png")
        return {"error": "Gambar tidak ditemukan"}

    @app.get("/result/kecamatan")
    def get_kecamatan_list():
        base_dir = os.path.abspath("backend/assets/image/Kecamatan")
        kecamatan_data = []

        for folder in os.listdir(base_dir):
            folder_path = os.path.join(base_dir, folder)
            if os.path.isdir(folder_path):
                image_file = os.path.join(folder_path, f"{folder}_final.png")
                if os.path.exists(image_file):
                    kecamatan_data.append({
                        "nama": folder.capitalize(),
                        "gambar": f"/static/kecamatan/{folder}/{folder}_final.png"
                    })

        return JsonResponse(content=kecamatan_data)

    @app.get("/status")
    def get_all_kecamatan_status():
        base_dir = os.path.abspath("backend/assets/image/Kecamatan")
        results = {}

        for folder in os.listdir(base_dir):
            folder_path = os.path.join(base_dir, folder)
            if os.path.isdir(folder_path):
                status_file = os.path.join(folder_path, f"{folder}_status.txt")
                if os.path.exists(status_file):
                    with open(status_file, "r", encoding="utf-8") as f:
                        results[folder.replace("_", " ").title()] = f.read().strip()
                else:
                    results[folder] = "Status tidak tersedia"

        return JsonResponse(content=results)

```

Lampiran 7 Kode image_scrap.py

```
import os
import re
import requests
from datetime import datetime
from lxml import HTML
from PIL import Image
from io import BytesIO

url = 'https://www.bmkg.go.id/cuaca/satelit/himawari-ir-enhanced/25'
output_dir = r'E:\CloudIdentification\backend\assets\image'

def sanitize_filename(filename):
    return re.sub(r'[\<>:"/\|?*]', '_', filename)

def scrape_images():
    try:
        print(f"[{datetime.now()}] Proses Scraping Dimulai...")
        headers = {
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
        }
        req = requests.get(url, headers=headers)

        if req.status_code != 200:
            print("Gagal mengambil gambar dari halaman web.")
            return None

        tree = HTML.fromstring(req.content)
        img_links = tree.XPath("//a[contains(@href, 'IMAGE/HIMA')]/@href")
        base_url = 'https://www.bmkg.go.id'

        if not img_links:
            print("Gambar citra tidak ditemukan.")
            return None

        os.makedirs(output_dir, exist_ok=True)

        latest_link = img_links[-1]
        if not latest_link.startswith('http'):
            latest_link = base_url + latest_link

        filename = sanitize_filename(latest_link.split('/')[-1])
        filename = filename.rsplit('.', 1)[0] + '.png'
        save_path = os.path.join(output_dir, filename)

        response = requests.get(latest_link, stream=True)
        if response.status_code == 200:
            image = Image.open(BytesIO(response.content))
            image = image.convert("RGB")
            image.save(save_path, format='PNG')
            print(f"Gambar terbaru disimpan di: {save_path}")
            return save_path
        else:
            print(f"Gagal mengunduh gambar dari: {latest_link}")
            return None

    except Exception as e:
        print(f"Error saat Scraping gambar: {e}")
        return None
```

Kode identifikasinocrop.py

```
import cv2
import numpy as np
import os

def proses_gambar(file_path):
    if not os.path.exists(file_path):
        print(f"Gambar tidak ditemukan: {file_path}")
        return None, None

    img = cv2.imread(file_path)

    if img is None:
        print("Gambar tidak bisa dibaca oleh OpenCV.")
        return None, None

    height, width = img.shape[:2]
    output_img = cv2.cvtColor(img, cv2.COLOR_BGR2BGRA)

    def identifikasi_awan(image):
        hsv_image = cv2.cvtColor(image[:, :, :3], cv2.COLOR_BGR2HSV)
        result_image = np.zeros_like(image)

        for i in range(hsv_image.shape[0]):
            for j in range(hsv_image.shape[1]):
                h, s, v = hsv_image[i, j]

                if (
                    (h == 0 and s == 255 and v == 254) or
                    (h == 0 and s == 165 and v == 250) or
                    (h == 12 and s == 71 and v == 255) or
                    (h == 14 and s == 114 and v == 255)
                ):
                    result_image[i, j] = [0, 0, 255, 255] # Awan Cb
                elif (
                    (h == 19 and s == 255 and v == 255) or
                    (h == 11 and s == 255 and v == 255) or
                    (h == 23 and s == 255 and v == 205)
                ):
                    result_image[i, j] = [0, 128, 255, 255] # Awan
                elif (
                    (h == 44 and s == 255 and v == 255) or
                    (h == 38 and s == 255 and v == 211) or
                    (h == 28 and s == 255 and v == 197)
                ):
                    result_image[i, j] = [0, 255, 255, 255] # Cloud
                elif (
                    (h == 77 and s == 255 and v == 230) or
                    (h == 83 and s == 255 and v == 192) or
                    (h == 103 and s == 188 and v == 255)
                ):
                    result_image[i, j] = [0, 255, 0, 255] # Cloud
                elif (
                    (h == 110 and s == 183 and v == 255) or
                    (h == 109 and s == 181 and v == 180) or
                    (h == 105 and s == 235 and v == 130)
                ):
                    result_image[i, j] = [255, 255, 0, 255] # Cloud
                elif (h == 0 and s == 0 and v == 0):
                    result_image[i, j] = [0, 0, 0, 255] # No Cloud

    Cu = identifikasi_awan(output_img)
    Low = identifikasi_awan(Cu)
    Middle = identifikasi_awan(Low)
    High = identifikasi_awan(Middle)
```

```

        else:
            result_image[i, j] = [255, 255, 255, 255] #
Garis peta/other

        return result_image

    processed_image = identifikasi_awan(output_img)

    output_path_gambar = file_path.replace(".png",
"_klasifikasi.png")
    cv2.imwrite(output_path_gambar, processed_image)
    print(f"Hasil Klasifikasi Tersimpan di: {output_path_gambar}")

    return output_path_gambar, processed_image

```

Lampiran 8 Kode Kabupaten_crop.py

```

import cv2
import numpy as np
from PIL import Image

def crop_kabupaten():

    # === Step 1: Load gambar utama dan mask ===
    img = cv2.imread(r'E:\CloudIdentification\backend\assets\image\H08_EH_Jateng_
klasifikasi.png')
    mask = cv2.imread(r'E:\CloudIdentification\backend\assets\image\map.png',
cv2.IMREAD_GRAYSCALE)

    # Resize mask ke ukuran gambar jika perlu
    if img.shape[:2] != mask.shape[:2]:
        mask = cv2.resize(mask, (img.shape[1], img.shape[0]),
interpolation=cv2.INTER_NEAREST)

    # Binarisasi dan inversi mask
    _, binary_mask = cv2.threshold(mask, 1, 255, cv2.THRESH_BINARY)
    binary_mask = cv2.bitwise_not(binary_mask)

    # Buat latar belakang abu-abu
    gray_background = np.full_like(img, 225)

    # Terapkan masking
    result_img = np.where(binary_mask[:, :, np.newaxis] == 255, img,
gray_background)

    # Temukan dan crop wilayah yang relevan
    coords = cv2.findNonZero(binary_mask)
    x, y, w, h = cv2.boundingRect(coords)
    cropped_result = result_img[y:y+h, x:x+w]

    # Konversi hasil crop ke PIL Image (RGBA)
    base_img = Image.fromarray(cv2.cvtColor(cropped_result,
cv2.COLOR_BGR2RGBA))

    # Resize (perbesar)
    scale_factor = 4.0
    new_size = (int(base_img.width * scale_factor),
int(base_img.height * scale_factor))
    base_img = base_img.resize(new_size, Image.NEAREST)

    # === Step 2: Load dan overlay peta tambahan ===

```

```

        overlay_img = overlay_img.resize(new_size, Image.NEAREST) #
        gunakan NEAREST untuk mask

        # Ubah alpha/transparansi overlay (opsional: 50% transparan)
        overlay_np = np.array(overlay_img)
        overlay_np[:, :, 3] = (overlay_np[:, :, 3] *
1.0).astype(np.uint8) # Bisa ubah ke 0.5 jika ingin 50%
        overlay_img = Image.fromarray(overlay_np)

        # Gabungkan dengan alpha compositing
        combined = Image.alpha_composite(base_img, overlay_img)

        # === Step 3: Simpan dan tampilkan ===
        output_path =
r'E:\CloudIdentification\backend\assets\image\Kabupaten_final.png'
        combined.save(output_path)

pass

```

Lampiran 9 Kode adipala_crop.py

```

import sys
import cv2
import os
import numpy as np
from PIL import Image

sys.path.append(os.path.abspath(os.path.dirname(__file__)))

def crop_adipala():

        img =
cv2.imread(r'E:\CloudIdentification\backend\assets\image\H08_EH_Jateng_
klasifikasi.png')

        mask =
cv2.imread(r'E:\CloudIdentification\backend\assets\image\Kecamatan\Adip
ala\Adipala.png', cv2.IMREAD_GRAYSCALE)

        base_dir =
r'E:\CloudIdentification\backend\assets\image\Kecamatan\Adipala'

        if img.shape[:2] != mask.shape[:2]:
            mask = cv2.resize(mask, (img.shape[1], img.shape[0]),
interpolation=cv2.INTER_NEAREST)

        _, binary_mask = cv2.threshold(mask, 1, 255, cv2.THRESH_BINARY)
        binary_mask = cv2.bitwise_not(binary_mask)

        gray_background = np.full_like(img, 225)
        result_img = np.where(binary_mask[:, :, np.newaxis] == 255, img,
gray_background)

        coords = cv2.findNonZero(binary_mask)
        x, y, w, h = cv2.boundingRect(coords)
        cropped_result = result_img[y:y+h, x:x+w]

        base_img = Image.fromarray(cv2.cvtColor(cropped_result,
cv2.COLOR_BGR2RGBA))

        scale_factor = 10.0

```

```

        new_size = (int(base_img.width * scale_factor),
int(base_img.height * scale_factor))
        base_img = base_img.resize(new_size, Image.NEAREST)
        output_path =
r'E:\CloudIdentification\backend\assets\image\Kecamatan\Adipala\adipala
_final.png'
        base_img.save(output_path)

        cropped_array = np.array(base_img)
        rgb_image = cropped_array[:, :, :3]

        warna_awan = {
            "Terdapat Awan CB": [255, 0, 0],
            "Terdapat Awan CU": [255, 128, 0],
            "Terdapat Awan Rendah": [255, 255, 0],
            "Terdapat Awan Tengah": [0, 255, 0],
            "Terdapat Awan Tinggi": [0, 255, 255],
            "Tidak Ada Awan": [0, 0, 0],
        }

        thresholds = {
            "Terdapat Awan CB": 0.10,
            "Terdapat Awan CU": 0.10,
            "Terdapat Awan Rendah": 0.10,
            "Terdapat Awan Tengah": 0.10,
            "Terdapat Awan Tinggi": 0.10,
        }

        abu_abu_rgb = [225, 225, 225]
        mask_valid = ~np.all(rgb_image == abu_abu_rgb, axis=-1)
        total_pixels = np.count_nonzero(mask_valid)

        terdeteksi = []

        for label, rgb in warna_awan.items():
            mask_awan = np.all(rgb_image == rgb, axis=-1)
            mask_awan_valid = mask_awan & mask_valid
            pixel_count = np.count_nonzero(mask_awan_valid)

            if label == "Tidak Ada Awan":
                if total_pixels > 0 and np.all(mask_awan[mask_valid]):
                    terdeteksi.append(label)
            else:
                if total_pixels > 0:
                    pixel_ratio = pixel_count / total_pixels
                    if pixel_ratio >= thresholds.get(label, 0):
                        terdeteksi.append(label)

        # Simpan status
        if not terdeteksi:
            status = "Awan Tidak Teridentifikasi"
        elif "Tidak Ada Awan" in terdeteksi and len(terdeteksi) == 1:
            status = "Tidak Ada Awan"
        else:
            status = " dan ".join([s.replace("Terdapat ", "") for s in
terdeteksi if s != "Tidak Ada Awan"])
            status = f"Terdapat {status}"

        status_path = os.path.join(base_dir, "adipala_status.txt")
        with open(status_path, "w", encoding="utf-8") as f:
            f.write(status)

        print(f"[STATUS] Adipala → {status}")

```

Lampiran 10 Kode page.jsx

```
"use client";
import React, { useState, useEffect } from "react";
import axios from "axios";

export default function ConvexView() {
  const [search, setSearch] = useState("");
  const [imageUrl, setImageUrl] = useState("");
  const [showAll, setShowAll] = useState(false);
  const [statusData, setStatusData] = useState({});
  const [showSidebar, setShowSidebar] = useState(false);
  const [kecamatanList, setKecamatanList] = useState([]);
  const [error, setError] = useState(null);
  const [highlightedIndex, setHighlightedIndex] = useState(-1);
  const [defaultImageUrl, setDefaultImageUrl] = useState("");
  const [showModal, setShowModal] = useState(false);
  const API_BASE = process.env.NEXT_PUBLIC_API_URL ||
"http://127.0.0.1:8000";
  const resolveImageUrl = (path) =>
    path.startsWith("http") ? path : `http://localhost:8000${path}`;

  useEffect(() => {
    fetch(`${API_BASE}/result/image`)
      .then((res) => {
        if (!res.ok) throw new Error("Gagal mengambil gambar");
        return res.blob();
      })
      .then((blob) => {
        const imgUrl = URL.createObjectURL(blob);
        setImageUrl(imgUrl);
        setDefaultImageUrl(imgUrl);
      })
      .catch((err) => {
        console.error("Gagal fetch gambar:", err);
        setError("Gagal fetch gambar:", err);
      });
  }, []);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const [kecRes, statusRes] = await Promise.all([
          axios.get(`${API_BASE}/result/kecamatan`),
          axios.get(`${API_BASE}/status`),
        ]);
        setKecamatanList(kecRes.data);
        setStatusData(statusRes.data);
      } catch (err) {
        console.error("Gagal memuat data:", err);
      }
    };
    fetchData();
  }, []);

  useEffect(() => {
    const handleClickOutside = (event) => {
      const dropdown = document.getElementById("kecamatan-
dropdown");
      if (dropdown && !dropdown.contains(event.target)) {
        setShowSidebar(false);
      }
    };
  }, []);
```

```

    }
    };
    document.addEventListener("mousedown", handleClickOutside);
    return () => document.removeEventListener("mousedown",
handleClickOutside);
}, []);

useEffect(() => {
    const interval = setInterval(() => {
        const now = new Date();
        const minutes = now.getMinutes();
        const seconds = now.getSeconds();

        const refreshMinutes = [8, 18, 28, 38, 48, 58];
        if (refreshMinutes.includes(minutes) && seconds === 0) {
            window.location.reload();
        }
    }, 1000); // cek setiap detik

    return () => clearInterval(interval);
}, []);

const filtered = kecamatanList.filter((kec) =>
kec.nama.toLowerCase().includes(search.toLowerCase())
);

const Header = () => {
    const [localTime, setLocalTime] = useState("");
    const [utcTime, setUtcTime] = useState("");
    const [hariTanggal, setHariTanggal] = useState("");

    useEffect(() => {
        const updateTime = () => {
            const now = new Date();
            setLocalTime(now.toLocaleTimeString("id-ID", { hour12: false,
timeZone: "Asia/Jakarta" }));
            setUtcTime(now.toLocaleTimeString("en-GB", { hour12: false,
timeZone: "UTC" }));
            setHariTanggal(now.toLocaleDateString("id-ID", {
                weekday: "long", year: "numeric", month: "long", day:
"numeric", timeZone: "Asia/Jakarta"
            }));
        };

        updateTime();
        const interval = setInterval(updateTime, 1000);
        return () => clearInterval(interval);
    }, []);

    return (
        <header className="bg-[#012f5c] text-green-400 font-bold py-2
px-4 lg:px-20 flex justify-between items-center text-xs">
            <div className="text-left ml-25">{hariTanggal}</div>
            <div className="text-right mr-25">Indonesia {localTime} /
{utcTime} UTC</div>
        </header>
    );
};

return (
    <div className="min-h-screen font-sans text-gray-800 bg-
gradient-to-b from-blue-200 to-white overflow-x-hidden">
        <Header />
    </div>
);

```

```

<div className="bg-[#012f5c] text-white flex flex-col sm:flex-
row items-center justify-between px-4 lg:px-20 py-3 gap-2 sm:gap-0 ">
  <div className="ml-25 flex items-center gap-4 w-20">
    
  </div>
  <div className="mr-25 text-sm text-center sm:text-right"
onClick={() => setShowModal(true)}
>TENTANG CONVEXVIEW</div>
</div>

{showModal && (
  <div className="fixed inset-0 bg-[#012f5c] bg-opacity-90 flex
justify-center items-center z-50 flex-col px-4">
    
    <button
      className="absolute top-2 right-4 text-white hover:text-
black text-4xl"
      onClick={() => setShowModal(false)}
    >
      &times;
    </button>
    <h2 className="text-3xl sm:text-5xl font-semibold text-
white mt-8 text-center">
      Tentang ConvexView
    </h2>
    <p className="text-white text-base sm:text-xl mt-6 text-
center max-w-md sm:max-w-2xl">
      ConvexView adalah aplikasi visualisasi dan identifikasi
awan berbasis citra satelit yang digunakan untuk mendeteksi awan
konvektif seperti awan Cumulonimbus (Cb) yang berpotensi menimbulkan
hujan lebat atau cuaca ekstrem di wilayah Cilacap.
    </p>
    <p className="text-white text-xs sm:text-sm mt-6 text-
center">
      - Developer ConvexView 2025 -
    </p>
    <p className="text-white text-xs sm:text-sm mt-2 text-
center">
      - Maamiii If 21 -
    </p>
  </div>
)}

<main className="relative py-8 px-4 sm:px-8 lg:px-20 bg-cover
bg-center bg-no-repeat" style={{ backgroundImage: "url('/bg.jpeg') " }}>
  <div className="absolute inset-0 bg-black opacity-10 z-
0"></div>
  <div className="z-10 relative max-w-screen-xl mx-auto">
    <h1 className="text-2xl md:text-3xl font-bold text-center
text-white mb-5 cursor-pointer">
      Awan Konvektif Kabupaten Cilacap
    </h1>

    <div className="relative w-full max-w-md mx-auto">

{/* Input */}
<input
  type="text"
  placeholder="Cari Kecamatan"
  value={search}
  onFocus={() => setShowSidebar(true)}
  onBlur={() => setTimeout(() => setShowSidebar(false),
200)}

  onChange={(e) => setSearch(e.target.value)}
  onKeyDown={(e) => {
    if (e.key === "Enter") {
      const matched = kecamatanList.find(
        (kec) => kec.nama.toLowerCase() ===
search.toLowerCase()
      );
      if (matched) {
        setImageUrl(resolveImageUrl(matched.gambar));
        setShowSidebar(false);
      } else {
        alert("Kecamatan tidak ditemukan");
      }
    }
  }}
  className="w-full pl-12 pr-12 py-2 rounded-full bg-white
text-black placeholder-gray-500 border border-white shadow
focus:outline-none focus:ring-2 focus:ring-white"
/>

{/* back button */}
{search && (
  <button
    onClick={() => {
      setImageUrl(defaultImageUrl);
      setSearch("");
      setShowSidebar(false);
    }}
    className="absolute right-3 top-1/2 -translate-y-1/2"
  >
    
  </button>
)}
</div>

{/* Sidebar */}
{showSidebar && (
  <div
    id="kecamatan-dropdown"
    className="absolute mt-2 w-full max-w-md bg-white
border border-gray-300 rounded shadow-lg z-20 left-1/2 transform -
translate-x-1/2"
  >
    <ul className="max-h-60 overflow-y-auto">
      {filtered.length > 0 ? (
        filtered.map((item, index) => (
          <li key={item.nama}>
            <button

```

```

        className={`w-full text-left px-4 py-2 text-sm ${
            highlightedIndex === index
                ? "bg-blue-100 text-blue-600 font-semibold"
                : "hover:bg-gray-100"
        }`}
        onMouseDown={() => {
            setImageUrl(resolveImageUrl(item.gambar));
            setSearch(item.nama);
            setShowSidebar(false);
        }}
        >
        {item.nama}
    </button>
</li>
))
) : (
    <li className="px-4 py-2 text-gray-500 text-sm">Tidak ditemukan</li>
    )}
</ul>
</div>
)}

{/* Map & Keterangan */}
<div className="bg-white/10 rounded-xl p-6 shadow-lg backdrop-blur mt-5">
    <h2 className="text-3xl text-center mb-6 font-bold text-white">Peta Awan</h2>
    {error && <div className="text-red-600 text-center mt-4">{error}</div>}
    <div className="flex flex-col lg:flex-row justify-between gap-6">
        <div className="flex-1 flex justify-center">
            {imageUrl ? (
                <img src={imageUrl} alt="Peta Awan" className="w-full max-w-md border-2 border-white rounded-lg shadow-md" onError={() => setError("Gagal memuat gambar awan")} />
            ) : (
                <p className="text-white">Memuat peta...</p>
            )}
        </div>
        <div className="flex-1 text-sm">
            <h2 className="font-semibold mb-4 text-white text-2xl md:text-3xl">Keterangan</h2>
            <ul className="space-y-2 text-white text-lg md:text-xl">
                <li><span className="inline-block w-5 h-5 bg-[rgb(255,0,0)] mr-2 rounded-sm"></span>: Awan Cumulonimbus (Cb)</li>
                <li><span className="inline-block w-5 h-5 bg-[rgb(255,128,0)] mr-2 rounded-sm"></span>: Awan Cumulus (Cu)</li>
                <li><span className="inline-block w-5 h-5 bg-[rgb(255,255,0)] mr-2 rounded-sm"></span>: Awan Rendah (Sc/St)</li>
                <li><span className="inline-block w-5 h-5 bg-[rgb(0,255,0)] mr-2 rounded-sm"></span>: Awan Tengah (Ac/As)</li>
                <li><span className="inline-block w-5 h-5 bg-[rgb(0,255,255)] mr-2 rounded-sm"></span>: Awan Tinggi (Ci/Cc)</li>
                <li><span className="inline-block w-5 h-5 bg-black mr-2 rounded-sm"></span>: Tidak Ada Awan</li>
            </ul>
        </div>
    </div>
</div>

```

```

<div className="flex justify-center mt-6">
  <button
    className="bg-blue-600 text-white px-6 py-2 rounded-
md hover:bg-blue-700 transition"
    onClick={() => {
      if (imageUrl) {
        const link = document.createElement("a");
        link.href = imageUrl;
        link.download = `ConvexView_${search} ||
"Kabupaten_Cilacap".png`;
        document.body.appendChild(link);
        link.click();
        document.body.removeChild(link)
      }
    }}
  >
    Download
  </button>
</div>

<div className="flex items-center text-center text-xs
text-gray-200 mt-2 relative z-10">
  
  <p className="text-black flex flex-col items-start
text-xs sm:text-sm font-bold">
    Sumber: Citra Himawari-9 IR Enhanced
    <span>Badan Meteorologi, Klimatologi, dan
Geofisika</span>
  </p>
</div>
</div>

{/* Status Kecamatan */}
<div className="grid grid-cols-2 sm:grid-cols-3 md:grid-
cols-4 lg:grid-cols-6 gap-4 mt-8">
  {(showAll ? filtered : filtered.slice(0, 12)).map((kec)
=> (
    <div key={kec.nama} className="bg-white rounded-lg
shadow-md p-4 text-center border border-blue-200">
      <p className="font-bold text-blue-800 text-base
md:text-lg">{kec.nama}</p>
      <p className="text-sm text-gray-700 mt-1">
        {statusData[kec.nama] || "Memuat status..."}
      </p>
    </div>
  )))
</div>

{filtered.length > 12 && (
  <div className="text-center mt-4">
    <button
      onClick={() => setShowAll(!showAll)}
      className="px-4 py-2 bg-blue-500 text-white rounded
hover:bg-blue-600 transition"
    >
      {showAll ? "Sembunyikan" : "Lihat Semua"}
    </button>
  </div>
)}
</div>
</main>

<footer className="w-full text-center text-xs text-[#8C8C8C]
py-4">

```

```

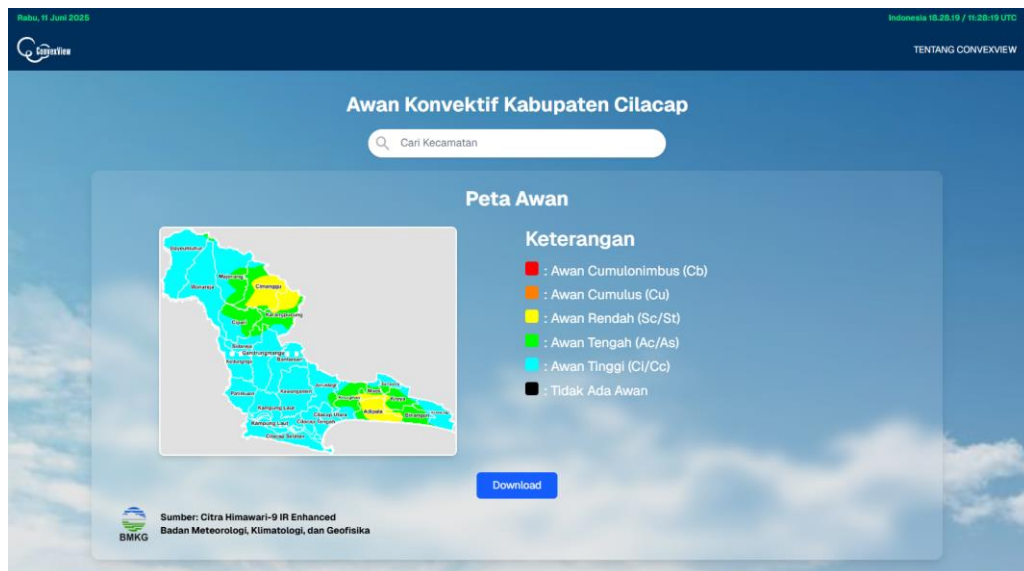
&copy; 2025 ConvexView
</footer>
</div>
);
}

```

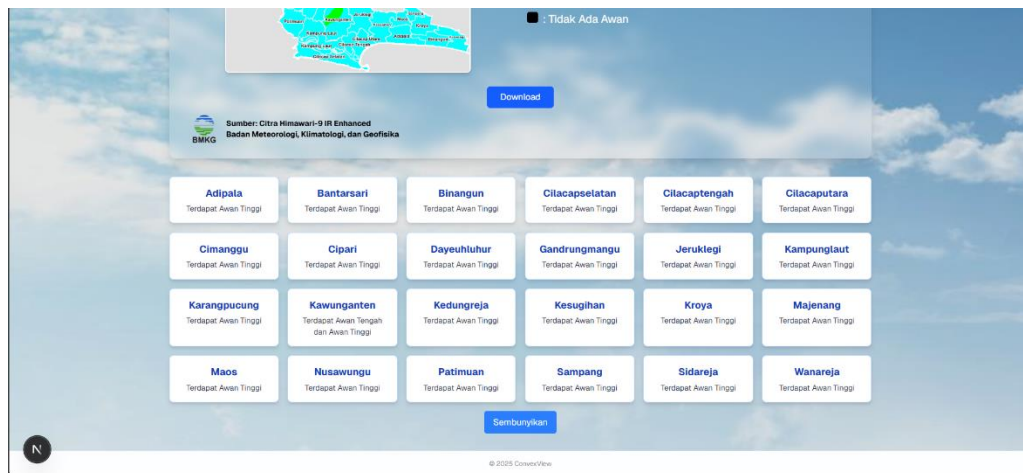
Lampiran 11 Logo *ConvexView*



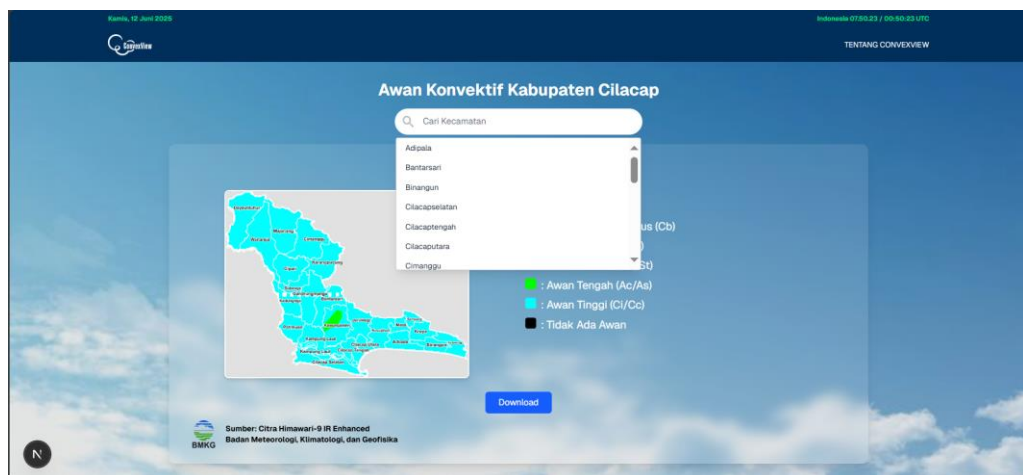
Lampiran 12 Tampilan Web *ConvexView*



Lampiran 13 Tampilan Card Kecamatan



Lampiran 14 Tampilan Cari Kecamatan



Lampiran 15 Tampilan Kecamatan

