LAMPIRAN

1. Source code menjalankan program

```
# Clone GFPGAN and enter the GFPGAN folder
%cd /content
!rm -rf GFPGAN
!git clone https://github.com/TencentARC/GFPGAN.git
%cd GFPGAN
!pip install basicsr
!pip install facexlib
!pip install -r requirements.txt
!python setup.py develop
!pip install realesrgan
!wget
https://github.com/TencentARC/GFPGAN/releases/download/v1.3.0/GFPGANv1.3.pth -
P experiments/pretrained_models

# upload citra
import os
from google.colab import files
import shutil
upload_folder = 'inputs/upload'
if os.path.isdir(upload_folder):
    shutil.rmtree(upload_folder)
os.mkdir(upload_folder)

# upload images
uploaded = files.upload()
for filename in uploaded.keys():
  dst_path = os.path.join(upload_folder, filename)
  print(f'move {filename} to {dst_path}')
  shutil.move(filename, dst_path)

# menghubungkan ke drive anda
```

```
from google.colab import drive
drive.mount('/content/drive')

# rekontruksi citra
!rm -rf results
!python inference_gfpgan.py -i inputs/ridha -o results -v 1.3 -s 2 --bg_upsampler
realesrgan
!ls results/cmp


# cropped faces
import cv2
import matplotlib.pyplot as plt
def display(img1, img2):
  fig = plt.figure(figsize=(25, 10))
  ax1 = fig.add_subplot(1, 2, 1)
  plt.title('Input image', fontsize=16)
  ax1.axis('off')
  ax2 = fig.add_subplot(1, 2, 2)
  plt.title('GFPGAN output', fontsize=16)
  ax2.axis('off')
  ax1.imshow(img1)
  ax2.imshow(img2)
def imread(img_path):
  img = cv2.imread(img_path)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  return img

# upload folder
import os
import glob

input_folder = 'results/cropped_faces'
result_folder = 'results/restored_faces'
input_list = sorted(glob.glob(os.path.join(input_folder, '*')))
output_list = sorted(glob.glob(os.path.join(result_folder, '*')))
```

```
for input_path, output_path in zip(input_list, output_list):
  img_input = imread(input_path)
  img_output = imread(output_path)
  display(img_input, img_output)

# visual seluruh citra (termasuk lingkungan sekitar (environtment))
import cv2
import matplotlib.pyplot as plt
def display(img1, img2):
  fig = plt.figure(figsize=(25, 10))
  ax1 = fig.add_subplot(1, 2, 1)
  plt.title('Input image', fontsize=16)
  ax1.axis('off')
  ax2 = fig.add_subplot(1, 2, 2)
  plt.title('GFPGAN output', fontsize=16)
  ax2.axis('off')
  ax1.imshow(img1)
  ax2.imshow(img2)
def imread(img_path):
  img = cv2.imread(img_path)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  return img

# display each image in the upload folder
import os
import glob

input_folder = 'inputs/upload'
result_folder = 'results/restored_imgs'
input_list = sorted(glob.glob(os.path.join(input_folder, '*')))
output_list = sorted(glob.glob(os.path.join(result_folder, '*')))
for input_path, output_path in zip(input_list, output_list):
  img_input = imread(input_path)
  img_output = imread(output_path)
  display(img_input, img_output)
```

```
# download hasil
!ls results
print('Download results')
os.system('zip -r download.zip results')
files.download("download.zip")
```

2. Source code Inference_gfpgan

   Sebelum menjalankan program pastikan memasukan souce kode ini pada folder GFPGAN

```
import argparse
import cv2
import glob
import numpy as np
import os
import torch
from basicsr.utils import imwrite

from gfpgan import GFPGANer


def main():
    """Inference demo for GFPGAN (for users).
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '-i',
        '--input',
        type=str,
        default='inputs/whole_imgs',
        help='Input image or folder. Default: inputs/whole_imgs')
    parser.add_argument('-o', '--output', type=str, default='results', help='Output folder.
Default: results')
```

```
    # we use version to select models, which is more user-friendly
    parser.add_argument(
        '-v', '--version', type=str, default='1.3', help='GFPGAN model version. Option: 1 | 1.2
| 1.3. Default: 1.3')
    parser.add_argument(
        '-s', '--upscale', type=int, default=2, help='The final upsampling scale of the image.
Default: 2')

    parser.add_argument(
        '--bg_upsampler', type=str, default='realesrgan', help='background upsampler.
Default: realesrgan')
    parser.add_argument(
        '--bg_tile',
        type=int,
        default=400,
        help='Tile size for background sampler, 0 for no tile during testing. Default: 400')
    parser.add_argument('--suffix', type=str, default=None, help='Suffix of the restored
faces')
    parser.add_argument('--only_center_face', action='store_true', help='Only restore the
center face')
    parser.add_argument('--aligned', action='store_true', help='Input are aligned faces')
    parser.add_argument(
        '--ext',
        type=str,
        default='auto',
        help='Image extension. Options: auto | jpg | png, auto means using the same
extension as inputs. Default: auto')
    parser.add_argument('-w', '--weight', type=float, default=0.5, help='Adjustable
weights.')
    args = parser.parse_args()

    args = parser.parse_args()

    # ------------------------ input & output ------------------------
    if args.input.endswith('/'):
        args.input = args.input[:-1]
    if os.path.isfile(args.input):
        img_list = [args.input]
    else:
        img_list = sorted(glob.glob(os.path.join(args.input, '*')))

    os.makedirs(args.output, exist_ok=True)

    # ------------------------ set up background upsampler ------------------------
    if args.bg_upsampler == 'realesrgan':
        if not torch.cuda.is_available():  # CPU
            import warnings
            warnings.warn('The unoptimized RealESRGAN is slow on CPU. We do not use
it. '
                          'If you really want to use it, please modify the corresponding codes.')
            bg_upsampler = None
        else:
            from basicsr.archs.rrdbnet_arch import RRDBNet
            from realesrgan import RealESRGANer
            model = RRDBNet(num_in_ch=3, num_out_ch=3, num_feat=64, num_block=23,
num_grow_ch=32, scale=2)
            bg_upsampler = RealESRGANer(
                scale=2,
                model_path='https://github.com/xinntao/Real-
ESRGAN/releases/download/v0.2.1/RealESRGAN_x2plus.pth',
                model=model,
                tile=args.bg_tile,
                tile_pad=10,
                pre_pad=0,
                half=True)  # need to set False in CPU mode
    else:
        bg_upsampler = None

    # ------------------------ set up GFPGAN restorer ------------------------
    if args.version == '1':
        arch = 'original'
        channel_multiplier = 1
        model_name = 'GFPGANv1'
        url =
'https://github.com/TencentARC/GFPGAN/releases/download/v0.1.0/GFPGANv1.pth'
    elif args.version == '1.2':
        arch = 'clean'
        channel_multiplier = 2
        model_name = 'GFPGANCleanv1-NoCE-C2'
        url =
'https://github.com/TencentARC/GFPGAN/releases/download/v0.2.0/GFPGANCleanv1-
NoCE-C2.pth'
    elif args.version == '1.3':
        arch = 'clean'
        channel_multiplier = 2
        model_name = 'GFPGANv1.3'
        url =
'https://github.com/TencentARC/GFPGAN/releases/download/v1.3.0/GFPGANv1.3.pth'
    elif args.version == '1.4':
        arch = 'clean'
        channel_multiplier = 2
        model_name = 'GFPGANv1.4'
        url =
'https://github.com/TencentARC/GFPGAN/releases/download/v1.3.0/GFPGANv1.4.pth'
    elif args.version == 'RestoreFormer':
        arch = 'RestoreFormer'
        channel_multiplier = 2
        model_name = 'RestoreFormer'
        url =
'https://github.com/TencentARC/GFPGAN/releases/download/v1.3.4/RestoreFormer.pth'
    else:
        raise ValueError(f'Wrong model version {args.version}.')

    # determine model paths
    model_path = os.path.join('experiments/pretrained_models', model_name + '.pth')
    if not os.path.isfile(model_path):
        model_path = os.path.join('gfpgan/weights', model_name + '.pth')
    if not os.path.isfile(model_path):
        # download pre-trained models from url
        model_path = url

    restorer = GFPGANer(
        model_path=model_path,
        upscale=args.upscale,
        arch=arch,
        channel_multiplier=channel_multiplier,
        bg_upsampler=bg_upsampler)

    # ------------------------ restore ------------------------
    for img_path in img_list:
        # read image
        img_name = os.path.basename(img_path)
        print(f'Processing {img_name} ...')
        basename, ext = os.path.splitext(img_name)
        input_img = cv2.imread(img_path, cv2.IMREAD_COLOR)

        # restore faces and background if necessary
        cropped_faces, restored_faces, restored_img = restorer.enhance(
            input_img,
            has_aligned=args.aligned,
            only_center_face=args.only_center_face,
            paste_back=True,
            weight=args.weight)

        # save faces
        for idx, (cropped_face, restored_face) in enumerate(zip(cropped_faces,
restored_faces)):
            # save cropped face
            save_crop_path = os.path.join(args.output, 'cropped_faces',
f'{basename}_{idx:02d}.png')
```

```
            imwrite(cropped_face, save_crop_path)
            # save restored face
            if args.suffix is not None:
                save_face_name = f'{basename}_{idx:02d}_{args.suffix}.png'
            else:
                save_face_name = f'{basename}_{idx:02d}.png'
            save_restore_path = os.path.join(args.output, 'restored_faces', save_face_name)
            imwrite(restored_face, save_restore_path)
            # save comparison image
            cmp_img = np.concatenate((cropped_face, restored_face), axis=1)
            imwrite(cmp_img, os.path.join(args.output, 'cmp', f'{basename}_{idx:02d}.png'))

        # save restored img
        if restored_img is not None:
            if args.ext == 'auto':
                extension = ext[1:]
            else:
                extension = args.ext
            if args.suffix is not None:
                save_restore_path = os.path.join(args.output, 'restored_imgs',
f'{basename}_{args.suffix}.{extension}')
            else:
                save_restore_path = os.path.join(args.output, 'restored_imgs',
f'{basename}.{extension}')
            imwrite(restored_img, save_restore_path)
    print(f'Results are in the [{args.output}] folder.')
if __name__ == '__main__':
    main()
```

3. Souce code penghitungan PSNR dan MSE

```
import os
from google.colab import files
import shutil
import cv2
import matplotlib.pyplot as plt
import numpy as np
import glob
import time
from skimage.metrics import structural_similarity as ssim

# Menonaktifkan GPU
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"

# Clone Real-ESRGAN and enter the Real-ESRGAN
!git clone https://github.com/xinntao/Real-ESRGAN.git
%cd Real-ESRGAN
# Set up the environment
!pip install basicsr
!pip install facexlib
!pip install gfpgan
!pip install -r requirements.txt
!python setup.py develop

# Set default PyTorch data type to float
import torch
torch.set_default_dtype(torch.float32)

upload_folder = 'upload'
result_folder = 'results'

if os.path.isdir(upload_folder):
    shutil.rmtree(upload_folder)
if os.path.isdir(result_folder):
    shutil.rmtree(result_folder)
os.mkdir(upload_folder)
os.mkdir(result_folder)

# upload images
uploaded = files.upload()
```

```
for filename in uploaded.keys():
    dst_path = os.path.join(upload_folder, filename)
    print(f'move {filename} to {dst_path}')
    shutil.move(filename, dst_path)

# if it is out of memory, try to use the `--tile` option
# We upsample the image with the scale factor X3.5
#!python inference_realesrgan.py -n RealESRGAN_x4plus -i upload --outscale 3.5 --
face_enhance
!python inference_realesrgan.py -n RealESRGAN_x4plus -i upload --fp32 --outscale 3.5
--face_enhance

# Function to calculate PSNR
def calculate_psnr(img1, img2):
    img1_resized = cv2.resize(img1, (img2.shape[1], img2.shape[0]))
    mse = np.mean((img1_resized - img2) ** 2)
    if mse == 0:
        return float('inf')
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr

# Function to calculate MSE (with normalization)
def calculate_mse(img1, img2):
    img1_resized = cv2.resize(img1.astype(np.float32) / 255.0, (img2.shape[1],
img2.shape[0]))  # Normalisasi
    img2 = img2.astype(np.float32) / 255.0  # Normalisasi
    mse = np.mean((img1_resized - img2) ** 2)
    return mse

# Function to process a single image pair
def process_image_pair(input_path, output_path):
    img_input = cv2.imread(input_path)
    img_output = cv2.imread(output_path)
    psnr_value = calculate_psnr(img_input, img_output)
    mse_value = calculate_mse(img_input, img_output)
    return psnr_value, mse_value

# Lists to store PSNR and MSE values
psnr_values = []
mse_values = []
start_times = []  # Menambahkan definisi start_times

# Process each image pair
input_list = sorted(glob.glob(os.path.join(upload_folder, '*')))
output_list = sorted(glob.glob(os.path.join(result_folder, '*')))

for input_path, output_path in zip(input_list, output_list):
    start_time = time.time()  # Menambahkan perhitungan waktu awal

    # Process the image pair
    psnr, mse = process_image_pair(input_path, output_path)

    # Append PSNR and MSE values to the lists
    psnr_values.append(psnr)
    mse_values.append(mse)
    start_times.append(start_time)  # Menambahkan waktu awal ke start_times

    # Calculate time taken for processing
    elapsed_time = time.time() - start_time
    print(f"Image: {input_path}, PSNR: {psnr:.2f}, MSE: {mse:.4f}, Time: 
{elapsed_time:.2f} seconds")

# Create a bar chart for PSNR values
plt.figure(figsize=(10, 5))
plt.bar(range(len(psnr_values)), psnr_values, color='blue')
plt.xlabel('Image Index')
plt.ylabel('PSNR Value')
plt.title('PSNR Values for Each Image')
plt.show()
```

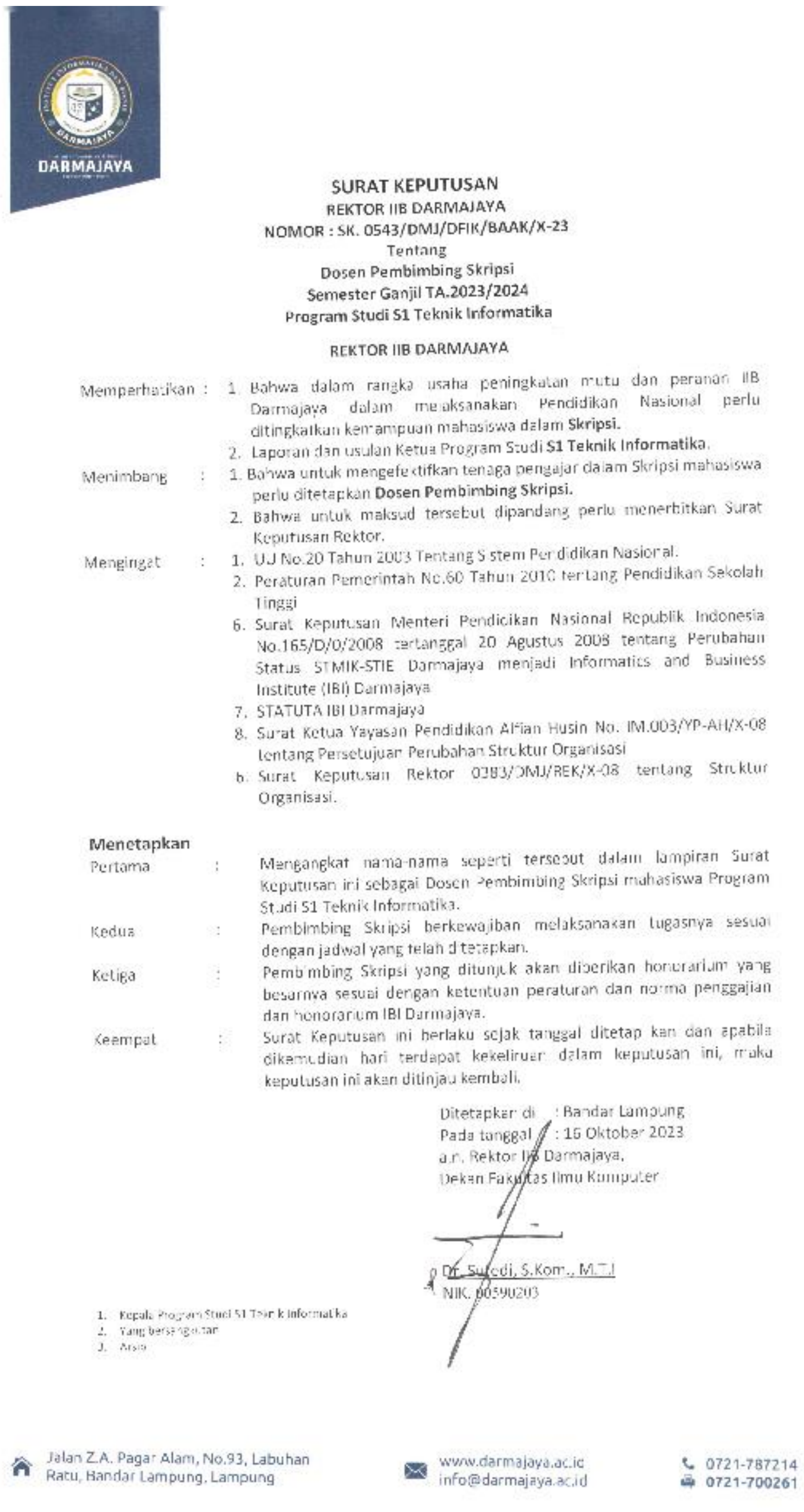
```
# Create a bar chart for MSE values
plt.figure(figsize=(10, 5))
plt.bar(range(len(mse_values)), mse_values, color='red')
plt.xlabel('Image Index')
plt.ylabel('MSE Value')
plt.title('MSE Values for Each Image')
plt.show()

# Variables for calculating averages
total_psnr = sum(psnr_values)
total_mse = sum(mse_values)
total_time = sum([time.time() - start_time for start_time in start_times])

# Calculate averages
average_psnr = total_psnr / len(input_list)
average_mse = total_mse / len(input_list)
average_time = total_time / len(input_list)

# Print overall results
print(f"\nAverage PSNR: {average_psnr:.2f}, Average MSE: {average_mse:.4f},
Average Time: {average_time:.2f} seconds per image.")
```

4.



DARMAJAYA

SURAT KEPUTUSAN
REKTOR IIB DARMAJAYA
NOMOR : SK. 0543/DMJ/DFIK/BAAK/X-23
Tentang
Dosen Pembimbing Skripsi
Semester Ganjil TA.2023/2024
Program Studi S1 Teknik Informatika

REKTOR IIB DARMAJAYA

Memperhatikan	:	1. Bahwa dalam rangka usaha peningkatan mutu dan peranan IIB Darmajaya dalam melaksanakan Pendidikan Nasional perlu ditingkatkan kemampuan mahasiswa dalam **Skripsi.**
		2. Laporan dan usulan Ketua Program Studi **S1 Teknik Informatika.**
Menimbang	:	1. Bahwa untuk mengefektifkan tenaga pengajar dalam Skripsi mahasiswa perlu ditetapkan **Dosen Pembimbing Skripsi.**
		2. Bahwa untuk maksud tersebut dipandang perlu menerbitkan Surat Keputusan Rektor.
Mengingat	:	1. UU No.20 Tahun 2003 Tentang Sistem Pendidikan Nasional.
		2. Peraturan Pemerintah No.60 Tahun 2010 tentang Pendidikan Sekolah Tinggi
		6. Surat Keputusan Menteri Pendidikan Nasional Republik Indonesia No.165/D/0/2008 tertanggal 20 Agustus 2008 tentang Perubahan Status STMIK-STIE Darmajaya menjadi Informatics and Business Institute (IBI) Darmajaya
		7. STATUTA IBI Darmajaya
		8. Surat Ketua Yayasan Pendidikan Alfian Husin No. IM.003/YP-AH/X-08 tentang Persetujuan Perubahan Struktur Organisasi
		6. Surat Keputusan Rektor 0383/DMJ/REK/X-08 tentang Struktur Organisasi.

Menetapkan

Pertama	:	Mengangkat nama-nama seperti tersebut dalam lampiran Surat Keputusan ini sebagai Dosen Pembimbing Skripsi mahasiswa Program Studi S1 Teknik Informatika.
Kedua	:	Pembimbing Skripsi berkewajiban melaksanakan tugasnya sesuai dengan jadwal yang telah ditetapkan.
Ketiga	:	Pembimbing Skripsi yang ditunjuk akan diberikan honorarium yang besarnya sesuai dengan ketentuan peraturan dan norma penggajian dan honorarium IBI Darmajaya.
Keempat	:	Surat Keputusan ini berlaku sejak tanggal ditetap kan dan apabila dikemudian hari terdapat kekeliruan dalam keputusan ini, maka keputusan ini akan ditinjau kembali.

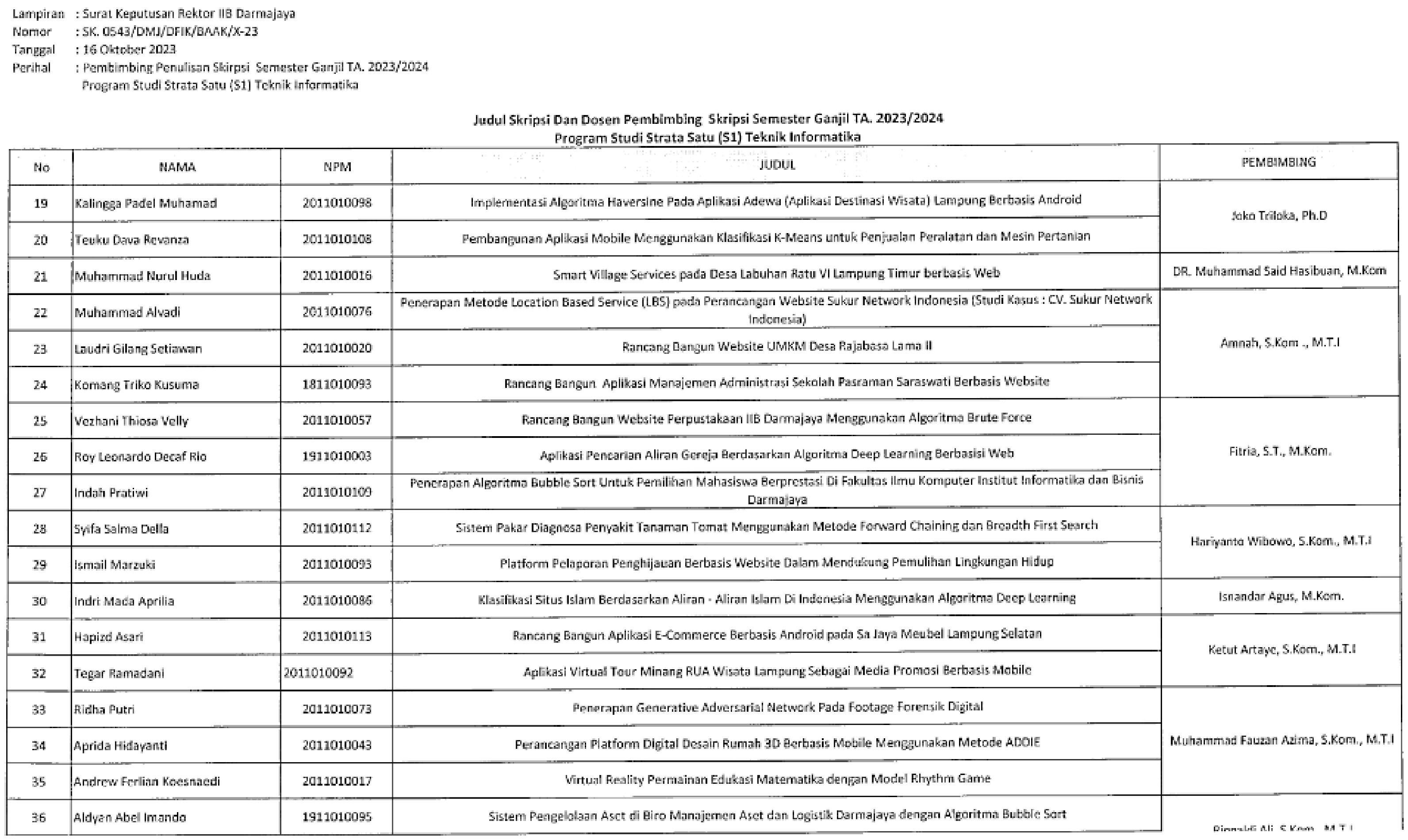
Ditetapkan di : Bandar Lampung
Pada tanggal : 16 Oktober 2023
a.n. Rektor IIB Darmajaya,
Dekan Fakultas Ilmu Komputer

[illegible]

Dr. Sutedi, S.Kom., M.T.I
NIK. 00590203

1. Kepala Program Studi S1 Teknik Informatika
2. Yang bersangkutan
3. Arsip

Jalan Z.A. Pagar Alam, No.93, Labuhan Ratu, Bandar Lampung, Lampung
www.darmajaya.ac.id
info@darmajaya.ac.id
0721-787214
0721-700261



Lampiran : Surat Keputusan Rektor IIB Darmajaya
Nomor : SK. 0543/DMJ/DFIK/BAAK/X-23
Tanggal : 16 Oktober 2023
Perihal : Pembimbing Penulisan Skirpsi Semester Ganjil TA. 2023/2024
Program Studi Strata Satu (S1) Teknik Informatika

Judul Skripsi Dan Dosen Pembimbing Skripsi Semester Ganjil TA. 2023/2024
Program Studi Strata Satu (S1) Teknik Informatika

No	NAMA	NPM	JUDUL	PEMBIMBING
19	Kalingga Padel Muhamad	2011010098	Implementasi Algoritma Haversine Pada Aplikasi Adewa (Aplikasi Destinasi Wisata) Lampung Berbasis Android	Joko Triloka, Ph.D
20	Teuku Dava Revanza	2011010108	Pembangunan Aplikasi Mobile Menggunakan Klasifikasi K-Means untuk Penjualan Peralatan dan Mesin Pertanian	
21	Muhammad Nurul Huda	2011010016	Smart Village Services pada Desa Labuhan Ratu VI Lampung Timur berbasis Web	DR. Muhammad Said Hasibuan, M.Kom
22	Muhammad Alvadi	2011010076	Penerapan Metode Location Based Service (LBS) pada Perancangan Website Sukur Network Indonesia (Studi Kasus : CV. Sukur Network Indonesia)	Amnah, S.Kom ., M.T.I
23	Laudri Gilang Setiawan	2011010020	Rancang Bangun Website UMKM Desa Rajabasa Lama II	
24	Komang Triko Kusuma	1811010093	Rancang Bangun Aplikasi Manajemen Administrasi Sekolah Pasraman Saraswati Berbasis Website	
25	Vezhani Thiosa Velly	2011010057	Rancang Bangun Website Perpustakaan IIB Darmajaya Menggunakan Algoritma Brute Force	Fitria, S.T., M.Kom.
26	Roy Leonardo Decaf Rio	1911010003	Aplikasi Pencarian Aliran Gereja Berdasarkan Algoritma Deep Learning Berbasisi Web	
27	Indah Pratiwi	2011010109	Penerapan Algoritma Bubble Sort Untuk Pemilihan Mahasiswa Berprestasi Di Fakultas Ilmu Komputer Institut Informatika dan Bisnis Darmajaya	
28	Syifa Salma Della	2011010112	Sistem Pakar Diagnosa Penyakit Tanaman Tomat Menggunakan Metode Forward Chaining dan Breadth First Search	Hariyanto Wibowo, S.Kom., M.T.I
29	Ismail Marzuki	2011010093	Platform Pelaporan Penghijauan Berbasis Website Dalam Mendukung Pemulihan Lingkungan Hidup	
30	Indri Mada Aprilia	2011010086	Klasifikasi Situs Islam Berdasarkan Aliran - Aliran Islam Di Indonesia Menggunakan Algoritma Deep Learning	Isnandar Agus, M.Kom.
31	Hapizd Asari	2011010113	Rancang Bangun Aplikasi E-Commerce Berbasis Android pada Sa Jaya Meubel Lampung Selatan	Ketut Artaye, S.Kom., M.T.I
32	Tegar Ramadani	2011010092	Aplikasi Virtual Tour Minang RUA Wisata Lampung Sebagai Media Promosi Berbasis Mobile	
33	Ridha Putri	2011010073	Penerapan Generative Adversarial Network Pada Footage Forensik Digital	Muhammad Fauzan Azima, S.Kom., M.T.I
34	Aprida Hidayanti	2011010043	Perancangan Platform Digital Desain Rumah 3D Berbasis Mobile Menggunakan Metode ADDIE	
35	Andrew Ferlian Koesnaedi	2011010017	Virtual Reality Permainan Edukasi Matematika dengan Model Rhythm Game	
36	Aldyan Abel Imando	1911010095	Sistem Pengelolaan Aset di Biro Manajemen Aset dan Logistik Darmajaya dengan Algoritma Bubble Sort	Rionaldi Ali, S.Kom., M.T.I