DAFTAR LAMPIRAN

Lampiran - 1 Listing Program

UI.py

```
import tkinter as tk
from tkinter import ttk
from subprocess import run

class BerandaApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Aplikasi Deteksi Kesegaran Cumi Menggunakan Algoritma KNN")
        self.root.geometry("400x400")  # Sesuaikan lebar dan tinggi sesuai keinginan

        style = ttk.Style()
        style.configure("TButton", relief="flat", background="#2196F3", foreground="black")

        frame = ttk.Frame(root)
        frame.pack(expand=True, fill="both")

        label_text = "Aplikasi Deteksi Kesegaran Cumi\nMenggunakan Algoritma KNN"
        label = ttk.Label(frame, text=label_text, font=("Helvetica", 16), background="#2196F3", foreground="black", justify="center")
        label.grid(row=0, column=0, columnspan=3, pady=(10, 20))

        # Menu Mulai Test
        mulai_test_button = ttk.Button(frame, text="Testing Realtime", command=self.mulai_test, style="TButton", width=15)
        mulai_test_button.grid(row=1, column=1, pady=(5, 10))

        # Menu Test Gambar
        test_gambar_button = ttk.Button(frame, text="Test Gambar", command=self.test_gambar, style="TButton", width=15)
        test_gambar_button.grid(row=2, column=1, pady=(5, 10))

        # Menu Logout
        logout_button = ttk.Button(frame, text="Logout", command=self.logout, style="TButton", width=15)
        logout_button.grid(row=3, column=1, pady=(5, 10))

        # Centang semua widget
        for child in frame.winfo_children():
            child.grid_configure(padx=10)

        # Menengahkan frame
        frame.columnconfigure(0, weight=1)
        frame.columnconfigure(2, weight=1)
        frame.rowconfigure(4, weight=1)

    def mulai_test(self):
        # Jalankan script testingreal.py
        run(["python", "testrealKNN.py"])

    def test_gambar(self):
```

```
        # Jalankan script Testinggambar.py
        run(["python", "histogram2.py"])

    def logout(self):
        # Keluar dari aplikasi
        self.root.destroy()

if __name__ == "__main__":
    root = tk.Tk()
    app = BerandaApp(root)
    root.mainloop()
```

ubahdata.py

```
import numpy as np
from PIL import Image

# Load gambar
image_path =
"/Users/macbook/cumi/cumisegar1.jpeg
"
img = Image.open(image_path)

# Convert gambar ke matriks NumPy
img_matrix = np.array(img)

# Flatten gambar menjadi vektor 1
dimensi
flattened_vector = img_matrix.flatten()

# Simpan vektor 1 dimensi ke dalam file
teks
output_file_path =
"/Users/macbook/cumi/segar1.txt"
np.savetxt(output_file_path,
flattened_vector)

# Cetak pesan sukses
print(f"Vektor 1 dimensi berhasil
disimpan ke dalam file:
{output_file_path}")
```

TestrealKNN.py

```
import cv2
import numpy as np
import joblib

# Load KNN model
knn_model =
joblib.load('KNN_modelnew.joblib')

cap = cv2.VideoCapture(0)

while True:
    # Baca frame dari kamera
    ret, frame = cap.read()

    # Resize frame menjadi dimensi yang
diharapkan (sesuaikan dengan dimensi
yang digunakan saat melatih model)
    resized_frame = cv2.resize(frame,
(100, 100))

    # Flatten dan normalisasi frame
    flattened_frame =
resized_frame.flatten() / 255.0

    # Lakukan prediksi menggunakan
model KNN
    prediction =
knn_model.predict([flattened_frame])

    # Konversi hasil prediksi ke integer
    prediction_int = int(prediction[0])

    # Map hasil prediksi ke nama kelas
    if prediction_int == 0:
        label = 'Segar'
    elif prediction_int == 1:
        label = 'Cumi_Busuk'
    else:
        label = 'Tidak_Segar'

    # Tampilkan hasil prediksi di layar
    cv2.putText(frame, f"Prediction:
{label}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 2)

    # Tampilkan frame pada jendela
    cv2.imshow('Real-Time Prediction',
frame)

    # Hentikan loop jika tombol 'q'
ditekan
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Tutup kamera dan jendela
cap.release()
cv2.destroyAllWindows()
```

```
import numpy as np
import cv2
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.preprocessing import
StandardScaler
from sklearn.model_selection import
train_test_split
from sklearn.metrics import
accuracy_score, classification_report
from joblib import dump, load

def preprocess_image(frame):
    # Preprocessing seperti sebelumnya

 def train_and_save_model():
    # Contoh data latihan
    X_train_fresh = np.random.rand(50,
384 * 512 * 3)  # Data untuk kelas
"Segar"
    X_train_not_fresh =
np.random.rand(50, 384 * 512 * 3)  #
Data untuk kelas "Tidak Segar"
    X_train_rotten = np.random.rand(10,
384 * 512 * 3)  # Data untuk kelas
"Busuk"

    # Label
    y_train_fresh = np.zeros(50)  # Label
untuk kelas "Segar"
    y_train_not_fresh = np.ones(50)  #
Label untuk kelas "Tidak Segar"
    y_train_rotten = 2 * np.ones(10)  #
Label untuk kelas "Busuk"

    # Gabungkan data dan label
    X_train =
np.concatenate([X_train_fresh,
X_train_not_fresh, X_train_rotten])
    y_train =
np.concatenate([y_train_fresh,
y_train_not_fresh, y_train_rotten])

    # Inisialisasi model KNN
    knn_model =
KNeighborsClassifier(n_neighbors=3)

    # Latih model
    knn_model.fit(X_train, y_train)

    # Simpan model
    dump(knn_model, 'knn_model.joblib')

def test_realtime_combined_model():
    # Load model gabungan
    combined_model_path =
"knn_model.joblib"
    loaded_combined_model =
load(combined_model_path)

    # Buka kamera
    cap = cv2.VideoCapture(0)  # 0 untuk
kamera default, ganti sesuai kebutuhan

    while True:
        # Ambil frame dari kamera
        ret, frame = cap.read()

        # Preprocess frame
        frame_data =
preprocess_image(frame)

        # Reshape frame_data menjadi
array 2D
        frame_data_2d =
frame_data.reshape(1, -1)



        # Menampilkan hasil prediksi di
layar
        if prediction[0] == 0:
            result_message = "Prediksi:
Segar"
        elif prediction[0] == 1:
            result_message = "Prediksi:
Tidak Segar"
        else:
```

```
            result_message = "Prediksi:
Busuk"

        cv2.putText(frame, result_message,
(10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 2, cv2.LINE_AA)


        # Keluar dari loop jika tombol 'q'
ditekan
        if cv2.waitKey(1) & 0xFF ==
ord('q'):
            break

    # Menutup kamera dan jendela
    cap.release()
    cv2.destroyAllWindows()
```

Testingmodel.py

```
import numpy as np
from PIL import Image, ImageTk
import easygui
import tkinter as tk
from tkinter import ttk
import matplotlib.pyplot as plt
from
matplotlib.backends.backend_tkagg
import FigureCanvasTkAgg
from sklearn.preprocessing import
StandardScaler
from joblib import load

def preprocess_image(image_path,
target_size=(100, 100)):
    # Load gambar
    image = Image.open(image_path)

    # Resize gambar
    resized_image =
image.resize(target_size)

    # Convert gambar ke array numpy
    image_array =
np.array(resized_image)

    # Flatten gambar
    flattened_array =
image_array.flatten()

    # Normalisasi nilai piksel antara 0 dan
1
    normalized_array = flattened_array /
255.0

    return image_array, normalized_array

def display_histogram(root, image):
    # Hitung histogram warna
    hist_red = np.histogram(image[:,:,0],
bins=256, range=[0,256])
    hist_green =
np.histogram(image[:,:,1], bins=256,
range=[0,256])
    hist_blue = np.histogram(image[:,:,2],
bins=256, range=[0,256])

    # Buat subplot untuk histogram
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3,
figsize=(6, 2))

    ax1.plot(hist_red[1][:-1], hist_red[0],
color='red')
    ax1.set_title('Red Histogram')

    ax2.plot(hist_green[1][:-1],
hist_green[0], color='green')
    ax2.set_title('Green Histogram')

    ax3.plot(hist_blue[1][:-1],
hist_blue[0], color='blue')
    ax3.set_title('Blue Histogram')

    # Menampilkan histogram warna
    canvas = FigureCanvasTkAgg(fig,
master=root)
    canvas_widget =
canvas.get_tk_widget()
    canvas_widget.pack()

    canvas.draw()

def test_new_image():
    # Membuka dialog untuk memilih file
gambar
    new_image_path =
easygui.fileopenbox(filetypes=["*.png",
"*.jpg", "*.jpeg"], title="Pilih file
gambar")

    try:
        if new_image_path:
            # Load model KNN
```

```
        knn_model_path =
"KNN_modelnew.joblib"
        loaded_knn_model =
load(knn_model_path)

        # Preprocess gambar baru
        new_image, new_image_data =
preprocess_image(new_image_path)

        # Menampilkan histogram
        root_hist = tk.Tk()
        root_hist.title("Histogram
Warna")
        display_histogram(root_hist,
new_image)
        root_hist.mainloop()

        # Mendapatkan prediksi dari
model KNN
        prediction =
loaded_knn_model.predict([new_image
_data])

        # Map hasil prediksi ke nama
kelas
        if prediction[0] == 0:
            label = 'Segar'
        elif prediction[0] == 1:
            label = 'Tidak Segar'
        else:
            label = 'Cumi Busuk'

        # Menampilkan hasil prediksi
menggunakan Tkinter
        root = tk.Tk()
        root.title("Hasil Prediksi")

        # Menampilkan prediksi
        result_message = f"Prediksi:
{label}"
        label_result = ttk.Label(root,
text=result_message, font=("Helvetica",
14), padding=10)
        label_result.pack()

        # Menampilkan gambar
        img =
ImageTk.PhotoImage(Image.fromarray(
new_image))
        img_label = ttk.Label(root,
image=img)
        img_label.pack()

        # Tombol Keluar untuk menutup
jendela
        keluar_button = ttk.Button(root,
text="Keluar", command=root.destroy)
        keluar_button.pack(pady=10)

        root.mainloop()
    else:
        easygui.msgbox("Tidak ada
gambar yang dipilih.",
title="Peringatan")

  except Exception as e:
    easygui.msgbox(f"Terjadi
kesalahan: {e}", title="Kesalahan")

# Uji gambar baru dengan model KNN
test_new_image()
```

testing2.py

```
import cv2
import numpy as np
from PIL import Image
from sklearn.preprocessing import StandardScaler
from joblib import load

# Fungsi untuk memproses gambar menjadi vektor 1 dimensi
def process_image(img, target_size=(256, 256)):
    img = cv2.resize(img, target_size)
    img_array = np.array(img)
    flattened_vector = img_array.flatten()
    return flattened_vector

# Load model yang sudah dilatih
loaded_knn_model = load('knn_model.joblib')

# Buka kamera
cap = cv2.VideoCapture(0)  # Gunakan 0 untuk kamera utama, bisa disesuaikan

while True:
    # Ambil frame dari kamera
    ret, frame = cap.read()

    # Proses frame menjadi vektor 1 dimensi
    image_vector = process_image(frame)

    # Normalisasi menggunakan Z-Score Scaling (Standardization)
    scaler = StandardScaler()
    image_vector_scaled = scaler.fit_transform(image_vector.reshape(1, -1))

    # Lakukan prediksi menggunakan model yang sudah dilatih
    prediction = loaded_knn_model.predict(image_vector_scaled)

    # Tampilkan hasil prediksi pada layar
    cv2.putText(frame, f'Prediction: {int(prediction[0])}', (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    # Tampilkan frame pada jendela
    cv2.imshow('Real-time Prediction', frame)

    # Hentikan jika tombol 'q' ditekan
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Tutup kamera dan jendela
cap.release()
cv2.destroyAllWindows()
```

Testing.py

```
import cv2
import numpy as np
from sklearn.preprocessing import StandardScaler
import joblib
import os

# Ganti path sesuai dengan lokasi file gambar Anda
current_directory = os.path.dirname(os.path.abspath(__file__))
image_uji_segar_path = os.path.join(current_directory, "segar/cumisegar1.jpeg")
image_uji_tidak_segar_path = os.path.join(current_directory, "tidak_segar/ts1.jpeg")

# Fungsi untuk membaca dan mengubah ukuran gambar
def read_and_resize_image(image_path, target_size=(100, 100)):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if image is not None:
        image_resized = cv2.resize(image, target_size, interpolation=cv2.INTER_AREA)
        return image_resized.flatten()
    else:
        print(f"Gagal membaca gambar di '{image_path}'. Pastikan path gambar benar.")
        return None

# Baca dan resize gambar uji segar
vector_uji_segar = read_and_resize_image(image_uji_segar_path)

# Baca dan resize gambar uji tidak segar
vector_uji_tidak_segar = read_and_resize_image(image_uji_tidak_segar_path)

# Periksa apakah kedua gambar berhasil dibaca dan di-resize
if vector_uji_segar is not None and vector_uji_tidak_segar is not None:
    # Gabungkan vektor fitur uji
    all_vectors_uji = np.vstack((vector_uji_segar, vector_uji_tidak_segar))

    # Normalisasi data uji menggunakan scaler yang sesuai
    scaler_filename = os.path.join(current_directory, "scaler.joblib")
    scaler = joblib.load(scaler_filename)
    all_vectors_uji_scaled = scaler.transform(all_vectors_uji)

    # Muat model
    loaded_model = joblib.load("knn_model.joblib")

    # Prediksi label pada data uji
    y_pred_uji = loaded_model.predict(all_vectors_uji_scaled)

    # Tampilkan hasil prediksi
    print("Hasil prediksi uji segar:", y_pred_uji[0])
    print("Hasil prediksi uji tidak segar:", y_pred_uji[1])
```

tes-2.html

```
<div>Teachable Machine Image Model</div>
<button type="button" onclick="init()">Start</button>
<div id="webcam-container"></div>
<div id="label-container"></div>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teachablemachine-image.min.js"></script>
<script type="text/javascript">
    // More API functions here:
    // https://github.com/googlecreativelab/teachablemachine-community/tree/master/libraries/image

    // the link to your model provided by Teachable Machine export panel
    const URL = "./my_model/";

    let model, webcam, labelContainer, maxPredictions;

    // Load the image model and setup the webcam
    async function init() {
        const modelURL = URL + "model.json";
        const metadataURL = URL + "metadata.json";

        // load the model and metadata
        // Refer to tmImage.loadFromFiles() in the API to support files from a file picker
        // or files from your local hard drive
        // Note: the pose library adds "tmImage" object to your window (window.tmImage)
        model = await tmImage.load(modelURL, metadataURL);
        maxPredictions = model.getTotalClasses();

        // Convenience function to setup a webcam
        const flip = true; // whether to flip the webcam
        webcam = new tmImage.Webcam(200, 200, flip); // width, height, flip
        await webcam.setup(); // request access to the webcam
        await webcam.play();

        window.requestAnimationFrame(loop);

        // append elements to the DOM

        document.getElementById("webcam-container").appendChild(webcam.canvas);
        labelContainer = document.getElementById("label-container");
        for (let i = 0; i < maxPredictions; i++) { // and class labels

            labelContainer.appendChild(document.createElement("div"));
        }
    }

    async function loop() {
        webcam.update(); // update the webcam frame
        await predict();
```

```
window.requestAnimationFrame(loop);
   }

   // run the webcam image through the
image model
   async function predict() {
      // predict can take in an image,
video or canvas html element
      const prediction = await
model.predict(webcam.canvas);
      for (let i = 0; i < maxPredictions;
i++) {
         const classPrediction =
            prediction[i].className + ": "
+ prediction[i].probability.toFixed(2);

labelContainer.childNodes[i].innerHTM
L = classPrediction;
      }
   }
</script>
```

Split.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import
accuracy_score
from sklearn.utils import shuffle

scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


knn =
KNeighborsClassifier(n_neighbors=5,
weights='distance')  # Menggunakan
bobot jarak

# Baca vektor 1 dimensi dan label kelas
dari file teks kelas_segar.txt
file_segar_path = "segar.txt"
data_segar =
np.genfromtxt(file_segar_path,
dtype=int)

# Baca vektor 1 dimensi dan label kelas
dari file teks kelas_tidak_segar.txt
file_tidak_segar_path =
"tidak_segar.txt"
data_tidak_segar =
np.genfromtxt(file_tidak_segar_path,
dtype=int)

   # Hapus kelas dengan sampel kurang
dari 2
   unique_labels, label_counts =
np.unique(labels, return_counts=True)
   small_classes =
unique_labels[label_counts < 2]
   indices_to_remove = np.isin(labels,
small_classes)
   all_vectors_array_filtered =
all_vectors_array[~indices_to_remove]
   labels_filtered =
labels[~indices_to_remove]

   # Bagi dataset menjadi set pelatihan
dan set pengujian (80% train, 20% test)
   X_train, X_test, y_train, y_test =
train_test_split(all_vectors_array_filtere
d, labels_filtered, test_size=0.2,
random_state=42,
stratify=labels_filtered)

   # Reshape data input menjadi matriks
dua dimensi
   X_train_reshaped = X_train.reshape(-
1, 1)
   X_test_reshaped = X_test.reshape(-1,
1)

   # Normalisasi data
   scaler = StandardScaler()
   X_train_reshaped =
scaler.fit_transform(X_train_reshaped)
   X_test_reshaped =
scaler.transform(X_test_reshaped)

   # Inisialisasi model KNN
   knn_model =
KNeighborsClassifier(n_neighbors=5)

   # Pelatihan model KNN
   knn_model.fit(X_train_reshaped,
y_train)

   # Prediksi label pada data pengujian
   y_pred =
knn_model.predict(X_test_reshaped)
```

```
    # Hitung akurasi
    accuracy = accuracy_score(y_test,
y_pred)
    print("Akurasi:
{:.2f}%".format(accuracy * 100))

    # Plot hasil prediksi
    plt.scatter(X_test_reshaped, y_test,
label="Data Asli")
    plt.scatter(X_test_reshaped, y_pred,
label="Prediksi")
    plt.xlabel("Fitur")
    plt.ylabel("Label Kelas")
    plt.legend()
    plt.show()
```

RealKNN.py

```
import cv2
import numpy as np
import joblib

# Load KNN model
knn_model =
joblib.load('KNN_modelnew.joblib')

# Inisialisasi kamera (biasanya kamera
utama laptop memiliki ID 0)
cap = cv2.VideoCapture(0)

while True:
    # Baca frame dari kamera
    ret, frame = cap.read()

    # Resize frame menjadi dimensi yang
diharapkan (sesuaikan dengan dimensi
yang digunakan saat melatih model)
    resized_frame = cv2.resize(frame,
(100, 100))

    # Flatten dan normalisasi frame
    flattened_frame =
resized_frame.flatten() / 255.0

    # Lakukan prediksi menggunakan
model KNN
    prediction =
knn_model.predict([flattened_frame])

    # Konversi hasil prediksi ke integer
    prediction_int = int(prediction[0])

    # Map hasil prediksi ke nama kelas
    if prediction_int == 0:
        label = 'Segarku'
    elif prediction_int == 1:
        label = 'TS'
    else:
        label = 'Cumi_Busuk'

    # Tampilkan hasil prediksi di layar
    cv2.putText(frame, f"Prediction:
{label}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 2)

    # Tampilkan frame pada jendela
    cv2.imshow('Real-Time Prediction',
frame)

    # Hentikan loop jika tombol 'q'
ditekan
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Tutup kamera dan jendela
cap.release()
cv2.destroyAllWindows()
```

prepprocessing.py

```
import numpy as np
from PIL import Image

# Load gambar
image_path =
"/Users/macbook/cumi/cumisegar1.jpeg
"
img = Image.open(image_path)

# Convert gambar ke matriks NumPy
img_matrix = np.array(img)

# Flatten gambar menjadi vektor 1
dimensi
flattened_vector = img_matrix.flatten()

# Preprocessing
# 1. Normalisasi
normalized_vector = (flattened_vector -
np.mean(flattened_vector)) /
np.std(flattened_vector)

# 2. Scaling
scaled_vector = flattened_vector / 255.0
# Scaling nilai ke rentang [0, 1]

# 3. Deteksi dan Penanganan Outlier
lower_limit = 0
upper_limit = 1
clipped_vector =
np.clip(flattened_vector, lower_limit,
upper_limit)

# 4. Filtering atau Smoothing
window_size = 3
smoothed_vector =
np.convolve(flattened_vector,
np.ones(window_size)/window_size,
mode='valid')

# 5. Ekstraksi Fitur Tambahan
mean_value =
np.mean(flattened_vector)
variance_value =
np.var(flattened_vector)

# Cetak hasil preprocessing
print("Original Vektor 1 Dimensi:")
print(flattened_vector)
print("\nNormalized Vektor:")
print(normalized_vector)
print("\nScaled Vektor:")
print(scaled_vector)
print("\nClipped Vektor:")
print(clipped_vector)
print("\nSmoothed Vektor:")
print(smoothed_vector)
print("\nMean Value:", mean_value)
print("Variance Value:",
variance_value)
```

```
import os
import cv2
import numpy as np
from sklearn.model_selection import
train_test_split
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import
accuracy_score
import joblib
import matplotlib.pyplot as plt
from skimage.filters import gabor
from skimage import transform as tf

def convert_to_hsv(image):
    hsv_image = cv2.cvtColor(image,
cv2.COLOR_BGR2HSV)
    return hsv_image

def apply_gabor_filter(image):
    gray_image = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
    _, gabor_image = gabor(gray_image,
frequency=0.6)
    return gabor_image

def deskew_image(image):
    gray_image = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
    coords =
np.column_stack(np.where(gray_image
> 0))
    angle = cv2.minAreaRect(coords)[-1]
    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle
    rotated = tf.rotate(gray_image, angle,
mode='edge', preserve_range=True)
    return rotated

def load_images_from_folder(folder):
    images = []
    labels = []
    for filename in os.listdir(folder):
        img_path = os.path.join(folder,
filename)
        img = cv2.imread(img_path,
cv2.IMREAD_COLOR)
        if img is not None:
            # Resize gambar ke dimensi
tertentu
            img_resized = cv2.resize(img,
(100, 100))

            # Tambahkan metode HSV
            hsv_img =
convert_to_hsv(img_resized)

            # Tambahkan metode Gabor
            gabor_img =
apply_gabor_filter(img_resized)

            # Tambahkan metode deskew
            deskewed_img =
deskew_image(img_resized)

            # Ganti img_resized dengan
salah satu metode di atas jika diperlukan
            images.append({'original':
img_resized, 'hsv': hsv_img, 'gabor':
gabor_img, 'deskewed':
deskewed_img})
            labels.append(folder.split('/')[-1])
# Label berdasarkan nama folder

            if len(images) >= 5:
                break

    return images, labels

# Load images for Cumi Segar
images_segar, labels_segar =
load_images_from_folder('segarku')

# Load images for Cumi Tidak Segar
images_tidak_segar, labels_tidak_segar
= load_images_from_folder('TS')
```

```
# Load images for Cumi Busuk
images_busuk, labels_busuk =
load_images_from_folder('Cumi_Busuk'
)

# Combine data
images = images_segar +
images_tidak_segar + images_busuk
labels = labels_segar +
labels_tidak_segar + labels_busuk

# Convert labels to numerical values
```

Newmodel.py

```
label_mapping = {'segarku': 0, 'TS': 1,
'Cumi_Busuk': 2}
labels_numeric = [label_mapping[label]
for label in labels]

# Convert images to NumPy array
images_flattened =
np.array([img['original'].reshape(-1) /
255.0 for img in images])

# Split data into training and testing sets
X_train, X_test, y_train, y_test =
train_test_split(images_flattened,
labels_numeric, test_size=0.2,
random_state=42)

# Initialize KNN model
knn_model =
KNeighborsClassifier(n_neighbors=5)

# Train the KNN model
knn_model.fit(X_train, y_train)

# Save the KNN model to file
joblib.dump(knn_model,
'KNN_modelnew.joblib')
print("KNN model has been saved in
KNN_modelnew.joblib")

# Test the model on the testing data
y_test_pred =
knn_model.predict(X_test)

# Calculate testing accuracy
test_accuracy = accuracy_score(y_test,
y_test_pred)
print("Model Accuracy (Testing):",
test_accuracy)

# Test the model on the training data
y_train_pred =
knn_model.predict(X_train)

# Calculate training accuracy
train_accuracy = accuracy_score(y_train,
y_train_pred)
print("Model Accuracy (Training):",
train_accuracy)

# Display bar chart of training and
testing accuracies
def plot_accuracy_bar(train_accuracy,
test_accuracy):
    labels = ['Training', 'Testing']
    accuracies = [train_accuracy,
test_accuracy]

    plt.bar(labels, accuracies,
color=['blue', 'orange'])
    plt.title('KNN Model Accuracy for
Training and Testing Data')
    plt.ylabel('Accuracy')
    plt.ylim(0, 1)
    plt.show()

# Plot the bar chart
plot_accuracy_bar(train_accuracy,
test_accuracy)
```

ModelKNN2.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import
accuracy_score, classification_report
from sklearn.preprocessing import
StandardScaler

# Fungsi untuk membaca data dari file
teks
def read_data(file_path):
    data = np.loadtxt(file_path,
dtype=float)
    return data

# Load data untuk kelas segar
cumisegar_data =
read_data("cumisegar2.txt")
cumisegar_labels =
np.zeros(cumisegar_data.shape[0])  #
Label 0 untuk kelas segar

# Load data untuk kelas tidak segar
cumitidaksegar_data =
read_data("cumitidaksegar3.txt")
cumitidaksegar_labels =
np.ones(cumitidaksegar_data.shape[0])
# Label 1 untuk kelas tidak segar

# Gabungkan data dari kedua kelas
all_data = np.vstack((cumisegar_data,
cumitidaksegar_data))
all_labels =
np.concatenate((cumisegar_labels,
cumitidaksegar_labels))

# Pisahkan data menjadi set pelatihan
dan pengujian
X_train, X_test, y_train, y_test =
train_test_split(all_data, all_labels,
test_size=0.2, random_state=42)

# Normalisasi menggunakan Z-Score
Scaling (Standardization)
scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Inisialisasi model KNN dengan
tetangga yang lebih sedikit
knn_model =
KNeighborsClassifier(n_neighbors=5)  #
Ganti dengan nilai yang lebih sesuai

# Latih model dengan set pelatihan
knn_model.fit(X_train_scaled, y_train)

# Lakukan prediksi pada set pelatihan
dan pengujian
y_train_pred =
knn_model.predict(X_train_scaled)
y_test_pred =
knn_model.predict(X_test_scaled)

# Evaluasi kinerja model pada set
pelatihan dan pengujian
accuracy_train = accuracy_score(y_train,
y_train_pred)
accuracy_test = accuracy_score(y_test,
y_test_pred)

# Visualisasi grafik nilai akurasi
plt.figure(figsize=(8, 5))
plt.bar(['Training', 'Testing'],
[accuracy_train, accuracy_test],
color=['blue', 'orange'])
plt.title('Accuracy on Training and
Testing Sets')
plt.ylabel('Accuracy')
```

```
plt.ylim(0, 1.0)
plt.show()

# Classification report on testing set
classification_rep =
classification_report(y_test, y_test_pred)
print('Classification Report on Testing
Set:')
print(classification_rep)
```

ModelKNN.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import
accuracy_score, classification_report
from sklearn.preprocessing import
StandardScaler
from joblib import dump, load

# Fungsi untuk membaca data dari file
teks
def read_data(file_path):
    data = np.loadtxt(file_path,
dtype=float)
    return data

# Load data untuk kelas segar
cumisegar_data =
read_data("cumisegar2.txt")
cumisegar_labels =
np.zeros(cumisegar_data.shape[0])  #
Label 0 untuk kelas segar

# Load data untuk kelas tidak segar
cumitidaksegar_data =
read_data("cumitidaksegar3.txt")
cumitidaksegar_labels =
np.ones(cumitidaksegar_data.shape[0])
# Label 1 untuk kelas tidak segar

# Load data untuk kelas busuk
cumibusuk_data =
read_data("cumibusuk.txt")
cumibusuk_labels =
np.full(cumibusuk_data.shape[0], 2)  #
Label 2 untuk kelas busuk

# Gabungkan data dari ketiga kelas
all_data = np.vstack((cumisegar_data,
cumitidaksegar_data, cumibusuk_data))
all_labels =
np.concatenate((cumisegar_labels,
cumitidaksegar_labels,
cumibusuk_labels))

# Pisahkan data menjadi set pelatihan
dan pengujian
X_train, X_test, y_train, y_test =
train_test_split(all_data, all_labels,
test_size=0.2, random_state=42)

# Normalisasi menggunakan Z-Score
Scaling (Standardization)
scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Inisialisasi model KNN dengan
tetangga yang lebih sedikit
knn_model =
KNeighborsClassifier(n_neighbors=5)  #
Ganti dengan nilai yang lebih sesuai

# Latih model dengan set pelatihan
knn_model.fit(X_train_scaled, y_train)

# Simpan model ke dalam file
dump(knn_model,
'knn_model6991.joblib')

# Load model dari file
loaded_knn_model =
load('knn_model6991.joblib')


# Visualisasi grafik nilai akurasi
plt.figure(figsize=(8, 5))
```

```
plt.bar(['Training', 'Testing'],
[accuracy_train, accuracy_test],
color=['blue', 'orange'])
plt.title('Akurasi Training dan Testing
Deteksi Kesegaran Cumi')
plt.ylabel('Accuracy')
plt.ylim(0, 1.0)
plt.show()

# Classification report on testing set
classification_rep =
classification_report(y_test, y_test_pred)
print('Classification Report on Testing
Set:')
print(classification_rep)
```

Modelbusuk.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
from joblib import dump, load

# Fungsi untuk membaca data dari file teks
def read_data(file_path):
    data = np.loadtxt(file_path, dtype=float)
    return data

# Load data untuk kelas busuk
cumibusuk_data = read_data("cumibusuk.txt")
cumibusuk_labels = np.full(cumibusuk_data.shape[0], 2)  # Label 2 untuk kelas busuk

# Pisahkan data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(cumibusuk_data, cumibusuk_labels, test_size=0.2, random_state=42)

# Normalisasi menggunakan Z-Score Scaling (Standardization)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Inisialisasi model KNN dengan tetangga yang lebih sedikit
knn_model = KNeighborsClassifier(n_neighbors=5)  # Ganti dengan nilai yang lebih sesuai

# Latih model dengan set pelatihan
knn_model.fit(X_train_scaled, y_train)

# Simpan model ke dalam file
dump(knn_model, 'knn_model_busuk.joblib')

# Load model dari file
loaded_knn_model = load('knn_model_busuk.joblib')

# Lakukan prediksi pada set pelatihan dan pengujian
y_train_pred = loaded_knn_model.predict(X_train_scaled)
y_test_pred = loaded_knn_model.predict(X_test_scaled)

# Evaluasi kinerja model pada set pelatihan dan pengujian
accuracy_train = accuracy_score(y_train, y_train_pred)
accuracy_test = accuracy_score(y_test, y_test_pred)

# Visualisasi grafik nilai akurasi
plt.figure(figsize=(8, 5))
plt.bar(['Training', 'Testing'], [accuracy_train, accuracy_test], color=['blue', 'orange'])
plt.title('Akurasi Training dan Testing Deteksi Kesegaran Cumi (Kelas Busuk)')
plt.ylabel('Accuracy')
plt.ylim(0, 1.0)
plt.show()
```

```
# Classification report on testing set
classification_rep =
classification_report(y_test, y_test_pred)
print('Classification Report on Testing
Set:')
print(classification_rep)
```

Makemodel.py

```
import numpy as np
from PIL import Image
import os

# Fungsi untuk membaca dan memproses gambar menjadi vektor 1 dimensi
def process_image(image_path, target_size=(256, 256)):
    img = Image.open(image_path).resize(target_size)
    img_array = np.array(img)
    flattened_vector = img_array.flatten()
    return flattened_vector

# Fungsi Min-Max Scaling
def min_max_scaling(vector):
    min_val = min(vector)
    max_val = max(vector)
    scaled_vector = [(x - min_val) / (max_val - min_val) for x in vector]
    return scaled_vector

# Fungsi Z-Score Scaling (Standardization)
def z_score_scaling(vector):
    mean_val = np.mean(vector)
    std_val = np.std(vector)
    scaled_vector = [(x - mean_val) / std_val for x in vector]
    return scaled_vector

# Contoh path ke folder dengan gambar cumi-cumi segar
folder_path = "Cumi_Busuk"

# List untuk menyimpan vektor 1 dimensi
all_vectors = []

# Dimensi target untuk gambar (contoh: 256 x 256)
target_size = (256, 256)

# Baca gambar-gambar dan tambahkan ke list vektor
for filename in os.listdir(folder_path):
    if filename.endswith(".jpeg"):
        image_path = os.path.join(folder_path, filename)
        flattened_vector = process_image(image_path, target_size)

all_vectors.append(flattened_vector)

# Konversi ke array NumPy dengan dtype=object
all_vectors_array = np.array(all_vectors, dtype=object)

# Normalisasi menggunakan Min-Max Scaling
normalized_vectors_min_max = [min_max_scaling(vector) for vector in all_vectors]

# Normalisasi menggunakan Z-Score Scaling
normalized_vectors_z_score = [z_score_scaling(vector) for vector in all_vectors]

# Simpan vektor 1 dimensi dan vektor normalisasi ke dalam satu file teks
output_file_path = "cumibusuk.txt"
np.savetxt(output_file_path, np.column_stack((all_vectors_array, normalized_vectors_min_max, normalized_vectors_z_score)), fmt='%s')
```

```
print(f"Vektor 1 dimensi dan hasil
normalisasi dari {len(all_vectors)}
gambar berhasil disimpan ke dalam file:
{output_file_path}")
```

makeload.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
StandardScaler
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import
accuracy_score
import joblib

# Baca vektor 1 dimensi dan label kelas
dari file teks kelas_segar.txt
file_segar_path = "segar.txt"
data_segar =
np.genfromtxt(file_segar_path,
dtype=int)

# Baca vektor 1 dimensi dan label kelas
dari file teks kelas_tidak_segar.txt
file_tidak_segar_path =
"tidak_segar.txt"
data_tidak_segar =
np.genfromtxt(file_tidak_segar_path,
dtype=int)

# Cek apakah data kosong
if data_segar.size == 0 or
data_tidak_segar.size == 0:
    print("Data kosong. Periksa file
input.")
else:
    # Gabungkan data dari kedua kelas
    all_vectors_array =
np.concatenate([data_segar,
data_tidak_segar])
    labels =
np.concatenate([np.ones_like(data_segar
), np.zeros_like(data_tidak_segar)])

    # Hapus kelas dengan sampel kurang
dari 2
    unique_labels, label_counts =
np.unique(labels, return_counts=True)
    small_classes =
unique_labels[label_counts < 2]
    indices_to_remove = np.isin(labels,
small_classes)
    all_vectors_array_filtered =
all_vectors_array[~indices_to_remove]
    labels_filtered =
labels[~indices_to_remove]

    # Bagi dataset menjadi set pelatihan
dan set pengujian (80% train, 20% test)
    X_train, X_test, y_train, y_test =
train_test_split(all_vectors_array_filtere
d, labels_filtered, test_size=0.2,
random_state=42,
stratify=labels_filtered)

    # Reshape data input menjadi matriks
dua dimensi
    X_train_reshaped = X_train.reshape(-
1, 1)
    X_test_reshaped = X_test.reshape(-1,
1)

    # Normalisasi data
    scaler = StandardScaler()
    X_train_reshaped =
scaler.fit_transform(X_train_reshaped)
    X_test_reshaped =
scaler.transform(X_test_reshaped)

    # Inisialisasi model KNN
    knn_model =
KNeighborsClassifier(n_neighbors=5)

    # Pelatihan model KNN
    knn_model.fit(X_train_reshaped,
y_train)
```

```
    # Prediksi label pada data pengujian
    y_pred =
knn_model.predict(X_test_reshaped)

    # Hitung akurasi
    accuracy = accuracy_score(y_test,
y_pred)
    print("Akurasi:
{:.2f}%".format(accuracy * 100))

    # Plot hasil prediksi
    plt.scatter(X_test_reshaped, y_test,
label="Data Asli")
    plt.scatter(X_test_reshaped, y_pred,
label="Prediksi")
    plt.xlabel("Fitur")
    plt.ylabel("Label Kelas")
    plt.legend()
```

Gabung.py

```
import joblib
from sklearn.neighbors import
KNeighborsClassifier

# Misalkan model pertama disimpan
dalam file 'model1.joblib'
model1 =
joblib.load('knn_model_busuk.joblib')

# Misalkan model kedua disimpan
dalam file 'model2.joblib'
model2 =
joblib.load('knn_model.joblib')

# Misalkan keduanya adalah
KNeighborsClassifier
if isinstance(model1,
KNeighborsClassifier) and
isinstance(model2,
KNeighborsClassifier):
   # Menggabungkan parameter atau
properti yang sesuai
   combined_model =
KNeighborsClassifier(n_neighbors=mod
el1.n_neighbors + model2.n_neighbors)
   # Anda bisa menambahkan properti
atau parameter lainnya sesuai kebutuhan

   # Training ulang model jika perlu
   # ...

   # Simpan model gabungan ke dalam
satu file
   joblib.dump(combined_model,
'combined_model.joblib')
else:
   print("Model yang diberikan bukan
KNeighborsClassifier.")
```

deteksi_kesegaran_cumi_gui_web.py

```
from flask import Flask,
render_template, redirect, url_for
from flask_socketio import SocketIO,
emit
import cv2
from sklearn.preprocessing import
StandardScaler
from joblib import load
import os
from threading import Thread

app = Flask(__name__)
socketio = SocketIO(app)

# Load model KNN
knn_model_path = "knn_model.joblib"
loaded_knn_model =
load(knn_model_path)

# Flag untuk menghentikan thread
kamera
camera_thread_running = False

# Fungsi untuk preprocessing frame
kamera
def preprocess_image(frame):
    frame_resized = cv2.resize(frame,
(384, 512))
    flattened_array =
frame_resized.flatten()
    normalized_array = flattened_array /
255.0
    return normalized_array

# Fungsi untuk testing kamera
def test_camera():
    global camera_thread_running

    # Buka kamera
    cap = cv2.VideoCapture(0)

    while camera_thread_running:
        ret, frame = cap.read()
        frame_data =
preprocess_image(frame)
        prediction =
loaded_knn_model.predict([frame_data]
)
        result_message = f"Prediksi:
{'Segar' if prediction[0] == 0 else 'Tidak
Segar'}"

        # Kirim hasil prediksi ke semua
klien yang terhubung
        socketio.emit('update_result',
{'result_message': result_message})

        _, jpeg = cv2.imencode('.jpg',
frame)
        frame_bytes = jpeg.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type:
image/jpeg\r\n\r\n' + frame_bytes +
b'\r\n')

    # Tutup kamera setelah pengujian
selesai
    cap.release()

# Fungsi untuk memulai thread kamera
def start_camera_thread():
    global camera_thread_running
    camera_thread_running = True
    app.config['CAMERA_THREAD'] =
Thread(target=test_camera)

app.config['CAMERA_THREAD'].start(
)

# Fungsi untuk menghentikan thread
kamera
def stop_camera_thread():
```

```
    global camera_thread_running
    camera_thread_running = False

# Fungsi untuk keluar aplikasi
def exit_app():
    # Hentikan thread kamera jika sedang
berjalan
    if 'CAMERA_THREAD' in
app.config:
        stop_camera_thread()

app.config['CAMERA_THREAD'].join(
)

    return 'Exiting application'

# Fungsi utama untuk halaman beranda
@app.route('/')
def home():
    return render_template('home.html')

# Route untuk testing kamera
@app.route('/test_camera')
def camera_page():
    start_camera_thread()
    return render_template('camera.html')

# Route untuk testing gambar (belum
diimplementasikan)
@app.route('/test_image')
def image_page():
    return render_template('image.html')

# Route untuk keluar aplikasi
@app.route('/exit')
def exit_page():
    return exit_app()

# Route untuk mengatasi koneksi
SocketIO
@socketio.on('connect')
def handle_connect():
    print('Client connected')

# Route untuk mengatasi disconnect
SocketIO
@socketio.on('disconnect')
def handle_disconnect():
    print('Client disconnected')

if __name__ == '__main__':
    socketio.run(app, debug=True)
```

deteksi_kesegaran_cumi_camera.py

```
from flask import Flask, Response
import cv2
from sklearn.preprocessing import StandardScaler
from joblib import load

app = Flask(__name__)

def preprocess_image(frame):
    # Resize frame sesuai dengan dimensi yang diharapkan oleh model
    frame_resized = cv2.resize(frame, (384, 512))  # Ganti ukuran sesuai kebutuhan

    # Meratakan frame
    flattened_array = frame_resized.flatten()

    # Normalisasi nilai piksel antara 0 dan 1 (jika diperlukan)
    normalized_array = flattened_array / 255.0

    return normalized_array

def test_realtime_knn():
    # Load model KNN
    knn_model_path = "knn_model.joblib"
    loaded_knn_model = load(knn_model_path)

    # Buka kamera
    cap = cv2.VideoCapture(0)  # 0 untuk kamera default, ganti sesuai kebutuhan

    while True:
        # Ambil frame dari kamera
        ret, frame = cap.read()

        # Preprocess frame
        frame_data = preprocess_image(frame)

        # Mendapatkan prediksi dari model KNN
        prediction = loaded_knn_model.predict([frame_data])

        # Menampilkan hasil prediksi di layar
        result_message = f"Prediksi: {'Segar' if prediction[0] == 0 else 'Tidak Segar'}"
        cv2.putText(frame, result_message, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

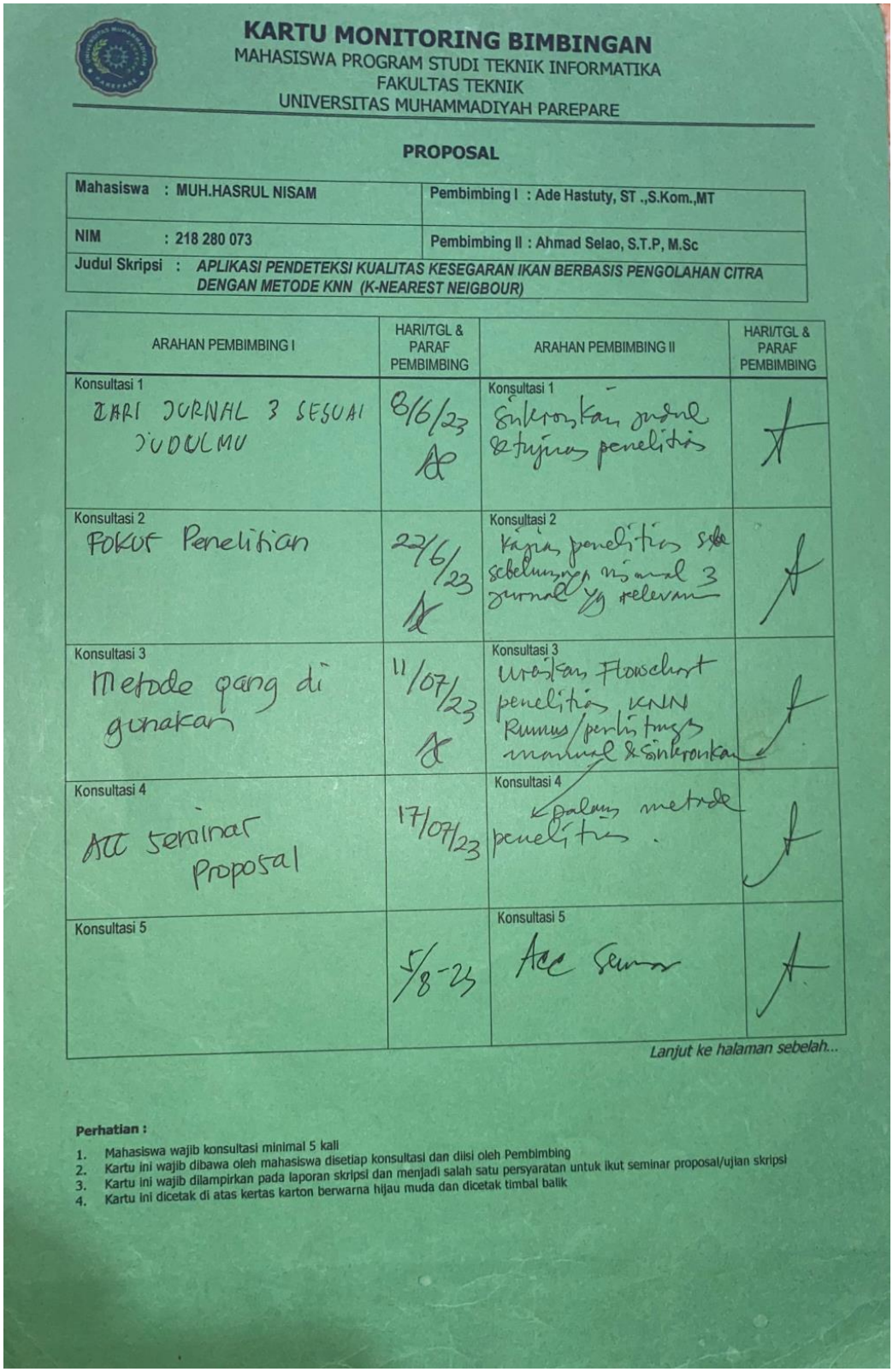
        # Mengirim frame sebagai respons ke browser
        _, jpeg = cv2.imencode('.jpg', frame)
        frame_bytes = jpeg.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/video_feed')
def video_feed():
    return Response(test_realtime_knn(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == "__main__":
    app.run(debug=True)
```

Lampiran - 2 Kartu Monitoring Bimbingan Proposal



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

PROPOSAL

Mahasiswa : MUH.HASRUL NISAM	Pembimbing I : Ade Hastuty, ST .,S.Kom.,MT
NIM : 218 280 073	Pembimbing II : Ahmad Selao, S.T.P, M.Sc
Judul Skripsi : *APLIKASI PENDETEKSI KUALITAS KESEGARAN IKAN BERBASIS PENGOLAHAN CITRA DENGAN METODE KNN (K-NEAREST NEIGBOUR)*	

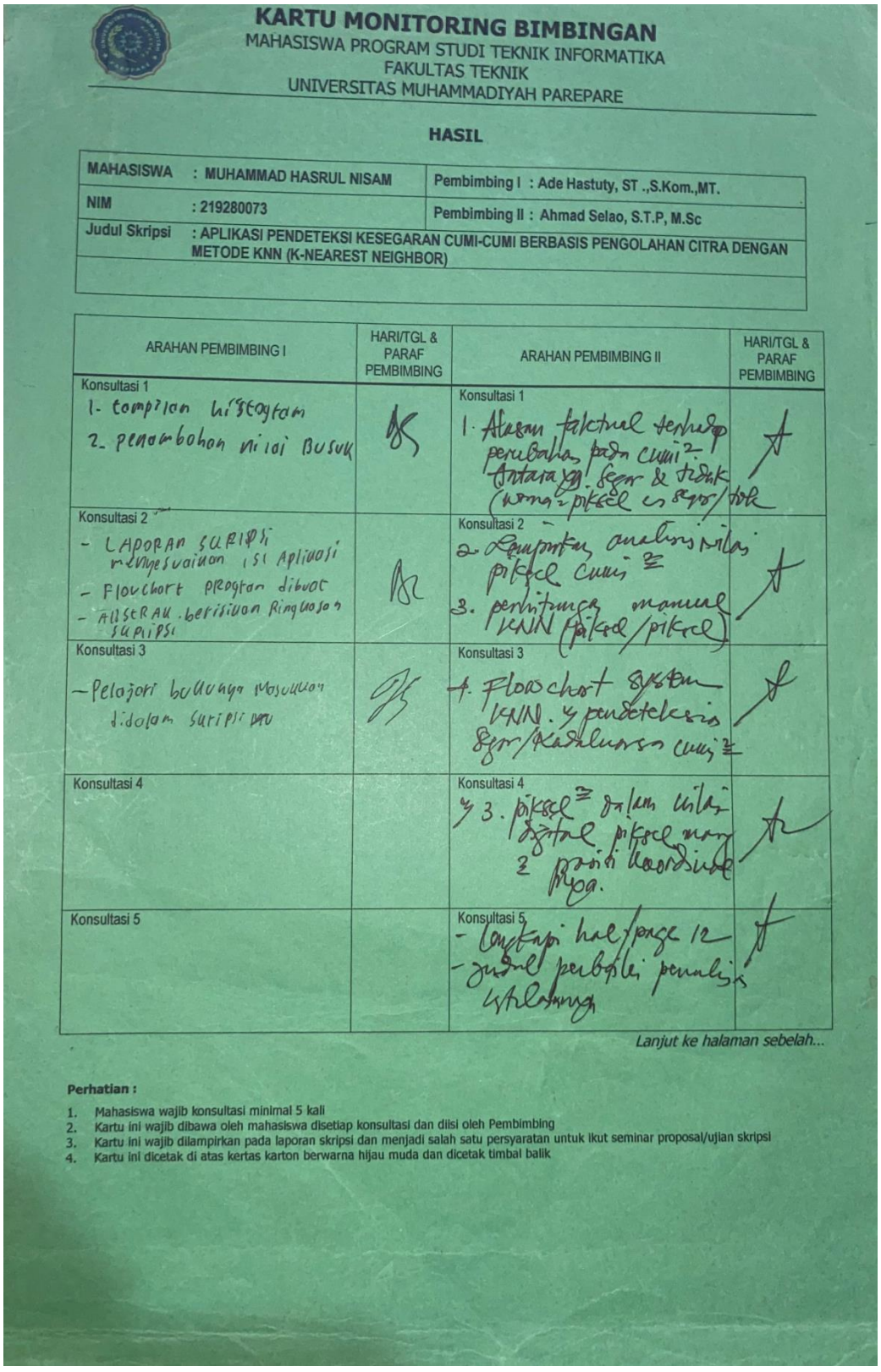
ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 DARI JURNAL 3 SESUAI JUDULMU	8/6/23	Konsultasi 1 Sinkronkan judul & tujuan penelitian	
Konsultasi 2 FOKUS Penelitian	22/6/23	Konsultasi 2 kajian penelitian sbb sebelumnya minimal 3 jurnal yg relevan	
Konsultasi 3 Metode yang di gunakan	11/07/23	Konsultasi 3 Urutkan Flowchart penelitian KNN Rumus/perhitungan manual & sinkronkan	
Konsultasi 4 ACC Seminar Proposal	17/07/23	Konsultasi 4 Dalam metode penelitian.	
Konsultasi 5	5/8-23	Konsultasi 5 Acc Semro	

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 Hasil



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

HASIL

MAHASISWA	: MUHAMMAD HASRUL NISAM	Pembimbing I	: Ade Hastuty, ST .,S.Kom.,MT.
NIM	: 219280073	Pembimbing II	: Ahmad Selao, S.T.P, M.Sc
Judul Skripsi	: APLIKASI PENDETEKSI KESEGARAN CUMI-CUMI BERBASIS PENGOLAHAN CITRA DENGAN METODE KNN (K-NEAREST NEIGHBOR)		

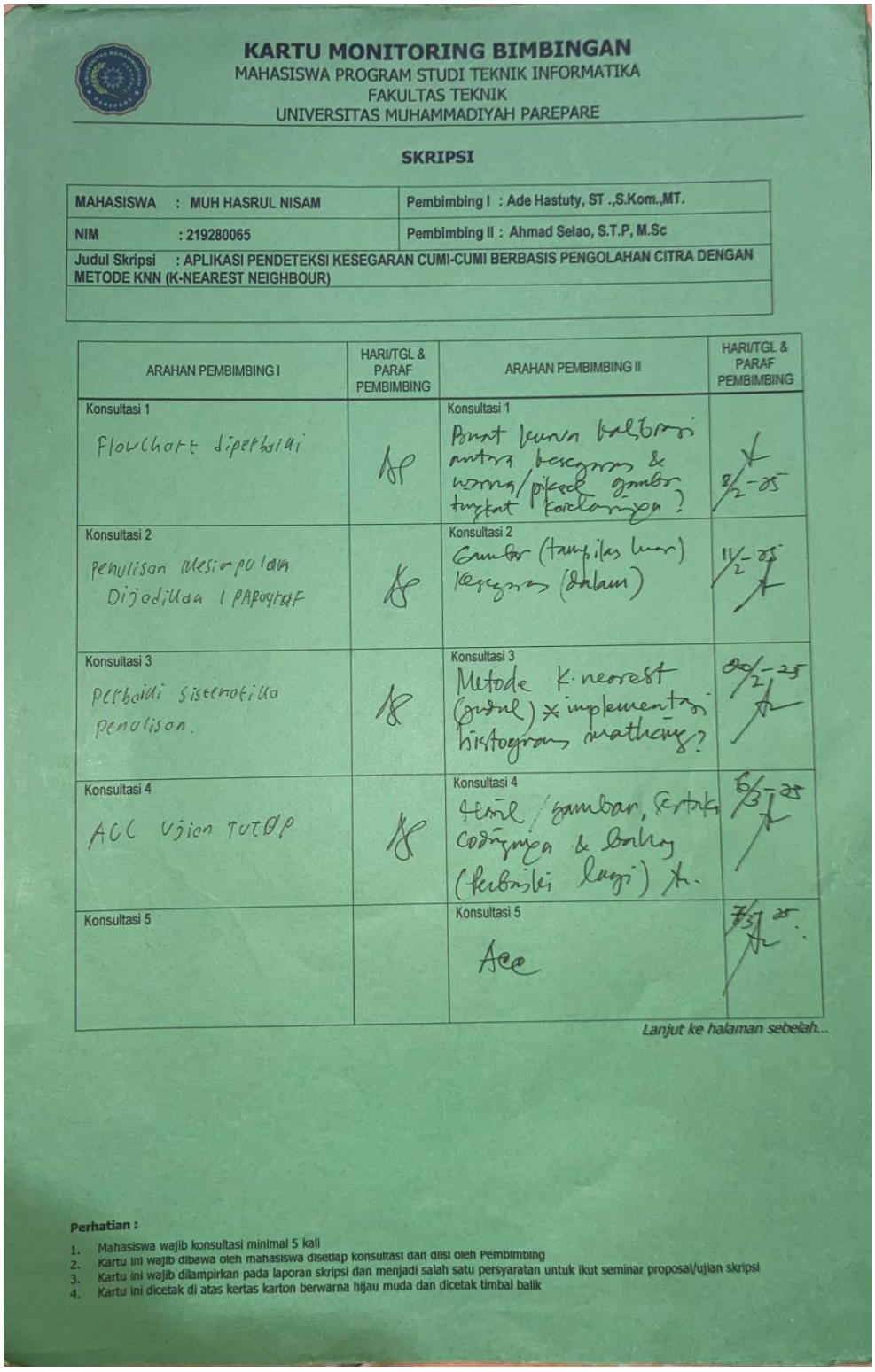
ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 1. tampilan lir Program 2. penambahan nilai Busuk		Konsultasi 1 1. Alasan faktual terhadap perubahan pada cumi2 Antara yg segar & tidak (warna, pixel cs segar/tdk	
Konsultasi 2 - LAPORAN SKRIPSI menyesuaikan ISI Aplikasi - Flowchart program dibuat - ABSTRAK. berisikan Ringkasan SKRIPSI		Konsultasi 2 2. Lampirkan analisis nilai pixel cumi2 3. perhitungan manual KNN (pixel/pixel)	
Konsultasi 3 - Pelajari bagaimana Mengukur didalam Skripsi nya		Konsultasi 3 4. Flowchart sistem KNN. yg pendeteksi segar/kadaluarsa cumi2	
Konsultasi 4		Konsultasi 4 3. pixel2 dalam nilai digital pixel mana2 posisi koordinat nya.	
Konsultasi 5		Konsultasi 5 - Lengkapi hal/page 12 - judul perbaiki penulisan	

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 - 4 Kartu Monitoring Bimbingan Skripsi



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

SKRIPSI

MAHASISWA : MUH HASRUL NISAM	Pembimbing I : Ade Hastuty, ST .,S.Kom.,MT.
NIM : 219280065	Pembimbing II : Ahmad Selao, S.T.P, M.Sc
Judul Skripsi : APLIKASI PENDETEKSI KESEGARAN CUMI-CUMI BERBASIS PENGOLAHAN CITRA DENGAN METODE KNN (K-NEAREST NEIGHBOUR)	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Flowchart diperbaiki		Konsultasi 1 Buat kurva kalibrasi antara kesegaran & warna/pixel gambar tingkat kecerahannya?	
Konsultasi 2 penulisan disesuaikan Dijadikan 1 PAPogtAF		Konsultasi 2 Gambar (tampilan luar) kesegaran (dalam)	
Konsultasi 3 perbaiki sistematika penulisan.		Konsultasi 3 Metode K-nearest (judul) x implementasi histogram matching?	
Konsultasi 4 ACC ujian TUTUP		Konsultasi 4 Hasil / gambar, serta coding & Buku (perbaiki lagi) x.	
Konsultasi 5		Konsultasi 5 Acc	

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 - 5 Sertifikat Laboratorium Praktikum V



SERTIFIKAT

Nomor : 461/PRK/LB-TI/II.3.AU/D/2023

Diberikan Kepada :

Muh. Hasrul Nisam

NIM. 219280073

Telah Menyelesaikan

PRAKTIKUM V

Laboratorium Teknik Informatika

Program Studi Teknik Informatika Fakultas Teknik

Universitas Muhammadiyah Parepare

Pada tanggal 16 Januari 2023 - 29 Maret 2023

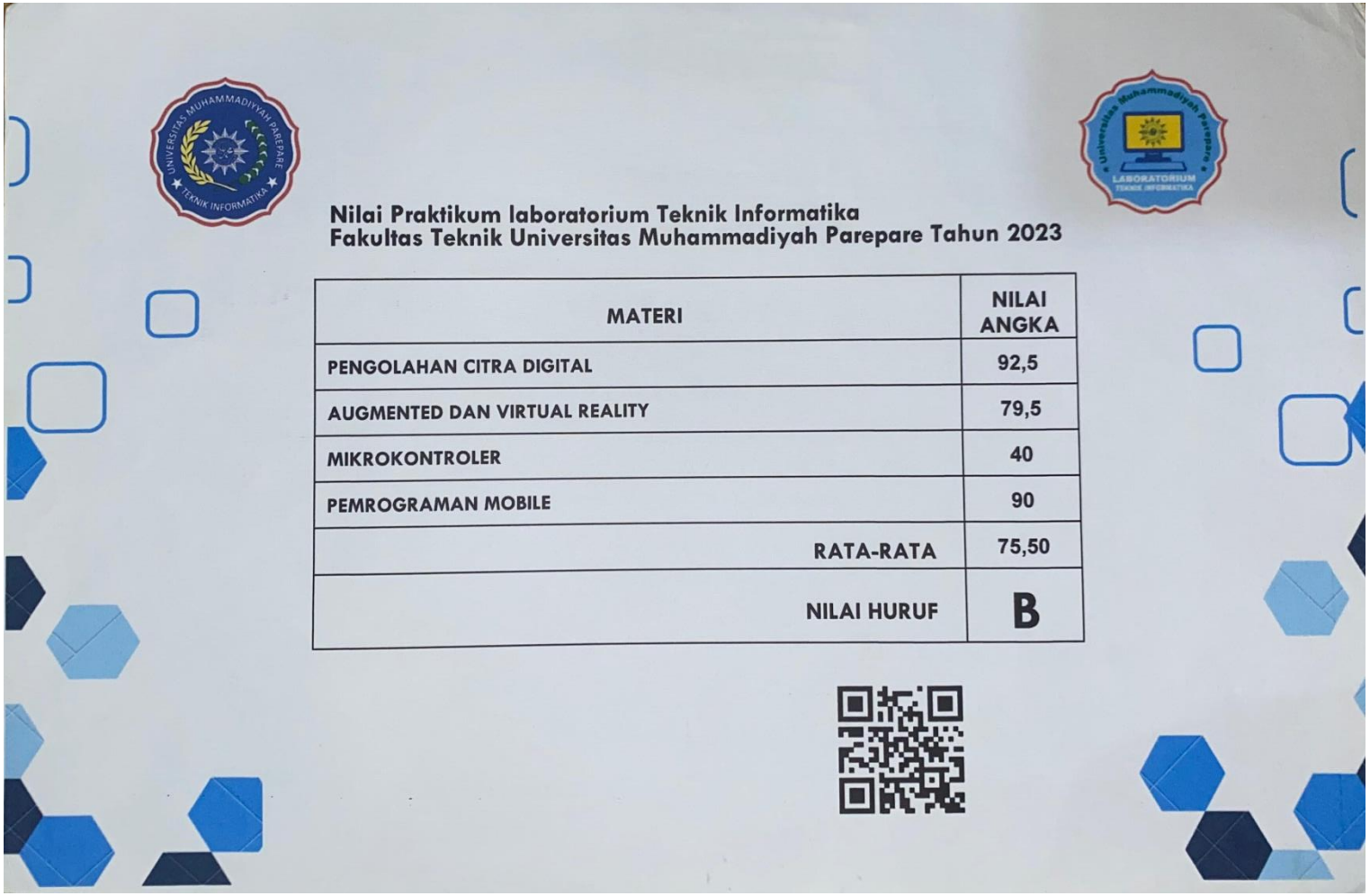
Parepare, 15 Juni 2023

KETUA PRODI
TEKNIK INOFORMATIKA

Marlina, S.Kom., M.kom
NBM. 1162 680

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	92,5
AUGMENTED DAN VIRTUAL REALITY	79,5
MIKROKONTROLER	40
PEMROGRAMAN MOBILE	90
RATA-RATA	75,50
NILAI HURUF	B