LAMPIRAN

SOURCE CODE IMPLEMENTASI ALGORITMA

Api Login

```
@app.route('/login', methods=['POST'])
def login():
conn = mysql.connector.connect(**db_config)
cursor = conn.cursor()
try:
data = request.json
username = data['username']
password = data['password']
# Melakukan enkripsi password yang diinput menggunakan MD5
hashed_password = hashlib.md5(password.encode()).hexdigest()
# Mengecek kecocokan username dan password di database
sql = "SELECT * FROM users WHERE username = %s AND password = %s"
values = (username, hashed_password)
cursor.execute(sql, values)
result = cursor.fetchone()
if result:
    # Login berhasil
    response = {
    "status": "Success",
    "message": "successfuly login to apps"
    }
else:
    # Username atau password salah
```

b. response = {"message": "email atau password yang anda masukkan salah"}

18. # Menutup kursor dan koneksi ke database
19. cursor.close()
20. conn.close()
21. return jsonify(response)
22. except Exception as e:
23. # Menutup kursor dan koneksi ke database
24. cursor.close()
25. conn.close()
26. return jsonify({"error": str(e)}),400

Api Exctract warna

1. @app.route('/api/extract_colors', methods=['POST'])
2. def testing():
3. # Ambil file gambar dari request
4. image_files = request.files.getlist('images')
5. # Periksa apakah ada file gambar dalam request
6. if not image_files:
7. return jsonify({"error": "No image files in the request."}), 400
8. # Inisialisasi list untuk menyimpan hasil ekstraksi warna
9. color_data = []
10. for image_file in image_files:
11. # Periksa tipe file apakah merupakan gambar
12. allowed_extensions = {'jpg', 'jpeg', 'png', 'gif'}
13. if '.' in image_file.filename and image_file.filename.rsplit('.', 1)[1].lower() not in allowed_extensions:
    a. continue # Lewati file yang bukan gambar

```
14. # Baca gambar menggunakan
    OpenCV
15. image_array =
    np.frombuffer(image_file.read(
    ), np.uint8)
16. image =
    cv2.imdecode(image_array,
    cv2.IMREAD_COLOR)
17. # Ekstraksi warna RGB
18. rgb_mean = np.mean(image,
    axis=(0, 1))
19. # Ekstraksi warna HSV
20. hsv = cv2.cvtColor(image,
    cv2.COLOR_BGR2HSV)
21. hsv_mean = np.mean(hsv,
    axis=(0, 1))
22. # Menyimpan hasil ekstraksi
    warna ke dalam list
23. color_data.append({
        a. 'HSV': {
        b. 'H': float(hsv_mean[0]),
        c. 'S': float(hsv_mean[1]),
        d. 'V': float(hsv_mean[2])
        e. },
        f. 'RGB': {
        g. 'R': float(rgb_mean[0]),
        h. 'G': float(rgb_mean[1]),
        i. 'B': float(rgb_mean[2])
        j. }
24. })
25. print(color_data)
26. # Mengembalikan data
    ekstraksi warna dalam respon
    JSON
27. return jsonify(color_data)
```

Api Mendapatkan Nilai Rata-Rata

```
1. @app.route('/api/hitung_rata',
   methods=['POST'])
2. def hitung_rata():
3. data = request.json
```

4. # Mengambil data kategori dan jenis dari payload
5. kategori = data['kategori']
6. jenis = data['jenis']
7. try:
    a. # Menghubungkan ke database
    b. conn = mysql.connector.connect(**db_config)
    c. cursor = conn.cursor()
    d. # Mengambil data dari database berdasarkan kategori dan jenis
    e. sql = "SELECT hasil_ekstrak FROM data WHERE jenis = %s AND kategori = %s"
    f. values = (jenis, kategori)
    g. cursor.execute(sql, values)
    h. results = cursor.fetchall()
    i. # Menginisialisasi variabel untuk menyimpan nilai rata-rata
    j. rata_h = 0
    k. rata_s = 0
    l. rata_v = 0
    m. rata_r = 0
    n. rata_g = 0
    o. rata_b = 0
    p. # Menghitung rata-rata nilai HSV dan RGB
    q. for result in results:
    r. hasil_ekstrak = eval(result[0]) # Mengubah string menjadi objek Python

```
s.  hsv_data =
    hasil_ekstrak.get('HSV',
    {})
t.  rgb_data =
    hasil_ekstrak.get('RGB',
    {})
u.  rata_h +=
    hsv_data.get('H', 0)
v.  rata_s +=
    hsv_data.get('S', 0)
w.  rata_v +=
    hsv_data.get('V', 0)
x.  rata_r +=
    rgb_data.get('R', 0)
y.  rata_g +=
    rgb_data.get('G', 0)
z.  rata_b +=
    rgb_data.get('B', 0)
aa. rata_h /= len(results)
bb. rata_s /= len(results)
cc. rata_v /= len(results)
dd. rata_r /= len(results)
ee. rata_g /= len(results)
ff. rata_b /= len(results)
dd. # Menutup kursor dan
    koneksi ke database
hh. cursor.close()
ii. conn.close()
jj. response = {
kk. "message": {
      i.  "jenis": jenis,
      ii. "keterangan":
          kategori,
      iii. "Hasil_rata-
          rata": {
      iv. "HSV": {
      v.  "H":
          round(rata_h, 6),
      vi. "S":
          round(rata_s, 6),
      vii. "V":
          round(rata_v, 6)
      viii. },
      ix. "RGB": {
      x.  "R":
          round(rata_r, 6),
```

xi. "G":
round(rata_g, 6),
xii. "B":
round(rata_b, 6)
xiii. }
xiv. }
ll. },
mm. "status":
"Success"
nn. }
oo. return jsonify(response)
8. except Exception as e:
a. # Menutup kursor dan
koneksi ke database jika
terjadi error
b. cursor.close()
c. conn.close()
d. response = {
e. "status": "Error",
f. "message": str(e)
g. }
h. return
jsonify(response), 400

Api Fuzzy

```
def fuzzyMembership(x, a, b, c):
    if x <= a or x >= c:
        return 0
    elif a < x < b:
        return (x - a) / (b - a)
    elif b <= x <= c:
        return (c - x) / (c - b)
    else:
        return 0

@app.route('/fuzzy',
methods=['POST'])
def fuzzy_logic():
    try:
        data = request.get_json()
```

1. rataH_rendah = data['Hasil_rata_rata_mentah']['HSV']['H']
2. rataS_rendah = data['Hasil_rata_rata_mentah']['HSV']['S']
3. rataV_rendah = data['Hasil_rata_rata_mentah']['HSV']['V']
4. rataH_tinggi = data['Hasil_rata_rata_matang']['HSV']['H']
5. rataS_tinggi = data['Hasil_rata_rata_matang']['HSV']['S']
6. rataV_tinggi = data['Hasil_rata_rata_matang']['HSV']['V']
7. rataH_data_uji = data['Hasil_rata_rata']['HSV']['H']
8. rataS_data_uji = data['Hasil_rata_rata']['HSV']['S']
9. rataV_data_uji = data['Hasil_rata_rata']['HSV']['V']
10. aRendahH = rataH_rendah - 2
11. bRendahH = rataH_rendah
12. cRendahH = rataH_rendah + 2
13. aTinggiH = rataH_tinggi - 2
14. bTinggiH = rataH_tinggi
15. cTinggiH = rataH_tinggi + 2
16. rendahH = fuzzyMembership(rataH_data_uji, aRendahH, bRendahH, cRendahH)

```
17. tinggiH =
    fuzzyMembership(rataH_data_
    uji, aTinggiH, bTinggiH,
    cTinggiH)

18. aRendahS = rataS_rendah - 25
19. bRendahS = rataS_rendah
20. cRendahS = rataS_rendah + 25

21. aTinggiS = rataS_tinggi - 25
22. bTinggiS = rataS_tinggi
23. cTinggiS = rataS_tinggi + 25

24. rendahS =
    fuzzyMembership(rataS_data_
    uji, aRendahS, bRendahS,
    cRendahS)
25. tinggiS =
    fuzzyMembership(rataS_data_
    uji, aTinggiS, bTinggiS,
    cTinggiS)

26. aRendahV = rataV_rendah - 18
27. bRendahV = rataV_rendah
28. cRendahV = rataV_rendah +
    18

29. aTinggiV = rataV_tinggi - 18
30. bTinggiV = rataV_tinggi
31. cTinggiV = rataV_tinggi + 18

32. rendahV =
    fuzzyMembership(rataV_data_
    uji, aRendahV, bRendahV,
    cRendahV)
33. tinggiV =
    fuzzyMembership(rataV_data_
```

uji, aTinggiV, bTinggiV,
cTinggiV)

34. rendah = (rendahH+ rendahS+
rendahV)
35. tinggi = (tinggiH+ tinggiS+
tinggiV)

36. if rendah > tinggi:
    a. tingkatKematangan =
    "Mentah"
37. elif rendah < tinggi:
    a. tingkatKematangan =
    "Matang"
38. else:
    a. tingkatKematangan =
    "Tidak Dapat
    Ditetapkan"

39. result = {
    a. "message": [
    b. {
        i. "Data_Uji": {
        ii. "HSV": {
        iii. "H":
        rataH_data_uji,
        iv. "S":
        rataS_data_uji,
        v. "V":
        rataV_data_uji
        vi. }
        vii. },
        viii. "tingkat_kemata
        ngan":
        tingkatKematan
        gan
    c. }
    d.],
    e. "status": "Success",
    f. "steps": [

g. {
    i. "variable": "H",
    ii. "a": aRendahH,
    iii. "b": bRendahH,
    iv. "c": cRendahH,
    v. "membership": rendahH
h. },
i. {
    i. "variable": "H",
    ii. "a": aTinggiH,
    iii. "b": bTinggiH,
    iv. "c": cTinggiH,
    v. "membership": tinggiH
j. },
k. {
    i. "variable": "S",
    ii. "a": aRendahS,
    iii. "b": bRendahS,
    iv. "c": cRendahS,
    v. "membership": rendahS
l. },
m. {
    i. "variable": "S",
    ii. "a": aTinggiS,
    iii. "b": bTinggiS,
    iv. "c": cTinggiS,
    v. "membership": tinggiS
n. },
o. {
    i. "variable": "V",
    ii. "a": aRendahV,
    iii. "b": bRendahV,
    iv. "c": cRendahV,
    v. "membership": rendahV
p. },
q. {
    i. "variable": "V",
    ii. "a": aTinggiV,
    iii. "b": bTinggiV,
    iv. "c": cTinggiV,

v. "membership":

tinggiV

r. }

s.]

40. }

41. return jsonify(result)

42. except Exception as e:

43. result = {

a. "message": "Error: " +

str(e),

b. "status": "Error",

c. "steps": []

44. }

45. return jsonify(result)