

BAB IV

PERANCANGAN SISTEM

4.1 Desain Prosedur

Berdasarkan uraian pada bab sebelumnya, untuk mengatasi permasalahan absensi dimasa pandemi covid-19 di Radar Cirebon, maka pada penelitian ini penulis akan mengimplementasikan pengenalan wajah menggunakan metode Local Binary Pattern Histogram berbasis website untuk mempermudah karyawan Radar Cirebon melakukan absensi sekaligus menerapkan protokol kesehatan.

4.1.1 Prosedur Pendaftaran Pengguna

1. Pengguna melakukan pendaftaran dengan menggunakan *email* dan *password*.
2. Pengguna akan menerima kode *verifikasi* yang akan dikirimkan melalui *email* yang digunakan untuk pendaftaran, masukan kode *verifikasi* ke *form* yang tersedia.
3. Setelah pendaftaran akun selesai pengguna dapat masuk ke aplikasi.

4.1.2 Prosedur Login Pengguna

1. Pengguna melakukan login dengan memasukan alamat *email* yang sudah terdaftar dan *password*.

2. Jika alamat *email/password* salah, silahkan masukan alamat *email/password* yang benar.
3. Jika pengguna berhasil *login* maka pengguna sudah bisa menggunakan aplikasi

4.1.3 Prosedur Kelola Akun Pengguna

1. Pengguna memilih menu untuk melengkapi data diri pengguna.
2. Pengguna mengisi *form* yang tersedia.
3. Pengguna mengisi nama, alamat, nomor *handphone*, dan video yang berisi wajah pengguna.
4. Pengguna menekan tombol *send data* untuk menyimpan perubahan ke *database*.

4.1.4 Prosedur Melatih Model

1. Pengguna memilih video yang berisi gambar wajahnya untuk di unggah ke server.
2. Membuat dataset wajah dari video menggunakan *library OpenCV*.
 - a. Langkah pertama yaitu mengunduh file video yang disimpan di S3.

- b. Langkah selanjutnya adalah menambahkan nama video ke database sesuai dengan data pengguna.
 - c. Kemudian muat video menggunakan perintah `'cv2.CaptureVideo'` yang bertujuan untuk memisahkan video *perframe* sehingga dapat menghasilkan foto dari sebuah video.
3. Membuat model dari *dataset* wajah menggunakan metode Local Binary Pattern Histogram.

- a. Langkah pertama adalah deteksi wajah.

Pada Langkah ini penulis mendeteksi wajah menggunakan bantuan dari *library OpenCV*. Gambar dari dataset akan dibandingkan dengan *Cascade Classifier* sehingga dapat diketahui bagian wajahnya. Gambar 4.1 merupakan contoh hasil yang diperoleh dari deteksi wajah.



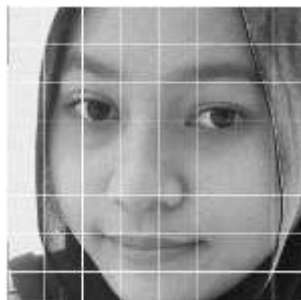
Gambar 4. 1 Hasil Deteksi Wajah

Setelah wajah terdeteksi maka gambar tersebut akan dirubah menjadi gambar abu-abu agar dapat dilatih menjadi sebuah model seperti pada gambar 4.2.



Gambar 4. 2 Merubah Warna Gambar

Langkah selanjutnya yaitu membagi wajah menjadi 64 bagian atau 8x8 seperti terlihat pada Gambar 4.3.



Gambar 4. 3 Pembagian Wilayah Wajah

b. Langkah kedua yaitu ekstraksi fitur.

Pada Langkah ini Gambar 4.3 akan dirubah menjadi bentuk array yang terdiri dari kepadatan/kecerahan sebuah piksel atau

bila dituliskan dalam angka akan dihasilkan angka antara 0-255. Tabel 4.1 merupakan contoh salah satu bagian dari 64 bagian yang ada.

Tabel 4. 1 Tingkat Kecerahan Gambar

85	170	227	28	57	227	28	85	227
57	0	227	227	170	227	0	57	57
170	0	28	227	227	0	170	170	113
0	0	0	0	0	0	0	227	0
0	0	57	0	0	0	57	0	28
28	0	0	0	57	227	142	57	57

Langkah selanjutnya akan dicari nilai LBP menggunakan Operator LBP yang telah dijelaskan pada BAB II. Tabel 4.1 akan dibagi lagi menjadi tabel kecil yang berukuran 3x3 seperti Tabel 4.2 sampai Tabel 4.7.

Tabel 4. 2 Operator LBP 1

85	170	227
57	0	227
170	0	28

Tabel 4. 3 Operator LBP 2

28	57	227
227	170	227
227	227	0

Tabel 4. 4 Operator LBP 3

28	85	227
0	57	57
170	170	113

Tabel 4. 5 Operator LBP 4

0	0	0
0	0	57
28	0	0

Tabel 4. 6 Operator LBP 5

0	0	0
0	0	0
0	57	227

Tabel 4. 7 Operator LBP 6

0	227	0
57	0	28
142	57	57

(1) Menghitung nilai LBP pada Tabel 4.2

Nilai 85 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.3 dan persamaan 1.4.

$$LBP(85) = (85 - 0)2^0 = 85 \quad (1.3)$$

$$S(85) = 1 \quad (1.4)$$

Nilai 170 akan dihitung menggunakan persamaan 1.1 kemudian persamaan 1.2 sehingga menghasilkan persamaan 1.5 dan persamaan 1.6.

$$LBP(170) = (170 - 0)2^1 = 340 \quad (1.5)$$

$$S(340) = 1 \quad (1.6)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.7 dan persamaan 1.8.

$$LBP(227) = (227 - 0)2^2 = 908 \quad (1.7)$$

$$S(908) = 1 \quad (1.8)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.9 dan persamaan 1.10.

$$LBP(227) = (227 - 0)2^3 = 1816 \quad (1.9)$$

$$S(1816) = 1 \quad (1.10)$$

Nilai 28 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.11 dan persamaan 1.12.

$$LBP(28) = (28 - 0)2^4 = 448 \quad (1.11)$$

$$S(448) = 1 \quad (1.12)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.13 dan persamaan 1.14.

$$LBP(0) = (0 - 0)2^5 = 0 \quad (1.13)$$

$$S(0) = 1 \quad (1.14)$$

Nilai 170 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.15 dan persamaan 1.16.

$$LBP(170) = (170 - 0)2^6 = 4480 \quad (1.15)$$

$$S(4480) = 1 \quad (1.16)$$

Nilai 57 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.17 dan persamaan 1.18.

$$LBP(57) = (57 - 0)2^7 = 7296 \quad (1.17)$$

$$S(7296) = 1 \quad (1.18)$$

Dari persamaan 1.3 sampai persamaan 1.18 dihasilkan nilai LBP berturut-turut yaitu 1,1,1,1,1,1,1,1 yang bila diubah ke bentuk biner maka senilai dengan 255 yang merupakan nilai asli atau nilai tengah. Hasil perhitungan LBP yang diperoleh akan dikalikan dengan bilangan biner antara 1 hingga 128 yang hasilnya dapat dilihat pada Tabel 4.8.

Tabel 4. 8 Hasil Operator LBP 1

1	2	4
8	255	16
32	64	128

(2) Menghitung nilai LBP pada Tabel 4.3

Nilai 28 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.19 dan persamaan 1.20.

$$LBP(28) = (28 - 170)2^0 = -142 \quad (1.19)$$

$$S(-142) = 0 \quad (1.20)$$

Nilai 57 akan dihitung menggunakan persamaan 1.1 kemudian persamaan 1.2 sehingga menghasilkan persamaan 1.21 dan persamaan 1.22.

$$LBP(57) = (57 - 170)2^1 = -226 \quad (1.21)$$

$$S(-226) = 0 \quad (1.22)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.23 dan persamaan 1.24.

$$LBP(227) = (227 - 170)2^2 = 228 \quad (1.23)$$

$$S(228) = 1 \quad (1.24)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.25 dan persamaan 1.26.

$$LBP(227) = (227 - 170)2^3 = 456 \quad (1.25)$$

$$S(456) = 1 \quad (1.26)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.27 dan persamaan 1.28.

$$LBP(0) = (0 - 170)2^4 = -2720 \quad (1.27)$$

$$S(-2720) = 0 \quad (1.28)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.29 dan persamaan 1.30.

$$LBP(227) = (227 - 170)2^5 = 1824 \quad (1.29)$$

$$S(1824) = 1 \quad (1.30)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.31 dan persamaan 1.32.

$$LBP(227) = (227 - 170)2^6 = 3648 \quad (1.31)$$

$$S(3648) = 1 \quad (1.32)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.33 dan persamaan 1.34.

$$LBP(227) = (227 - 170)2^7 = 7286 \quad (1.33)$$

$$S(7296) = 1 \quad (1.34)$$

Dari persamaan 1.19 sampai persamaan 1.34 dihasilkan nilai LBP berturut-turut yaitu 0,0,1,1,0,1,1,1 yang bila diubah ke bentuk biner maka senilai dengan 55 yang merupakan nilai asli atau nilai tengah. Hasil perhitungan LBP yang diperoleh akan dikalikan dengan bilangan biner antara 1 hingga 128 yang hasilnya dapat dilihat pada Tabel 4.9.

Tabel 4. 9 Hasil Operator LBP 2

0	0	4
8	55	16
32	64	0

(3) Menghitung nilai LBP pada Tabel 4.4

Nilai 28 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.35 dan persamaan 1.36.

$$LBP(28) = (28 - 57)2^0 = -29 \quad (1.35)$$

$$S(-29) = 0 \quad (1.36)$$

Nilai 85 akan dihitung menggunakan persamaan 1.1 kemudian persamaan 1.2 sehingga menghasilkan persamaan 1.37 dan persamaan 1.38.

$$LBP(85) = (85 - 57)2^1 = 56 \quad (1.37)$$

$$S(56) = 1 \quad (1.38)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.39 dan persamaan 1.40.

$$LBP(227) = (227 - 57)2^2 = 680 \quad (1.39)$$

$$S(680) = 1 \quad (1.40)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.41 dan persamaan 1.42.

$$LBP(57) = (57 - 57)2^3 = 0 \quad (1.41)$$

$$S(0) = 1 \quad (1.42)$$

Nilai 113 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.43 dan persamaan 1.44.

$$LBP(113) = (113 - 57)2^4 = 896 \quad (1.43)$$

$$S(896) = 1 \quad (1.44)$$

Nilai 170 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.45 dan persamaan 1.46.

$$LBP(170) = (170 - 57)2^5 = 3616 \quad (1.45)$$

$$S(3616) = 1 \quad (1.46)$$

Nilai 170 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.47 dan persamaan 1.48.

$$LBP(170) = (170 - 57)2^6 = 7232 \quad (1.47)$$

$$S(7232) = 1 \quad (1.48)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.49 dan persamaan 1.50.

$$LBP(0) = (0 - 57)2^7 = -7296 \quad (1.49)$$

$$S(-7296) = 0 \quad (1.50)$$

Dari persamaan 1.35 sampai persamaan 1.50 dihasilkan nilai LBP berturut-turut yaitu 0,1,1,1,1,1,0 yang bila diubah ke bentuk biner maka senilai dengan 126 yang merupakan nilai asli atau nilai tengah. Hasil perhitungan LBP yang diperoleh akan dikalikan dengan bilangan biner antara 1 hingga 128 yang hasilnya dapat dilihat pada Tabel 4.10.

Tabel 4. 10 Hasil Operator LBP 3

0	2	4
0	126	0
32	64	128

(4) Menghitung nilai LBP pada Tabel 4.5

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.51 dan persamaan 1.52.

$$LBP(0) = (0 - 0)2^0 = 0 \quad (1.51)$$

$$S(0) = 1 \quad (1.52)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 kemudian persamaan 1.2 sehingga menghasilkan persamaan 1.53 dan persamaan 1.54.

$$LBP(0) = (0 - 0)2^1 = 0 \quad (1.53)$$

$$S(0) = 1 \quad (1.54)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.55 dan persamaan 1.56.

$$LBP(0) = (0 - 0)2^2 = 0 \quad (1.55)$$

$$S(0) = 1 \quad (1.56)$$

Nilai 57 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.57 dan persamaan 1.58.

$$LBP(57) = (57 - 0)2^3 = 228 \quad (1.57)$$

$$S(228) = 1 \quad (1.58)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.59 dan persamaan 1.60.

$$LBP(0) = (0 - 0)2^4 = 0 \quad (1.59)$$

$$S(0) = 1 \quad (1.60)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.61 dan persamaan 1.62.

$$LBP(0) = (0 - 0)2^5 = 0 \quad (1.61)$$

$$S(0) = 1 \quad (1.62)$$

Nilai 28 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.63 dan persamaan 1.64.

$$LBP(28) = (28 - 0)2^6 = 1792 \quad (1.63)$$

$$S(1792) = 1 \quad (1.64)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.65 dan persamaan 1.66.

$$LBP(0) = (0 - 0)2^7 = 0 \quad (1.65)$$

$$S(0) = 1 \quad (1.66)$$

Dari persamaan 1.51 sampai persamaan 1.66 dihasilkan nilai LBP berturut-turut yaitu 0,0,0,1,0,0,1,0 yang bila diubah ke bentuk biner maka senilai dengan 18 yang merupakan nilai asli atau nilai tengah. Hasil perhitungan LBP yang diperoleh akan dikalikan dengan bilangan biner antara 1 hingga 128 yang hasilnya dapat dilihat pada Tabel 4.11.

Tabel 4. 11 Hasil Operator LBP 4

0	0	0
0	18	16
32	0	0

(5) Menghitung nilai LBP pada Tabel 4.6

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.67 dan persamaan 1.68.

$$LBP(0) = (0 - 0)2^0 = 0 \quad (1.67)$$

$$S(0) = 1 \quad (1.68)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 kemudian persamaan 1.2 sehingga menghasilkan persamaan 1.69 dan persamaan 1.70.

$$LBP(0) = (0 - 0)2^1 = 0 \quad (1.69)$$

$$S(0) = 1 \quad (1.70)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.71 dan persamaan 1.72.

$$LBP(0) = (0 - 0)2^2 = 0 \quad (1.71)$$

$$S(0) = 1 \quad (1.72)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.73 dan persamaan 1.74.

$$LBP(0) = (0 - 0)2^3 = 0 \quad (1.73)$$

$$S(0) = 1 \quad (1.74)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.75 dan persamaan 1.76.

$$LBP(227) = (227 - 0)2^4 = 3632 \quad (1.75)$$

$$S(3632) = 1 \quad (1.76)$$

Nilai 57 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.77 dan persamaan 1.78.

$$LBP(57) = (57 - 0)2^5 = 1824 \quad (1.77)$$

$$S(1824) = 1 \quad (1.78)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.79 dan persamaan 1.80.

$$LBP(0) = (0 - 0)2^6 = 0 \quad (1.79)$$

$$S(0) = 1 \quad (1.80)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.81 dan persamaan 1.82.

$$LBP(0) = (0 - 0)2^7 = 0 \quad (1.81)$$

$$S(0) = 1 \quad (1.82)$$

Dari persamaan 1.67 sampai persamaan 1.82 dihasilkan nilai LBP berturut-turut yaitu 0,0,0,0,1,1,0,0 yang bila diubah ke bentuk biner maka senilai dengan 12. Hasil perhitungan LBP

yang diperoleh akan dikalikan dengan bilangan biner antara 1 hingga 128 yang hasilnya dapat dilihat pada Tabel 4.12.

Tabel 4. 12 Hasil Operator LBP 5

0	0	0
0	12	0
0	64	128

(6) Menghitung nilai LBP pada Tabel 4.7

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.83 dan persamaan 1.84.

$$LBP(0) = (0 - 0)2^0 = 0 \quad (1.83)$$

$$S(0) = 1 \quad (1.84)$$

Nilai 227 akan dihitung menggunakan persamaan 1.1 kemudian persamaan 1.2 sehingga menghasilkan persamaan 1.85 dan persamaan 1.86.

$$LBP(227) = (227 - 0)2^1 = 454 \quad (1.85)$$

$$S(454) = 1 \quad (1.86)$$

Nilai 0 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.87 dan persamaan 1.88.

$$LBP(0) = (0 - 0)2^2 = 0 \quad (1.87)$$

$$S(0) = 1 \quad (1.88)$$

Nilai 28 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.89 dan persamaan 1.90.

$$LBP(28) = (28 - 0)2^3 = 224 \quad (1.89)$$

$$S(224) = 1 \quad (1.90)$$

Nilai 57 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.91 dan persamaan 1.92.

$$LBP(57) = (57 - 0)2^4 = 912 \quad (1.91)$$

$$S(912) = 1 \quad (1.92)$$

Nilai 57 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.93 dan persamaan 1.94.

$$LBP(57) = (57 - 0)2^5 = 1824 \quad (1.93)$$

$$S(1824) = 1 \quad (1.94)$$

Nilai 142 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.95 dan persamaan 1.96.

$$LBP(142) = (142 - 0)2^6 = 9088 \quad (1.95)$$

$$S(9088) = 1 \quad (1.96)$$

Nilai 57 akan dihitung menggunakan persamaan 1.1 dan 1.2 sehingga menghasilkan persamaan 1.97 dan persamaan 1.98.

$$LBP(57) = (57 - 0)2^7 = 7296 \quad (1.97)$$

$$S(7296) = 1 \quad (1.98)$$

Dari persamaan 1.83 sampai persamaan 1.98 dihasilkan nilai LBP berturut-turut yaitu 0,1,0,1,1,1,1,1 yang bila diubah ke bentuk biner maka senilai dengan 95. Hasil perhitungan LBP yang diperoleh akan dikalikan dengan bilangan biner antara 1 hingga 128 yang hasilnya dapat dilihat pada Tabel 4.13.

Tabel 4. 13 Hasil Operator LBP 6

0	2	0
8	95	16
32	64	128

Tabel 4.8 hingga Tabel 4.13 dapat digabungkan kembali menjadi satu tabel namun hanya titik pusatnya saja yang digunakan dan dapat dilihat pada Tabel 4.14.

Tabel 4. 14 Nilai LBP Dari Satu Bagian Gambar Wajah

1	2	4	0	0	4	0	2	4
8	255	16	8	55	16	0	126	0
32	64	128	32	64	0	32	64	128
0	0	0	0	0	0	0	2	0
0	18	16	0	12	0	8	95	16
32	0	0	0	64	128	32	64	128

Langkah diatas diulang hingga telah diketahui nilai LBP dari setiap bagiannya. Selanjutnya Nilai LBP dari setiap bagian disatukan hingga membentuk histogram seperti pada Gambar 2.4.

- c. Membuat *dataset* yang berisi nilai LBP dari setiap gambar wajah.

Dalam tahap ini penulis menggunakan *library OpenCV* untuk membuat dataset yang berisi nilai LBP dari setiap gambar wajah yang ada di *dataset* wajah.

4.1.5 Prosedur Memprediksi Wajah

1. Pengguna mengambil gambar wajah.
2. Pengguna mengunggah foto wajah ke s3.
3. Sistem akan mengunduh gambar yang dikirimkan pengguna.
4. Pengenalan wajah menggunakan metode Local Binary Pattern Histogram.
 - a. Mendeteksi wajah.

Pada Langkah ini penulis mendeteksi wajah menggunakan bantuan dari *library OpenCV*. Gambar yang diunduh akan dibandingkan dengan *Cascade Classifier* sehingga dapat diketahui bagian wajahnya. Gambar 4.1 merupakan contoh hasil yang diperoleh dari deteksi wajah.

Setelah wajah terdeteksi maka gambar tersebut akan dirubah menjadi gambar abu-abu agar dapat dilatih menjadi sebuah

model seperti pada gambar 4.2. Langkah selanjutnya yaitu membagi wajah menjadi 64 bagian atau 8x8 seperti terlihat pada Gambar 4.3. Langkah selanjutnya sama seperti Langkah-langkah ekstraksi fitur pada Prosedur Memprediksi Wajah.

Selanjutnya setelah melakukan ekstraksi fitur dan mendapatkan nilai Histogram LBP maka dilakukan pengukuran jarak antara histogram LBP dengan histogram dari dataset yang sudah dilatih sebelumnya. Mengukur jarak dari dua histogram dapat menggunakan persamaan *euclidean distance* seperti pada persamaan 1.99.

$$D = \sqrt{\sum_{i=1}^n (hist1_i - hist2_i)^2} \quad (1.99)$$

Hasil dari persamaan 1.99 merupakan jarak antara kedua histogram, nilai ini tidak memiliki batas namun semangin kecil (mendekati 0) nilainya maka tingkat kemiripan semakin tinggi maupun sebaliknya jika nilainya semakin besar (menjauhi 0) maka tingkat kemiripannya semakin kecil.

4.2 Desain Dokumen dan informasi

4.2.1 Desain Dokumen

Desain dokumen baru pada *website* website Absensi Online Radar Cirebon setelah diimplementasikan dengan teknik pengenalan wajah adalah sebagai berikut:

1. Dokumen pengguna

Nama dokumen : Data Pengguna
 Fungsi : Untuk melengkapi data pengguna
 Sumber : Pengguna
 Bentuk : Formulir
 Atribut : nama, *email*, nomor *handphone*, id wajah dan alamat

2. Dokumen absensi

4.2.2 Desain Informasi

1. Informasi Data Pengguna

Nama dokumen : Data Pengguna
 Fungsi : Untuk melengkapi data pengguna
 Sumber : Pengguna
 Bentuk : Formulir
 Atribut : nama, *email*, nomor *handphone*, id wajah dan alamat

2. Informasi Absensi

Nama dokumen : Data Absensi

Fungsi : Untuk bukti absensi

Sumber : Pengguna/ Sistem

Bentuk : Formulir

Atribut : *_typename, confidence, face_id, time, user_id, year, dan location.*

Nama dokumen : Data Absensi

Fungsi : Untuk bukti absensi

Sumber : Pengguna/ Sistem

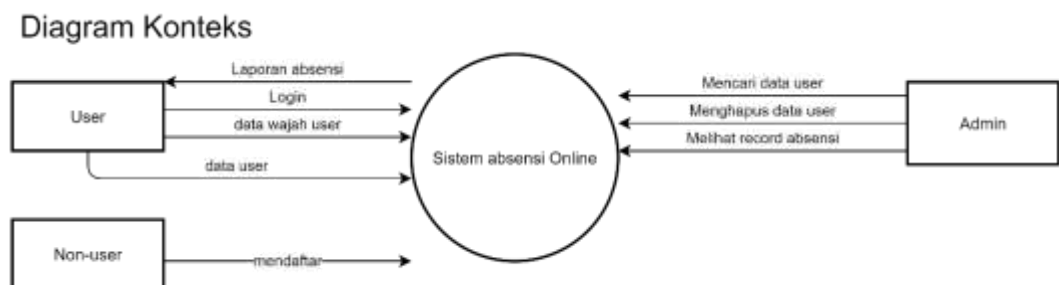
Bentuk : Formulir

Atribut : *_typename, confidence, face_id, time, user_id, year, dan location.*

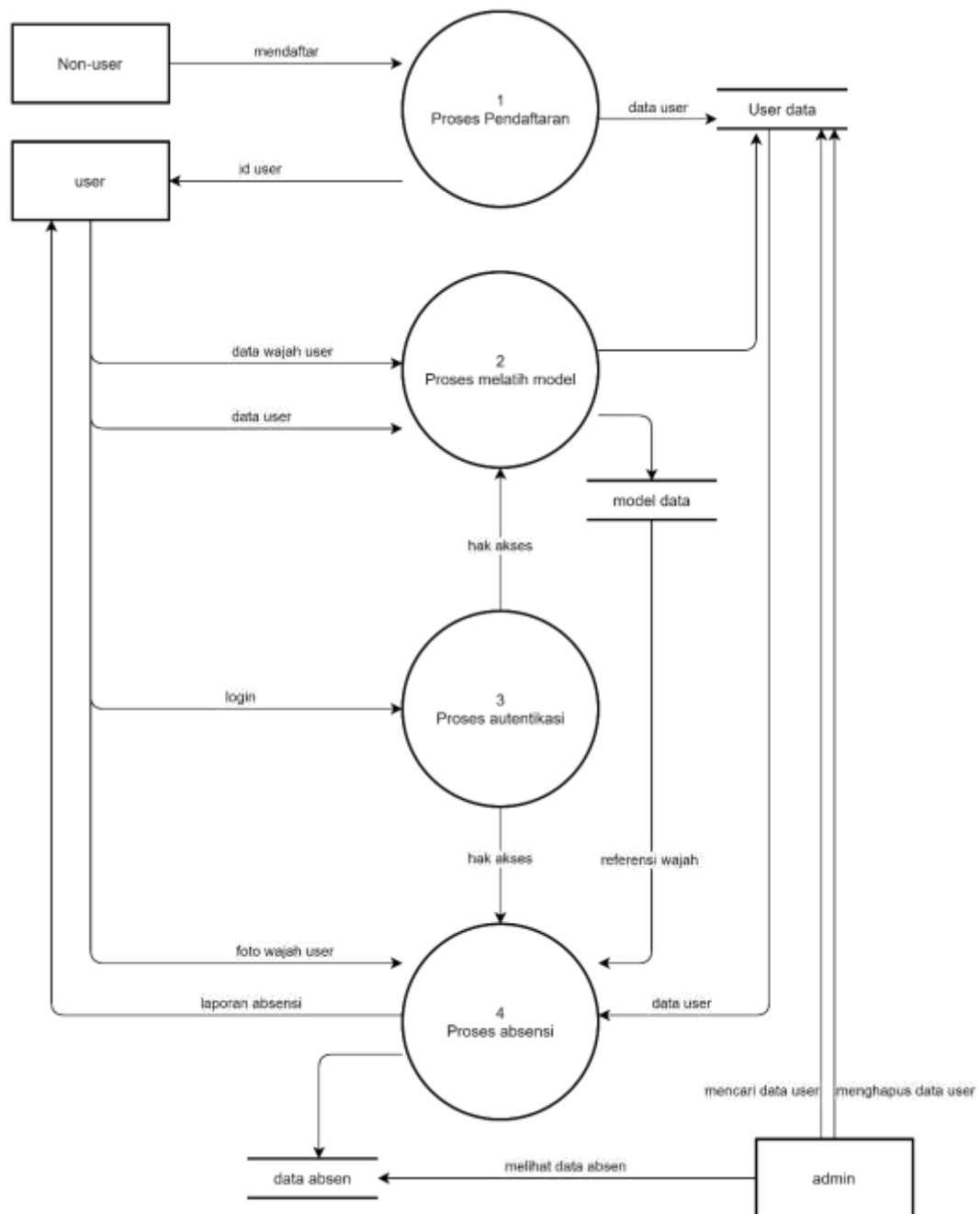
4.3 Desain Aliran Data

4.3.1 Data Flow Diagram

Berikut adalah Diagram Konteks yang penulis buat pada penelitian ini yang dapat dilihat pada Gambar 4.4.



Gambar 4. 4 Diagram Konteks



Gambar 4. 5 DFD Level 1

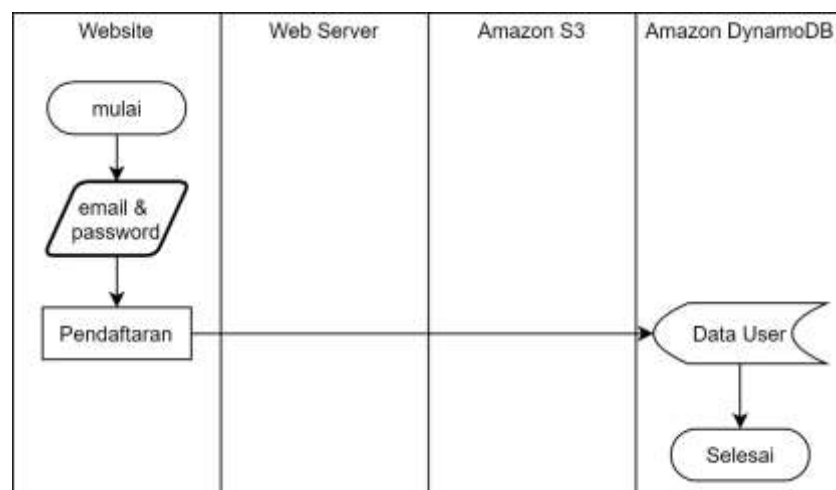
Dari Gambar 4.4 dapat diturunkan menjadi *Data Flow Diagram* level 1 yang dapat dilihat pada Gambar 4.5.

4.3.2 Flowmap

Untuk mempermudah dalam pembuatan program penulis menggunakan *flowmap* sebagai acuan aliran data dalam sebuah program. *Flowmap* aliran data program dibagi berdasarkan prosedur sebagai berikut:

1. Flowmap Pendaftaran

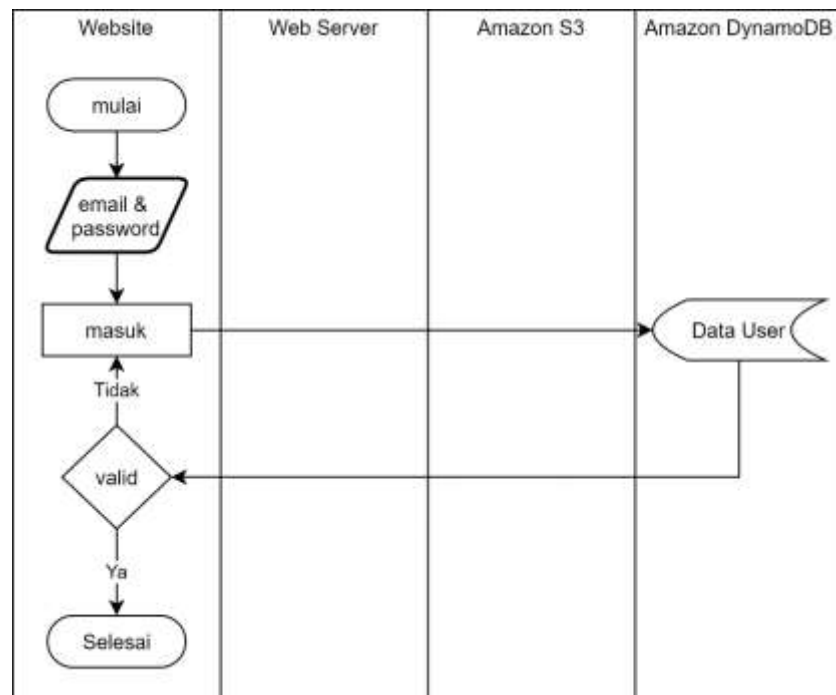
Berikut adalah *Flowmap* Pendaftaran yang dapat dilihat pada Gambar 4.6.



Gambar 4. 6 Flowmap Pendaftaran

2. Flowmap Login

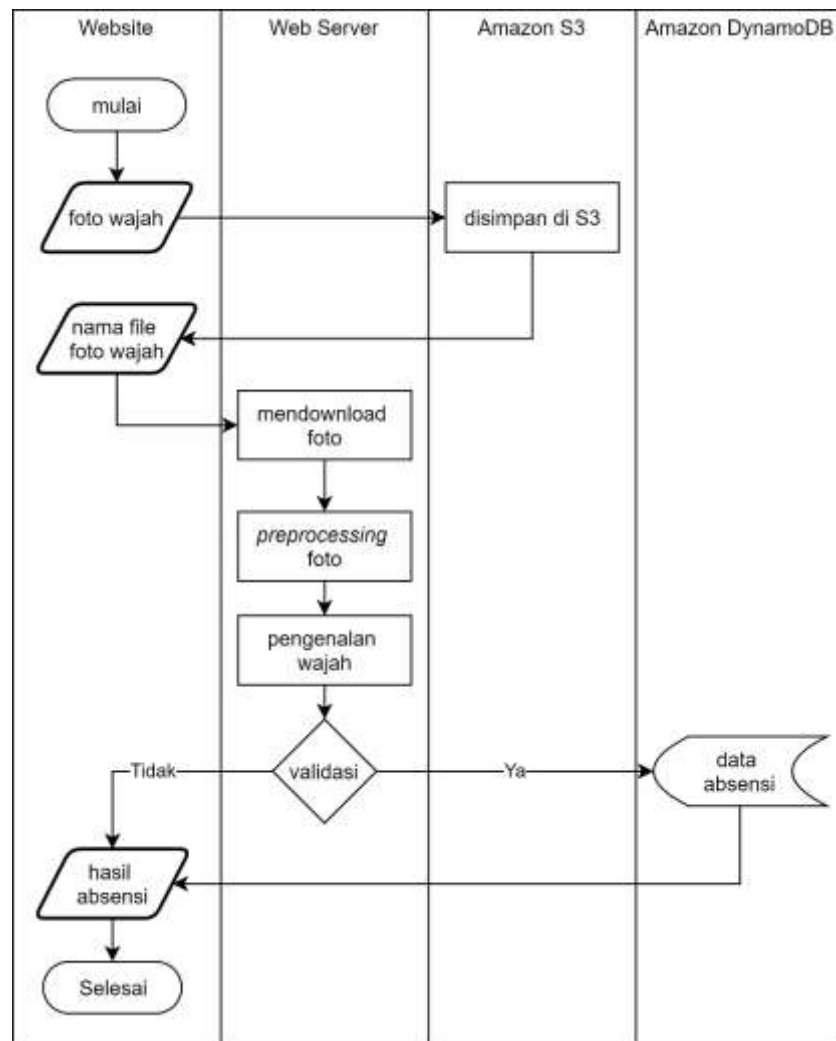
Berikut adalah *flowmap login* yang dapat dilihat pada Gambar 4.7.



Gambar 4. 7 Flowmap Login

3. Flowmap Memprediksi Wajah (Absensi)

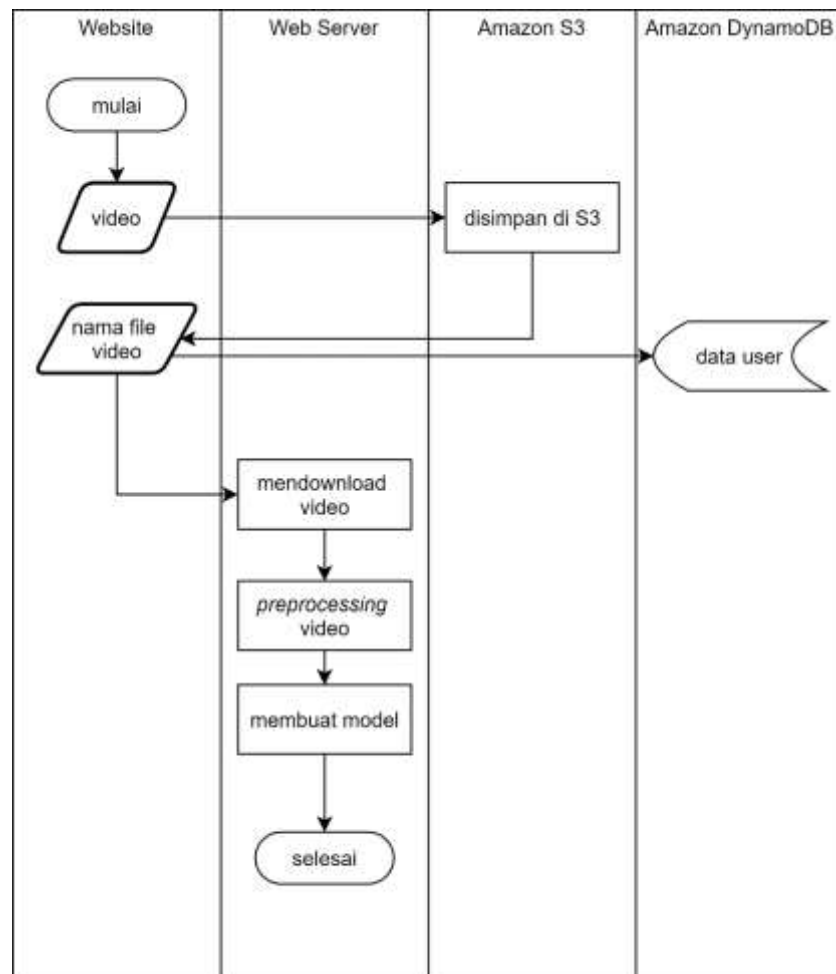
Berikut adalah *flowmap* memprediksi wajah yang dapat dilihat pada Gambar 4.8.



Gambar 4. 8 Flowmap Memprediksi Wajah

4. Flowmap Melatih Model

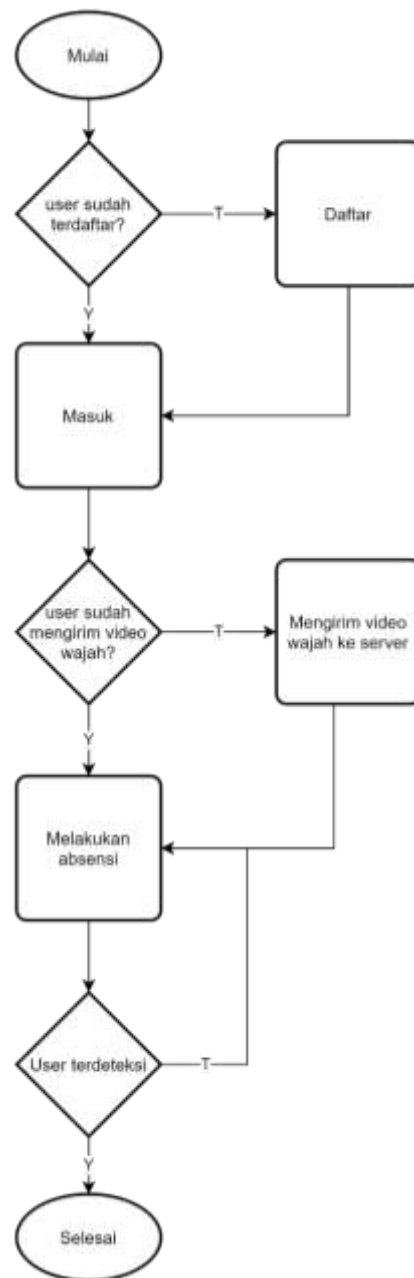
Berikut adalah *flowmap* melatih model yang dapat dilihat pada Gambar 4.9.



Gambar 4. 9 Flowmap Melatih Model

4.3.3 Flowchart

Untuk mempermudah dalam pembuatan program penulis menggunakan *flowchart* sebagai acuan alur program. *Flowchart* alur program ditunjukkan oleh Gambar 4.10 berikut:



Gambar 4. 10 Flowchart Program

4.3.4 Skema Database

Database nosql adalah *database* yang tidak memiliki perintah SQL dan konsep penyimpanannya semistuktural atau tidak struktural dan tidak harus memiliki relasi layaknya tabel-

tabel MySQL. Dalam penelitian ini penulis memilih DynamoDB sebagai *database nosql* karena DynamoDB memiliki kecepatan baca yang konstan sehingga cocok untuk aplikasi *realtime*.

1. Skema Tabel Pengguna

Berikut adalah skema yang penulis buat dalam bentuk json untuk tabel pengguna yang dapat dilihat pada Tabel 4.15.

Tabel 4. 15 Skema Tabel Pengguna

```
{
  "address": {
    "S": ""
  },
  "complete": {
    "BOOL": ""
  },
  "edited_at": {
    "S": ""
  },
  "face_id": {
    "S": ""
  },
  "id": {
    "S": ""
  },
  "name": {
    "S": ""
  },
  "phone_number": {
    "S": ""
  },
  "picture": {
    "S": ""
  }
}
```

2. Skema Tabel Data Absensi

Berikut adalah skema yang penulis buat dalam bentuk json untuk tabel data absensi yang dapat dilihat pada Tabel 4.16.

Tabel 4. 16 Skema Tabel Data Absensi

```
{
  "__typename": {
    "S": ""
  },
  "confidence": {
    "S": ""
  },
  "face_id": {
    "S": ""
  },
  "time": {
    "S": ""
  },
  "user_id": {
    "S": ""
  },
  "year": {
    "S": ""
  },
  "location": {
    "S": ""
  }
}
```

3. Skema Model

Berikut adalah skema yang dihasilkan oleh OpenCV dalam bentuk yaml yang dapat dilihat pada Tabel 4.17.

Tabel 4. 17 Skema Model

```
opencv_lbphfaces:
  threshold:
  radius:
  neighbors:
  grid_x:
  grid_y:
  histogram:
    - !!opencv-matrix
      rows:
      cols:
      dt:
      data:
  labels: !!opencv-matrix
    rows:
    cols:
    dt:
    data:
  labelsInfo:
    []
```

4.4 Desain Interface dan Struktur Menu

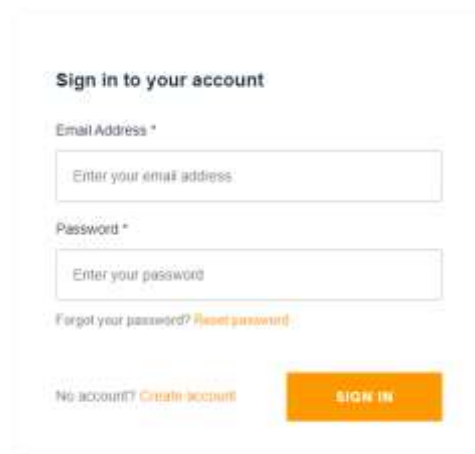
4.4.1 Desain Interface

Desain adalah sesuatu yang sangat dibutuhkan dalam membuat sebuah sistem, tujuannya agar sistem tersebut mudah untuk digunakan oleh pengguna. Adapun desain *interface* untuk

website Absensi Online Radar Cirebon setelah diimplementasikan dengan teknik pengenalan wajah adalah sebagai berikut:

1. Desain Halaman Login

Berikut adalah Desain Halaman *Login* yang dapat dilihat pada Gambar 4.11:



The image shows a login form titled "Sign in to your account". It contains two input fields: "Email Address *" with a placeholder "Enter your email address" and "Password *" with a placeholder "Enter your password". Below the password field is a link "Forgot your password? [Reset password](#)". At the bottom left, there is a link "No account? [Create account](#)". At the bottom right, there is an orange button labeled "SIGN IN".

Gambar 4. 11 Desain Halaman Login

2. Desain Halaman Dashboard

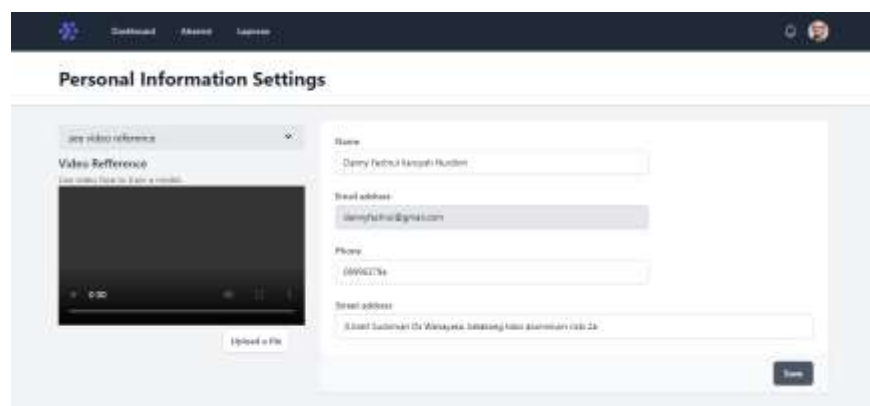
Berikut adalah Desain Halaman Setting yang dapat dilihat pada Gambar 4.12:



Gambar 4. 12 Desain Halaman Dashboard

3. Desain Halaman Setting

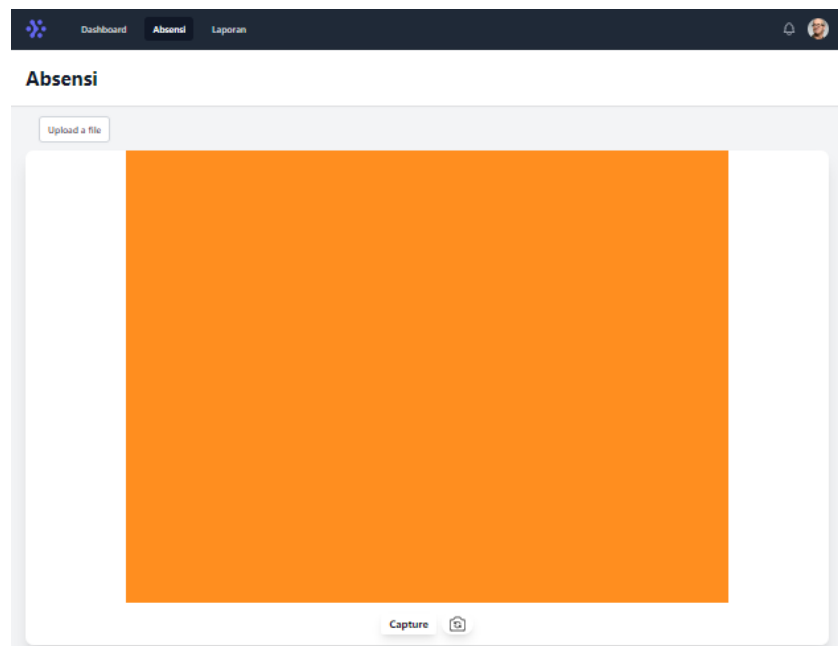
Berikut adalah Desain Halaman *Setting* yang dapat dilihat pada Gambar 4.13:



Gambar 4. 13 Desain Halaman Setting

4. Desain Halaman Absensi

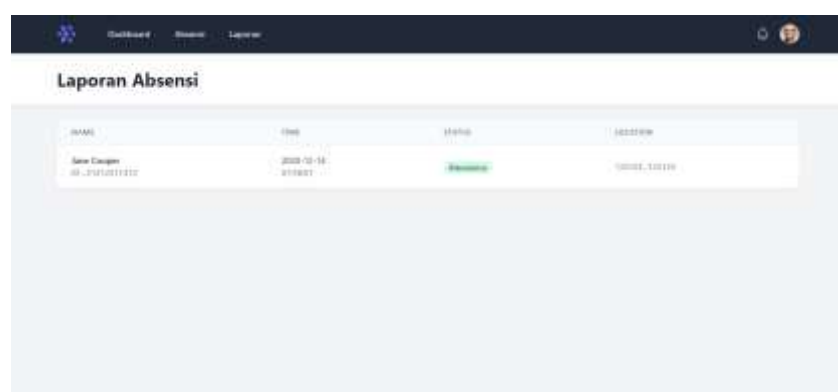
Berikut adalah Desain Halaman Absensi yang dapat dilihat pada Gambar 4.14:



Gambar 4. 14 Desain Halaman Absensi

5. Desain Halaman Laporan

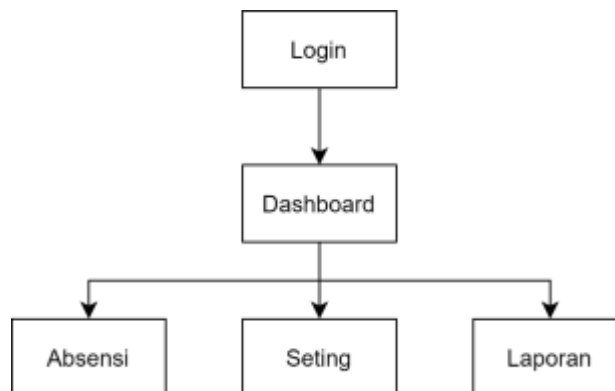
Berikut adalah Desain Halaman Laporan yang dapat dilihat pada Gambar 4.15:



Gambar 4. 15 Desain Halaman Laporan

4.4.2 Struktur Menu

Berikut adalah struktur dari website Absensi Online Radar Cirebon setelah diimplementasikan dengan teknik pengenalan wajah dapat dilihat pada Gambar 4.16:



Gambar 4. 16 Struktur Menu Sistem

4.5 Implementasi Sistem

4.5.1 Konfigurasi Perangkat Lunak

Kebutuhan perangkat lunak untuk mengelola server pengenalan wajah guna mendukung *website* Unit Robotika STIKOM Poltek Cirebon yang dijalankan dalam Amazon EC2 adalah sebagai berikut:

1. Linux Ubuntu 18.04.
2. Nginx.
3. Flask.
4. Unicorn.

Sedangkan perangkat lunak yang digunakan untuk pengembangan website adalah Visual Studio Code dan Postman.

4.5.2 Konfigurasi Perangkat Keras

Dalam pembuatan *website* Absensi Online Radar Cirebon penulis hanya menggunakan perangkat keras pada saat pengembangan *website*, karena penulis menggunakan Amazon Web Service sebagai sarana *hosting* sehingga tidak membutuhkan perangkat keras secara langsung. Berikut perangkat keras yang digunakan pada saat pengembangan *website*.

1. Processor Intel Core i3 7100.
2. Ram 4 GB.
3. Penyimpanan 250 GB.
4. Mouse, keyboard, monitor, dan webcam.

4.5.3 Implementasi Program

1. Sintaks Pembuatan Dataset

Berikut adalah Sintaks Pembuatan Dataset yang dapat dilihat pada Tabel 4.18:

Tabel 4. 18 Sintaks Membuat Dataset

```
import cv2
import os
import numpy as np
```

Tabel 4. 18 Sintaks Membuat Dataset

```

import time
import matplotlib.pyplot as plt
import sys
import argparse as arg

class Train():

    def __init__(self,
face_cascade,config,username):
        self.username = username
        # self.current_dir = current_dir
        self._Face_Cascade =
cv2.CascadeClassifier(face_cascade)
        self.dataset_path("dataset/")
        self.recognizer =
cv2.face.LBPHFaceRecognizer_create(config[0],
config[1], config[2], config[3],config[4])

    def dataset_path(self, path):
        dir = os.path.dirname(path)
        if not os.path.exists(dir):
            os.makedirs(dir)

    def ReadName(self):
        NAME = []
        with open ("users.txt", "r") as f:
            for line in f:

NAME.append(line.split(",") [1].rstrip())
        return NAME

    def AddUser(self):
        # Name = input('\n[INFO] Masukkan nama
user : ')
        Name = self.username
        info = open('users.txt', "a+")
        ID = len(open("users.txt").readlines(
))
        info.write(str(ID) + "," + Name + "\n")
        print("\n[INFO] Tambah user berhasil,
ID:" + str(ID))
        info.close
        return ID

    def getImageWithLabels(self,path):
        imagePath = [os.path.join(path, f) for
f in os.listdir(path)]
        faceSamples = []
        ids = []
        for imagePath in imagePath:

```

Tabel 4. 18 Sintaks Membuat Dataset

```

        img = cv2.imread(imagePath, 0)
        img_numpy = np.array(img, 'uint8')
        id = int(os.path.split(imagePath)[-1].split('.')[1])
        faceSamples.append(img_numpy)
        ids.append(id)
    return faceSamples, ids

    def train(self, path, file_name):
        # os.chdir('tmp')
        # real_path =
        '{0}/{1}'.format(self.current_dir, path)
        print("\n[INFO] Training wajah sedang dimulai...")
        time.sleep(1)
        faces, ids =
self.getImageWithLabels(path)

self.recognizer.update(faces, np.array(ids))
self.recognizer.write(file_name)
print("\n[INFO] Model sukses melatih user ID: {0}".format(len(np.unique(ids))))
print("\n[INFO] Menutup program")
return len(np.unique(ids))

    def create_Rect(self, Image, face, color):
        x,y,w,h = face
        cv2.line(Image, (x, y), (int(x + (w/5)),y), color, 2)
        cv2.line(Image, (int(x+((w/5)*4)), y), (x+w, y), color, 2)
        cv2.line(Image, (x, y), (x,int(y+(h/5))), color, 2)
        cv2.line(Image, (x+w, y), (x+w, int(y+(h/5))), color, 2)
        cv2.line(Image, (x, int(y+(h/5*4))), (x, y+h), color, 2)
        cv2.line(Image, (x, int(y+h)), (x + int(w/5) ,y+h), color, 2)
        cv2.line(Image, (x+int((w/5)*4), y+h), (x + w, y + h), color, 2)
        cv2.line(Image, (x+w, int(y+(h/5*4))), (x+w, y+h), color, 2)

    def
createDataset(self, samples, cam, dataset_name):
        fig, axs =
plt.subplots(10,5,figsize=(20,20),
facecolor='w', edgecolor='k')
        fig.subplots_adjust(hspace=.5,
wspace=.001)

```

Tabel 4. 18 Sintaks Membuat Dataset

```

self.dataset_path(dataset_name)
count = 0
face_id = self.AddUser()
print('\n[INFO] Membuat dataset')
while(True):
    success, image = cam.read()
    # convert image to grayscale
    gray = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
    faces =
self._Face_Cascade.detectMultiScale(gray,
scaleFactor = 1.098, minNeighbors = 6, minSize
= (50, 50))
    if(len(faces)> 1):
        print('\n[WARNING] Terdeteksi
lebih dari 1 wajah')
        continue
    try:
        for _,face in enumerate(faces):
            x, y, w, h = face
            gray_chunk = gray[y-30: y +
h + 30, x-30: x + w + 30]
            image_chunk = image[y: y +
h, x: x + w]
            self.create_Rect(image,
face, [0,255,0])
            # cv2.imshow("Video",
image)
            #get center image
            # image_center =
tuple(np.array(gray_chunk.shape) / 2)
            # rot_mat =
cv2.getRotationMatrix2D(image_center,
angle_degree, 1.0)
            # rotated_image =
cv2.warpAffine(gray_chunk, rot_mat,
gray_chunk.shape, flags=cv2.INTER_LINEAR)
            print("\n[INFO] Adding
image number {} to the dataset".format(count))
            # Save image
            cv2.imwrite("dataset/User."
+ str(face_id) + '.' + str(count) + ".jpg " ,
                image)

axs[int(count/5)][count%5].imshow(image,cmap='g
ray', vmin=0, vmax=255)

axs[int(count/5)][count%5].set_title("Person."
+ str(face_id) + '.' + str(count) + ".jpg ",
                fontdict={'fontsize':
15,'fontweight': 'medium'})

```


Tabel 4. 18 Sintaks Membuat Dataset

```

    axs[int(count/5)][count%5].axis('off')
        count += 1
    except Exception as e:
        print(e)
        print('[WARNING] Ada error')
        continue
    if cv2.waitKey(1) & 0xff == 27:
        break
    elif count >= samples:
        break
    print('\n[INFO] Dataset berhasil
dibuat')
    # cam.release()
    # cv2.destroyAllWindows()
    # plt.show()
def Arg_Parse():
    Arg_Par = arg.ArgumentParser()
    Arg_Par.add_argument("-v", "--video",
                        help = "path of the
video or if not then webcam")
    Arg_Par.add_argument("-c", "--camera",
                        help = "Id of the
camera")
    arg_list = vars(Arg_Par.parse_args())
    return arg_list

if __name__ == "__main__":
    if len(sys.argv) == 1:
        print("Masukan argumen yang valid!")
        sys.exit()
    Arg_list = Arg_Parse()
    face_cascade =
'lib/haarcascade_frontalface_default.xml'
    if not (os.path.isfile(face_cascade)):
        raise RuntimeError("%s: not found" %
face_cascade)
    samples = 2
    dataset_name = 'dataset/'
    file_name = 'train.yml'
    radius = 1
    neighbour = 8
    grid_x = 8
    grid_y = 8
    treshold = 140
    var =
list([radius,neighbour,grid_x,grid_y,treshold])
    model = Train(face_cascade,var)
    if Arg_list["video"] != None :
        video =
cv2.VideoCapture(Arg_list["video"])

```

Tabel 4. 18 Sintaks Membuat Dataset

```

        #create a dataset for further model
training
        print('{0} {1}
{2}'.format(samples,video,dataset_name))

model.createDataset(samples,video,dataset_name)
        #Training the model
        model.train(dataset_name,file_name)
        if Arg_list["camera"] != None :
            camera =
cv2.VideoCapture(eval(Arg_list["camera"]))
            camera.set(3, 640)
            camera.set(4, 480)

model.createDataset(samples,camera,dataset_name
)
        #Training the model
        model.train(dataset_name,file_name)

```

2. Sintaks Membuat Model

Berikut adalah Sintaks Membuat Model yang dapat dilihat pada

Tabel 4.19:

Tabel 4. 19 Sintaks Membuat Model

```

import numpy as np
import cv2
import os

#Face detection is done
def faceDetection(test_img):

gray_img=cv2.cvtColor(test_img,cv2.COLOR_BGR2GRAY)

face_haar=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
        faces=face_haar.detectMultiScale(gray_img,
scaleFactor=1.2,
        minNeighbors=4,
        minSize=(30, 30))

```

Tabel 4. 19 Sintaks Membuat Model

```

        return faces,gray_img

# labeling dataset
def labels_for_training_data(directory):
    faces=[]
    faceID=[]
    file_count = 0

    for path,subdirnames,filenames in
os.walk(directory):
        for filename in filenames:
            if filename.startswith("."):
                print("skipping system file")
                continue
            file_count += 1
            print(f'{filename} with id:
{filename.split(".")[1]}')
            id = filename.split(".")[1]

            image =
cv2.imread(f'dataset/{filename}')
            if image is None:
                print(f'{filename} not exist!')
                continue

faces_rect,gray_img=faceDetection(image)
            if len(faces_rect)!=1:
                print(f'{filename} ->
no/multiple face detected')

                continue

            (x,y,w,h)=faces_rect[0]
            tiny_face=gray_img[y:y+w,x:x+h]

            faces.append(tiny_face)

            faceID.append(int(id))

    print(f'{file_count=}')
    return faces,faceID

#Here training Classifier is called
def train_classifier(faces,faceID):

    face_recognizer=cv2.face.LBPHFaceRecognizer_create()

    face_recognizer.train(faces,np.array(faceID))

```

Tabel 4. 19 Sintaks Membuat Model

```

face_recognizer.write(f'{os.path.dirname(os.pat
h.realpath(__file__))}/model.yml')
return face_recognizer

#Drawing a Rectangle on the Face Function
def draw_rect(test_img,face):
    (x,y,w,h)=face

cv2.rectangle(test_img,(x,y),(x+w,y+h),(0,255,0
),thickness=3)

#Putting text on images
def put_text(test_img,text,x,y):

cv2.putText(test_img,text,(x,y),cv2.FONT_HERSHE
Y_DUPLEX,3,(255,0,0),6)

```

3. Sintaks Pengenalan Wajah

Berikut adalah Sintaks Pengenalan Wajah yang dapat dilihat pada

Tabel 4.20:

Tabel 4. 20 Sintaks Pengenalan Wajah

```

import cv2
import numpy as np
import csv
import os
import logging

logging.basicConfig(filename='predict.log',
format='%(asctime)s %(levelname)-8s
%(message)s',level=logging.DEBUG, datefmt='%Y-
%m-%d %H:%M:%S')

# global recognizer =
cv2.face.LBPHFaceRecognizer_create()
# recognizer.read('train.yml')

```

Tabel 4. 20 Sintaks Pengenalan Wajah

```

# global faceCascade =
cv2.CascadeClassifier('haarcascade_frontalface_
default.xml')
# global font = cv2.FONT_HERSHEY_SIMPLEX

class Predict():
    def __init__(self, image):
        self.image = image
        # self.recognizer =
cv2.face.LBPHFaceRecognizer_create()
        # self.recognizer.read('train.yml')
        # self.faceCascade =
cv2.CascadeClassifier('haarcascade_frontalface_
default.xml')
        # self.font = cv2.FONT_HERSHEY_SIMPLEX

    def print(self):
        return os.path.join('dataset',
self.image)

    def predict(self):
        logging.info('-----
----')
        logging.info('Predict Start')
        # print(self.image)
        # print(os.path.join('dataset',
self.image))
        # print('{} '.format(self.image))

        #
logging.info('{} '.format(os.path.join('dataset'
, self.image)))
        # logging.info('{} '.format('dataset/{}
'.format(self.image)))

        # img = cv2.imread('image/59b53e2d-
0c03-44ec-a4a9-23d496caf6e0.jpg')
        img = cv2.imread(str(self.image))
        # img =
cv2.imread(os.path.join('dataset', self.image)+
" ")
        logging.info('Recognizer start')
        recognizer =
cv2.face.LBPHFaceRecognizer_create()
        recognizer.read('model.yml')
        faceCascade =
cv2.CascadeClassifier('haarcascade_frontalface_
default.xml')
        font = cv2.FONT_HERSHEY_SIMPLEX
        ID = None
        CONF = None

```

Tabel 4. 20 Sintaks Pengenalan Wajah

```

        # recognizer =
cv2.face.LBPHFaceRecognizer_create()
        # recognizer.read('train.yml')
        # faceCascade =
cv2.CascadeClassifier('haarcascade_frontalface_
default.xml')
        # font = cv2.FONT_HERSHEY_SIMPLEX

        # img =
cv2.imread('/home/ubuntu/skripsi/dataset/User.3
.0.jpg ')
        # img = cv2.imread(self.image)

        ### resize image
        imgs =
cv2.resize(img, (0,0),None,0.25,0.25)

        # convert to gray scale
        gray = cv2.cvtColor(imgs,
cv2.COLOR_BGR2GRAY)

        ### detect the face
        faces =
faceCascade.detectMultiScale(gray,scaleFactor=1
.05,minNeighbors=4,minSize=(30, 30))
        threshold =
cv2.face_LBPHFaceRecognizer.getThreshold
        print(threshold)

        for (x,y,w,h) in faces:
            Id,conf =
recognizer.predict(gray[y:y+h,x:x+w])
            # cv2.rectangle(imgS, (x,y), (x+w,
y+h), (255,0,0), 2)
            # cv2.rectangle(img, (x, y), (x +
w, y + h), (0, 255, 0), 2)
            # cv2.putText(imgS, str(Id), (x,y-
40),font, 2, (255,255,255), 3)
            print('ID {0}, confidence
{1}'.format(Id, conf))
            ID = Id
            CONF = conf
            logging.info('Predict result: {} with
{} distance'.format(ID, CONF))
            print('ID {0}, confidence
{1}'.format(ID, CONF))
            return ID,CONF

```

4. Sintaks Server Pengenalan Wajah

Berikut adalah Sintaks Server Pengenalan Wajah yang dapat dilihat pada Tabel 4.21:

Tabel 4. 21 Sintaks Server Pengenalan Wajah

```
from flask import Flask, request, make_response
from flask_cors import CORS
from file_downloader import downloader,
getData, updateData, updateDataNew, putData
import cv2
from train import Train
from predict import Predict
import uuid
import os
import datetime
import json
import logging
import math
import recognize as re

logging.basicConfig(filename='skripsi.log',
format='% (asctime)s % (levelname)-8s
%(message)s', level=logging.INFO, datefmt='%Y-
%m-%d %H:%M:%S')
app = Flask(__name__)
cors = CORS(app, resources={r"/*": {"origins":
"*"}})

@app.route('/')
def test():
    current_time = datetime.datetime.now()
    response = make_response('<h1>Success at {}
(server time)</h1>'.format(current_time))
    response.headers['Content-Security-Policy']
= 'upgrade-insecure-requests'
    return response

@app.route('/recognize/predict',
methods=['POST'])
def predict():
    logging.info('halo')
    request_data = request.get_json()

    if request_data:
        image = request_data['image']
```

Tabel 4. 21 Sintaks Server Pengenalan Wajah

```

        expect_username =
request_data['username']
        location = request_data['location']
        username = None
        conf = None

        debug_id = None
        debug_sim = None
        debug_dis = None
        debug_username = None

        result_id = None
        result_sim = None
        result_dis = None
        result_username = None

        debug_user_data = None
        result_user_data = None

        minSim = 40 # 75%

        image_path =
'image/{}/.jpg'.format(uuid.uuid4())

        # download video from s3
        downloader_client =
downloader(key=image,bucket='dannynurdin',
destination=image_path)
        downloader_client.download()

        model = Predict(image_path)

        res = model.print()
        id,c = model.predict()
        logging.info('DATA => id from model ==
', id)
        print('DATA => id from model == ', id)
        logging.info('success {} -
{}'.format(id, c))
        if c:
            disMax = 140.0
            simMax = 100
            similarity = simMax -
simMax/disMax*c
            conf = similarity

        with open('users.txt') as f:
            datafile = f.readlines()
            for line in datafile:

```


Tabel 4. 21 Sintaks Server Pengenalan Wajah

```

        Id = line.split(',')[0]
        print('DATA => Id from model ==
', Id)
        logging.info('DATA => Id from
model == ', Id)
        if Id == id:
            dump= line.split(',')[1]
            debug_username =
dump.split('\n')[0]
            debug_id = id
            debug_sim = conf
            debug_dis = c

            dynamodb_client =
getData(id = debug_username)
            debug_response =
dynamodb_client.get()
            debug_user_data =
debug_response['Item'] or None

            # add record to database
test-v2
            record = putData(id =
debug_username,face_id=debug_id,conf=debug_sim,
location='test location')
            record.put()

            if conf and conf >= minSim:
                result_username =
dump.split('\n')[0]
                result_sim = conf
                result_id = id
                result_dis = c

                dynamodb_client =
getData(id = result_username)
                result_response =
dynamodb_client.get()
                result_user_data =
result_response['Item'] or None

        response = make_response({
            "debug": {
                "id": debug_id,
                "dis": debug_dis,
                'sim': debug_sim,
                "username": debug_username,
                "user_data": debug_user_data,
                'expect_user': expect_username

```

Tabel 4. 21 Sintaks Server Pengenalan Wajah

```

        },
        'result': {
            "username": result_username,
            "dis": result_dis,
            "sim": result_sim,
            "id": result_id,
            "user_data": result_user_data
        },
    })
    response.headers['Content-Security-
Policy'] = 'upgrade-insecure-requests'
    return response

return 'kosong'

# if image is None:
#     value = {
#         statusCode : 400,
#         message : 'Image required!',
#     }
#     return json.dumps(value)
# model = Predict(image)
# id, conf = model.predict()

# response = {
#     statusCode : 200,
#     data : {
#         id : id,
#         conf : conf,
#         status : success,
#     }
# }
# return json.dumps(response)

@app.route('/recognize/train',
methods=['POST'])
def train():
    # grab data from post request
    request_data = request.get_json()

    key = None
    username = None

    # config
    face_cascade =
'lib/haarcascade_frontalface_default.xml'
    dataset_name = 'dataset/'
    samples = 15

```

Tabel 4. 21 Sintaks Server Pengenalan Wajah

```

file_name = 'model.yml'
video_path =
'video/{}.mp4'.format(uuid.uuid4())

# LBPH variables
radius = 1
neighbour = 8
grid_x = 8
grid_y = 8
treshold = 140
var =
list([radius,neighbour,grid_x,grid_y,treshold])

if request_data:

    if 'key' in request_data:
        key = request_data['key']

    if 'username' in request_data:
        username = request_data['username']

    if 'from' in request_data:
        status = request_data['from']

    if status == 'development':
        bucket_name = 'skripsi200053-dev'
    else:
        bucket_name = 'skripsi132739-prod'

    # download video from s3
    downloader_client =
downloader(key=key,bucket=bucket_name,
destination=video_path)
    downloader_client.download()

    # get data user
    dynamodb_client = updateData(id =
username, key = key)
    ress = dynamodb_client.update()
    print(ress)

    # start recognize using opencv
    model =
Train(face_cascade,var,username) # create
instance train
    video = cv2.VideoCapture(video_path) #
load video

```

Tabel 4. 21 Sintaks Server Pengenalan Wajah

```

model.createDataset(samples,video,dataset_name)
# create dataset
    # id =
model.train(dataset_name,file_name)

    faces,faceID =
re.labels_for_training_data('dataset')

face_recognizer=re.train_classifier(faces,faceID)

face_recognizer.save(f'{os.path.dirname(os.path
.realpath(__file__))}/model.yml')
    print(f'faces: {len(faces)} , id:
{len(faceID)}')

    response = {
        "success": True,
        "face_id": id,
        "username": username
    }

    response = make_response({
        "success": True,
        "face_id": id,
        "username": username
    })
    response.headers['Content-Security-
Policy'] = 'upgrade-insecure-requests'
    return response
else:
    return 'Key required!'

@app.route('/update', methods=['POST'])
def update():
    request_data = request.get_json()
    AttrName = {}
    AttrValue = {}
    Expression = []

    if request_data:
        if 'id' in request_data:
            id = request_data['id']

        if 'name' in request_data:
            # key = request_data['name']
            AttrName['#N'] = 'name'
            AttrValue[':N'] = { 'S' :
request_data['name']}

```

Tabel 4. 21 Sintaks Server Pengenalan Wajah

```

        Expression.append('#N = :N')

        if 'phone' in request_data:
            # status = request_data['phone']
            AttrName['#P'] = 'phone_number'
            AttrValue[':P'] = { 'S' :
request_data['phone']}
            Expression.append('#P = :P')

        if 'address' in request_data:
            # status = request_data['address']
            AttrName['#A'] = 'address'
            AttrValue[':A'] = { 'S' :
request_data['address']}
            Expression.append('#A = :A')

        expression = 'Set ' + ','.join([str(elem)
for elem in Expression])

        dynamodb_client = updateDataNew(id)
        ress =
dynamodb_client.update(AttrName,AttrValue,
expression)

        response = make_response({
            "res": request_data,
            'AttrName': json.dumps(AttrName),
            'AttrValue': json.dumps(AttrValue),
            'Expression': expression

        })
        response.headers['Content-Security-Policy']
= 'upgrade-insecure-requests'
        return response
if __name__ == '__main__':
    # run app in debug mode on port 5000
    app.run(host='0.0.0.0',port=80)

```

4.5.4 Pedoman Pengoperasian Perangkat Lunak

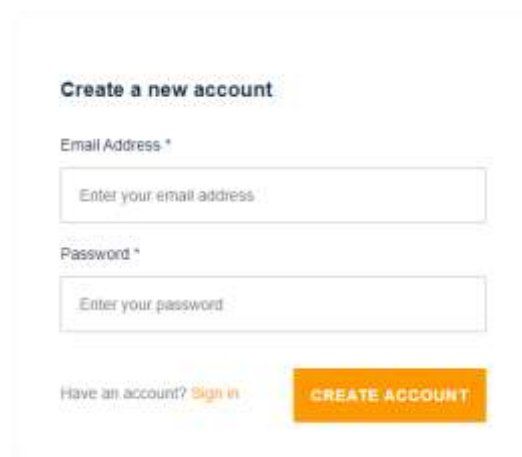
Pedoman pengoperasian berfungsi tatacara untuk membantu pengguna dalam menggunakan sebuah sistem, pedoman lebih membantu pengguna dibandingkan harus mencoba semua menu yang ada. Adapun pedoman dalam menggunakan *website* Absensi Online

Radar Cirebon setelah diimplementasikan dengan teknik pengenalan wajah adalah sebagai berikut:

1. Proses Pendaftaran

- a. Masukkan *email* yang akan didaftarkan.
- b. Masukkan *password* yang akan anda gunakan, pastikan *password* harus merupakan *alphanumeric*.
- c. Setelah pendaftaran pengguna harus mengkonfirmasi akun menggunakan kode yang diberikan ke *email* yang terdaftar.

Berikut adalah tampilan halaman pendaftaran yang dapat dilihat pada Gambar 4.17:



Gambar 4. 17 Halaman Pendaftaran

2. Proses *Login*

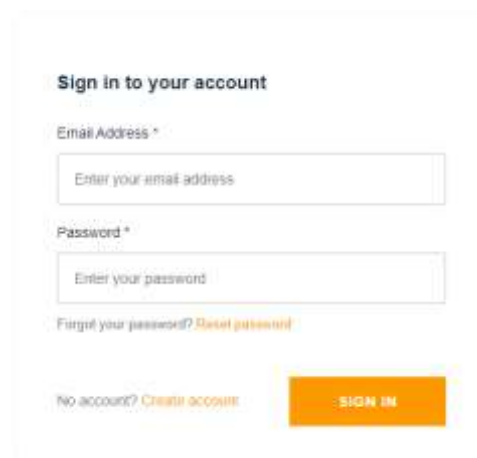
- a. Masukkan *email* dan *password* yang telah terdaftar.

b. Klik tombol masuk

c. Jika *email* dan *password* benar, maka akan dialihkan ke halaman *dashboard*.

Berikut adalah tampilan halaman *login* yang dapat dilihat pada

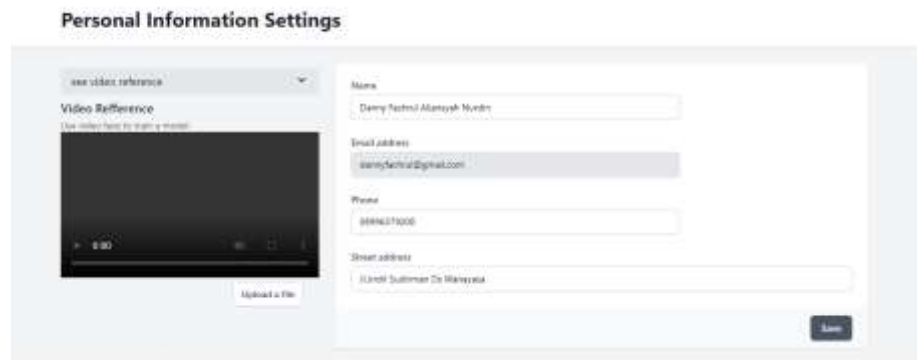
Gambar 4.18:



Gambar 4. 18 Halaman Login

3. Proses Menambahkan Informasi Pengguna

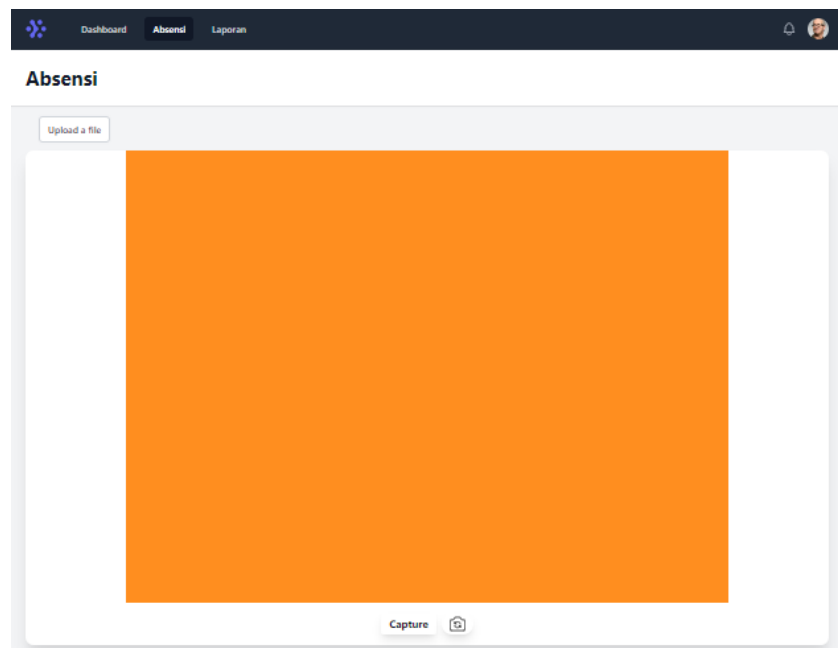
Untuk menambahkan informasi pengguna maka pilih menu *setting*. Pada halaman *setting* berisi informasi dari pengguna seperti nama, alamat *email*, alamat dan nomor telepon. Di halaman ini juga pengguna bisa memasukkan data wajahnya yang berupa file video. Tampilan halaman *setting* dapat dilihat pada Gambar 4.19 berikut:



Gambar 4. 19 Tampilan Setting

4. Proses Absensi

Untuk melakukan absensi maka pilih menu absensi. Pada halaman absensi terdapat tombol untuk mengunggah foto dan kamera yang bisa digunakan untuk menangkap gambar wajah pengguna. Tampilan halaman absensi dapat dilihat pada Gambar 4.20 berikut:



Gambar 4. 20 Halaman Absensi