DAFTAR PUSTAKA

[1] "Indonesia: Coronavirus Pandemic Country Profile - Our World in Data." https://ourworldindata.org/coronavirus/country/indonesia (accessed Apr. 30, 2021).

[2] WHO, "Transmisi SARS-CoV-2: implikasi terhadap kewaspadaan pencegahan infeksi," pp. 1–10, 2020.

[3] V. Bruce and A. Young, "Understanding face recognition," *Br. J. Psychol.*, vol. 77, no. 3, pp. 305–327, 1986, doi: 10.1111/j.2044-8295.1986.tb02199.x.

[4] Q. Mutiara and E. Prasetyo, "Perbandingan Metode Eigenface, Fisherface, dan LBPH pada Sistem Pengenalan Wajah," *J. Ilm. Komputasi*, vol. 18, no. 4, 2019, doi: 10.32409/jikstik.18.4.2675.

[5] Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif, dan Tindakan*. 2012.

[6] H. Hasanah, "TEKNIK-TEKNIK OBSERVASI (Sebuah Alternatif Metode Pengumpulan Data Kualitatif Ilmu-ilmu Sosial)," *At-Taqaddum*, vol. 8, no. 1, p. 21, 2017, doi: 10.21580/at.v8i1.1163.

[7] A. Mirzaqon, "Studi Kepustakaan Mengenai Landasan Teori Dan Praktik Konseling Expressive Writing Library," *J. BK UNESA*, no. 1, pp. 1–8, 2018.

[8] S. Z. Li and A. K. Jain, *Handbook of Face Recognition*. 2005.

[9] H. Jaya *et al.*, *Kecerdasan Buatan*, vol. 53, no. 9. 2018.

[10] A. Ahmed, F. Ali, and A. Ahmed, “LBPH based improved face recognition at low resolution LBPH Based Improved Face Recognition At Low Resolution,” *2018 Int. Conf. Artif. Intell. Big Data*, no. October 2019, pp. 144–147, 2018, doi: 10.1109/ICAIBD.2018.8396183.

[11] A. P. Singh, S. S. Manvi, P. Nimbal, and G. K. Shyam, “Face recognition system based on LBPH algorithm,” *Int. J. Eng. Adv. Technol.*, vol. 8, no. 5 Special Issue, pp. 26–30, 2019.

[12] D. A. Wahyudi and I. H. Kartowisastro, “Menghitung Kecepatan Menggunakan Computer Vision,” *J. Tenik Komput.*, vol. 19, no. 2, pp. 89–101, 2011.

[13] E. Gorelik, “Cloud Computing Models,” 2013.

[14] R. S. Pressman, *Software Quality Engineering: A Practitioner’s Approach*, vol. 9781118592. 2014.

[15] D. S. M. Sc, M. Math, and D. Ph, “LOGIKA PEMROGRAMAN Disusun,” no. September, 2002.

[16] H. Bagir and B. E. Putro, “Analisis Perancangan Sistem Informasi Pergudangan di CV. Karya Nugraha,” *J. Media Tek. dan Sist. Ind.*, vol. 2, no. 1, p. 30, 2018, doi: 10.35194/jmtsi.v2i1.274.

[17] “Data Flow Diagram (DFD)s: An Agile Introduction.” http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm (accessed Jun. 14, 2021).

[18] D. Vassallo and J. Pschorr, “The Good Parts of AWS.”

LAMPIRAN-LAMPIRAN

Lampiran 1 Sintaks Membuat Dataset

```
import cv2
import os
import numpy as np
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] Masukan 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):
        imagePaths = [os.path.join(path, f) for f in
os.listdir(path)]
        faceSamples = []
        ids = []
        for imagePath in imagePaths:
            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)
        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='gray', vmin=0,
vmax=255)

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

Lampiran 2 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_defa
ult.xml')
    faces=face_haar.detectMultiScale(gray_img,
scaleFactor=1.2,
        minNeighbors=4,
        minSize=(30, 30))
    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))

face_recognizer.write(f'{os.path.dirname(os.path.realpath(__f
ile__))}/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_HERSHEY_DUPLEX,3,(25
5,0,0),6)
```

Lampiran 3 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')
# 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
        ### 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,minNeighbo
rs=4,minSize=(30, 30))
        treshold = cv2.face_LBPHFaceRecognizer.getThreshold
        print(treshold)
        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,
260, 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
```

Lampiran 4 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']
        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:
                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
            },
            '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'
@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
    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
        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(__fi
le__))}/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']}
            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='0.0.0.0.0',port=80)
```

Lampiran 5 Hasil Pengujian Gambar Acak

Pengujian ke	Id Pengguna	Hasil prediksi	Jarak	Prediksi
1	Tidak diketahui	1	89	Salah
2	Tidak diketahui	1	113	Salah
3	Tidak diketahui	-	-	benar
4	2	2	121	Benar
5	1	1	97	Benar
6	Tidak diketahui	1	78	Salah
7	1	1	76	Benar
8	2	2	117	Benar
9	1	-	-	Salah
10	Tidak diketahui	1	83	Salah
11	1	-	-	Salah
12	2	2	85	Benar
13	1	1	95	Benar
14	1	1	110	Benar
15	1	1	118	Benar
16	1	1	111	Benar
17	2	2	76	Benar
18	Tidak diketahui	1	89	salah
19	1	1	167	Benar
20	2	2	125	Benar
21	2	2	117	Benar
22	1	1	109	Benar
23	1	0	163	Benar
24	1	-	-	Salah
25	Tidak diketahui	1	84	Salah
26	Tidak diketahui	1	117	Salah
27	Tidak diketahui	2	108	Salah
28	2	2	159	Benar
29	2	2	85	Benar
30	1	1	122	Benar
31	1	-	-	Salah
32	1	1	116	Benar

33	1	-	-	Salah
34	1	1	77	Benar
35	2	2	113	Benar
36	2	2	96	Benar
37	1	1	90	Benar
38	1	1	79	Benar
39	2	2	122	Benar
40	Tidak diketahui	1	143	Salah
41	1	-	-	Salah
42	1	1	82	Benar
43	1	1	79	Benar
44	Tidak diketahui	1	66	Salah
45	1	-	-	Salah
46	1	1	111	Benar
47	1	-	-	Salah
48	Tidak diketahui	1	65	Salah
49	Tidak diketahui	1	89	Salah
50	Tidak diketahui	1	132	Salah

Lampiran 6 Pengujian Dengan Id Diketahui

Pengujian ke	Id Pengguna	Hasil prediksi	Jarak	Prediksi
1	2	2	121	Benar
2	1	1	97	Benar
3	1	1	76	Benar
4	2	2	117	Benar
5	1	-	-	Salah
6	1	-	-	Salah
7	2	2	85	Benar
8	1	1	95	Benar
9	1	1	110	Benar
10	1	1	118	Benar
11	1	1	111	Benar
12	2	2	76	Benar
13	1	1	167	Benar
14	2	2	125	Benar
15	2	2	117	Benar
16	1	1	109	Benar
17	1	0	163	Benar
18	1	-	-	Salah
19	2	2	159	Benar
20	2	2	85	Benar
21	1	1	122	Benar
22	1	-	-	Salah
23	1	1	116	Benar
24	1	-	-	Salah
25	1	1	77	Benar
26	2	2	113	Benar
27	2	2	96	Benar
28	1	1	90	Benar
29	1	1	79	Benar
30	2	2	122	Benar
31	1	-	-	Salah
32	1	1	82	Benar

33	1	1	79	Benar
34	1	-	-	Salah
35	1	1	111	Benar
36	1	-	-	Salah

Lampiran 7 Hasil pengujian Muka Senyum

No	Id Pengguna	Hasil Prediksi	Jarak	Prediksi
1	1	1	110	Benar
2	1	1	111	Benar
3	1	1	109	Benar
4	1	1	122	Benar
5	1	1	116	Benar
6	1	-	-	Salah
7	1	1	90	Benar
8	1	1	79	Benar
9	1	-	-	Salah
10	1	-	-	Salah
11	1	1	112	Benar
12	1	1	103	Benar
13	1	1	100	Benar
14	1	1	118	Benar
15	1	1	114	Benar
16	1	-	-	Salah
17	1	1	81	Benar
18	1	1	68	Benar
19	1	-	-	Salah
20	1	-	-	Salah

Lampiran 8 Hasil pengujian Muka Datar

No	Id Pengguna	Hasil Prediksi	Jarak	Prediksi
1	1	1	76	Benar
2	1	-	-	Salah
3	1	1	95	Benar
4	1	1	163	Benar
5	2	2	85	Benar
6	1	1	77	Benar
7	2	2	113	Benar
8	2	2	96	Benar
9	1	1	79	Benar
10	1	-	-	Salah
11	1	1	88	Benar
12	1	1	152	Benar
13	2	2	81	Benar
14	1	1	66	Benar
15	2	2	102	Benar
16	2	2	82	Benar
17	1	1	69	Benar
18	2	2	111	Benar
19	2	2	92	Benar
20	1	-	-	Salah