BAB III

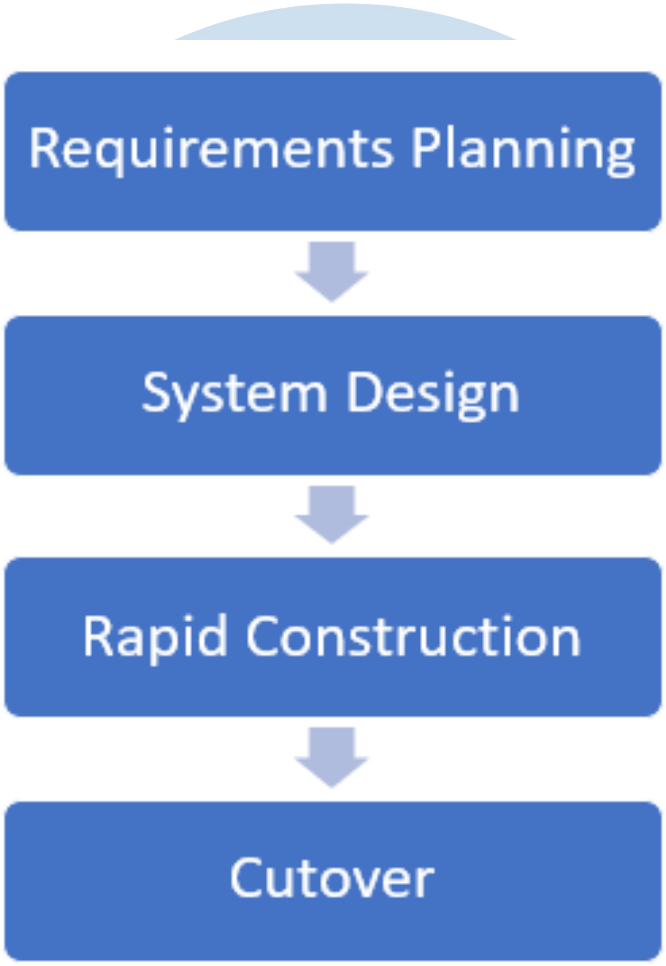
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan penelitian independen ini dibimbing oleh Ibu Friska Natalia selaku dosen dari program studi Sistem Informasi di Universitas Multimedia Nusantara. *Job Description* pada penelitian independent ini adalah membuat suatu sistem program yang bisa diimplementasikan di lingkungan Universitas Multimedia Nusantara untuk mendeteksi penggunaan masker dimana sistem program yang dibuat bisa mendeteksi siapa yang menggunakan masker dan yang tidak menggunakan masker.

3.2 Tugas yang dilakukan

Tugas yang dilakukan pada penelitian independen ini adalah berdasarkan metode *Rapid Application Development* (RAD), yaitu metode pengembangan sistem informasi yang dibuat oleh *James Martin* yang, jika dibandingkan dengan tahapan lain seperti *waterfall*, lebih memfokuskan pada tahap pengembangan program atau aplikasi. [4]



Gambar 3. 1 **Urutan** ***Rapid Application Development***

Dari gambar 3.1, Ada 4 tahapan utama dalam *Rapid Application Development*, yaitu :

1. *Requirement Planning*

   Tahapan untuk menjelaskan apa saja yang dibutuhkan dalam menjalankan penelitian ini. Tahapan ini diperlukan untuk mencegah hal-hal yang bisa menghambat berjalannya program. Tahapan ini

akan dipecah menjadi 3 bagian, pertama adalah *Hardware Requirement,* yaitu perangkat keras yang dibutuhkan untuk membuat sistem program, seperti OS dan RAM. Kedua adalah *Software Requirement,* yaitu perangkat lunak untuk membuat sistem program. Bagian terakhir adalah *Data Requirement* yang membahas pengolahan data yang akan digunakan sehingga bisa digunakan sebagai dataset untuk sistem program yang dibuat.

2. *System Design*

   Tahapan untuk membuat bagan sistem yang dibuat, seperti Use Case Diagram, Class Diagram, dan Activity Diagram. Tahap ini bertujuan untuk memberikan gambaran dalam pembuatan sistem program.

3. *Rapid Construction*

   Tahapan dimana hasil sistem program yang sudah dibuat diuji melalui skenario-skenario untuk membantu menentukan apakah sistem program sudah berhasil dalam menangani skenario yang ada atau tidak. Jika sistem program yang dijalankan tidak memenuhi ekspektasi atau kurang bagus, maka tahapan ini akan terus diulang sampai mendapat hasil yang optimal.

4. *Cutover*

   Tahapan implementasi setelah pengujian skenario pada tahapan sebelumnya, yaitu *Rapid Construction*.

3.3 Uraian Pelaksanaan Kerja Magang

Tabel 3.1 menjelaskan mengenai uraian pelaksanaan kerja magang, dengan jangka waktu 12 minggu. Pelaksanaan kerja magang ini meliputi dari tahapan-tahapan pada *Rapid Application Development*

Tabel 3. 1 Uraian Pelaksanaan Kerja Magang

Minggu ke-	Kegiatan	Mulai	Selesai
1 sampai 3	*Requirement Planning*	11 September 2021	22 September 2021
3 sampai 4	*System Design*	22 September 2021	27 September 2021
4 sampai 12	*Rapid Construction*	29 September 2021	28 November 2021
12, masih berlanjut	*Cutover*	30 November 2021	Masih berlanjut.

3.3.1 *Requirement Planning*

Requirement Planning adalah tahapan untuk menentukan apa saja yang dibutuhkan dalam menjalankan penelitian ini [5] , mulai dari *Hardware Requirement*, yaitu spesifikasi komputer atau laptop, *Software Requirement* dalam pembuatan sistem program, dan *Data Requirement* untuk mempersiapkan dataset yang bisa digunakan untuk membantu pembuatan. Untuk *Hardware Requirement*, spesifikasi komputer atau laptop yang digunakan adalah sebagai berikut :

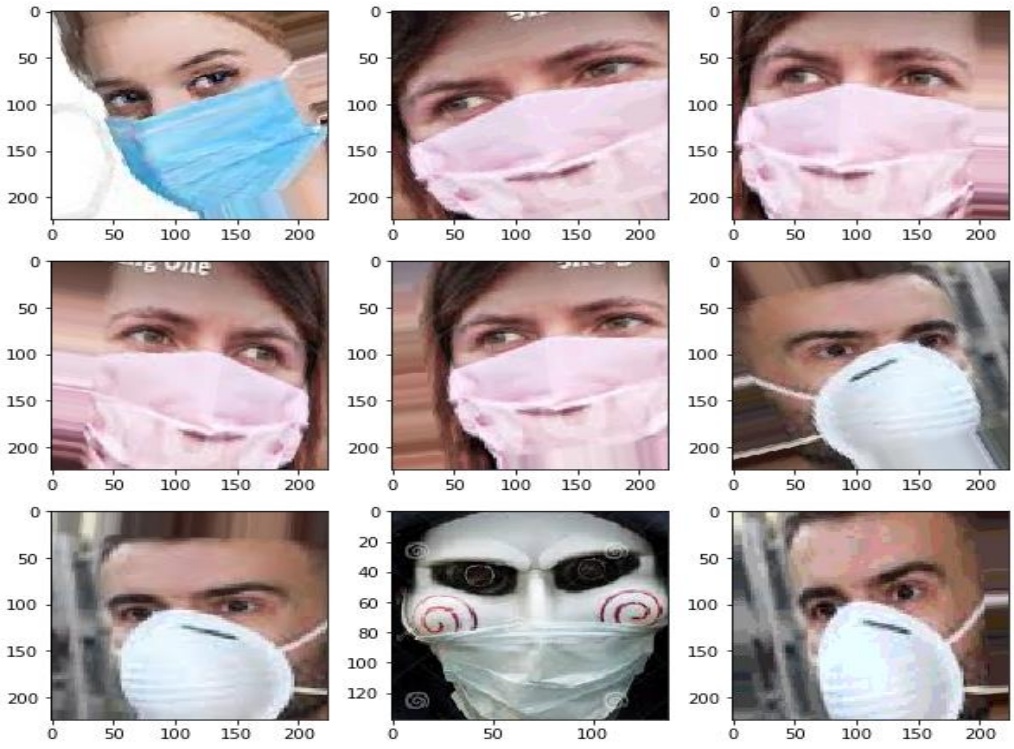
- RAM dengan ukuran minimal 8GB
- Kapasitas penyimpanan minimal 150GB
- Prosesor *intel core i-7* minimal generasi 7 atau setara

- Bisa terhubung dengan jaringan *internet*, jika menggunakan aplikasi pengendalian jarak jauh seperti *AnyDesk.*
- *Webcam* untuk melakukan pengambilan gambar atau video

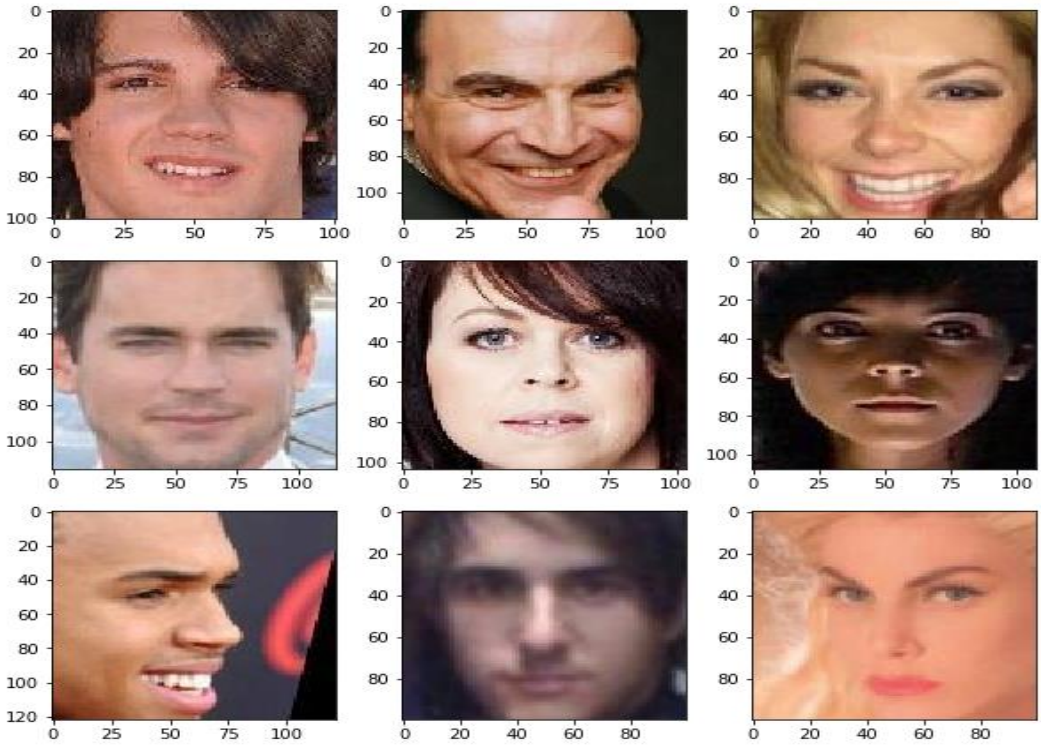
Sedangkan untuk *Software Requirement*, yaitu aplikasi yang digunakan dalam pembuatan sistem program adalah sebagai berikut :

- Sistem operasi *Windows 10*
- *Jupyter Notebook* untuk menjalankan *Python*
- *Browser* apa saja untuk membuka *Jupyter Notebook*

Untuk *Data Requirement,* dataset akan digunakan untuk menghasilkan sebuah program yang dijalankan melalui *jupyter notebook* untuk mendeteksi apakah orang yang tertangkap di dalam kamera menggunakan masker atau tidak.



***Gambar 3. 2* Contoh data gambar yang menggunakan masker**



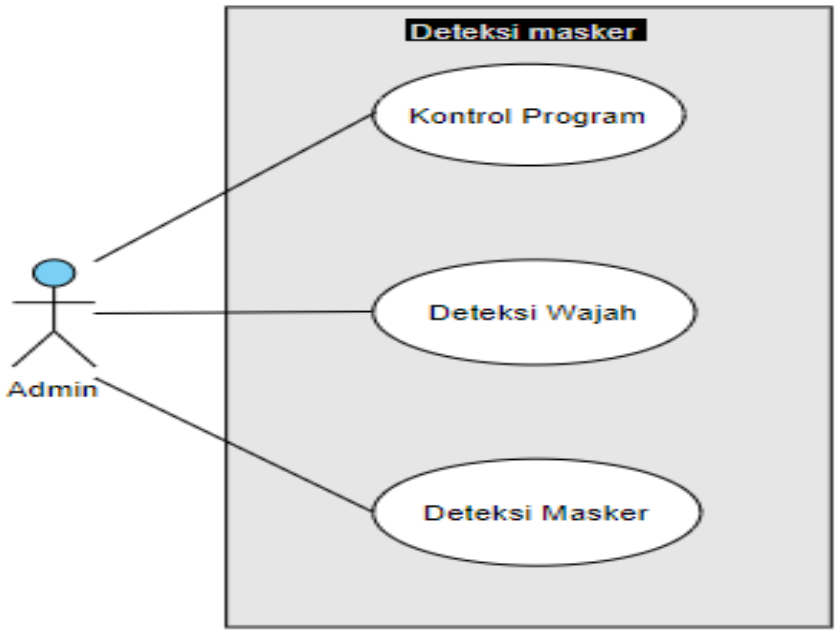
***Gambar 3. 3* Contoh data gambar yang tidak memakai masker**

Data yang diperlukan adalah data orang yang menggunakan masker dan yang tidak menggunakan masker. Data yang digunakan pada penelitian ini didapat dari *pyimagesearch.com*, lebih tepatnya dibuat oleh Prajna Bhandary, dengan mengumpulkan kumpulan gambar wajah, lalu memasang masker pada wajah beberapa data melalui pemasangan *landmark* pada wajah tersebut, sehingga terciptanya dataset buatan untuk wajah yang menggunakan masker. Gambar 3.2 dan 3.3 adalah beberapa contoh data gambar yang ada pada dataset.

Dengan membagi data menggunakan rasio 75% untuk *training* dan 25% untuk *testing*, dataset lalu dilatih menggunakan *MobileNet* dengan total 10 EPOCH untuk pelatihan. *Tensorflow* kemudian digunakan dalam pengujian deteksi gambar.

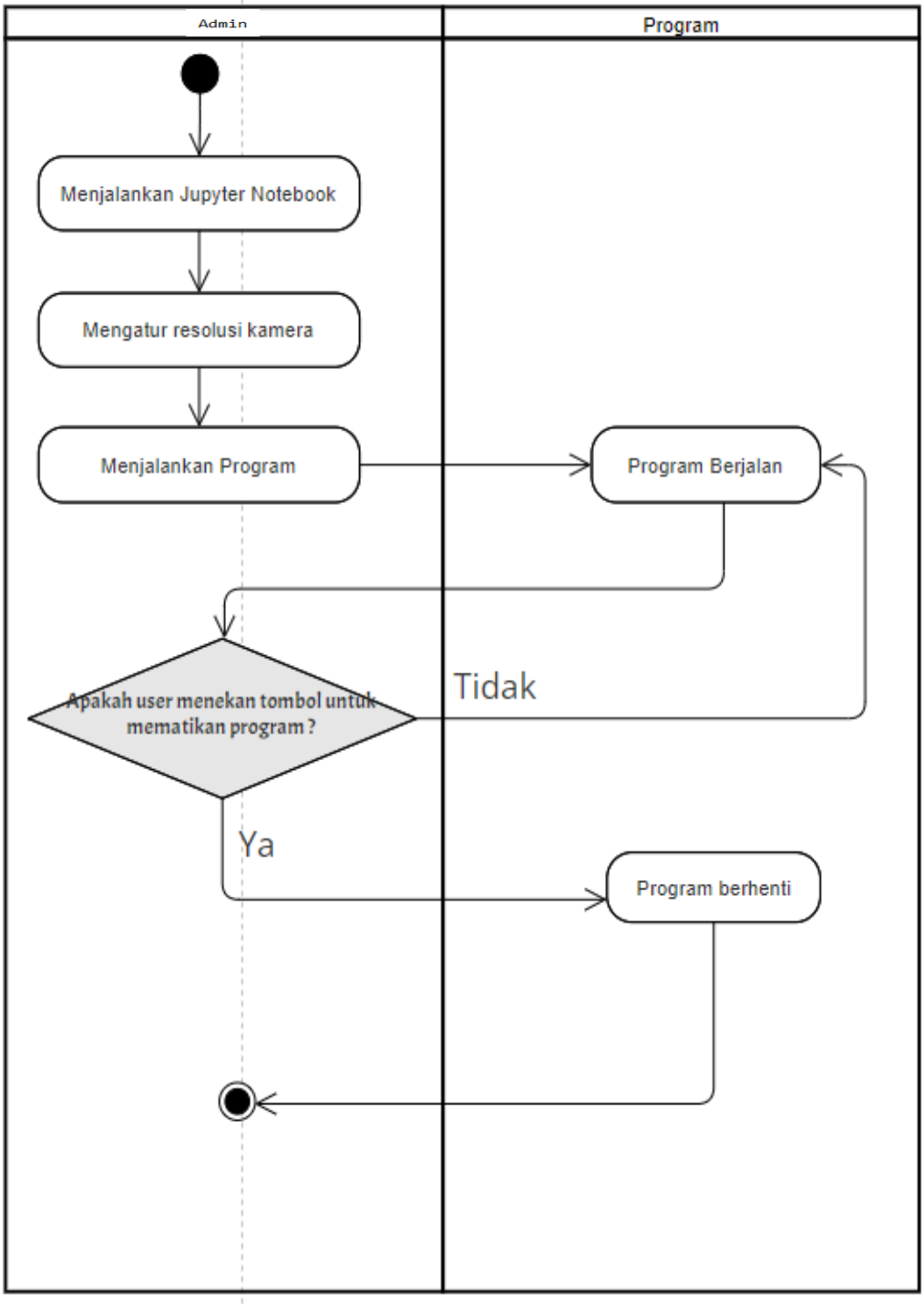
3.3.2 *System Design*

Dalam tahapan ini, sistem program dijelaskan melalui *Use Case Diagram, Activity Diagram,* dan *Class Diagram.*



Gambar 3. 4 *Use Case Diagram*

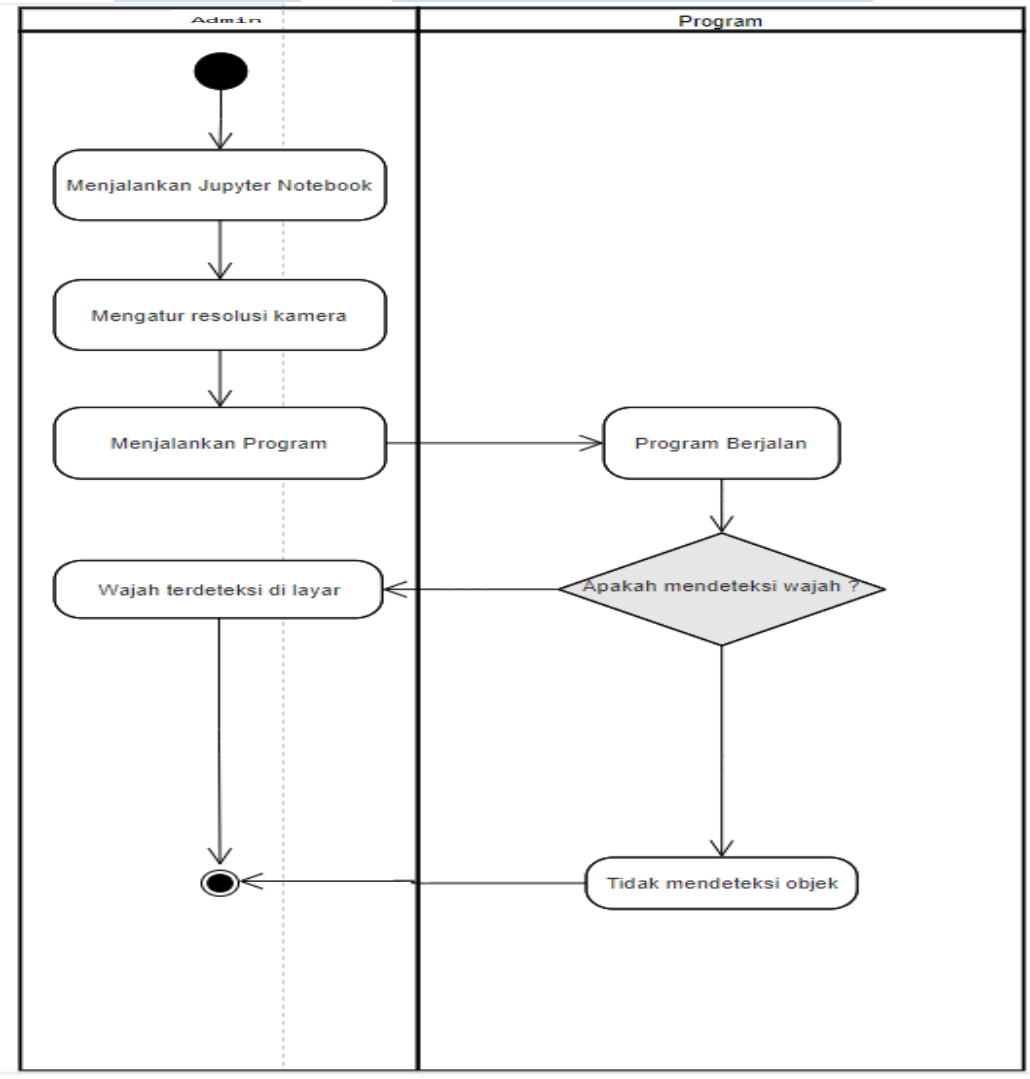
Gambar 3.4 menggambarkan *Use Case Diagram* dengan penjelasan penggunaan program ini yang meliputi 3 *use case*, yaitu kontrol program, deteksi wajah, dan deteksi masker. Pada kontrol program, pengguna program bisa menjalankan dan mematikan program. Deteksi wajah dan deteksi masker, yaitu mendeteksi wajah yang tertangkap di kamera dan juga menentukan apakah wajah tersebut menggunakan masker atau tidak.



Gambar 3. 5 *Activity Diagram* untuk kontrol program

Gambar 3.5 menjelaskan *Activity Diagram* untuk kontrol program. Setelah menjalankan *Jupyter Notebook*, *user* bisa mengatur kamera beserta resolusi jika

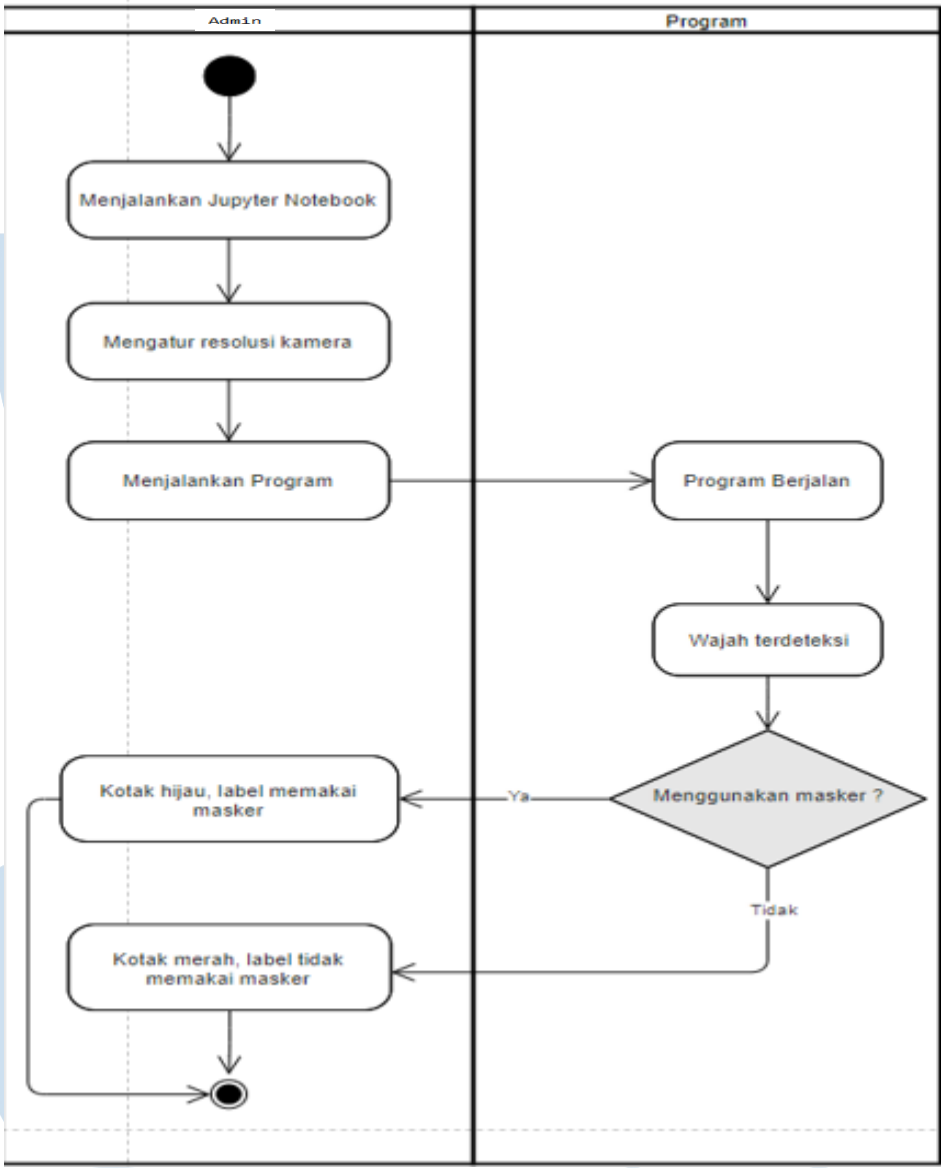
diperlukan, lalu program bisa dijalankan. Disaat program masih berjalan, jika *user* menekan tombol untuk mematikan program, dimana tombol *default* nya adalah *escape* di *keyboard*, maka program akan berhenti.



Gambar 3. 6 *Activity Diagram* untuk deteksi wajah

Untuk gambar 3.6 yang menjelaskan *activity diagram* untuk deteksi wajah, ketika kamera mendeteksi adanya wajah, maka akan memunculkan *feedback*

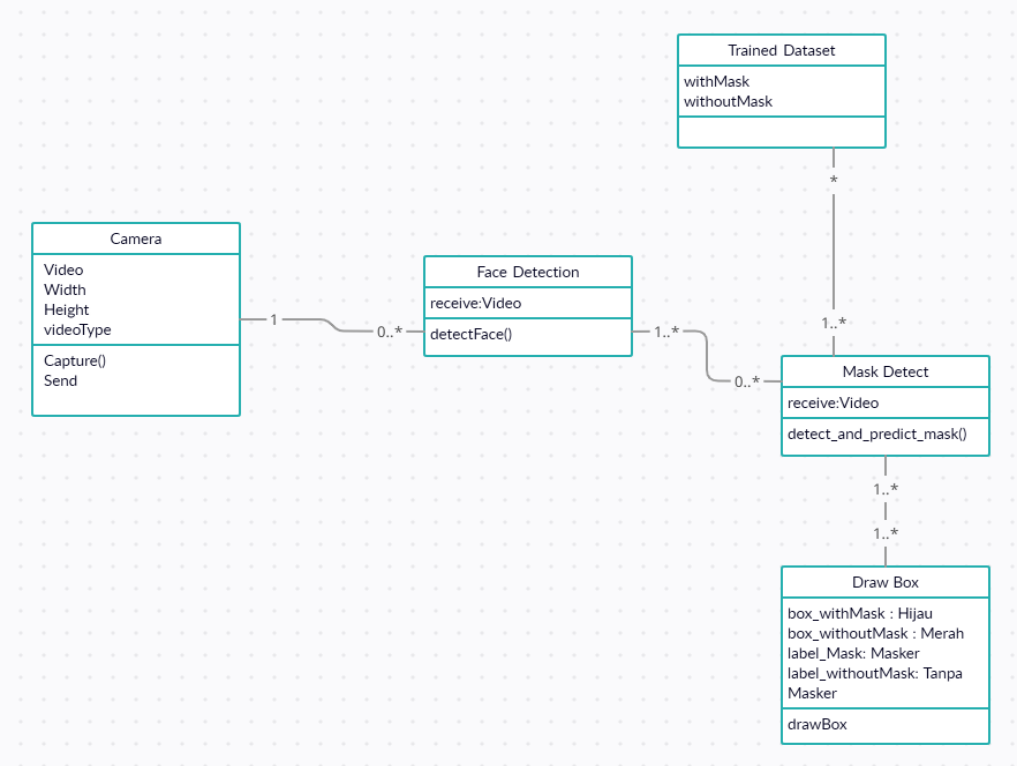
berupa kotak berwarna hijau ataupun merah di layar. Namun jika tidak mendeteksi adanya wajah, maka tidak akan memunculkan *feedback*.



Gambar 3. 7 *Activity Diagram* untuk deteksi masker

Gambar 3.7 menggambarkan *Activity Diagram* untuk deteksi masker. Ketika program mendeteksi wajah, jika wajah tersebut menggunakan masker, maka akan memunculkan kotak hijau di layar beserta label bahwa wajah tersebut

memakai masker. Namun, jika wajah yang terdeteksi tidak menggunakan masker, maka akan memunculkan kotak merah di layar beserta label bahwa wajah tersebut tidak memakai masker.



Gambar 3. 8 *Class Diagram*

Gambar 3.8 menjelaskan *Class Diagram* yang mempunyai 5 *class*, yaitu *Camera, Face Detection*, *Trained Dataset*, *Mask Detect,* dan *Draw Box* dengan variabel dan *operation* masing-masing.

Pada *class camera*, kamera yang digunakan untuk program akan memproses hasil rekaman yang didapat secara *real-time* dengan hasil resolusi yang ditentukan melalui variabel *width* dan *height*.

Class face detection mendeteksi wajah yang tertangkap dari hasil video yang didapat dengan bantuan dari *library OpenCV*.

Deteksi masker dilakukan di *class mask detect*, dimana setelah menangkap wajah yang terdeteksi di kamera, akan menentukan apakah wajah tersebut menggunakan masker atau tidak berdasarkan hasil dataset yang sebelumnya sudah di latih, yaitu dari *class trained dataset*, dengan variabel masker dan tanpa masker.

Setelah mendeteksi apakah menggunakan masker atau tidak, maka *class draw box* akan membuat sebuah kotak di area yang terdeteksi. Untuk yang terdeteksi menggunakan masker, kotak akan berwarna hijau dengan label tulisan yang menandakan bahwa wajah tersebut menggunakan masker. Namun, jika yang terdeteksi tidak menggunakan masker, kotak akan berwarna merah dengan label tulisan tidak menggunakan masker.

3.3.3 *Rapid Construction*

Rapid Construction adalah tahapan untuk membangun aplikasi dan melakukan perbaikan atau optimisasi sehingga menghasilkan aplikasi yang optimal . Tahapan ini akan membahas mengenai dataset yang akan dilatih sehingga bisa digunakan untuk deteksi masker, hingga pembuatan

```
In [1]:
import matplotlib.pyplot as plt
import numpy as np
import os
```

```
In [2]:
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
```

***Gambar 3. 9 Library* yang digunakan untuk proses dataset dan model**

Gambar 3.9 menunjukkan *library* yang digunakan untuk pengolahan data dan model dengan penjelasan sebagai berikut :

- *Matplotlib*, yaitu *library Python* yang digunakan untuk *plotting* dan pembacaan gambar [6] dengan *sub library pyplot* digunakan untuk membuat plot agar bisa menunjukkan gambar yang akan digunakan.
- *Numpy*, *library* yang berfungsi untuk memberikan fungsi dasar dalam membaca *array* [7] digunakan untuk menjadikan dataset yang ada menjadi *array* versi *numpy* agar memudahkan pengolahan data
- *Os* digunakan agar *Python* bisa berinteraksi dengan sistem operasi yang dijalankan, sehingga bisa membaca dan melakukan interaksi dengan apa yang ada di dalam komputer atau laptop yang digunakan [8], contohnya adalah agar bisa menggabungkan beberapa folder yang terpisah dengan *command os.path.join*.

- *Tensorflow*, yaitu sebuah *library* untuk *machine intelligence* digunakan untuk pengolahan dataset dengan bantuan *keras*, yaitu API bawaan *Tensorflow* yang sebelumnya merupakan *library* terpisah, namun digabungkan pada *Tensorflow 2.0* [9].

Setelah membagi data dengan ratio 75% dan 25% untuk *training* dan *testing*, dataset kemudian dilatih menggunakan *MobileNet* dengan total 10 EPOCH. Tensorflow lalu digunakan untuk menguji apakah model yang dibuat sudah cukup untuk mendeteksi masker. Setelah selesai memproses EPOCH, model beserta *weights* lalu disimpan agar bisa digunakan kembali saat uji coba menggunakan kamera. Gambar 3.10 merupakan salah satu kode untuk menyimpan model dan *weights* sehingga bisa digunakan nantinya.

```
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
200/200 [==============================] - 33s 166ms/step - loss: 0.0096 - accuracy: 0.9972

In [39]:  1 model.save('deteksi_masker.h5')

In [40]:  1 model.save_weights("weights/deteksi_masker_weights.temp.hd5",overwrite=True)
```

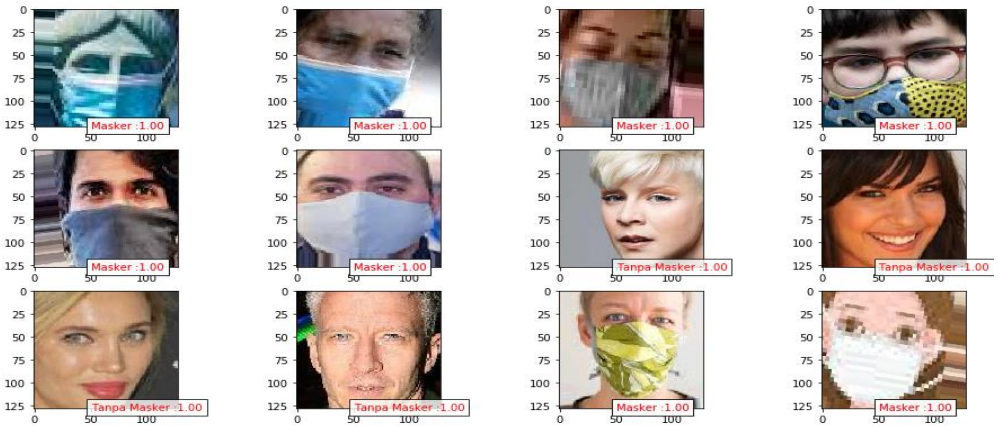
Gambar 3. 10 **Model dan *Weights* disimpan untuk digunakan saat uji coba kamera**

```
# prediksi
result = model.predict(image)[0]
if (result[0]>0.5):
  print("Gambar tes adalah wajah dengan masker")
else:
  print("Gambar tes adalah wajah tanpa masker")
```

```
Gambar tes adalah wajah dengan masker
```

Gambar 3. 11 Hasil ujicoba satu gambar

Untuk percobaan satu jenis gambar, seperti pada gambar 3.11, model membuat sebuah kondisi *if*, dimana jika hasil prediksi melebihi 50%, maka akan menentukan bahwa gambar tersebut adalah gambar wajah yang menggunakan masker. Namun jika kurang dari 50%, maka model akan menentukan bahwa gambar tersebut tidak memakai masker. Dari hasil tersebut, maka bisa ditentukan bahwa model sudah bisa memprediksi apakah gambar yang diberikan merupakan orang yang menggunakan masker atau tidak.



Gambar 3. 12 Hasil ujicoba lebh dari satu gambar

Untuk percobaan beberapa gambar sekaligus pada gambar 3.12, model juga masih bisa memprediksi gambar-gambar tersebut dengan hasil optimal.

Skenario untuk deteksi masker di sini ada 3, yaitu :

- Jika seseorang terdeteksi menggunakan masker, maka akan memunculkan kotak berwarna hijau, disertai tulisan ‘Masker’ dan akurasi dari prediksi
- Jika seseorang terdeteksi tidak menggunakan masker, maka akan memunculkan kotak berwarna merah, disertai tulisan ‘Tanpa Masker’ dan akurasi dari prediksi
- Jika tidak mendeteksi adanya wajah dalam tangkapan kamera, maka tidak akan memunculkan hasil prediksi apapun, hanya tangkapan kamera.

Setelah dataset berhasil diolah dan model disimpan, maka langkah selanjutnya adalah mengaplikasikan model yang telah disimpan kepada gambar selain yang termasuk ke dalam dataset.

Gambar 3. 13 Gambar yang akan diujicoba

Pada gambar 3.13 sebagai gambar yang akan diujicoba, dengan memanfaatkan *library cv* untuk kegunaannya yang bisa membaca gambar ataupun vidio secara *real-time* digabungkan dengan model yang sudah disimpan sebelumnya, *library cv* digunakan untuk membaca gambar yang akan digunakan, yaitu gambar 3.13, dengan mengikuti resolusi awal dari gambar tersebut.

```
In [9]:
#membaca gambar
image = cv2.imread(input_image_path)
orig = image.copy()
#resolusi diatur sesuai ukuran asli gambar
(h, w) = image.shape[:2]
print(h,w)

402 715
```

***Gambar 3. 14 Code* untuk membaca gambar dengan resolusi 402x715**

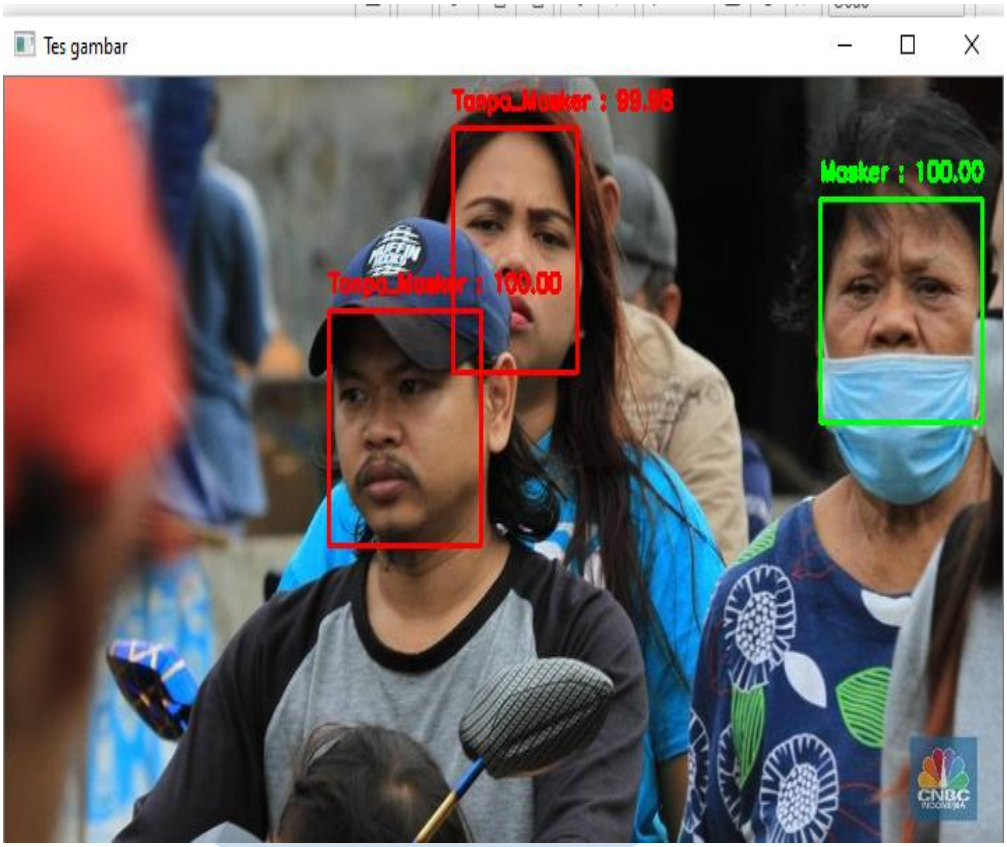
Setelah gambar sudah dibaca dan ditentukan resolusinya, seperti pada gamabr 3.14, maka langkah selanjutnya adalah membaca model yang sebelumnya sudah disimpan sebelumnya, dan memanfaatkan *blob*, yaitu sebuah

objek binary yang menyerupai satu sama lain [10], untuk bisa mendeteksi wajah yang ada pada gambar tersebut, maka prediksi bisa mulai dilakukan.

```
#melakukan pengulangan untuk menentukan jarak dan tingkat confidence prediksi
for i in range(0, detections.shape[2]):
    confidence = detections[0,0,i,2]
    #jika tingkaat confidence melebihi batas confidence 20%,
    #maka akan lanjut dalam melakukan prediksi untuk menentukan deteksi masker
    if confidence > thresh_confidence:
        box = detections[0, 0, i, 3:7]*np.array([w,h,w,h])
        startX,startY,endX,endY = box.astype("int")

        (startX,startY) = (max(0,startX),max(0,startY))
        (endX,endY) = (min(w-1,endX), min(h-1,endY))

        #print(startX,startY,endX,endY)

        face = image[startY:endY,startX:endX]

        #set tangkapan kamera wajah dalam bentuk berwarna dan merubah ukuran agar bisa lebih terbaca
        face = cv2.cvtColor(face , cv2.COLOR_BGR2RGB)
        face = cv2.resize(face, (128,128))

        #Mengubah hasil tangkapan menjadi bentuk array lalu dilakukan preproses
        face = img_to_array(face)
        face = preprocess_input(face)

        #menambahkan ukuran array, dengan variabel face selaku array, dan axis diawali pada 0
        face = np.expand_dims(face , axis=0)

        #melakukan prediksi menggunakan model dan membaca hasil dari variabel face
        (mask,withoutMask) = model.predict(face)[0]

        #memberi label atau tulisan
        label = "Masker" if mask > withoutMask else "Tanpa_Masker"

        #memberikan warna hijau jika menggunakan masker, jika tidak akan berwarna merah
        color = (0,255,0) if label == "Masker" else (0,0,255)

        label = "{} : {:.2f}".format(label , max(mask,withoutMask)*100)

        #Meletakkan label di bagian atas kiri kotak
        cv2.putText(image,label,(startX,startY-10),cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)

        #Memberikan feedback kepada user berupa kotak yang mengelilingi wajah.
        cv2.rectangle(image,(startX,startY),(endX,endY),color,2)

        #digunakan jika ingin menghentikan aplikasi, dengan menekan tombol q
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cv2.imshow('Tes gambar',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

***Gambar 3. 15 Code* untuk melakukan prediksi pada gambar**

Gambar 3.15 merupakan salah satu kode yang digunakan untuk membangun algoritma deteksi masker. Setelah menjalankan *code* tersebut, maka akan muncul *windows* baru yang memunculkan hasil prediksi.



Gambar 3. 16 **Hasil prediksi gambar**

Dari hasil yang ditunjukkan pada gambar 3.16, model bisa mendeteksi yang mana saja wajah yang menggunakan masker dan tidak menggunakan masker, dengan satu kesalahan dimana wajah yang tidak sepenuhnya tertutup masker masih terhitung bahwa dia menggunakan masker. Hal ini disebabkan karena model tidak membaca wajah tersebut secara keseluruhan, dimana bagian leher tertutup masker, sehingga model membaca bagian wajah sebagian dan melihat bahwa masker telah digunakan.

Untuk tangkapan secara *real-time*, bisa dilakukan dengan *code* yang sama digunakan pada hasil prediksi gambar, yang ditunjukkan pada gambar 3.17 dan 3.18

```python
def detect_and_predict_mask(frame, faceNet, maskNet):
    # Mengambil dimensi dari gambar yang digunakan, dan membuat blob dari gambar tersebut
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (150,150),
        (104.0, 177.0, 123.0))

    # blob dioper ke model deteksi
    faceNet.setInput(blob)
    detections = faceNet.forward()

    # initialisasi list gambar, lokasi, dan prediksi
    faces = []
    locs = []
    preds = []

    # Melakukan Pengulangan untuk deteksi
    for i in range(0, detections.shape[2]):
        # Mengambil confidence untuk digunakan
        confidence = detections[0, 0, i, 2]

        # Membuat batasan confidence, jika diatas dari batasan maka bisa melanjutkan prediksi
        if confidence > thresh_confidence:
            # Menghitung koordinat x dan y untuk membuat kotak mengelilingi wajah
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # Memastikan ukuran kotak tidak melebihi resolusi
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            # Merubah hasil tangkapan menjadi berwarna,  lalu merubah resolusi tangkapan menjadi 128x128 agar lebih terbaca
              # kemudian Memproses hasil tangkapan tersebut ke bentuk array
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (128, 128))
            face = img_to_array(face)
            face = preprocess_input(face)

            # Memasukan hasil ke dalam list
            faces.append(face)
            locs.append((startX, startY, endX, endY))

    # Jika mendeteksi satu atau lebih wajah,
    if len(faces) > 0:
        # Akan memasukan semua tangkapan ke bentuk array lalu memprediksi semua hasil tangkapan
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    # return lokasi dari wajah yang diprediksi
    return (locs, preds)
```

***Gambar 3. 17* Kode untuk deteksi masker**

```
#Untuk capture video webcam
cap = cv2.VideoCapture(0)

#Mendapat resolusi dari webcam yang digunakan
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH) + 0.5)
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT) + 0.5)

#menentukan ukuran resolusi tangkapan berdasarkan hasil tangkapan sebelumnya
size = (width, height)

#Menentukan jenis video hasil untuk penyimpanan, disini menggunakan format .avi
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('your_video.avi', fourcc, 20.0, size)
writer = None

#Selagi program berjalan, akan melakukan :
while(True):
    #membaca hasil tangkapan, dan kemudian merubah resolusi dari hasil tangkapan tersebut
    _, frame = cap.read()
    frame = imutils.resize(frame, 900)

    #
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
#     print(locs,preds)
    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Masker" if mask > withoutMask else "Tanpa Masker"
        color = (0, 255, 0) if label == "Masker" else (0, 0, 255)

        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        cv2.putText(frame, label, (startX, startY - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    cv2.imshow('Recording...', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

#    if writer is None:
#        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
#        writer = cv2.VideoWriter('output.avi', fourcc, 25,
#                                 (frame.shape[1], frame.shape[0]), True)

    if writer is not None:
        writer.write(frame)

cap.release()
out.release()
cv2.destroyAllWindows()
```

***Gambar 3. 18 code* untuk mendeteksi masker secara *real-time* menggunakan kamera**

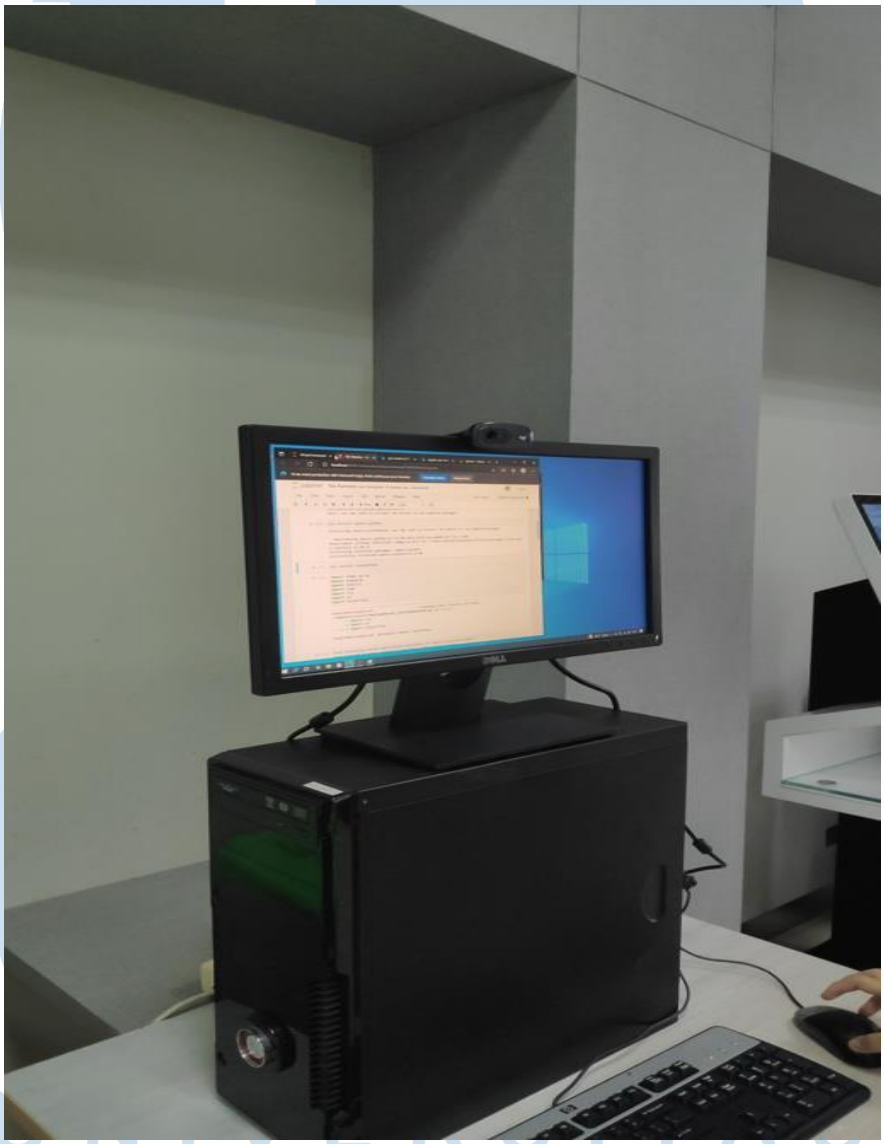
3.3.4 *Cutover*

Pada tahap *Cutover*, implementasi dilakukan di perpustakaan Universitas Multimedia Nusantara. *Setup* untuk menjalankan program, yaitu sebuah komputer yang disediakan oleh perpustakaan Universitas Multimedia Nusantara, selesai dilakukan pada tanggal 30 November 2021 oleh pihak perpustakaan Universitas Multimedia Nusantara. *Setup* ini diletakkan di bagian kanan pintu masuk perpustakaan UMN, tepatnya di sebelah mesin untuk pengembalian buku, sehingga mengawasi setiap orang yang masuk atau keluar dari perpustakaan UMN, dan jika ada yang menggunakan mesin pengembalian buku. Spesifikasi komputer yang dipakai adalah sebagai berikut :

- OS *Windows 10* 64-bit
- Prosesor *Intel I-5 7600K*
- *RAM 8GB*
- Kapasitas penyimpanan 150gb

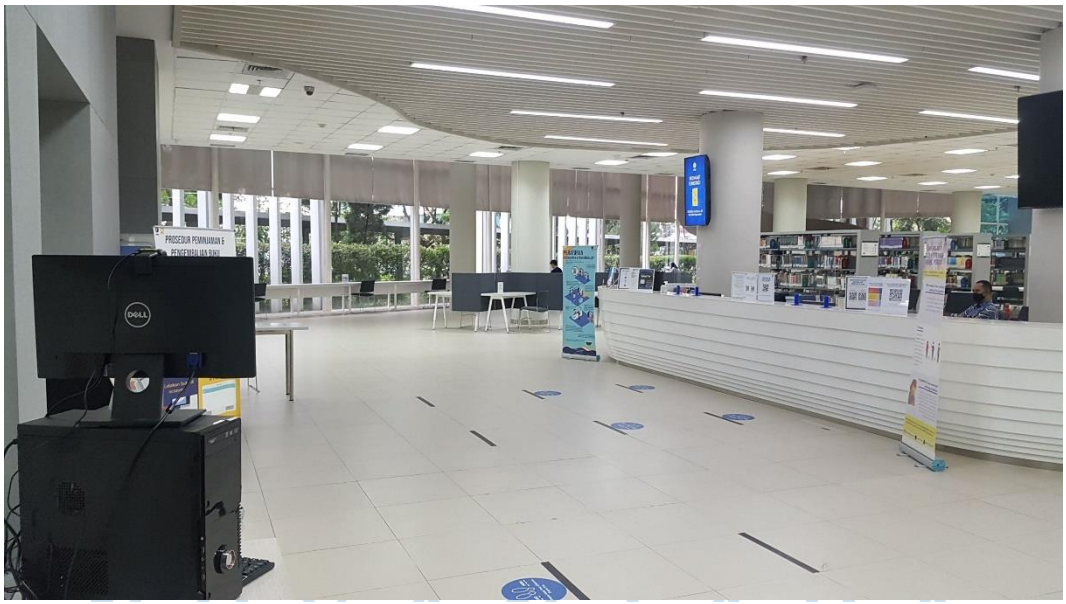
Selain itu, webcam *logitech C270* juga disediakan untuk merekam area perpustakaan. Untuk penggunaan jarak jauh, aplikasi *AnyDesk* digunakan sehingga tidak perlu ke area perpustakaan Universitas Multimedia Nusantara setiap hari kecuali untuk menyalakan komputer, dan bisa dikontrol di mana saja asalkan mempunyai akses.

Gambar 3.19 adalah gambar *setup* komputer yang ada di area perpustakaan Universitas Multimedia Nusantara, sedangkan gambar 3.19 sampai 3.21 merupakan area sekitar pemasangan *setup* komputer di perpustakaan Universitas Multimedia Nusantara.



Gambar 3. 19 **Setup komputer yang digunakan**

Gambar 3. 20 **Tampak samping kiri area *setup***



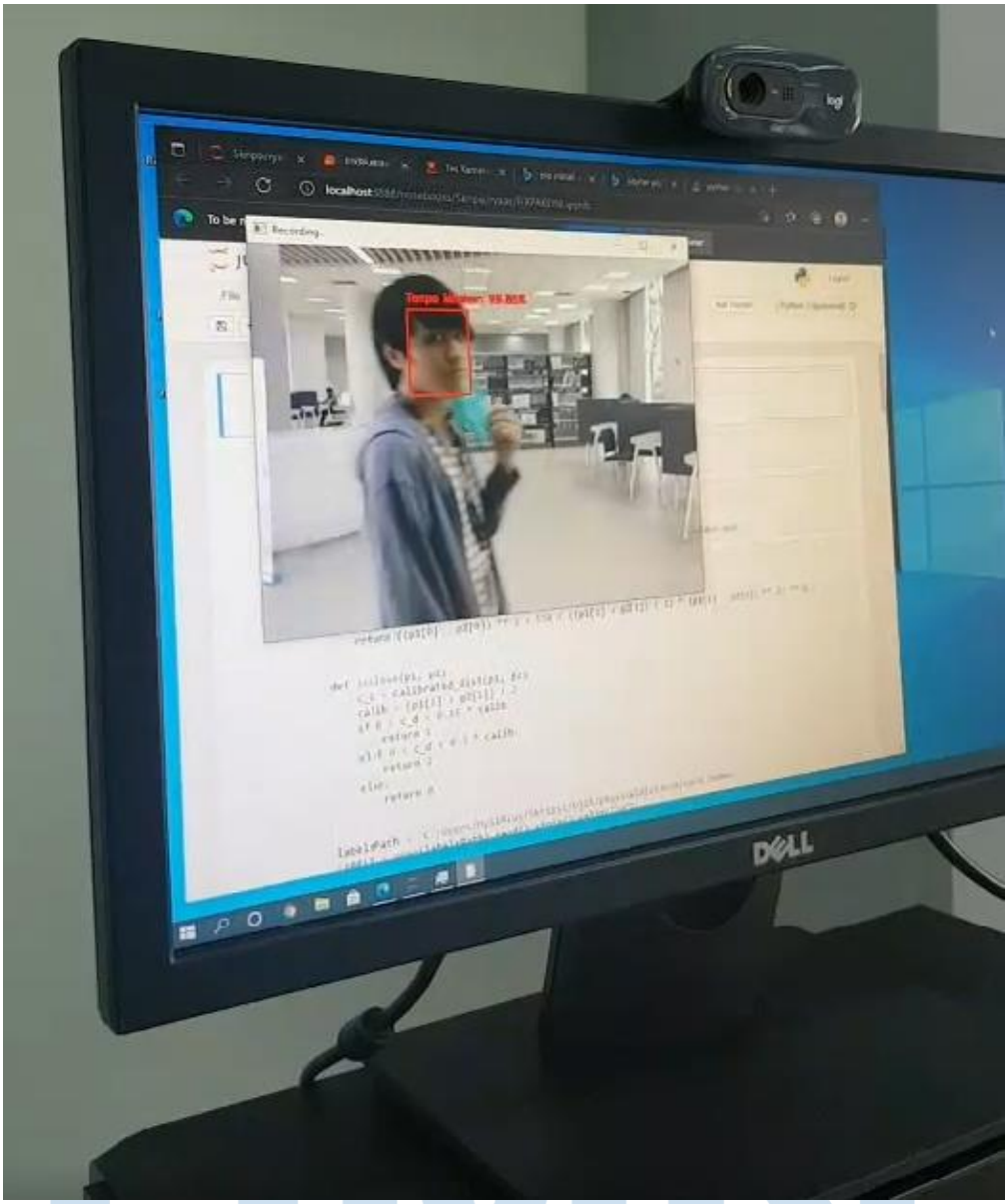
Gambar 3. 21 **Tampak samping kanan area setup, mengawasi pintu masuk dan jika ada yang menggunakan mesin *return* buku**

3.4 Hasil Penelitian

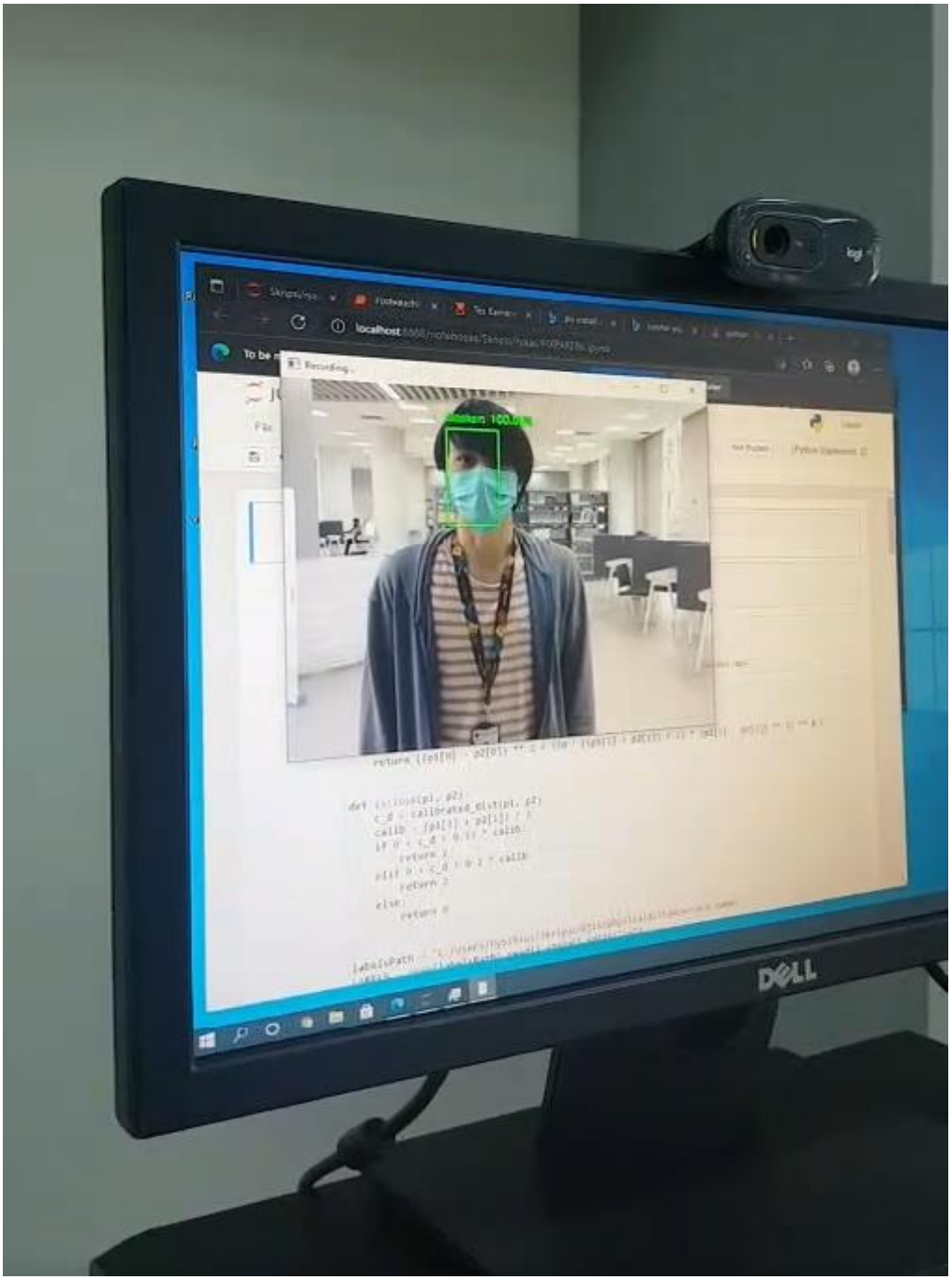
Pada sub bab hasil penelitian akan membahas hasil yang didapat setelah melakukan uji coba di area perpustakaan Universitas Multimedia Nusantara.

Walaupun kegiatan implementasi di perpustakaan Universitas Multimedia Nusantara masih berjalan, program yang sudah dibuat mendapat hasil denganberhasil mendeteksi wajah yang menggunakan masker dan tidak menggunakan masker.

Gambar 3.22 menunjukkan bahwa program, melalui *webcam,* bisa mendeteksi apakah orang yang melewati *webcam* menggunakan masker atau tidak. Dalam skenario ini, program memberikan *feedback* berupa kotak warna merah yang mengelilingi wajah tersebut dengan tulisan ‘tanpa masker’. Sedangkan untuk gambar 3.23 menunjukkan kotak hijau dengan label ‘Masker’ yang menunjukkan bahwa wajah yang terdeteksi dalam kamera sudah memakai masker dan ikut serta dalam melaksanakan salah satu protokol kesehatan, yaitu menggunakan masker.



Gambar 3. 22 **Terdeteksi tidak menggunakan masker**



Gambar 3. 23 **Terdeteksi menggunakan masker**

3.5 Kendala yang dihadapi

Penelitian ini tentu tidak lepas dari kendala-kendala yang berpotensi menghambah jalannya penelitian. Kendala yang muncul saat melaksanakan program penelitian independen adalah sebagai berikut :

1. Terdapat kurangnya informasi atau materi yang dibutuhkan untuk melanjutkan penelitian, seperti *library* yang mana saja yang cukup optimal untuk digunakan, sehingga diharuskan untuk mempelajari hal-hal yang sekiranya di luar pembelajaran program studi Sistem Informasi
2. Mendapatkan dataset yang mempunyai jumlah banyak namun bisa meningkatkan akurasi pelatihan data dan model agar program berjalan dengan optimal.

3.6 Solusi atas Kendala

Walaupun penelitian mempunyai beberapa kendala, solusi atas kendala juga didapatkan sehingga tidak menjadi sebuah halangan dalam penelitian. Solusi yang didapat berdasarkan kendala yang dihapadi adalah :

1. Membaca jurnal-jurnal yang membahas *library* yang bisa digunakan untuk pembuatan program pada penelitian.
2. Dataset didapat dari *pyimagesearch.com*, lebih tepatnya dibuat oleh Prajna Bhandary, yang memanfaatkan *facial landmark* untuk membuat dataset pengguna masker secara artifisial. Walaupun dibuat secara artifisial, dataset yang berjumlah 8805 ini masih bisa digunakan untuk kasus nyata sehingga bisa diterapkan pada program penelitian ini.