BAB IV
PERANCANGAN

Pada bab ini, penelitian mejelaskan tentang kebututhan spesifikasi sistem, cara kerja sistem, rancangan antarmuka sistem, dan rancangan pengujian sistem yang dibuat.

4.1 Analisis Sistem Terdahulu

Dalam penelitian kali ini, peneliti merujuk pada studi sebelumnya yang dilakukan oleh Khusnul Khotimah, M. Taqijjuddin Alawiy, dan Bambang Minto Basuki. Mereka mengembangkan aplikasi untuk mendeteksi helm dan mengklasifikasikan pengendara motor berdasarkan penggunaan helm menggunakan Convolutional Neural Network (CNN) dengan algoritma You Only Look Once (YOLO). Studi ini akan menjadi pembanding penting dalam penelitian terkait identifikasi plat nomor kendaraan.

4.2 Spesifikasi Kebutuhan Sistem Baru

Spesifikasi kebutuhan sistem baru mengharuskan sistem dibagi menjadi empat bagian, yaitu spesifikasi kebutuhan perangkat lunak, perangkat keras, input, dan output.

4.2.1 Spesifikasi Kebutuhan Perangkat Lunak

Persyaratan Perangkat Lunak menjelaskan persyaratan yang diperlukan bagi peneliti untuk mengembangkan aplikasi. Di bawah ini adalah spesifikasi perangkat lunak yang diperlukan.

Tabel 4.1 Spesifikasi Kebutuhan Perangkat Lunak

No	Nama Perangkat	Keterangan
1	Windows 10	Sistem operasi komputer yang digunakan.
2	Visual Studio Code	Apikasi editor yang digunakan untuk membuat kode program menggunakan python.

3	Google Collaboratory	*IDE* online untuk melakukan pembuatan model Yolo.
4	*Web Browser*	Aplikasi yang digunakan untuk mengakses *web* di internet.
5	Google Drive	Penyimpanan dataset.

4.2.2 Spesifikasi Kebutuhan Perangkat Keras

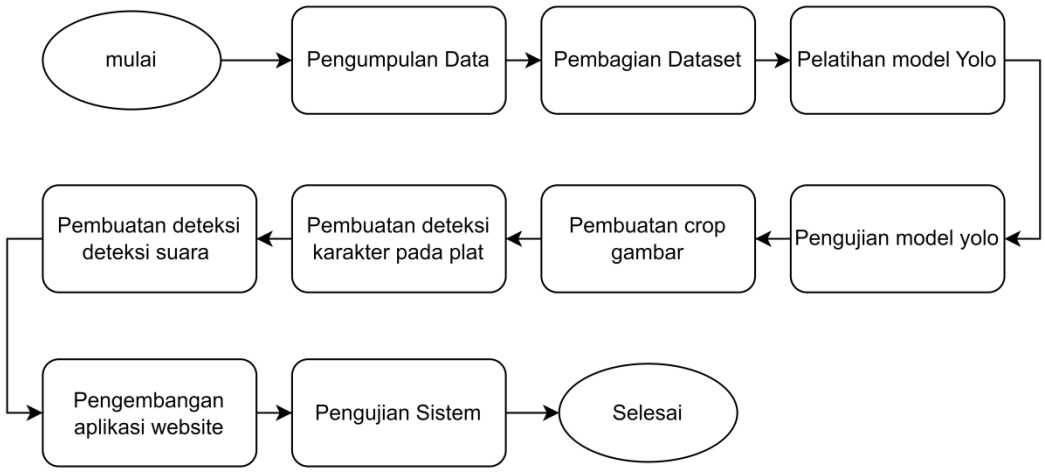
Spesifikasi kebutuhan perangkat keras yang dibutuhkan dalam penelitian ini meliputi prosesor, hard disk, dan memori. Berikut adalah rincian spesifikasi perangkat keras yang dibutuhkan:

Tabel 4.2 Spesifikasi Kebutuhan Perangkat Keras

No	Nama Perangkat	Keterangan
1	*Processor*	Intel i7-9750H
2	*Harddisk*	256 GB SSD & 1 TB HDD
3	*RAM*	16 GB
4	*VGA*	NVIDIA GeForce GTX 880M
5	Kamera	Webcam Namesis Model A50
6	Microfon	Realtek HighDefinition Audio

4.3 Perancangan Sistem

Tahap perancangan aplikasi indentifikasi kendaraan bermotor bersuara bising disertai dengan fitur pengenalan nomor kendaraan berbasis yolo yang akan diimplementasikan untuk mendeteksi adanya kendaraan bersuara bising pada suatu tempat dengan menangkap gambar serta mendeteksi nomor plat kendaraan tersebut terdiri dari beberapa tahap. Berikut tahapan pembuatan model YOLO dan pengembangan sistem untuk mendeteksi nomor plat kendaraan bersuara bising.



Gambar 4.1 Tahapan Pembuatan Model *Yolo*

4.3.1 Pengumpulan Dataset

Dataset digunakan untuk keperluan analisis dan deteksi model yolo yang akan dibuat. Dataset ini akan dibuat menjadi data training untuk melatih model dan data testing untuk menguji model yang sudah dilatih. Dataset didapatkan melalui website kaggle yang bersifat *open source*. Dataset berisi 500 gambar dengan berbagai macam jenis plat nomor kendaraan di Indonesia yang memiliki warna hitam, merah dan putih. Berikut untuk contoh dataset yang dijelaskan pada tabel 4.3 sebagai berikut.

Tabel 4.3 Contoh Gambar Dataset Aplikasi Identifikasi plat nomor kendaraan

No	Gambar	Keterangan
1		Gambar plat nomor kendaraan berwarna hitam dengan nomor E 6586 PAM yang dianotasikan sebagai objek plat nomor kendaraan.
2		Gambar plat nomor kendaraan berwarna merah dengan nomor E 9967 P yang dianotasikan sebagai objek plat nomor kendaraan.

3		Gambar plat nomor kendaraan berwarna putih dengan nomorB 2156 TOR yang dianotasikan sebagai objek plat nomor kendaraan.

4.3.2 Pengolahan Dataset

Dataset yang sudah disediakan maka akan diolah dengan membagi gambar menjadi data training, data validation dan data testing dengan rasio pembagian data training 60%, data validation 20% dan data testing 20%. Berikut proses dan penjelasannya:

```
ano_paths=[]
for dirname, _, filenames in os.walk('/content/datasets'):
    for filename in filenames:
        if filename[-4:]=='.txt' and filename[0:3]=='IMG' :
            ano_paths+=[(os.path.join(dirname, filename))]

n=len(ano_paths)
print(n)
N=list(range(n))
random.shuffle(N)
```

Gambar 4.2 Pembagian dataset

Pada Gambar 4.2 dilakukan pembagian menjadi data training, validation dan testing data akan diambil dari *storage google drive* yang kemudian gambar akan diacak sebelum dilakukan pembagian dataset.

```
train_ratio = 0.6
valid_ratio = 0.2
test_ratio = 0.2

train_size = int(train_ratio*n)
valid_size = int(valid_ratio*n)

train_i = N[:train_size]
valid_i = N[train_size:train_size+valid_size]
test_i = N[train_size+valid_size:]
```

Gambar 4.3 Pembuatan rasio pembagian data

Pada Gambar 4.3 menentukan rasio pembagian data yang mana variabel tersebut akan digunakan sebagai acuan banyaknya data yang akan dibagi kepada masing masing data *training*, *validation* dan *testing*.

```
for i in train_i:
    ano_path=ano_paths[i]
    img_path=os.path.join('/content/datasets',
                          ano_path.split('/')[-1][0:-4]+'.jpg')

    try:
        !cp {ano_path} {train_path}
        !cp {img_path} {train_path}
    except:
        continue
print(len(os.listdir(train_path)))
```

Gambar 4.4 Pembagian data *training*

Pada Gambar 4.4 pembagian data training dilakukan dengan melakukan *copyfile* dari *storage google drive ke dalam IDE Google Colaboratory* dengan jumlah yang sudah ditentukan yaitu 60% dari 500 data gambar yang sudah diacak.

```
for i in valid_i:
    ano_path=ano_paths[i]
    img_path=os.path.join('/content/datasets',
                          ano_path.split('/')[-1][0:-4]+'.jpg')

    try:
        !cp {ano_path} {valid_path}
        !cp {img_path} {valid_path}
    except:
        continue
print(len(os.listdir(valid_path)))
```

Gambar 4.5 Pembagian data *validation*

Pada Gambar 4.5 pembagian data validation dilakukan dengan melakukan *copyfile* dari *storage google drive* ke dalam *IDE Google Colaboratory* dengan jumlah yang sudah ditentukan yaitu 20% dari 500 data gambar yang sudah diacak.

```
for i in test_i:
    ano_path=ano_paths[i]
    img_path=os.path.join('/content/datasets',
                          ano_path.split('/')[-1][0:-4]+'.jpg')

    try:
        !cp {ano_path} {test_path}
        !cp {img_path} {test_path}
    except:
        continue
print(len(os.listdir(test_path)))
```

Gambar 4.6 Pembagian data testing

Pada Gambar 4.6 pembagian data testing dilakukan dengan melakukan *copyfile* dari *storage google drive* ke dalam *IDE Google Colaboratory* dengan

jumlah yang sudah ditentukan yaitu 20% dari 500 data gambar yang sudah diacak.

4.3.4 Training model Yolo

Pata tahap ini dilakukan pembuatan dan pelatihan model Yolo yang sudah disediakan oleh library pada bahasa pythn dengan nama ultralytics. Model tersebut kemudian dilatih menggunakan dataset training agar model tersebut mampu mengenali objek plat noor kendaraan yang akan dideteksi.

```
model = YOLO("yolov8x.pt")
!yolo task=detect mode=train model=yolov8x.pt data=data.yaml epochs=20 imgsz=480
```

Gambar 4.7 Training model Yolo

Model yolo yang sudah disediakan kemudian akan dilatih dengan beberapa konfigurasi diantaranya menggunakan *task= detect* artinya model akan melakukan deteksi. Mode=train maka model akan diarahkan untuk mempelajari data. Model=yolov8x adalah versi model yolo yang digunakan. Data= data.yaml merupakan dataset yang sudah di anotasikan pada proses sebelumnya. *Epoch*=20 yang artinya model akan mempelajari seluruh gambar pada data training sebanyak 20 kali iterasi. Selanjutnya konfigurasi imgsz = 480 yang artinya gambar input akan di ubah ukurannya sebesar 480pixel pada masing masing gambar yang akan dipelajari oleh model Yolo.

4.3.5 Testing model Yolo

Melakukan testing menggunakan model yolo akan menghasilkan boundary box yang akan mendeteksi keberadaan plat nomor kendaraan pada suatu gambar yang akan diujikan, berikut untuk cara melakukan testing pada model yolo yang sudah dilatih.

```
best_path0='runs/detect/train/weights/best.pt'
model2 = YOLO(best_path0)
image_path = '/content/6509428b1406d.jpeg'
results = model2.predict(image_path, conf=0.4)
```

Gambar 4.8 *Testing model yolo*

Pada Gambar 4.8 proses Testing yang dilakukan menggunakan modl yolo yang sudah disimpan dari hasil pelatihan pada proses sebelumnya yakni file dengan nama best.pt, kemudian akan dicoba melakukan deteksi gambar dengan nama 6509428b1406d.jpeg yang akan menghasilkan gambar dan boundary box hasil deteksi plat kendaraan pada Gambar 4.9 sebagai berikut.

Gambar 4.9 Hasil testing model Yolo

4.3.6 Crop gambar hasil deteksi yolo

Gambar yang sudah dideteksi sebelumnya menggunakan modle yolo akan menghasilkan boundary box yang mendetesi keberadaan plat pada suatu gambar yang selanjutnya akan dilakukan croping guna untuk menghasilkan output gambar yang sudah dideteksi oleh model.

```
def draw_box(img_path, boxes):
    image = cv2.imread(img_path)
    crops = []
    for box in boxes:
        x1, y1, x2, y2, conf, cls = box
        x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
        label = class_map[int(cls)]
        cv2.putText(image, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

        # Crop the detected plate number
        crop = image[y1:y2, x1:x2]
        crops.append(crop)

    return image, crops

result_img, crops = draw_box(image_path, results[0].boxes.data.cpu().numpy())
plt.imshow(cv2.cvtColor(result_img, cv2.COLOR_BGR2RGB))
plt.show()
```

Gambar 4.10 Proses Crop gambar hasil deteksi plat kendaraan

Pada Gambar 4.10 merupakan proses dilakukannya *croping* gambar pada *boundary box* hasil dari deteksi model yolo terhadap suatu gambar yang kemudaian gambar tersebut akan ditampilkan pada gambar 4.11 sebagai berikut.

Gambar 4.11 hasil *crop* gambar

4.3.7 Deteksi text pada gambar plat kendaraan

Deteksi text pada plat kendaraan pada penelitian ini menggunakan library pytesseract (Tesseract-OCR) yang dapat mendeteksi karakter huruf atau angka pada suatu gambar berikut penggunaan library tersebut dijelaskan pada Gambar 4.12 sebagai berikut.

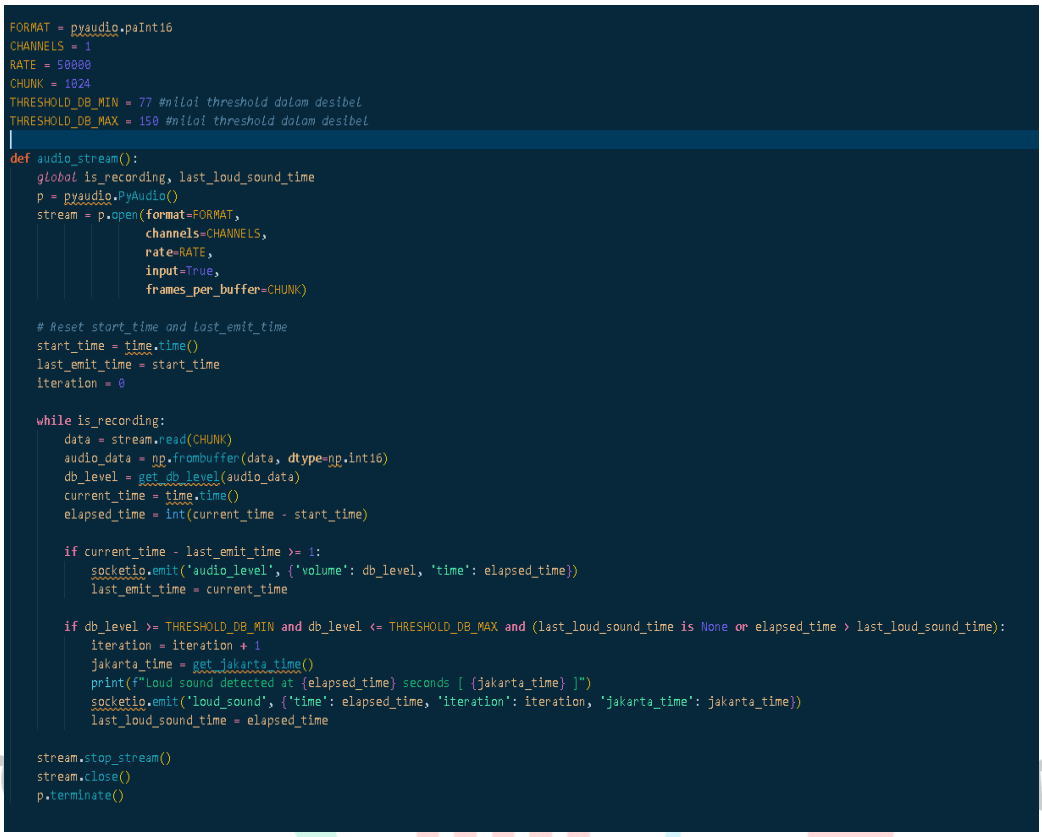
```
import pytesseract
def ocr_plate_images(crops):
    plate_texts = []
    for i, crop in enumerate(crops):
        gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

        # Use pytesseract to perform OCR on the binary image
        text = pytesseract.image_to_string(binary, config='--psm 8')
        plate_texts.append(text.strip())

        # Save the cropped image
        crop_filename = os.path.splitext(os.path.basename(image_path))[0] + f'_crop_{i}.jpg'
        crop_path = os.path.join(output_dir, crop_filename)
        cv2.imwrite(crop_path, crop)
        print(f'Saved cropped image: {crop_path}')

    return plate_texts
```

Gambar 4.12 Deteksi text plat kendaraan

Hasil deteksi teks plat nomor kendaraan terdapat pada table aplikasi dan table excel yang dapat diunduh didalam aplikasi. Tabel tersebut digunakan pengguna untuk melapor kepada pihak berwajib.

4.3.8 Deteksi Suara Bising

Pada penelitian ini menerapkan deteksi suara bising menggunkan library *python* bernama *pyaudio* yang digunakan untuk mendeteksi suara bising dan menentukan kondisi saat aplikasi berjalan untuk memotret kendaraan sebelum gambar kendaraan yang diambil oleh kamera lalu diproses oleh model yolo untuk dideteksi plat nomornya. Berikut adalah penerapan pengguaaan deteksi suara bising menggunakan *pyaudio* yang disajikan pada Gambar 4.13 sebagai berikut.



```
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 50000
CHUNK = 1024
THRESHOLD_DB_MIN = 77 #nilai threshold dalam desibel
THRESHOLD_DB_MAX = 150 #nilai threshold dalam desibel

def audio_stream():
    global is_recording, last_loud_sound_time
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

    # Reset start_time and last_emit_time
    start_time = time.time()
    last_emit_time = start_time
    iteration = 0

    while is_recording:
        data = stream.read(CHUNK)
        audio_data = np.frombuffer(data, dtype=np.int16)
        db_level = get_db_level(audio_data)
        current_time = time.time()
        elapsed_time = int(current_time - start_time)

        if current_time - last_emit_time >= 1:
            socketio.emit('audio_level', {'volume': db_level, 'time': elapsed_time})
            last_emit_time = current_time

        if db_level >= THRESHOLD_DB_MIN and db_level <= THRESHOLD_DB_MAX and (last_loud_sound_time is None or elapsed_time > last_loud_sound_time):
            iteration = iteration + 1
            jakarta_time = get_jakarta_time()
            print(f"Loud sound detected at {elapsed_time} seconds [ {jakarta_time} ]")
            socketio.emit('loud_sound', {'time': elapsed_time, 'iteration': iteration, 'jakarta_time': jakarta_time})
            last_loud_sound_time = elapsed_time

    stream.stop_stream()
    stream.close()
    p.terminate()
```

Gambar 4.13 Deteksi suara bising

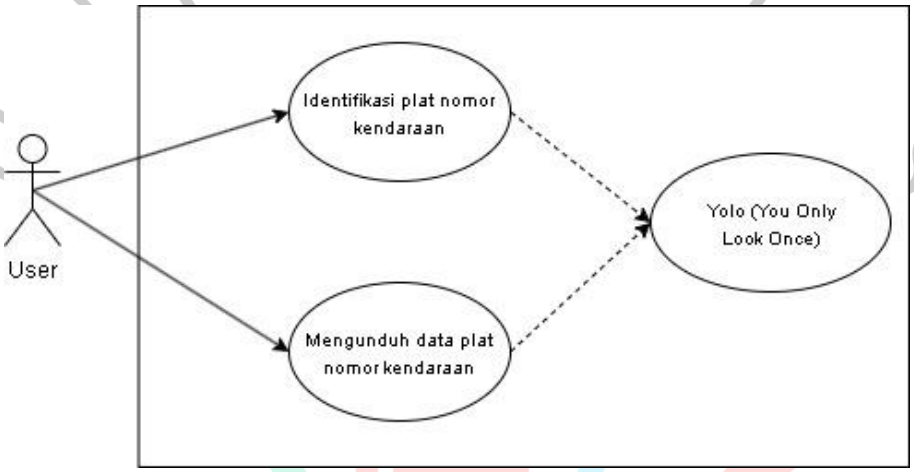
Pada gambar 4.13 Kode tersebut bertujuan untuk merekam audio, mendeteksi suara keras, dan mengirimkan informasi terkait melalui socket menggunakan PyAudio dan SocketIO. Pada awalnya, stream audio diinisialisasi dan dibuka dengan parameter tertentu, kemudian dalam loop utama, data audio dibaca dalam chunk, dikonversi menjadi array integer 16-bit, dan level desibel dihitung. Setiap detik, level audio terkini dikirimkan melalui socket. Jika suara keras terdeteksi dalam ambang batas desibel yang ditentukan, 'loud_sound' dikirimkan beserta waktu terdeteksi, iterasi, dan waktu lokal Jakarta. Setelah proses perekaman selesai, stream audio dihentikan dan ditutup, serta instance PyAudio diakhiri. Kode ini juga mempertimbangkan penanganan potensi pengecualian dan penggunaan variabel global secara minimal untuk memastikan operasi yang stabil dan efisien.

Suara keras yang dimaksud adalah suara yang melebihi ambang batas yang telah ditentukan oleh pemerintah. Salah satu peraturan yang mengatur hal ini adalah Peraturan Menteri Negara Lingkungan Hidup Nomor 7 Tahun 2009 tentang Ambang Batas Kebisingan Kendaraan Bermotor Tipe Baru. Pasal 4 menetapkan tingkat kebisingan maksimum untuk kendaraan bermotor tipe baru sebagai berikut:

1. Sepeda motor dengan kapasitas mesin hingga 80 cc: 77 dB(A).
2. Sepeda motor dengan kapasitas mesin lebih dari 80 cc hingga 175 cc: 80 dB(A).
3. Sepeda motor dengan kapasitas mesin lebih dari 175 cc: 83 dB(A).
4. Mobil penumpang dan kendaraan barang: 82 dB(A).

Peraturan ini dirancang untuk mengurangi polusi suara yang dihasilkan oleh kendaraan bermotor dan menjaga kualitas lingkungan hidup.

4.3.9 *Use Case Diagram*

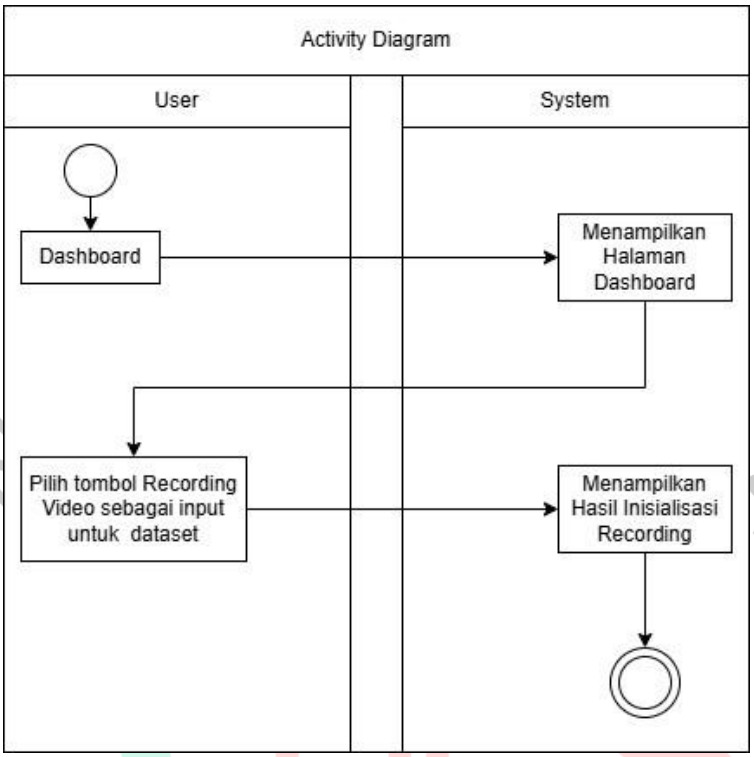


Gambar 4.14 Use Case Diagram Aplikasi

Berdasarkan Gambar 4.14, user atau pengguna setelah membuka aplikasi identifikasi plat nomor kendaraan dapat melihat tombol start, stop, reset, dan print excel secara langsung. Jika user menekan tombol start, maka aplikasi akan melakukan pengambilan gambar untuk diolah yolo, tombol stop digunakan oleh user untuk menghentikan aplikasi untuk mempermudah dalam mengunduh file excel.

4.3.10 *Activity Diagram*

Activity diagram merupakan gambaran alur aktivitas dari sebuah sistem yang dirancang (Kurniawan & Syarifuddin, 2020). Diagram ini berisi urutan proses dari aplikasi yang dibuat oleh peneliti. Gambar di bawah ini adalah activity diagram dari aplikasi identifikasi plat nomor kendaraan.



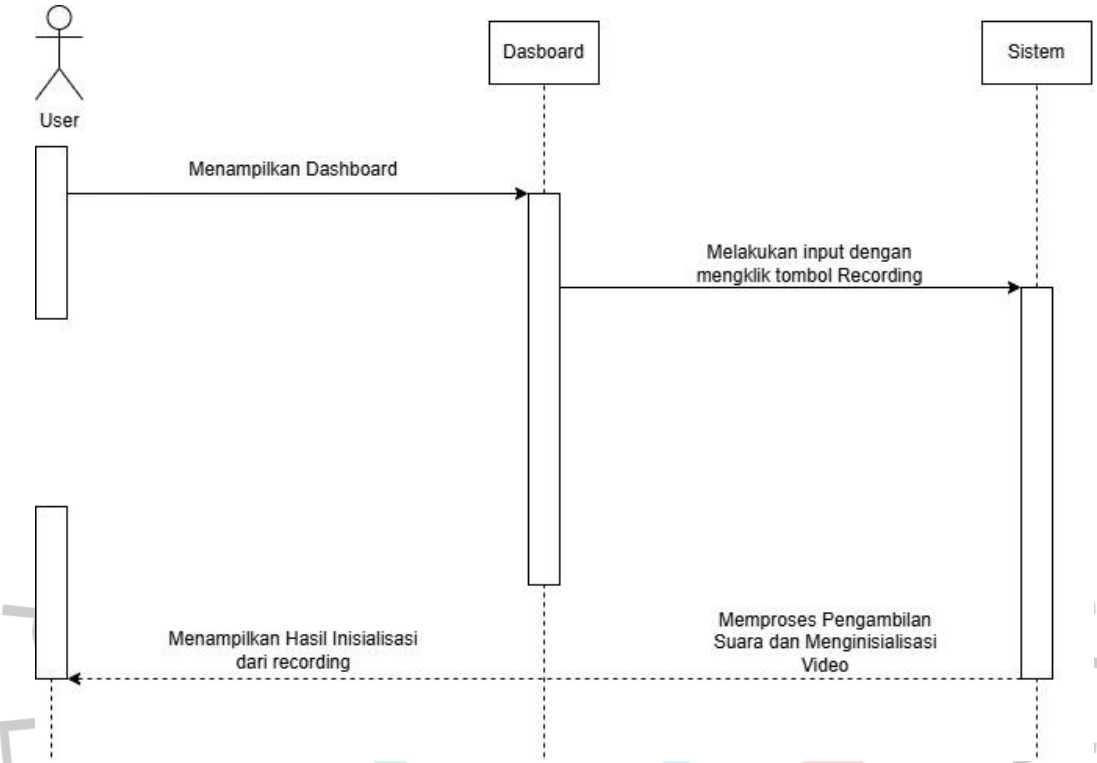
Gambar 4.15 *Activity Diagram*

Gambar 4.15 adalah activity diagram identifikasi plat nomor kendaraan untuk memulain tahap identifikasi plat nomor kendaraan hingga langkah-langkah pengguna dari masuk aplikasi sampai menampilkan hasil identifikasinya.

4.3.11 *Squence Diagram*

Sequence diagram adalah representasi visual dari urutan interaksi antara objek-objek yang terlibat dalam proses berjalannya sebuah sistem atau aplikasi. Diagram ini membantu memahami bagaimana objek-objek saling berinteraksi untuk mencapai tujuan akhir atau output yang diinginkan. Untuk aplikasi identifikasi plat nomor kendaraan, sequence diagram dapat menunjukkan interaksi antara komponen-komponen seperti kamera CCTV atau sensor gambar dengan sistem deteksi plat nomor. Ini mencakup langkah-langkah seperti pengambilan gambar, pengolahan citra untuk deteksi plat nomor, dan pengenalan karakter plat nomor. Diagram ini akan memvisualisasikan secara jelas bagaimana

informasi dan kontrol mengalir antara berbagai komponen dalam sistem untuk menghasilkan output identifikasi plat nomor kendaraan yang akurat.

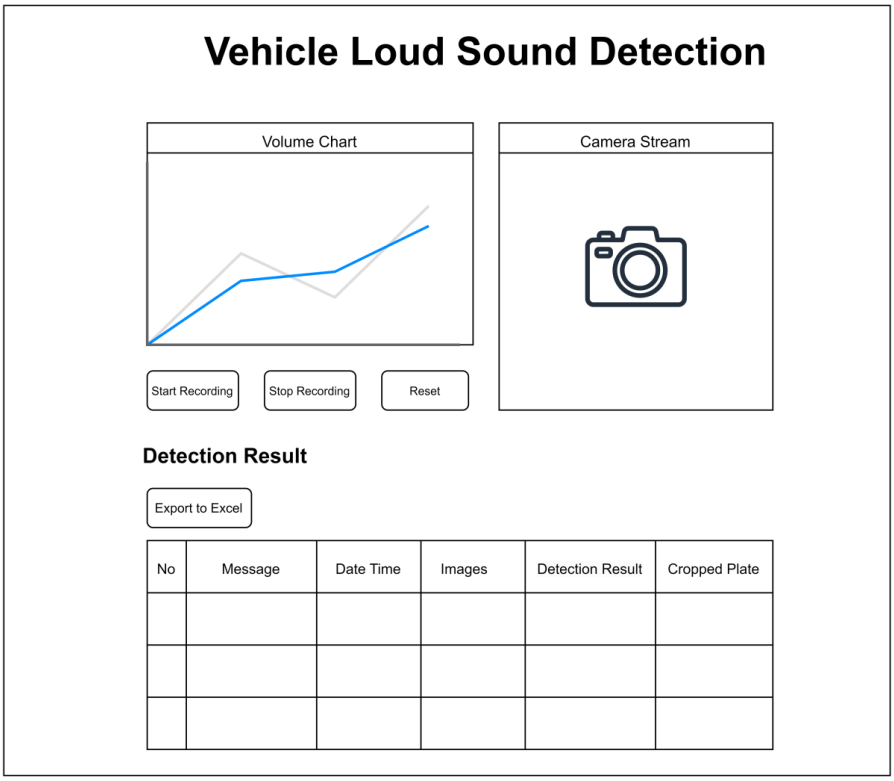


Gambar 4.16 Squence Diagram

Gambar 4.16 adalah sequence diagram identifikas plat nomor kendaraan yang menjelaskan proses dari menekan tombol start hingga memproses gambar plat nomor kendaraan yang ingin diidentifikasi serta menampilkan hasil identifikasi plat nomor kendaraan.

4.3.12Perancangan Antarmuka

Perancangan antarmuka merupakan suatu proses yang memiliki tujuan untuk menciptakan tampilan visual dan interaksi yang efektif antara pengguna dan sistem. Selain itu, perancangan antarmuka juga mempermudah tugas programmer dalam merancang aplikasi indentifikasi kendaraan bermotor bersuara bising disertai dengan fitur pengenalan nomor kendaraan berbasis yolo.



Gambar 4.17 Rancangan antar muka

Gambar diatas merupakan perancangan antarmuka halaman utama. Pada halaman utama juga user dapat melihat semua fitur seperti tombol start untuk memulai program, stop untuk memberhentikan program, reset untuk menghapus daftar table yang sudah di print, dan print untuk membuat file yang berisi plat nomor kendaraan.

4.3.12.1 *Black Box*

Pengujian *blackbox* berfokus pada output yang dihasilkan berdasarkan input yang diberikan. Berikut adalah contoh perencanaan pengujian Balck box yang akan dilakukan pada penelitian ini dijelaskan pada tabel 4.4 sebagai berikut.

Tabel 4.4 *Black Box*

No	Fitur Aplikasi	Input	Output
1	Halaman Utama	Akses url aplikasi	Menampilkan halaman utama
2	Tombol Start Recording	Klik tombol Start Recording	Menjalankan deteksi plat nomor kendaraan yang bersuara kencang.
3	Tombol stop recording	Klik tombol Stop Recording	Menghentikan deteksi plat nomor kendaraan yang bersuara kencang.
4	Tombol Reset	Klik tombol Reset	Menghapus histori deteksi plat nomor kendaraan yang bersuara kencang sebelumnya
5	Tombol Export to Excel	Klik tombol Export to Excel	Mendownload hasil deteksi plat nomor kendaraan yang bersuara kencang yang tertera pada tabel hasil deteksi

4.3.12.2 *White Box*

Pengujain *white box* berfokus pada analisa yang dilakukan pada aplikasi hingga kedalam kode program untuk memastikan kode program berjalan dengan baik untuk menjalankan aplikasi identifikasi kendaraan bersuara bising menggunakan Yolo