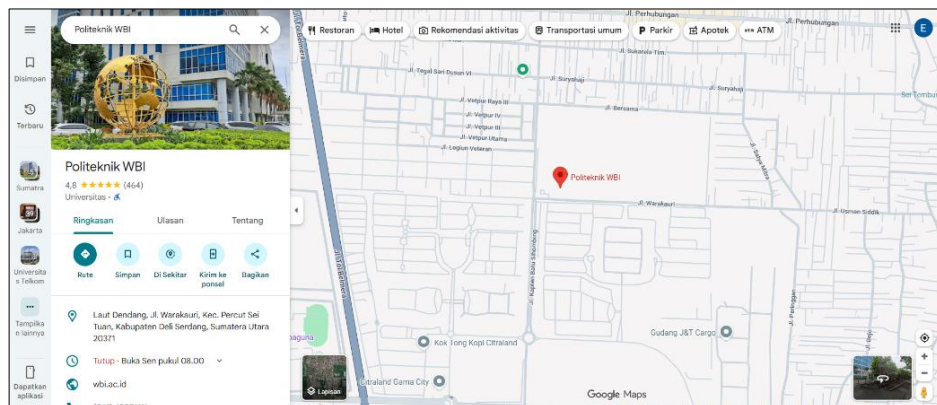


BAB III. METODOLOGI

3.1. Waktu dan Tempat Penelitian

Penelitian ini berlangsung pada tahun ajaran 2026/2027, dengan proses pengembangan dan pengujian sistem dikerjakan secara mandiri oleh penulis menggunakan perangkat pribadi. Implementasi aplikasi, pelatihan model, serta evaluasi kinerja klasifikasi dilaksanakan di lingkungan kampus Politeknik Wilmar Bisnis Indonesia

Untuk memastikan sistem dapat bekerja di kondisi nyata, pengujian aplikasi dilakukan langsung di lahan pohon jeruk milik kampus, terutama pada tahap pengambilan citra daun via perangkat mobile. Adapun data latih model dihimpun dari dataset citra daun jeruk daring, dilengkapi data pendukung hasil dokumentasi langsung di kebun jeruk kampus.

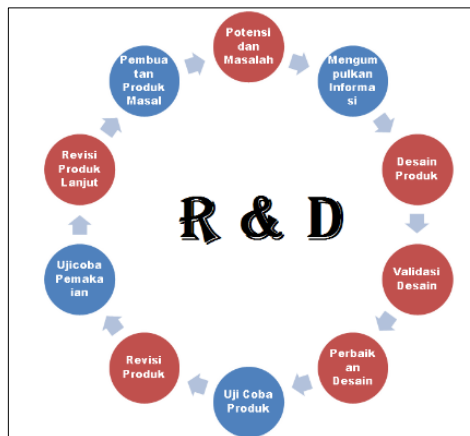


Gambar 3.1 Waktu dan Tempat Penelitian

(Sumber: Google Maps, 2026)

3.2. Jenis Penelitian

Penelitian ini merupakan penelitian dan pengembangan *Research and Development* (R&D) dengan pendekatan kuantitatif untuk merancang, mengembangkan, dan menguji aplikasi *mobile* berbasis CNN yang mendeteksi penyakit daun jeruk (CNN) (Okpatrioka 2023).

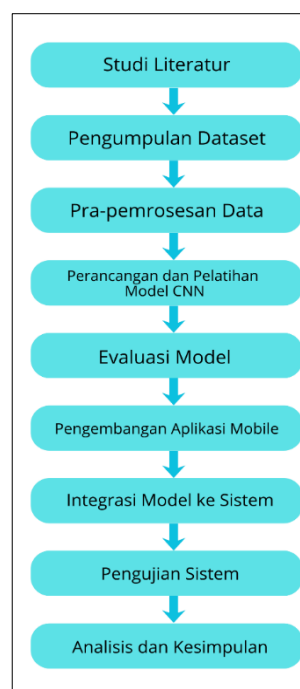


Gambar 3.2 Siklus *Research and Development* (R&D)

(Sumber: Jurnalhala.id, 2023)

3.3. Tahapan Penelitian

Tahapan penelitian dimulai dari studi literatur, pengumpulan dataset, dan pra-pemrosesan data, kemudian dilanjutkan dengan perancangan serta pelatihan model CNN berarsitektur MobileNetV2. Model yang telah dilatih dievaluasi, diintegrasikan ke aplikasi mobile, lalu diuji fungsionalitasnya. Seluruh hasil dianalisis untuk menarik kesimpulan.



Gambar 3.3 Tahapan Penelitian

3.4. Pengumpulan Data

Pengumpulan data dilakukan melalui dua kegiatan, yaitu studi literatur dan pengumpulan dataset.

3.4.1. Dataset

Dataset bersumber dari Kaggle berupa citra daun jeruk yang telah berlabel penyakit. Dataset ini dipilih karena jumlah datanya memadai, kondisi daunnya bervariasi, dan label kelasnya jelas; anotasi dan augmentasi yang sudah tersedia juga mempermudah pelatihan model CNN MobileNetV2 untuk klasifikasi penyakit daun jeruk.

- Nama Dataset: *Orange Leaf Disease*
- Sumber: www.kaggle.com/datasets/shuvokumarbask4004/orange-leaf-disease-dataset

3.4.2. Struktur Dataset

Struktur dataset ini tersusun dalam lima folder kategori. Untuk memperkaya variasi data dan generalisasi model, seluruh gambar telah melalui berbagai teknik augmentasi seperti *blurred, brightened, cropped, flipped, gamma corrected, grayscale, inverted, noisy, rotated, scaled, sharpened, sheared, shifted, dan zoomed*.

3.4.3. Kriteria Dataset

Adapun kriteria data penelitian harus memenuhi kriteria berikut:

- Objek utama berupa daun jeruk terlihat dominan pada gambar.
- Latar belakang polos sehingga tidak mengganggu proses identifikasi.
- Setiap gambar telah memiliki label kelas yang sesuai.
- Format berbentuk JPEG/JPG
- Ukuran gambar kecil yang mendukung pelatihan model (Acarya, Muhaimin, and Hindrayani 2024).

3.5. Metode Analisis

Metode analisis pada penelitian ini mencakup pengolahan dataset, pembangunan model, serta evaluasi kinerjanya sebelum diimplementasikan ke aplikasi, dengan seluruh proses analisis dijalankan pada platform Google Colab.

3.5.1. Pra-pemrosesan Data

Tahap pra-pemrosesan bertujuan menyiapkan dataset sebelum masuk ke pelatihan model, meliputi pembagian data menjadi *training, validation, dan testing*, penyeragaman ukuran citra menjadi 224×224 piksel, serta pemrosesan lewat

fungsi *preprocess_input* dan *ImageDataGenerator* agar data siap dipakai untuk pelatihan (Soleha, Hidayat, and Zulus 2025).

Tabel 3.1 Parameter Pra-pemrosesan Data

No.	Parameter	Keterangan
1.	Metode Pembagian	<i>train-val-test</i>
2.	Ukuran Citra	224 × 224 piksel
3.	Metode Pra-pemrosesan	<i>preprocess_input</i> (MobileNetV2)
4.	Data Generator	<i>ImageDataGenerator</i>
5.	Batch Size	32

3.5.2. Perancangan Model

Model dirancang dengan pendekatan *Transfer Learning*. MobileNetV2 berbobot awal (*pre-trained weights*) dari ImageNet berperan sebagai *feature extractor*, kemudian ditambahkan lapisan klasifikasi yang terdiri atas *GlobalAveragePooling2D*, *Dropout*, dan *Dense* berfungsi aktivasi *Softmax* untuk mengelompokkan citra daun jeruk ke dalam lima kelas (Soleha, Hidayat, and Zulus 2025).

Tabel 3.2 Parameter Perancangan Model

No.	Parameter	Keterangan
1.	Metode	<i>Transfer Learning</i>
2.	Bobot Awal	<i>ImageNet</i>
3.	Input Citra	224 × 224 × 3
4.	Base Model	<i>Freeze</i>
5.	<i>Global Pooling</i>	<i>GlobalAveragePooling2D</i>
6.	<i>Dropout</i>	Mengurangi <i>overfitting</i>
7.	<i>Output Layer</i>	<i>Dense</i> (5 neuron)
8.	Fungsi Aktivasi	<i>Softmax</i>
9.	Jumlah Kelas	5

3.5.3. Pelatihan Model

Model dilatih menggunakan dataset hasil pra-pemrosesan melalui pendekatan *Transfer Learning* pada arsitektur MobileNetV2, dengan proses pelatihan dijalankan di Google Colab menggunakan TensorFlow sampai diperoleh model dengan performa terbaik (Soleha, Hidayat, and Zulus 2025).

Tabel 3.3 Parameter Pelatihan Model

No.	Parameter	Keterangan
1.	Framework	<i>TensorFlow</i>
2.	<i>Optimizer</i>	Adam
3.	<i>Learning Rate</i>	0,0001
4.	<i>Loss Function</i>	<i>Categorical Crossentropy</i>
5.	Metrik	<i>Accuracy</i>
6.	<i>Epoch</i>	10
7.	<i>Callback</i>	<i>EarlyStopping, ModelCheckpoint</i>

3.5.4. Evaluasi Model

Kinerja model dievaluasi dengan data testing menggunakan metrik *accuracy*, *loss*, *precision*, *recall*, *F1-score*, dan *confusion matrix* (M. F. A. Maulana, Anggadimas, and Sani 2025).

Tabel 3.4 Parameter Evaluasi Model

No.	Metrik	Rumus	Fungsi
1.	<i>Accuracy</i>	$(TP + TN) / (TP + TN + FP + FN)$	Mengukur ketepatan prediksi secara keseluruhan
2.	<i>Precision</i>	$TP / (TP + FP)$	Menilai ketepatan prediksi pada tiap kelas penyakit
3.	<i>Recall (Sensitivity)</i>	$TP / (TP + FN)$	Menilai kemampuan model menemukan seluruh data yang relevan
4.	<i>F1-Score</i>	$2 \times (Precision \times Recall) / (Precision + Recall)$	Menyeimbangkan nilai precision dan recall
5.	<i>Confusion Matrix</i>	-	Merinci kesalahan klasifikasi pada tiap kelas

3.5.5. Implementasi Model

Pada tahap implementasi, model hasil pelatihan dikonversi ke format TensorFlow Lite (.tflite) agar dapat dijalankan pada aplikasi *mobile*, lalu diintegrasikan ke aplikasi Flutter. Dengan demikian, pengguna dapat mendeteksi

penyakit daun jeruk dari foto yang diambil melalui kamera atau galeri (Soleha, Hidayat, and Zulus 2025).

Tabel 3.5 Parameter Implementasi Model

No.	Parameter	Keterangan
1.	Format Model	<i>TensorFlow Lite</i> (.tflite)
2.	Format Label	.txt

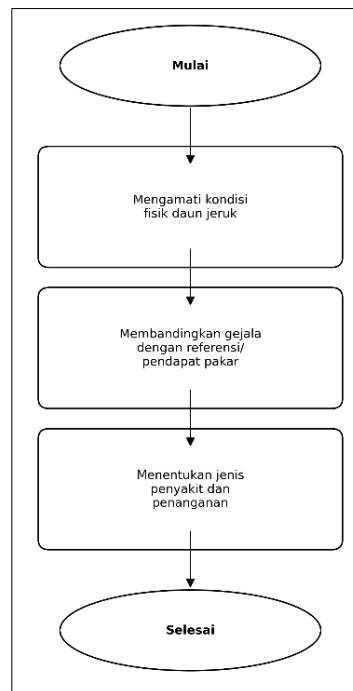
BAB IV. DESAIN DAN SISTEM

4.1. Keadaan Awal Sistem

Keadaan sistem awal digunakan sebagai identifikasi serta permasalahan yang menjadi dasar pengembangan aplikasi.

4.1.1. Identifikasi Penyakit Secara Manual

Identifikasi penyakit daun jeruk masih dilakukan secara manual melalui pengamatan visual terhadap daun, proses dimulai dengan pengamatan kondisi daun, kemudian gejala yang ditemukan dibandingkan dengan referensi atau dikonsultasikan kepada pakar untuk menentukan jenis penyakit. Selanjutnya hasil digunakan sebagai dasar dalam menentukan penanganan yang sesuai.



Gambar 4.1 Keadaan Awal Sistem

4.1.2. Analisis Permasalahan Sistem

Berdasarkan proses secara manual, terdapat beberapa permasalahan dalam menentukan jenis penyakit. Analisis dilakukan untuk mengidentifikasi kelemahan yang berjalan serta menjadi dasar dalam perancangan sistem.

Tabel 4.1 Analisis Permasalahan Sistem

No.	Permasalahan	Solusi
1.	Identifikasi manual	Model akan mengidentifikasi penyakit secara otomatis.
2.	Proses identifikasi bergantung pada pengetahuan dan pengalaman pengguna	Hasil deteksi secara otomatis disesuaikan dari dataset
3.	Membutuhkan waktu lama	Memberikan hasil deteksi secara cepat.

4.2. Keadaan Umum Objek Penelitian

Objek penelitian digunakan untuk membantu proses identifikasi kondisi daun jeruk secara otomatis dan dapat memberikan hasil deteksi yang cepat dan mudah digunakan oleh pengguna.

4.2.1. Objek Penelitian

Objek yang digunakan yakni data daun jeruk yang diklasifikasikan berdasarkan kondisi kesehatannya. Penelitian difokuskan pada lima jenis kondisi/penyakit.

Tabel 4.2 Keterangan 5 Kelas Kondisi dan Penyakit Daun

No.	Kondisi/Penyakit	Keterangan
1.	<i>Citrus_Canker_Diseases_Leaf_Orange</i>	Terinfeksi penyakit kanker jeruk
2.	<i>Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange</i>	Kekurangan nutrisi
3.	<i>Multiple_Diseases_Leaf_Orange</i>	Lebih dari satu gejala penyakit
4.	<i>Healthy_Leaf_Orange</i>	Sehat dewasa tanpa gejala penyakit
5.	<i>Young_Healthy_Leaf_Orange</i>	Sehat muda tanpa gejala penyakit

4.2.2. Data Penelitian

Dataset bersumber dari platform Kaggle. Dataset terdiri telah diberi label sesuai kondisi daun. Dataset terdiri atas lima kelas dengan total 38.432 citra yang telah melalui proses augmentasi.

Tabel 4.3 Pembagian dan Total Jumlah Citra Dataset

No.	Kondisi/Penyakit	Jumlah Citra
1.	<i>Citrus_Canker_Diseases_Leaf_Orange</i>	11.248
2.	<i>Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange</i>	12.800
3.	<i>Multiple_Diseases_Leaf_Orange</i>	4.800
4.	<i>Healthy_Leaf_Orange</i>	6.384
5.	<i>Young_Healthy_Leaf_Orange</i>	3.200
	Total	38.432

Setelah dataset diperoleh, dilakukan *splitting* menjadi data *training*, *validation*, dan *testing* dengan rasio 80:10:10. Tujuan pembagian ini agar model dapat dilatih menggunakan data *training*, divalidasi selama pelatihan menggunakan data *validation*, serta dievaluasi menggunakan data *testing* yang tidak pernah digunakan pada proses pelatihan.

Tabel 4.4 Pembagian *Training*, *Validation* dan *Testing*

No.	Kondisi/Penyakit	Jumlah Citra
1.	<i>Training (80%)</i>	30.745
2.	<i>Validation (10%)</i>	3.843
3.	<i>Testing (10%)</i>	3.844
	Total	38.432

4.3. Proses Penyusunan Sistem

Proses penyusunan sistem dimulai dengan pembangunan model MobileNetV2, konversi model ke format *TensorFlow Lite*, hingga pengembangan aplikasi *mobile* menggunakan *framework* yang sudah ditentukan

4.3.1. Persiapan Dataset

1. Koneksi ke Google Drive

Tahap pertama yang dilakukan yakni menghubungkan Google Drive ke Google Colab agar dataset dapat diakses selama proses pembangunan model.

```
[ ] # =====
# MOUNT GOOGLE DRIVE
# =====

from google.colab import drive

drive.mount('/content/drive')

Mounted at /content/drive
```

Gambar 4.2 Import Google Drive

2. Memuat Dataset

Setelah Google Drive terhubung, dataset dimuat dari folder penyimpanan.

```
[ ] # =====
# CEK FILE DATASET
# =====

import os

dataset_zip = "/content/drive/MyDrive/ENRICO OCTADARMA HULU/TUGAS AKHIR/DATASET DAUN JERUK/orange-leaf-disease-dataset.zip"

if os.path.exists(dataset_zip):
    print("✅ Dataset ditemukan")
    print("Lokasi:", dataset_zip)
    print("Ukuran:", round(os.path.getsize(dataset_zip)/(1024*1024),2), "MB")
else:
    print("❌ Dataset tidak ditemukan")

✅ Dataset ditemukan
Lokasi: /content/drive/MyDrive/ENRICO OCTADARMA HULU/TUGAS AKHIR/DATASET DAUN JERUK/orange-leaf-disease-dataset.zip
Ukuran: 252.39 MB
```

Gambar 4.3 Memuat Dataset

3. Analisis Dataset

Analisis mengetahui karakteristik kelas mulai dari jumlah kelas, jumlah data per kelas, ukuran gambar dan contoh visualisasi sampel. Karena semua gambar sudah 256×256 dan MobileNetV2 membutuhkan 224×224, kita tidak perlu *resize* permanen dataset. *Resize* dilakukan otomatis pada saat loading data.

```
[ ] # =====
# JUMLAH GAMBAR PER KELAS
# =====

import os
import pandas as pd

dataset_path = "/content/orange_leaf_dataset/DATASET DAUN JERUK/train/train"
class_counts = []

for class_name in sorted(os.listdir(dataset_path)):
    class_path = os.path.join(dataset_path, class_name)

    if os.path.isdir(class_path):
        total_images = len(os.listdir(class_path))

        class_counts.append({
            "Class": class_name,
            "Total Images": total_images
        })

df = pd.DataFrame(class_counts)

print(df)
```

	Class	Total Images
0	Citrus_Canker_Diseases_Leaf_Orange	11248
1	Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange	12800
2	Healthy_Leaf_Orange	6384
3	Multiple_Diseases_Leaf_Orange	4800
4	Young_Healthy_Leaf_Orange	3200

Gambar 4.4 Mengecek Jumlah Gambar Setiap Kelas

```
[ ] # =====
# CEK UKURAN GAMBAR
# =====

from PIL import Image

for class_name in classes:

    class_path = os.path.join(dataset_path, class_name)

    image_name = random.choice(os.listdir(class_path))

    image_path = os.path.join(class_path, image_name)

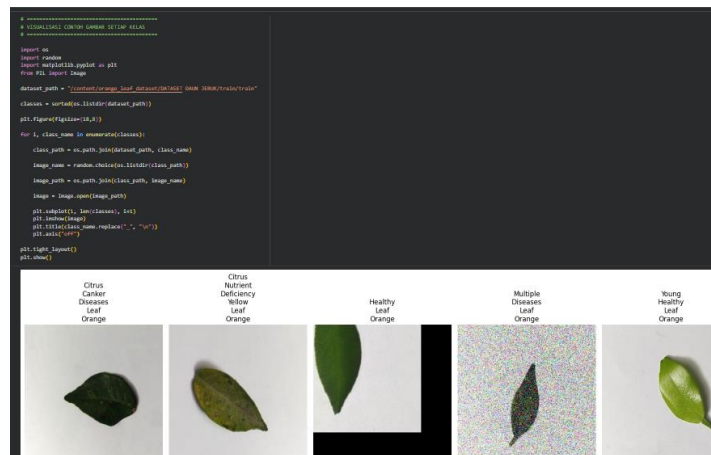
    image = Image.open(image_path)

    print(f"{class_name}")
    print(f"Ukuran : {image.size}")
    print("-"*50)
```

```

Citrus_Canker_Diseases_Leaf_Orange
Ukuran : (256, 256)
-----
Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange
Ukuran : (256, 256)
-----
Healthy_Leaf_Orange
Ukuran : (256, 256)
-----
Multiple_Diseases_Leaf_Orange
Ukuran : (256, 256)
-----
Young_Healthy_Leaf_Orange
Ukuran : (256, 256)
-----
```

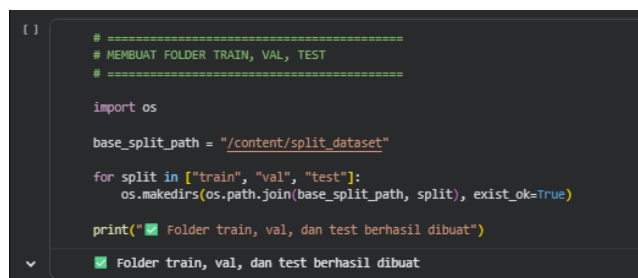
Gambar 4.5 Mengecek Ukuran Gambar Awal



Gambar 4.6 Contoh Visualisasi Setiap Kelas

4. Pembagian Dataset

Dataset dibagi menjadi data *training*, *validation*, dan *testing* dengan rasio 80:10:10 menggunakan metode *train-test split*. Ini bertujuan agar model dapat dilatih, divalidasi, dan diuji menggunakan data yang berbeda.



Gambar 4.7 Pembuatan Folder *Training*, *Validation* dan *Testing*

```

# SPLIT DATASET 80% TRAIN
# 10% VALIDATION
# 10% TEST
# =====

import os
import shutil
import random
from sklearn.model_selection import train_test_split

source_path = "/content/orange_leaf_dataset/DATASET DAUN JERUK/train/train"
split_path = "/content/split_dataset"

random.seed(42)

classes = sorted(os.listdir(source_path))

for class_name in classes:

    class_dir = os.path.join(source_path, class_name)

    images = os.listdir(class_dir)

    # Train 80%
    train_imgs, temp_imgs = train_test_split(
        images,
        test_size=0.20,
        random_state=42
    )

    # Val 10% dan Test 10%
    val_imgs, test_imgs = train_test_split(
        temp_imgs,
        test_size=0.50,
        random_state=42
    )

    # Copy Train
    for img in train_imgs:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(split_path, "train", class_name, img)
        )

    # Copy Validation
    for img in val_imgs:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(split_path, "val", class_name, img)
        )

    # Copy Test
    for img in test_imgs:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(split_path, "test", class_name, img)
        )

    print(f"✅ (class_name) selesai")

✅ Citrus_Canker_Diseases_Leaf_Orange selesai
✅ Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange selesai
✅ Healthy_Leaf_Orange selesai
✅ Multiple_Diseases_Leaf_Orange selesai
✅ Young_Healthy_Leaf_Orange selesai

```

Gambar 4.8 *Split Dataset*

```

[] # =====
# CEK JUMLAH DATA HASIL SPLIT
# =====

for split in ["train", "val", "test"]:

    print(f"\n===== {split.upper()} =====")

    total = 0

    split_dir = os.path.join("/content/split_dataset", split)

    for class_name in sorted(os.listdir(split_dir)):

        class_path = os.path.join(split_dir, class_name)

        count = len(os.listdir(class_path))

        total += count

        print(f"{class_name}: {count}")

    print(f"Total {split}: {total}")

===== TRAIN =====
Citrus_Canker_Diseases_Leaf_Orange: 8998
Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange: 10240
Healthy_Leaf_Orange: 5107
Multiple_Diseases_Leaf_Orange: 3840
Young_Healthy_Leaf_Orange: 2560
Total train: 30745

===== VAL =====
Citrus_Canker_Diseases_Leaf_Orange: 1125
Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange: 1280
Healthy_Leaf_Orange: 639
Multiple_Diseases_Leaf_Orange: 480
Young_Healthy_Leaf_Orange: 320
Total val: 3843

===== TEST =====
Citrus_Canker_Diseases_Leaf_Orange: 1125
Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange: 1280
Healthy_Leaf_Orange: 639
Multiple_Diseases_Leaf_Orange: 480
Young_Healthy_Leaf_Orange: 320
Total test: 3844

```

Gambar 4.9 Mengecek Jumlah Data Hasil *Split*

5. Pra-pemrosesan Dataset

Pra-pemrosesan menggunakan *Image Data Generator* dan *fungsi preprocess_input* dari MobileNetV2. Seluruh citra diproses dengan ukuran 224×224 piksel sehingga sesuai dengan kebutuhan arsitektur MobileNetV2 sebelum memasuki proses pelatihan.

```
[ ]  
# =====  
# DATA GENERATOR  
# =====  
  
IMG_SIZE = (224, 224)  
BATCH_SIZE = 32  
  
train_datagen = ImageDataGenerator(  
    preprocessing_function=preprocess_input  
)  
  
val_datagen = ImageDataGenerator(  
    preprocessing_function=preprocess_input  
)  
  
test_datagen = ImageDataGenerator(  
    preprocessing_function=preprocess_input  
)
```

Gambar 4.10 Pra-pemrosesan Menggunakan *Data Generator*

```
[ ]  
print("\nJumlah Kelas :", len(train_generator.class_indices))  
print("\nLabel Kelas:")  
for k, v in train_generator.class_indices.items():  
    print(f"{v} -> {k}")  
  
Jumlah Kelas : 5  
  
Label Kelas:  
0 -> Citrus_Canker_Diseases_Leaf_Orange  
1 -> Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange  
2 -> Healthy_Leaf_Orange  
3 -> Multiple_Diseases_Leaf_Orange  
4 -> Young_Healthy_Leaf_Orange
```

Gambar 4.11 Hasil *Data Generator*

4.3.2. Pembangunan Model CNN

1. Import Library

Diawali dengan mengimpor *library* yang dibutuhkan pada Google Colab. Beberapa *library* pendukung lainnya digunakan untuk proses dan pelatihan model.

```

() # IMPORT LIBRARY
# =====

# Operasi sistem dan file
import os
import shutil
import random

# Operasi numerik
import numpy as np
import pandas as pd

# Visualisasi
import matplotlib.pyplot as plt
import seaborn as sns

# Pengolahan gambar
from PIL import Image

# TensorFlow & Keras
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (
    Dense,
    GlobalAveragePooling2D,
    Dropout
)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import (
    EarlyStopping,
    ModelCheckpoint
)

# Evaluasi Model
from sklearn.metrics import (
    confusion_matrix,
    classification_report
)

# Menampilkan versi library
print("TensorFlow Version :", tf.__version__)
print("NumPy Version      :", np.__version__)
print("Pandas Version      :", pd.__version__)

TensorFlow Version : 2.20.0
NumPy Version      : 2.0.2
Pandas Version     : 2.2.2

```

Gambar 4.12 Import *Library*

2. Base Model MobileNetV2

Pembuatan *base* model menggunakan arsitektur MobileNetV2 dengan bobot awal (*pre-trained weights*) dari *ImageNet*. Lapisan dasar model dibekukan (*freeze*) sehingga hanya lapisan klasifikasi yang akan diperbarui selama proses pelatihan.

```

() # =====
# BASE MODEL MOBILENETV2
# =====

from tensorflow.keras.applications import MobileNetV2

base_model = MobileNetV2(
    weights="imagenet",
    include_top=False,
    input_shape=(224, 224, 3)
)

# Freeze Layer MobileNetV2
base_model.trainable = False

print("MobileNetV2 loaded")

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
5486464/5486464 1s 0us/step
MobileNetV2 loaded

```

Gambar 4.13 Pembuatan *Base Model MobileNETV2*

Model dikonfigurasi menggunakan ukuran masukan $224 \times 224 \times 3$ tanpa lapisan klasifikasi bawaan (*include_top=False*) serta menggunakan bobot awal dari *ImageNet*.

3. Penambahan Classification Head

Setelah *base* model berhasil dibuat, ditambahkan *classification head*. Penambahan ini berfungsi mengklasifikasi citra daun jeruk ke dalam lima kelas berdasarkan hasil ekstraksi fitur dari MobileNetV2.

```
[ ] # =====  
# CLASSIFICATION HEAD  
# =====  
  
from tensorflow.keras.layers import (  
    GlobalAveragePooling2D,  
    Dense,  
    Dropout  
)  
from tensorflow.keras.models import Model  
  
x = base_model.output  
x = GlobalAveragePooling2D()(x)  
x = Dropout(0.3)(x)  
  
x = Dense(  
    128,  
    activation='relu'  
) (x)  
  
x = Dropout(0.3)(x)  
  
predictions = Dense(  
    5,  
    activation='softmax'  
) (x)  
  
model = Model(  
    inputs=base_model.input,  
    outputs=predictions  
)  
  
print("✅ Model berhasil dibuat")  
✅ Model berhasil dibuat
```

Gambar 4.14 Penambahan *Classification Head*

4. Kompilasi Model

Model kemudian dikompilasi dengan *optimizer Adam*, *loss function Categorical Crossentropy*, dan metrik *accuracy* sebelum memasuki tahap pelatihan.

```
[ ] # =====
# COMPILE MODEL
# =====

from tensorflow.keras.optimizers import Adam

model.compile(
    optimizer=Adam(
        learning_rate=0.0001
    ),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

print("✅ Model berhasil di-compile")

✅ Model berhasil di-compile
```

Gambar 4.15 Kompilasi Model

5. Konfigurasi Callback

Konfigurasi ini menghentikan pelatihan ketika performa model tidak lagi meningkat dan *ModelCheckpoint* berfungsi menyimpan model dengan performa terbaik.

```
[ ] # =====
# CALLBACK
# =====

from tensorflow.keras.callbacks import (
    EarlyStopping,
    ModelCheckpoint
)

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True
)

checkpoint = ModelCheckpoint(
    'best_mobilenetv2.h5',
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)

print("✅ Callback siap")

✅ Callback siap
```

Gambar 4.16 Konfigurasi Callback

4.3.3. Pelatihan dan Evaluasi Model

1. Pelatihan Model

Dilakukan menggunakan fungsi *model.fit()* dengan parameter yang telah ditentukan, yakni jumlah *epoch*, *batch size*, serta *callback* yang disiapkan.

```
[ ]  
# =====  
# TRAINING MODEL  
# =====  
  
EPOCHS = 10  
  
history = model.fit(  
    train_generator,  
    validation_data=val_generator,  
    epochs=EPOCHS,  
    callbacks=[  
        early_stop,  
        checkpoint  
    ]  
)
```

Gambar 4.17 Pelatihan Model

2. Evaluasi Model

Setelah pelatihan selesai, model dievaluasi menggunakan data testing melalui *funcmodel.evaluate()* untuk pengukuran kinerja sebelum diimplementasikan ke *mobile*.

```
[ ]  
# =====  
# EVALUASI TEST SET  
# =====  
  
test_loss, test_accuracy = model.evaluate(  
    test_generator,  
    verbose=1  
)  
  
print(f"\nTest Accuracy : {test_accuracy:.4f}")  
print(f"Test Loss      : {test_loss:.4f}")
```

Gambar 4.18 Evaluasi Model

4.3.4. Konversi Model

1. Simpan Model

Model dengan performa terbaik terlebih dahulu disimpan dalam format Keras, bertujuan agar dapat digunakan kembali tanpa pelatihan ulang.

```
[ ]  
# =====  
# SAVE MODEL (.h5)  
# =====  
  
model.save("orange_tesf_mobilenetv2.h5")  
print("Model berhasil disimpan")  
WARNING:absl:You are saving your model as an HDF5 file via "model.save()" or "keras.saving.save_model(model)". This file format is considered legacy. We recommend using  
Model berhasil disimpan
```

Gambar 4.19 Simpan Model

```
[ ] # =====
# CONVERT TO TFLITE
# =====

import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_keras_model(model)

tflite_model = converter.convert()

with open(
    "orange_leaf_mobilenetv2.tflite",
    "wb"
) as f:
    f.write(tflite_model)

print("✅ TFLite berhasil dibuat")

▼ Saved artifact at '/tmp/tmp4ddbgq6'. The following endpoints are available:

* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 224, 224, 3), dtype=tf.float32, name=
Output Type:
  TensorSpec(shape=(None, 5), dtype=tf.float32, name=None)
Captures:
  135479681885264: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135479681885840: TensorSpec(shape=(), dtype=tf.resource, name=None)
  135479681888528: TensorSpec(shape=(), dtype=tf.resource, name=None)
```

Gambar 4.20 Konversi Model ke TFLite

2. Konversi ke TensorFlow Lite

Model ini digunakan sebagai model inferensi pada aplikasi *mobile*.

```
[ ] labels = [
    "Citrus_Canker_Diseases_Leaf_Orange",
    "Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange",
    "Healthy_Leaf_Orange",
    "Multiple_Diseases_Leaf_Orange",
    "Young_Healthy_Leaf_Orange"
]

with open("labels.txt", "w") as f:
    for label in labels:
        f.write(label + "\n")

print("✅ labels.txt berhasil dibuat")

▼ ✅ labels.txt berhasil dibuat
```

Gambar 4.21 Penyimpanan Label Kelas

3. Penyimpanan Label Kelas

Selain model dikonversi, dibuat juga *labels.txt* yang berisi lima kelas penyakit daun jeruk. Dua Berkas ini digunakan oleh aplikasi untuk menampilkan nama kelas sesuai dengan hasil prediksi model.

4.3.5. Implementasi Model Pada Aplikasi

Model dan label yang sudah disimpan kemudian dipindahkan kedalam direktori proyek flutter, yakni pada folder *assets/model*, kemudian untuk memproses file tersebut dibuat

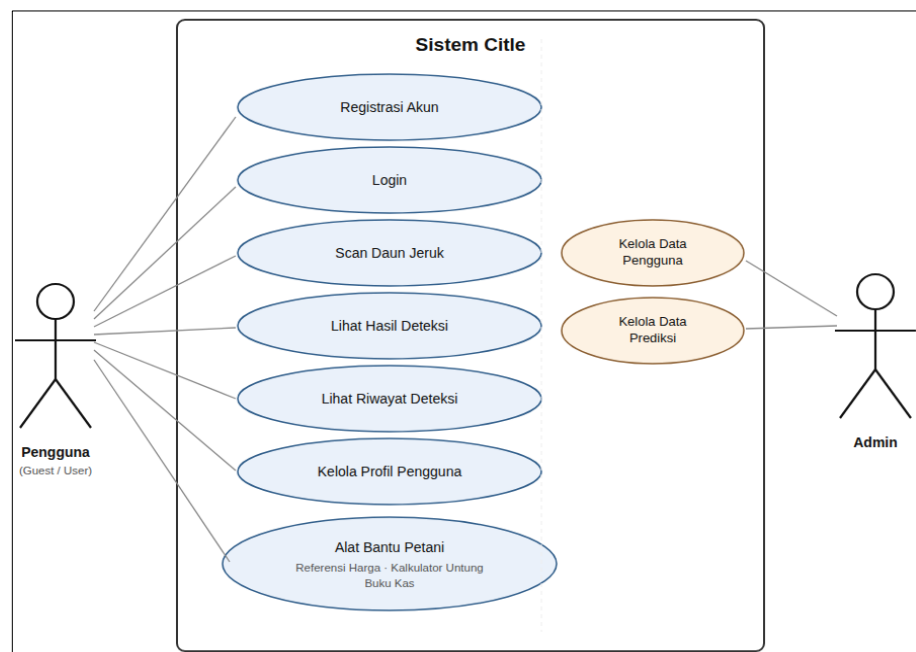
file *tflite_services.dart*.

Potongan kode yang tertera pada lampiran 1 diimplementasi dalam proses klasifikasi citra menggunakan model. Kode memproses dengan memuat model dan label, kemudian membaca citra yang dipilih pengguna. Selanjutnya citra diubah ukurannya menjadi 224×224 piksel dan dilakukan normalisasi nilai piksel ke rentang -1 hingga 1 sesuai kebutuhan arsitektur MobileNetV2. Data hasil prapemrosesan kemudian diproses oleh model melalui fungsi *run()* untuk melakukan inferensi. Hasil inferensi berupa label kelas dengan nilai probabilitas (*confidence*) tertinggi serta skor probabilitas seluruh kelas dikembalikan dalam bentuk *Map<String, dynamic>* untuk ditampilkan pada halaman hasil.

4.4. Deskripsi Sistem

4.4.1. Perancangan Use Case Diagram

Use case diagram dapat dilihat pada gambar dibawah ini.



Gambar 4.22 Perancangan Use Case Diagram

Pada gambar diatas menggambarkan interaksi antara aktor dengan sistem aplikasi Citle. Sistem melibatkan dua aktor, yaitu Pengguna (*Guest/User*) dan *Admin*.

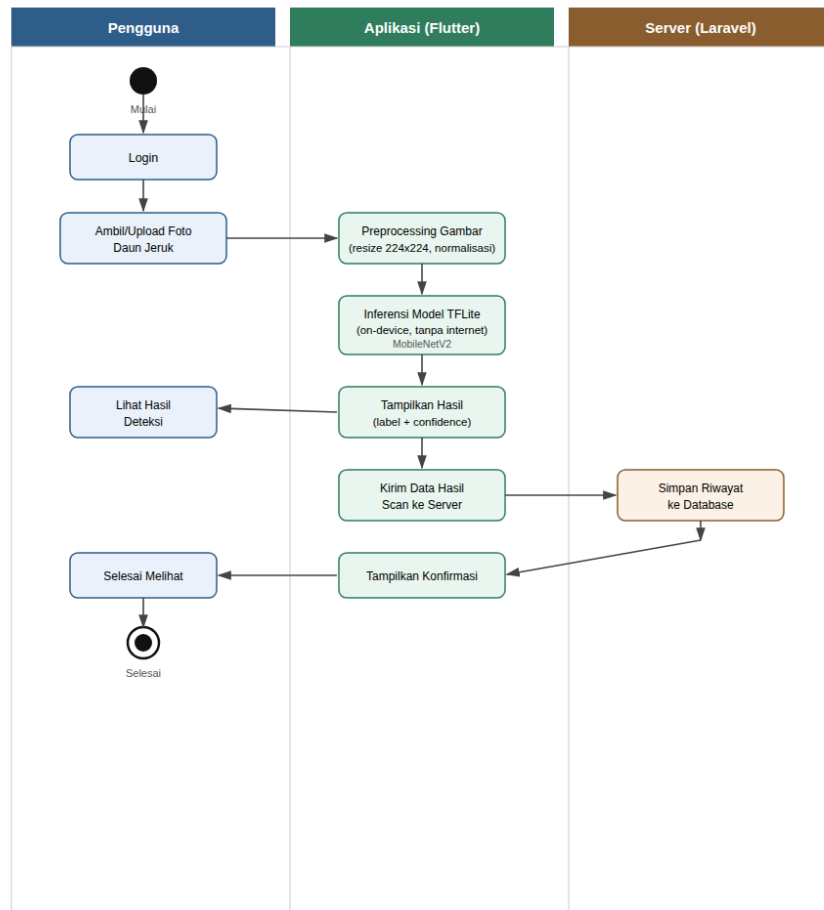
- Pengguna dapat melakukan registrasi dan *login*, melakukan scan daun jeruk untuk deteksi penyakit, melihat hasil deteksi, melihat riwayat deteksi,

mengelola profil, serta mengakses fitur Alat Bantu Petani yang meliputi referensi harga, kalkulator keuntungan panen, dan buku kas.

- Admin memiliki hak akses untuk mengelola data pengguna dan data hasil prediksi yang tersimpan dalam sistem.

4.4.2. Perancangan Activity Diagram

Activity diagram dapat dilihat pada gambar dibawah ini.



Gambar 4.23 Perancangan *Activity Diagram*

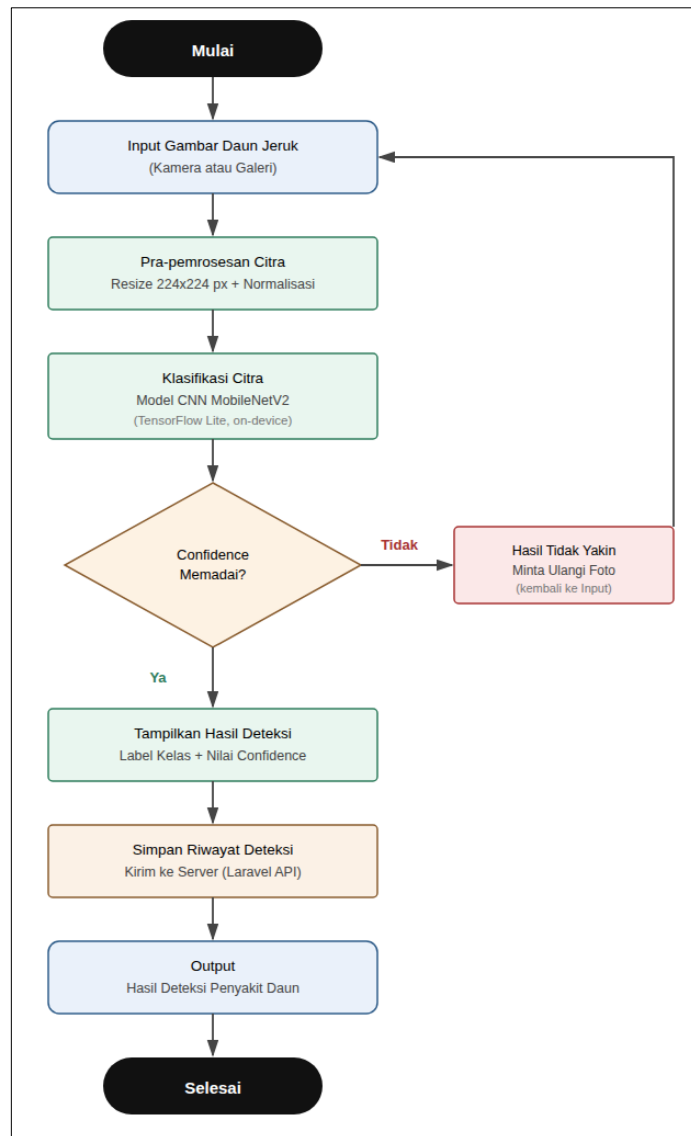
Pada gambar diatas melibatkan tiga entitas yakni Pengguna, Aplikasi (Flutter), dan Server (Laravel). Proses dimulai dari Pengguna melakukan login, mengambil atau mengunggah foto daun jeruk, kemudian aplikasi melakukan pra-pemrosesan citra dan inferensi model secara *on-device* menggunakan model tanpa memerlukan koneksi internet, hingga hasil klasifikasi ditampilkan kepada Pengguna.

Proses penyimpanan akan berjalan dengan mensyaratkan pengguna dalam status *login* (memiliki akun)

dan terhubung ke internet. Apabila kondisi tersebut tidak terpenuhi, hasil deteksi tetap dapat ditampilkan kepada Pengguna, namun tidak tersimpan sebagai riwayat pada *server*.

4.4.3. Perancangan Diagram Alur Sistem (Flowchart)

Diagram alur sistem dapat dilihat pada gambar dibawah ini.



Gambar 4.24 Perancangan Diagram Alur Sistem

Flowchart pada gambar merupakan alur sistem, dimulai dari input gambar melalui kamera atau galeri, pra-pemrosesan citra (*resize* dan *normalisasi*), hingga klasifikasi menggunakan model. Sistem kemudian memeriksa nilai *confidence* hasil klasifikasi. Hasil ditampilkan dan disimpan sebagai riwayat ke *server*, Proses berakhir setelah hasil deteksi ditampilkan

4.5. Perancangan Basis Data

Basis data dikelola oleh backend *Laravel* melalui *ORM Eloquent*, Relasi antar tabel digambarkan melalui *Entity Relationship Diagram* (ERD) berikut.



Gambar 4.25 Relasi Antar Table Menggunakan *Entity Relationship Diagram*

Relasi antar tabel menggunakan konsep *one-to-many*, di mana satu pengguna dapat memiliki banyak data pada tabel *scan_histories*, *profit_histories*, *cashbook_entries*, dan *price_references*, sedangkan setiap data hanya dimiliki oleh satu pengguna. Sementara itu, tabel *app_settings* bersifat global sehingga tidak memiliki relasi dengan tabel *users*.

4.5.1. Struktur Tabel Pengguna

Tabel *users* menyimpan data akun pengguna, terdiri atas kolom *id* sebagai *primary key*, *name* sebagai identitas *login*, *phone*, *password* yang disimpan dalam bentuk ter-enkripsi

(hashed), *photo* untuk foto profil, serta *role* bertipe enum (*user*, *admin*, *default user*) yang menentukan hak akses *user* dapat melakukan registrasi, *login*, scan deteksi penyakit daun jeruk, melihat riwayat deteksi, mengelola profil, dan menggunakan fitur Alat Bantu Petani, sedangkan *admin* memiliki hak akses tambahan untuk mengelola data pengguna dan data hasil prediksi. Dilengkapi juga kolom *created_at* dan *updated_at* sebagai penanda waktu.

#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default	Comment	Collation
1	id	BIGINT	20	☒	☐	☐	AUTO_INCREMENT		
2	name	VARCHAR	255	☐	☐	☐	No default		utf8mb4_unicode_ci
3	phone	VARCHAR	255	☐	☒	☐	NULL		utf8mb4_unicode_ci
4	password	VARCHAR	255	☐	☒	☐	No default		utf8mb4_unicode_ci
5	photo	VARCHAR	255	☐	☒	☐	NULL		utf8mb4_unicode_ci
6	role	ENUM	'user','admin'	☐	☐	☐	'user'		utf8mb4_unicode_ci
7	remember_tok...	VARCHAR	100	☐	☒	☐	NULL		utf8mb4_unicode_ci
8	created_at	TIMESTAMP		☐	☒	☐	NULL		
9	updated_at	TIMESTAMP		☐	☒	☐	NULL		

Gambar 4.26 Struktur Tabel Pengguna

4.5.2. Struktur Table Token Akses Pengguna

Tabel *personal_access_tokens* menyimpan token autentikasi API yang digunakan *Laravel Sanctum* untuk memverifikasi pengguna yang sedang *login*, terdiri atas kolom *id* sebagai *primary key*, *tokenable_id* sebagai *foreign key* yang merujuk ke tabel *users*, token yang menyimpan token dalam bentuk enkripsi, *abilities* yang menentukan hak akses token, *last_used_at* untuk mencatat waktu penggunaan terakhir, *expires_at* sebagai batas masa berlaku token, serta *created_at* dan *updated_at* sebagai penanda waktu.

#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default	Comment	Collation
1	id	BIGINT	20	☒	☐	☐	AUTO_INCREMENT		
2	tokenable_type	VARCHAR	255	☐	☐	☐	No default		utf8mb4_unicode_ci
3	tokenable_id	BIGINT	20	☒	☐	☐	No default		utf8mb4_unicode_ci
4	name	TEXT		☐	☐	☐	No default		utf8mb4_unicode_ci
5	token	VARCHAR	64	☐	☐	☐	No default		utf8mb4_unicode_ci
6	abilities	TEXT		☐	☒	☐	NULL		utf8mb4_unicode_ci
7	last_used_at	TIMESTAMP		☐	☒	☐	NULL		
8	expires_at	TIMESTAMP		☐	☒	☐	NULL		
9	created_at	TIMESTAMP		☐	☒	☐	NULL		
10	updated_at	TIMESTAMP		☐	☒	☐	NULL		

Gambar 4.27 Struktur Table Token Akses Pengguna

4.5.3. Struktur Table Riwayat Deteksi

Tabel *scan_histories* menyimpan riwayat hasil deteksi penyakit daun jeruk yang dilakukan pengguna, terdiri atas kolom *id* sebagai *primary key*, *user_id* sebagai *foreign key*

yang merujuk ke tabel *users*, *image* untuk menyimpan *path* gambar daun yang di-scan, label sebagai hasil klasifikasi model, *confidence* yang menyatakan tingkat keyakinan prediksi, serta *scores* yang menyimpan probabilitas seluruh kelas dalam format JSON, dilengkapi kolom *created_at* dan *updated_at* sebagai penanda waktu.

#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default	Comment	Collation
1	id	BIGINT	20	☒	☐	☐	AUTO_INCREMENT		
2	user_id	BIGINT	20	☒	☐	☐	No default		
3	image_path	VARCHAR	255	☐	☐	☐	No default		utf8mb4_unicode_ci
4	label	VARCHAR	255	☐	☐	☐	No default		utf8mb4_unicode_ci
5	confidence	DECIMAL	5,4	☐	☐	☐	No default		
6	scores	LONGTEXT		☐	☒	☐	NULL		utf8mb4_bin
7	created_at	TIMESTAMP		☐	☒	☐	NULL		
8	updated_at	TIMESTAMP		☐	☒	☐	NULL		

Gambar 4.28 Table Riwayat Deteksi

4.5.4. Struktur Table Riwayat Keuntungan

Tabel *profit_histories* menyimpan riwayat perhitungan keuntungan panen pengguna, terdiri atas kolom *id* sebagai *primary key*, *user_id* sebagai *foreign key* yang merujuk ke tabel *users*, *harvest_kg* sebagai jumlah hasil panen dalam kilogram, *price_per_kg* sebagai harga jual per kilogram, serta *cost* sebagai biaya produksi yang dikeluarkan, dilengkapi kolom *created_at* dan *updated_at* sebagai penanda waktu.

#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default	Comment	Collation
1	id	BIGINT	20	☒	☐	☐	AUTO_INCREMENT		
2	user_id	BIGINT	20	☒	☐	☐	No default		
3	harvest_kg	DECIMAL	10,2	☐	☐	☐	No default		
4	price_per_kg	DECIMAL	12,2	☐	☐	☐	No default		
5	cost	DECIMAL	12,2	☐	☐	☐	No default		
6	revenue	DECIMAL	14,2	☐	☐	☐	No default		
7	profit	DECIMAL	14,2	☐	☐	☐	No default		
8	margin	DECIMAL	12,2	☐	☐	☐	No default		
9	created_at	TIMESTAMP		☐	☒	☐	NULL		
10	updated_at	TIMESTAMP		☐	☒	☐	NULL		

Gambar 4.29 Table Riwayat Keuntungan

4.5.5. Struktur Table Buku Kas

Tabel *cashbook_entries* menyimpan catatan pemasukan dan pengeluaran usaha tani pengguna, terdiri atas kolom *id* sebagai *primary key*, *user_id* sebagai *foreign key* yang merujuk ke tabel *users*, *type* bertipe *enum* (masuk, keluar)

yang menandai jenis transaksi, *amount* sebagai nominal transaksi, serta *note* sebagai catatan atau keterangan tambahan yang bersifat opsional, dilengkapi kolom *created_at* dan *updated_at* sebagai penanda waktu.

#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default	Comment	Collation
1	id	BIGINT	20	☒	☐	☐	AUTO INCREME...		
2	user_id	BIGINT	20	☒	☐	☐	No default		
3	type	ENUM	'masuk','kelu...	☐	☐	☐	No default		utf8mb4_unicode_ci
4	amount	DECIMAL	14,2	☐	☐	☐	No default		
5	note	VARCHAR	255	☐	☒	☐	NULL		utf8mb4_unicode_ci
6	created_at	TIMESTAMP		☐	☒	☐	NULL		
7	updated_at	TIMESTAMP		☐	☒	☐	NULL		

Gambar 4.30 Struktur Table Buku Kas

4.5.6. Struktur Table Referensi Harga

Tabel *price_references* menyimpan referensi harga jual jeruk yang dicatat pengguna, terdiri atas kolom *id* sebagai *primary key*, *user_id* sebagai *foreign key* yang merujuk ke tabel *users*, serta *price* sebagai nominal harga yang dicatat, dilengkapi kolom *created_at* dan *updated_at* sebagai penanda waktu.

#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default
1	id	BIGINT	20	☒	☐	☐	AUTO INCREME...
2	user_id	BIGINT	20	☒	☐	☐	No default
3	price	DECIMAL	12,2	☐	☐	☐	No default
4	created_at	TIMESTAMP		☐	☒	☐	NULL
5	updated_at	TIMESTAMP		☐	☒	☐	NULL

Gambar 4.31 Struktur Table Referensi Harga

4.5.7. Struktur Table Pengaturan

Tabel *app_settings* menyimpan pengaturan dan informasi umum aplikasi yang ditampilkan pada halaman Tentang Aplikasi, terdiri atas kolom *id* sebagai *primary key*, *app_name* dan *app_version* sebagai identitas aplikasi, *description* sebagai deskripsi aplikasi, serta *contact_email* dan *contact_phone* sebagai informasi kontak pengembang, dilengkapi kolom *created_at* dan *updated_at* sebagai penanda waktu. Tabel ini tidak berelasi ke tabel *users* karena datanya bersifat global dan dapat diakses oleh seluruh pengguna tanpa memerlukan proses autentikasi.

Citle Production DB Database: citle_jaravel Table: app_settings Data Query									
Columns: Add Remove Up Down									
#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zero fill	Default	Comment	Collation
1	id	BIGINT	20	☒	☐	☐	AUTO_INCREME...		
2	app_name	VARCHAR	100	☐	☐	☐	No default		utf8mb4_unicode_ci
3	app_version	VARCHAR	20	☐	☐	☐	'1.0.0'		utf8mb4_unicode_ci
4	admin_contact	VARCHAR	100	☐	☒	☐	NULL		utf8mb4_unicode_ci
5	about	TEXT		☐	☒	☐	NULL		utf8mb4_unicode_ci
6	created_at	TIMESTAMP		☐	☒	☐	NULL		
7	updated_at	TIMESTAMP		☐	☒	☐	NULL		

Gambar 4.32 Struktur Table Pengaturan

4.6. Implementasi Antarmuka Aplikasi

4.6.1. Halaman User Mode

Halaman *User Mode* merupakan tampilan awal yang saat aplikasi pertama kali dibuka yang mengarahkan pengguna ke salah satu dari tiga jalur akses aplikasi. Halaman ini menampilkan logo dan nama aplikasi, disertai tiga pilihan tombol, yaitu Masuk/Daftar untuk pengguna yang memiliki atau ingin membuat akun guna mengakses seluruh fitur aplikasi, Lanjutkan sebagai Tamu untuk pengguna yang hanya ingin menggunakan fitur scan deteksi penyakit tanpa perlu melakukan registrasi, serta Masuk sebagai Admin untuk pengguna dengan hak akses administratif.



Gambar 4.33 Halaman User Mode

4.6.2. Halaman Login

1. Login Pengguna

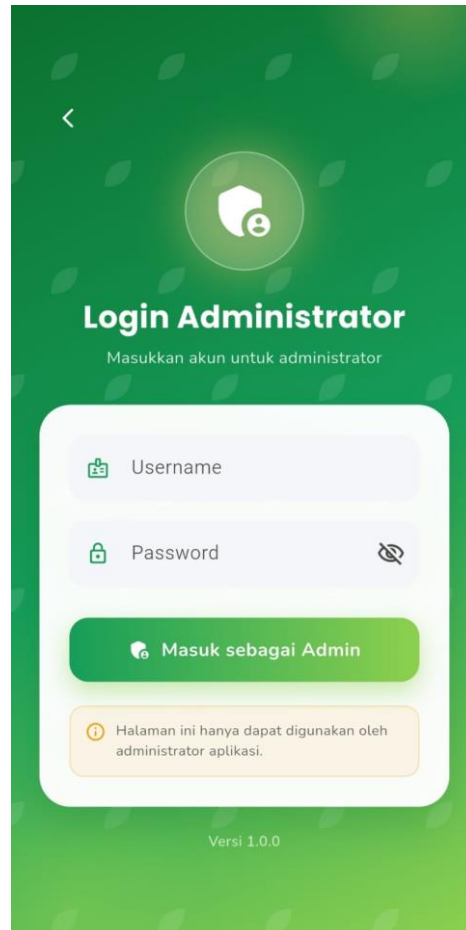
Halaman *Login* Pengguna digunakan untuk proses autentikasi bagi pengguna yang telah memiliki akun, dengan memasukkan username dan password yang telah didaftarkan sebelumnya. Halaman ini menampilkan kolom *input username* dan *input password*, tombol Masuk untuk memproses autentikasi, serta tautan menuju halaman Registrasi bagi pengguna yang belum memiliki akun. Setelah kredensial berhasil diverifikasi oleh sistem, pengguna diarahkan menuju halaman *Dashboard* untuk mengakses seluruh fitur aplikasi.



Gambar 4.34 Halaman Login Pengguna

2. Login Admin

Halaman Login Admin digunakan untuk proses autentikasi khusus bagi pengguna dengan hak akses administrator, dengan memasukkan username dan password yang terdaftar dengan role admin. Halaman ini menampilkan kolom input username dan password, tombol Masuk sebagai Admin, serta keterangan yang menegaskan bahwa halaman ini hanya dapat digunakan oleh administrator aplikasi. Setelah kredensial diverifikasi, admin diarahkan ke halaman Dashboard Admin untuk mengelola data pengguna dan data hasil prediksi.



Gambar 4.35 Halaman *Login Admin*

4.6.3. Halaman Registrasi

Halaman Registrasi digunakan untuk membuat akun pengguna baru, dengan mengisi *username*, nomor HP, dan *password*. Halaman ini menyertakan keterangan bahwa *username* harus unik dan cukup panjang agar tidak bentrok dengan pengguna lain, serta tautan menuju halaman *login* bagi pengguna yang telah memiliki akun. Setelah data berhasil disimpan, sistem akan mengarahkan pengguna untuk masuk ke aplikasi menggunakan akun yang baru dibuat.

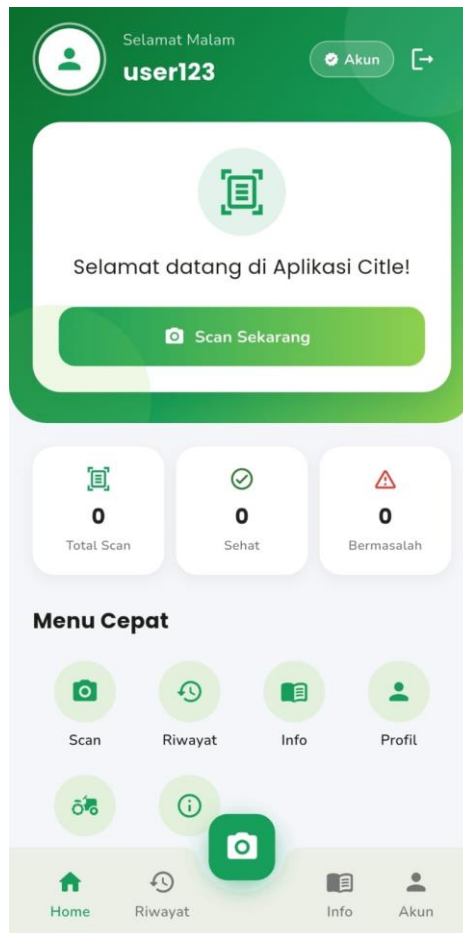


Gambar 4.36 Halaman Registrasi

4.6.4. Halaman Dashboard

1. Dashboard Pengguna

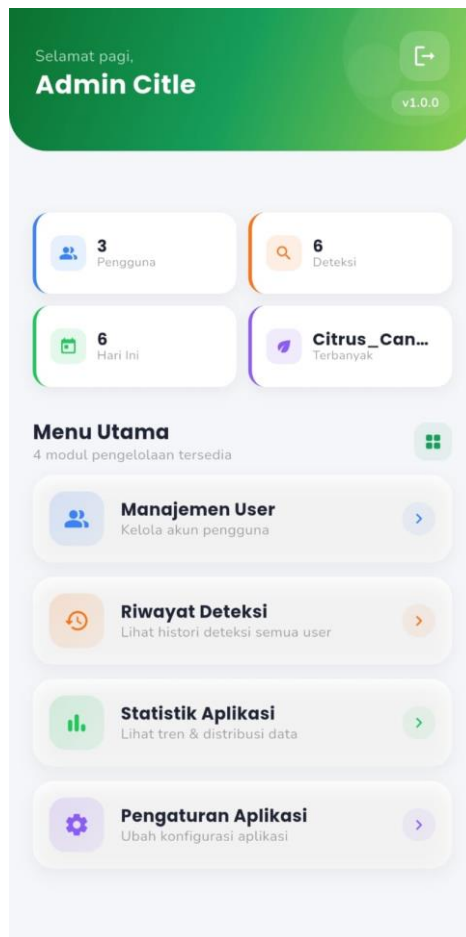
Halaman Dashboard Pengguna merupakan halaman utama yang ditampilkan setelah pengguna berhasil masuk ke aplikasi, baik sebagai pengguna terdaftar maupun sebagai tamu (*guest*). Halaman ini menampilkan sapaan beserta nama pengguna, tombol Scan Sekarang untuk memulai proses deteksi penyakit daun jeruk, ringkasan statistik berupa Total Scan, jumlah daun Sehat, dan jumlah daun Bermasalah, serta Menu Cepat yang menyediakan akses langsung ke fitur Scan, Riwayat, Info, Profil, dan Alat Bantu Petani. Navigasi utama aplikasi juga tersedia melalui *bottom navigation bar* berupa Home, Riwayat, Info, dan Akun.



Gambar 4.37 Halaman *Dashboard* Pengguna

2. Dashboard Admin

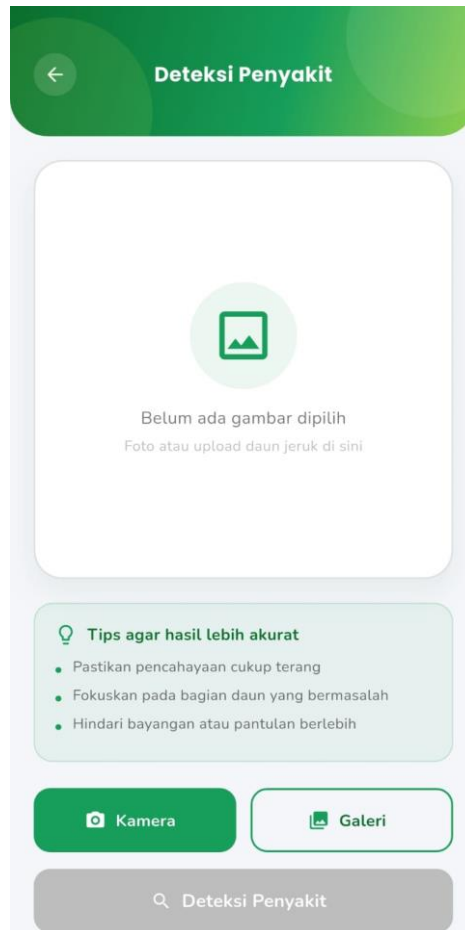
Halaman *Dashboard Admin* merupakan halaman utama yang ditampilkan setelah admin berhasil masuk ke aplikasi, menampilkan ringkasan statistik sistem berupa jumlah Pengguna, jumlah Deteksi, jumlah deteksi Hari Ini, serta kelas penyakit dengan jumlah deteksi Terbanyak. Halaman ini juga menyediakan Menu Utama yang terdiri atas empat modul pengelolaan, yaitu Manajemen User untuk mengelola akun pengguna, Riwayat Deteksi untuk melihat histori deteksi seluruh pengguna, Statistik Aplikasi untuk melihat tren dan distribusi data, serta Pengaturan Aplikasi untuk mengubah konfigurasi aplikasi.



Gambar 4.38 Halaman *Dashboard Admin*

4.6.5. Halaman Scan Penyakit

Halaman Scan Penyakit digunakan untuk memilih citra daun jeruk yang akan dianalisis oleh sistem. Halaman ini menampilkan area pratinjau gambar yang masih kosong sebelum pengguna memilih foto, disertai tips pengambilan gambar agar hasil deteksi lebih akurat, seperti memastikan pencahayaan cukup terang, memfokuskan pada bagian daun yang bermasalah, dan menghindari bayangan atau pantulan berlebih. Pengguna dapat mengambil gambar melalui tombol Kamera atau memilih gambar melalui tombol Galeri, kemudian menekan tombol Deteksi Penyakit untuk memulai proses analisis.



Gambar 4.39 Halaman Scan Penyakit

4.6.6. Halaman Proses Analisis

Halaman Proses Analisis ditampilkan setelah pengguna memilih gambar daun jeruk dan menekan tombol Deteksi Penyakit. Pada tahap ini, aplikasi menjalankan proses inferensi model *TensorFlow Lite* secara *on-device* untuk menganalisis pola dan tekstur citra, disertai indikator progres berupa persentase yang menunjukkan bahwa proses klasifikasi sedang berlangsung.



Gambar 4.40 Halaman Proses Analisis

4.6.7. Halaman Hasil Deteksi

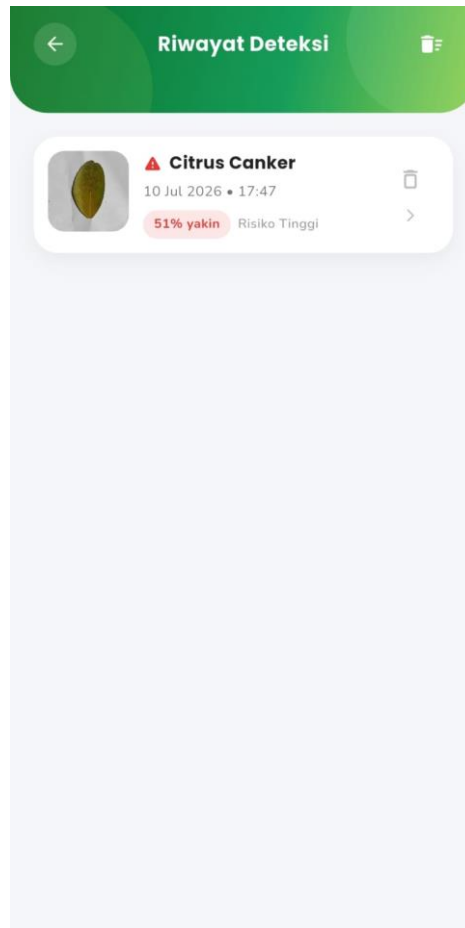
Halaman Hasil Deteksi menampilkan hasil klasifikasi penyakit daun jeruk yang diperoleh dari model. Informasi yang disajikan meliputi nama kelas hasil prediksi, tingkat keyakinan (*confidence*) dalam bentuk persentase, tingkat risiko kondisi daun, penjelasan mengenai penyakit yang terdeteksi, serta dampak terhadap pertumbuhan tanaman, sehingga pengguna dapat memahami hasil deteksi beserta implikasinya secara lebih menyeluruh.



Gambar 4.41 Halaman Hasil Deteksi

4.6.8. Halaman Riwayat Deteksi

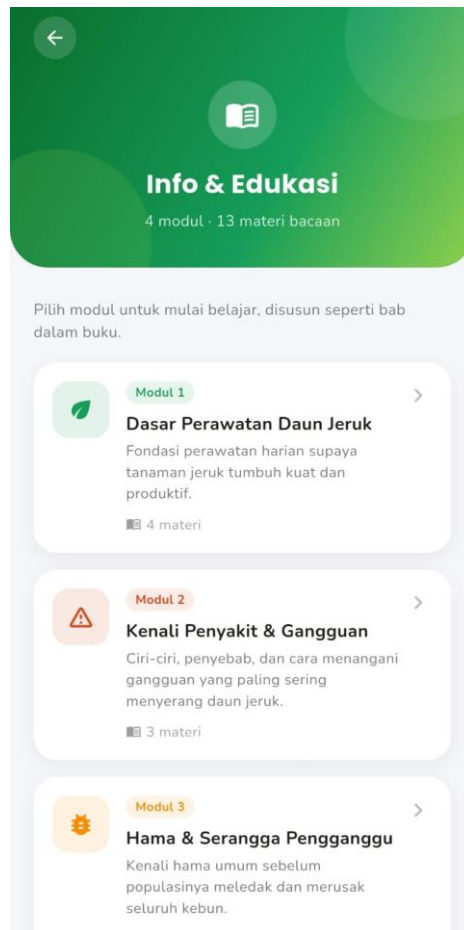
Halaman Riwayat menampilkan daftar hasil deteksi penyakit daun jeruk yang pernah dilakukan pengguna, meliputi gambar daun, nama kelas hasil klasifikasi, tanggal dan waktu deteksi, tingkat keyakinan (*confidence*), serta tingkat risiko. Pengguna dapat membuka detail hasil deteksi atau menghapus riwayat melalui ikon tempat sampah pada setiap data maupun secara keseluruhan.



Gambar 4.42 Halaman Riwayat Deteksi

4.6.9. Halaman Informasi dan Edukasi

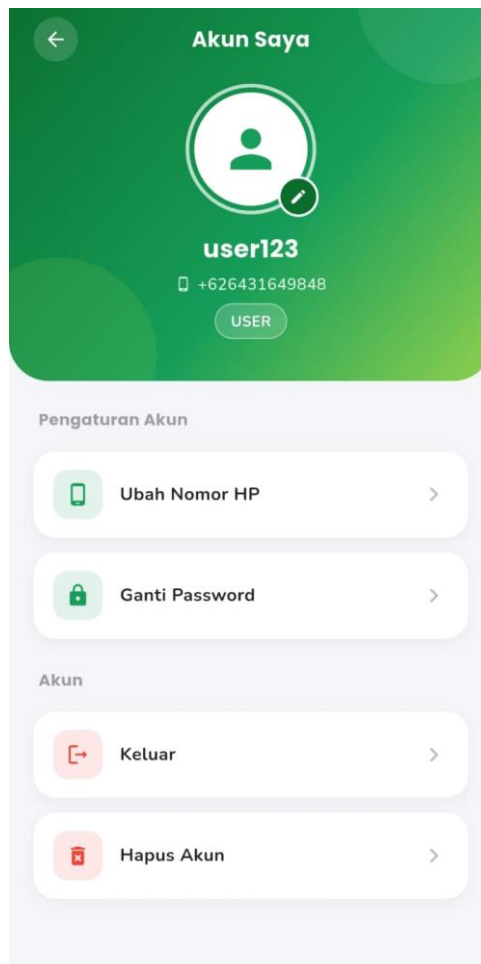
Halaman Informasi menyediakan materi edukasi seputar perawatan dan penyakit daun jeruk, disusun dalam beberapa modul menyerupai bab dalam buku, terdiri atas Dasar Perawatan Daun Jeruk, Kenali Penyakit dan Gangguan, Hama dan Serangga Pengganggu, serta modul lainnya, dengan total 13 materi bacaan.



Gambar 4.43 Halaman Informasi dan Edukasi

4.6.10. Halaman Profile

Halaman *Profile* menampilkan informasi akun pengguna, meliputi foto profil, *username*, nomor HP, dan status *role*. Halaman ini juga menyediakan menu pengaturan akun berupa ubah nomor HP, ganti *password*, keluar (logout), dan hapus akun.



Gambar 4.44 Halaman *Profile*

4.6.11. Halaman Alat Petani

Halaman Alat Bantu Petani menyediakan fitur pendukung bagi pengguna dalam mengelola budidaya dan usaha tani jeruk, terdiri atas empat fitur berikut:

1. Referensi harga jeruk yang menyimpan dan melihat riwayat harga jual buah jeruk sebagai acuan pengguna.
2. Kalkulator keuntungan panen yang menghitung estimasi keuntungan berdasarkan hasil panen, harga jual, dan biaya produksi.
3. Estimasi pupuk dan pestisida berfungsi memperkirakan kebutuhan pupuk dan pestisida berdasarkan jumlah tanaman jeruk.
4. Buku kas sederhana untuk mencatat pemasukan dan pengeluaran usaha tani secara sederhana.
- 5.



Gambar 4.45 Halaman Alat Petani

4.6.12. Halaman Tentang

Halaman Tentang Aplikasi menampilkan identitas dan deskripsi aplikasi *Citle* beserta versinya, serta informasi kontak pengembang melalui WhatsApp dan Instagram bagi pengguna yang memiliki pertanyaan atau kendala.



Gambar 4.46 Halaman Tentang

4.7. Spesifikasi Perangkat Keras dan Perangkat Lunak

Pengembangan dan pengujian sistem pada penelitian ini menggunakan perangkat keras dan perangkat lunak sebagai berikut

Tabel 4.5 Spesifikasi Perangkat Keras

No.	Perangkat Keras	Fungsi
1.	Laptop <i>Lenovo IdeaPad Slim 5</i>	Media pengembangan aplikasi dan pengolahan data
2.	<i>Smartphone Xiaomi Redmi Note 12 Pro</i>	Pengujian aplikasi pada sistem operasi Android
3.	<i>Smartphone iPhone 11</i>	Pengujian aplikasi pada sistem Operasi iOS

4.	<i>VPS Online</i>	Media hosting backend <i>Laravel</i> dan basisdata produksi
----	-------------------	---

Tabel 4.6 Spesifikasi Perangkat Lunak

No.	Perangkat Keras	Fungsi
1.	Google Colab	Melatih dan menguji model CNN
2.	Google Drive	Media penyimpanan
3.	<i>Python</i>	Bahasa pemrograman untuk membangun model CNN
4.	<i>TensorFlow dan Keras</i>	Library untuk merancang, melatih, dan mengevaluasi model
5.	Visual Studio Code	Editor kode
6.	<i>Flutter SDK</i>	Framework aplikasi mobile
7.	Android Studio	Emulator pengujian aplikasi
8.	<i>Laravel</i>	Framework backend dan API
9.	<i>MySQL</i>	Sistem manajemen basis data
10.	<i>HeidiSQL</i>	Monitoring basis data
11.	cPanel/WHM	Panel pengelolaan server dan deployment aplikasi
12.	<i>Extension Reqbin</i>	Pengujian endpoint API

BAB V. HASIL DAN PEMBAHASAN

5.1. Prosedur Pengujian

Prosedur pengujian dalam penelitian ini terbagi menjadi dua tahap: pengujian model CNN dan pengujian aplikasi mobile. Pengujian model berfokus pada evaluasi menggunakan data *testing* untuk mengukur kinerja klasifikasi secara keseluruhan, sementara pengujian aplikasi memakai metode *Black Box Testing* guna memastikan setiap fitur berfungsi baik.

5.1.1. Metode Pengujian Antarmuka Aplikasi

Pengujian aplikasi mobile memanfaatkan metode *Black Box Testing*, yaitu teknik pengujian yang menitikberatkan pada kesesuaian antara *input* dan *output* aplikasi tanpa perlu menelaah struktur maupun kode program di baliknya. Tujuannya adalah memastikan setiap fitur berjalan sesuai kebutuhan fungsional yang telah dirancang sebelumnya. Seluruh fitur diuji berdasarkan skenario penggunaan untuk mengetahui apakah sistem memberikan keluaran yang sesuai dengan hasil yang diharapkan (Khaira, Weni, and Wilia 2024).

5.1.2. Metode Pengujian Robustness

Pengujian *robustness* bertujuan untuk mengukur ketahanan model klasifikasi dalam mempertahankan konsistensi prediksi ketika dihadapkan pada variasi kondisi citra yang umum terjadi pada pengambilan gambar secara nyata di lapangan, seperti citra buram, pencahayaan tidak ideal, derau (*noise*), dan perbedaan jarak pengambilan gambar. Pengujian menggunakan lima kategori citra, yaitu *original*, *blur*, *brightness*, *noisy*, dan *zoom*, masing-masing berjumlah 208 citra, sehingga total citra uji berjumlah 1.040 citra.

Pengujian dilakukan pada dua tahap, yaitu *model leaf detector* untuk memastikan objek terdeteksi sebagai daun, kemudian *model disease classifier* untuk mengklasifikasikan kondisi daun. *Robustness* diukur melalui konsistensi prediksi, yaitu membandingkan hasil prediksi pada citra hasil augmentasi terhadap hasil prediksi pada citra original dari gambar yang sama sebagai acuan. Prediksi dinyatakan konsisten apabila kelas yang dihasilkan pada citra augmentasi sama dengan kelas pada citra originalnya.

Persentase konsistensi dihitung untuk tiap kategori augmentasi, baik pada tahap *leaf detector* maupun *disease classifier*, guna mengetahui kondisi citra yang masih dapat ditangani model dengan baik dan kondisi yang

menyebabkan penurunan performa.

5.1.3. Metode Pengujian Performa Aplikasi

Pengujian ini berfokus pada aspek performa teknis aplikasi menggunakan *Flutter DevTools* pada mode *profile*, yaitu mode eksekusi yang dioptimalkan mendekati kondisi aplikasi sesungguhnya namun tetap menyediakan akses data untuk keperluan analisis. Pengujian mencakup tiga aspek utama, yaitu *Performance* untuk mengukur *frame rate* dan mendeteksi *jank* (frame rendering yang lambat), *CPU Profiler* untuk mengukur waktu eksekusi proses pada aplikasi, serta *memory* untuk memantau penggunaan memori dan mendeteksi potensi kebocoran memori (*memory leak*) (Kartika and Priyawati 2026).

5.2. Hasil Pengujian

Hasil pengujian dilakukan untuk mengevaluasi kinerja model dalam mengklasifikasikan penyakit serta memastikan bahwa seluruh fitur aplikasi telah berfungsi sesuai dengan kebutuhan yang telah ditetapkan.

5.2.1. Hasil Pelatihan Model

Hasil meliputi perkembangan nilai *accuracy* dan *loss* selama proses pelatihan sebagai dasar dalam menentukan performa model terbaik.

1. Epoch

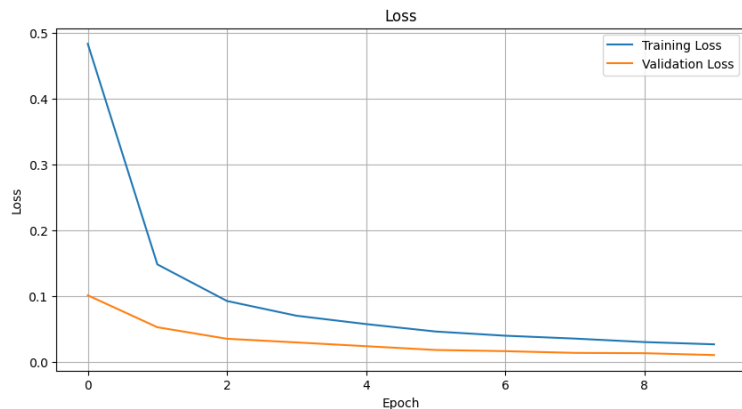
Dari proses iterasi *epoch*, nilai *training accuracy* mengalami peningkatan secara bertahap, dari 82,53% pada *epoch* ke-1 menjadi 99,15% pada *epoch* ke-10. Sementara itu, *validation accuracy* meningkat dari 97,27% menjadi 99,74%. Di sisi lain, nilai *training loss* dan *validation loss* terus menurun hingga mencapai 0,0265 dan 0,0101 pada akhir proses pelatihan.

Gambar 5.1 Iterasi *Epoch*

Kedua kurva mengalami peningkatan pada setiap *epoch* hingga mencapai nilai di atas 99%. Nilai *validation accuracy* yang tetap tinggi dan tidak berbeda jauh dengan *training accuracy*.



Kedua kurva mengalami penurunan yang konsisten pada setiap *epoch* hingga mencapai nilai yang rendah pada akhir pelatihan.



Gambar 5.3 Grafik Loss

5.2.2. Hasil Evaluasi Model

Model dievaluasi untuk mengetahui kemampuan model dalam mengklasifikasikan penyakit. Evaluasi dilakukan menggunakan metrik yang sudah ditentukan.

1. Model Accuracy

Berdasarkan hasil evaluasi, model memperoleh nilai *accuracy* sebesar 99,87% dengan *loss* sebesar 0,0092.

```
[ ] # =====
# EVALUASI TEST SET
# =====

test_loss, test_accuracy = model.evaluate(
    test_generator,
    verbose=1
)

print(f"\nTest Accuracy : {test_accuracy:.4f}")
print(f"Test Loss      : {test_loss:.4f}")

121/121 ————— 17s 140ms/step - accuracy: 0.9987 - loss: 0.0092

Test Accuracy : 0.9987
Test Loss      : 0.0092
```

Gambar 5.4 Akurasi Model

2. Precision, Recall, dan F1-Score

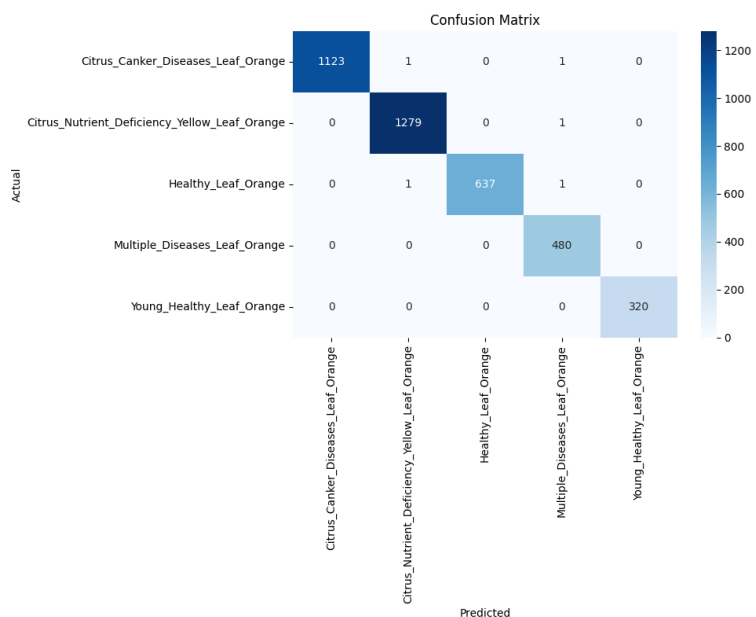
Model menunjukkan nilai *precision*, *recall*, dan *f1-score* pada hampir seluruh kelas konsisten mencapai 1,00, sedangkan kelas *Multiple Diseases Leaf Orange* memiliki nilai *precision* sebesar 0,99 dengan *recall* dan *f1-score* tetap 1,00

	precision	recall	f1-score	support
Citrus_Canker_Diseases_Leaf_Orange	1.00	1.00	1.00	1125
Citrus_Nutrient_Deficiency_Yellow_Leaf_Orange	1.00	1.00	1.00	1280
Healthy_Leaf_Orange	1.00	1.00	1.00	639
Multiple_Diseases_Leaf_Orange	0.99	1.00	1.00	480
Young_Healthy_Leaf_Orange	1.00	1.00	1.00	320
accuracy			1.00	3844
macro avg	1.00	1.00	1.00	3844
weighted avg	1.00	1.00	1.00	3844

Gambar 5.5 Precision, Recall, dan F1-Score

3. Confusion Matrix

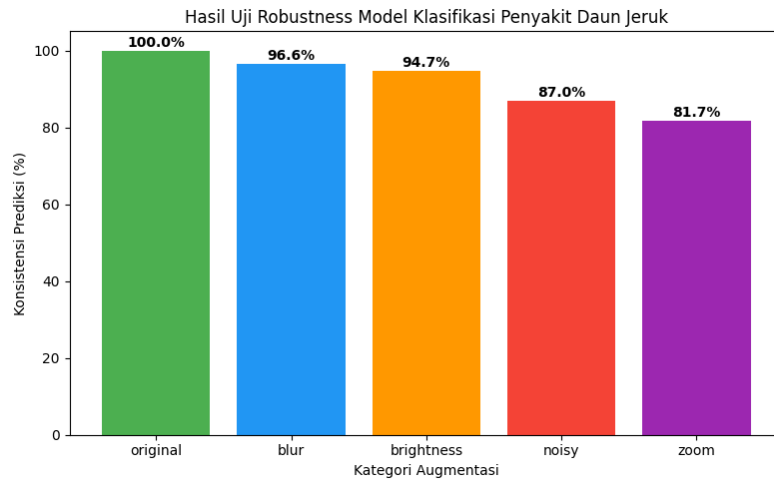
Pada *confusion matrix*, sebagian besar citra diklasifikasikan ke kelas yang benar yang ditunjukkan oleh nilai dominan pada diagonal utama matriks. Dari total, hanya terdapat beberapa kesalahan klasifikasi, yaitu dua citra pada kelas *Citrus Canker Diseases Leaf Orange*, satu citra pada kelas *Citrus Nutrient Deficiency Yellow Leaf Orange*, serta dua citra pada kelas *Healthy Leaf Orange*. Sementara itu, seluruh citra pada kelas *Multiple Diseases Leaf Orange* dan *Young Healthy Leaf Orange* berhasil diklasifikasikan dengan benar. Hasil ini menunjukkan bahwa model memiliki kemampuan klasifikasi yang sangat baik dengan tingkat kesalahan yang sangat rendah



Gambar 5.6 Confusion Matrix

5.2.3. Hasil Pengujian Robustness

Pengujian *robustness* dilakukan terhadap 1.040 citra uji yang terdiri atas lima kategori, yaitu *original*, *blur*, *brightness*, *noisy*, dan *zoom*, dengan 208 citra pada masing-masing kategori. Hasil pengujian pada model *disease classifier* ditampilkan pada tabel berikut.



Gambar 5.7 Hasil Pengujian *Robustness*

Tabel 5.1 Hasil Pengujian *Robustness*

No.	Jenis Augmentasi	Konsistensi Leaf Detector	Konsistensi Diseases Classifier
1.	Original	100%	100%
2.	Blur	100%	96,6%
3.	Brightness	100%	94,7%
4.	Noisy	100%	87%
5.	Zoom	100%	81,7%
	Rata-rata	100%	90%

Dari hasil pengujian, model *leaf detector* menunjukkan tingkat konsistensi sebesar 100% pada seluruh kategori augmentasi, yang mengindikasikan bahwa tahap deteksi daun tidak terpengaruh oleh perubahan kondisi citra apa pun. Sementara itu, model *disease classifier* menunjukkan tingkat konsistensi sebesar 96,6% pada kategori *blur*, 94,7% pada *brightness*, 87,0% pada *noisy*, dan 81,7% pada *zoom*, dengan rata-rata konsistensi sebesar 90,0% di luar kategori *original*. Tingkat konsistensi menurun secara bertahap mengikuti urutan *blur*, *brightness*, *noisy*, hingga *zoom*, yang menunjukkan bahwa model relatif tahan terhadap perubahan

ketajaman dan pencahayaan citra, namun mulai menurun performanya pada citra yang mengandung derau, dan menunjukkan penurunan paling signifikan pada citra dengan perubahan skala. Penurunan konsistensi pada kategori *zoom* kemungkinan disebabkan oleh berubahnya proporsi area gejala penyakit pada daun akibat proses pembesaran maupun pemotongan pada citra, sehingga fitur visual yang menjadi acuan klasifikasi model menjadi kurang representatif dibandingkan pada citra aslinya. Sementara itu, penurunan pada kategori *noisy* diduga disebabkan oleh derau yang mengaburkan tekstur dan pola warna pada permukaan daun, yang merupakan fitur utama yang digunakan model dalam membedakan kondisi penyakit.

5.2.4. Hasil Black Box Testing

Pengujian aplikasi dilakukan terhadap seluruh fitur utama, mencakup fitur pada sisi Pengguna maupun *Admin*, dengan skenario pengujian kondisi normal (input valid) maupun kondisi tidak valid untuk memastikan sistem menampilkan respons yang sesuai.

Tabel 5.2 Hasil *Black Box Testing*

No.	Halaman dan Fitur yang Diuji	Skenario Pengujian	Output	Hasil
1.	Registrasi Pengguna	Mengisi <i>username</i> , nomor hp dan <i>password</i> dengan benar	Akun berhasil didaftarkan	Berhasil
2.	Registrasi Pengguna	Mengisi <i>Username</i> kurang dari 6 karakter	Sistem menampilkan pesan " <i>username</i> minimal 6 karakter"	Berhasil
3.	Registrasi Pengguna	Mengisi Nomor HP kurang dari 8 digit	Sistem menampilkan pesan "Nomor HP tidak valid"	Berhasil
4.	Registrasi Pengguna	Mengisi Nomor HP sudah terdaftar	Sistem menampilkan pesan nomor sudah	Berhasil

			digunakan	
5.	Registrasi Pengguna	Mengisi <i>password</i> kurang dari 6 karakter	Sistem menampilkan pesan " <i>password</i> minimal 6 karakter"	Berhasil
6.	Login Pengguna	Mengisi <i>username</i> dan <i>password</i> yang benar	Pengguna masuk ke <i>Dashboard</i>	Berhasil
7.	Login Pengguna	Mengisi <i>username</i> yang belum terdaftar	Sistem menampilkan pesan " <i>Username</i> belum terdaftar. Silakan daftar dahulu"	Berhasil
8.	Login Pengguna	Mengisi <i>password</i> yang salah	Sistem menampilkan pesan <i>password</i> salah	Berhasil
9.	Login Admin	Mengisi <i>username</i> dan <i>password</i> admin yang valid	<i>Admin</i> masuk ke <i>Dashboard Admin</i>	Berhasil
10.	Login Admin	Login akun ber-role <i>user</i>	Sistem menolak akses sebagai <i>admin</i>	Berhasil
11.	Lanjutkan sebagai Tamu	Menekan tombol Lanjutkan sebagai Tamu	Pengguna masuk ke <i>Dashboad</i> dan hanya bisa akses fitur deteksi	Berhasil
12.	Dashboard Pengguna	Membuka halaman dashboard	<i>Dashboard</i> tampil beserta seluruh menu	Berhasil

13.	<i>Dashboard Admin</i>	Membuka halaman <i>dashboard admin</i>	Statistik pengguna, deteksi, dan data terbanyak tampil	Berhasil
14.	<i>Scan Kamera</i>	Mengambil foto daun melalui kamera	Gambar berhasil diambil	Berhasil
15.	<i>Scan Galeri</i>	Memilih gambar dari galeri	Gambar berhasil dipilih	Berhasil
16.	Deteksi Penyakit	Menekan tombol Deteksi Penyakit setelah memilih gambar	Model melakukan klasifikasi dan menampilkan hasil	Berhasil
17.	Deteksi Penyakit	Menekan tombol Deteksi Penyakit tanpa memilih gambar	Sistem menampilkan pesan "Silakan ambil atau upload foto daun terlebih dahulu"	Berhasil
18.	Hasil Deteksi	Membuka halaman hasil deteksi	Informasi hasil klasifikasi ditampilkan	Berhasil
19.	Riwayat Deteksi	Menyimpan hasil deteksi	Riwayat deteksi terbaru ditampilkan	Berhasil
20.	Riwayat Deteksi	Membuka salah satu detail riwayat	Detail hasil deteksi ditampilkan	Berhasil
21.	Riwayat Deteksi	Menghapus data riwayat	Riwayat berhasil dihapus	Berhasil
22.	Riwayat Deteksi (Admin)	Melihat riwayat deteksi seluruh pengguna	Data riwayat seluruh pengguna ditampilkan	Berhasil

23.	Manajemen User	Membuka menu Manajemen User	Daftar seluruh pengguna ditampilkan	Berhasil
24.	Statistik Aplikasi	Membuka menu Statistik Aplikasi	Distribusi data ditampilkan	Berhasil
25.	Pengaturan Aplikasi	Mengubah informasi aplikasi	Perubahan tersimpan	Berhasil
26.	Halaman Informasi	Membuka menu informasi	Materi berisi modul informasi ditampilkan	Berhasil
27.	<i>Profile</i>	Membuka halaman <i>profile</i>	Data dan menu pengaturan <i>profile</i> ditampilkan	Berhasil
28.	<i>Profile</i>	Mengganti nomor HP	Nomor HP berhasil diperbarui	Berhasil
29.	<i>Profile</i>	Mengganti password	<i>Password</i> berhasil diperbarui	Berhasil
30.	<i>Profile</i>	Menambah atau mengganti foto profil	Foto berhasil ditambah/ diganti	Berhasil
31.	<i>Profile</i>	Menghapus akun	Akun dan data terkait terhapus	Berhasil
32.	Alat Bantu Petani	Menyimpan data Referensi Harga	Data harga tersimpan dan tampil di riwayat	Berhasil
33.	Alat Bantu Petani	Menghitung estimasi pupuk dan pestisida	Hasil estimasi pupuk dan pestisida ditampilkan	Berhasil
34.	Alat Bantu Petani	Menghitung Kalkulator Keuntungan Panen	Estimasi keuntungan ditampilkan sesuai input	Berhasil
35.	Alat Bantu Petani	Mencatat transaksi Buku	Data pemasukan/	Berhasil

		Kas	pengeluaran tersimpan	
36.	Logout	Menekan tombol Logout	Pengguna kembali ke halaman <i>Us Mode</i>	Berhasil

5.2.5. Hasil Pengujian Flutter DevTools

1. Performance

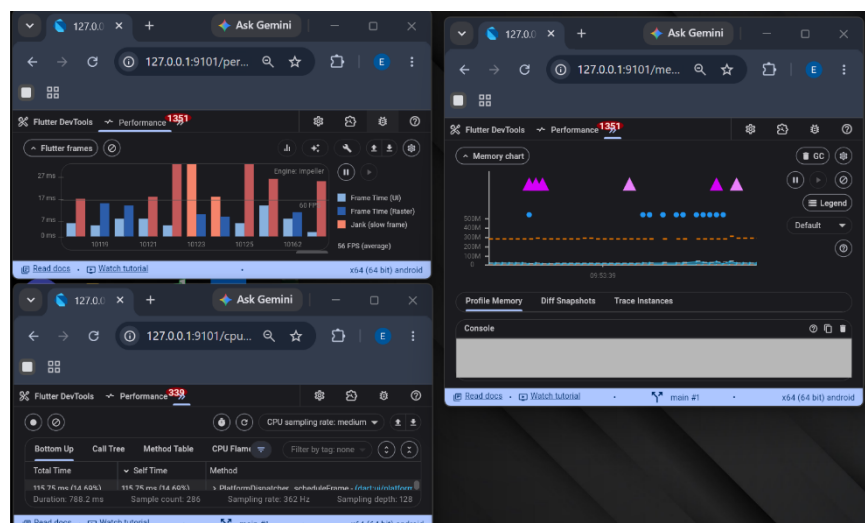
Rata-rata 56 FPS, mendekati 60 FPS (*Standart Android*). Ada beberapa bar oranye/merah yang menandakan *jank* (*frame* lambat), kelihatan di *frame* 10119, 10123, 10125, 10162, tingginya sampai 27ms.

2. CPU Profiler

Method yang paling makan waktu *PlatformDispatcher._scheduleFrame* 115,75 ms (14,69% dari total waktu). Sample count 286 dengan durasi rekaman 788,2 ms cukup singkat.

3. Memory

Grafik heap Dart relatif stabil di bawah 100M (garis biru datar di bawah). Ada beberapa segitiga ungu, itu penanda GC (*Garbage Collection*) *Event* yaitu momen aplikasi otomatis "beberes" memori yang udah nggak dipakai.



Gambar 5.8 Pengujian *Flutter DevTools*

5.3. Pembahasan Hasil Pengujian

Pembahasan difokuskan pada analisis kinerja model dalam mengklasifikasikan penyakit daun jeruk serta implementasi aplikasi berdasarkan hasil pengujian yang telah diperoleh.

5.3.1. Analisis Kinerja Model

Berdasarkan hasil pelatihan, evaluasi model, dan pengujian dataset *testing*, diperoleh beberapa temuan sebagai berikut.

1. Akurasi Model

Model memperoleh training *accuracy* sebesar 99,15%, *validation accuracy* sebesar 99,74%, dan test *accuracy* sebesar 99,87%. Hasil tersebut menunjukkan bahwa model mampu mengenali pola citra daun jeruk dengan tingkat ketepatan yang sangat tinggi.

2. Nilai Loss

Nilai training *loss* terus menurun hingga 0,0265, sedangkan *validation loss* mencapai 0,0101 pada akhir pelatihan. Pada proses evaluasi menggunakan data *testing* diperoleh test *loss* sebesar 0,0092. Nilai *loss* yang rendah menunjukkan bahwa kesalahan prediksi model sangat kecil sehingga proses pembelajaran berjalan dengan baik.

3. Hasil Classification Report

Hasil evaluasi menunjukkan bahwa model memperoleh nilai *precision*, *recall*, dan *F1-score* sebesar 1,00 pada hampir seluruh kelas penyakit daun jeruk. Hal ini menunjukkan bahwa model mampu mengklasifikasikan setiap kategori dengan tingkat ketepatan dan konsistensi yang sangat baik.

4. Hasil Confusion Matrix

Confusion matrix memperlihatkan bahwa sebagian besar citra berhasil diklasifikasikan ke kelas yang benar, dengan hanya lima kesalahan prediksi dari total 3.844 data testing, yaitu dua citra pada kelas *Citrus Canker Diseases Leaf Orange*, satu citra pada kelas *Citrus Nutrient Deficiency Yellow Leaf Orange*, serta dua citra pada kelas *Healthy Leaf Orange*. Mayoritas data berada pada diagonal utama matriks, yang menunjukkan tingkat klasifikasi yang sangat tinggi.

5. Kesimpulan Kinerja Model

Secara keseluruhan, hasil pengujian menunjukkan bahwa model CNN berbasis MobileNetV2 memiliki performa yang sangat baik untuk mengklasifikasikan

lima kategori kondisi daun jeruk. Tingginya nilai *accuracy*, rendahnya nilai *loss*, serta hasil *classification report* dan *confusion matrix* menunjukkan bahwa model layak diimplementasikan pada aplikasi *mobile* sebagai sistem deteksi.

5.3.2. Analisis Kinerja Aplikasi

Berdasarkan hasil pengujian *Black Box* terhadap aplikasi *mobile*, diperoleh beberapa hasil sebagai berikut.

1. Fungsionalitas Aplikasi

Pengujian dilakukan terhadap seluruh fitur utama aplikasi melalui 36 skenario pengujian, mencakup fitur pada sisi Pengguna maupun *Admin*, meliputi registrasi, *login* (pengguna dan *admin*), akses sebagai tamu, deteksi penyakit, pengelolaan riwayat, profil, Alat Bantu Petani, hingga fitur manajemen data pada sisi admin. Seluruh skenario, baik dengan input valid maupun tidak valid, memperoleh status Berhasil sesuai dengan output yang diharapkan.

2. Proses Deteksi Penyakit

Aplikasi mampu mengambil citra daun melalui kamera maupun galeri, kemudian menjalankan proses inferensi menggunakan model *TensorFlow Lite* secara langsung pada perangkat (*on-device*). Hasil deteksi berupa nama kelas, nilai *confidence*, dan informasi penyakit berhasil ditampilkan tanpa kendala.

3. Integrasi Model dengan Aplikasi

Model berhasil diintegrasikan dengan aplikasi Flutter melalui layanan *TFLiteService*. Proses pemanggilan model, *preprocessing* citra berukuran 224×224 piksel, inferensi, hingga pengiriman hasil ke halaman hasil deteksi dapat berjalan dengan baik.

4. Antarmuka Pengguna

Antarmuka aplikasi berhasil menampilkan seluruh halaman sesuai rancangan, meliputi *dashboard* (pengguna dan *admin*), halaman scan, proses analisis, hasil deteksi, riwayat, informasi, profil, dan Alat Bantu Petani. Navigasi antarhalaman juga berjalan dengan baik sehingga memudahkan pengguna dalam mengoperasikan aplikasi.

5. Kesimpulan Kinerja Aplikasi

Hasil pengujian menunjukkan bahwa aplikasi berhasil mengimplementasikan model CNN berbasis

MobileNetV2 pada platform *mobile*. Seluruh fitur utama, baik pada sisi pengguna maupun *admin*, dapat dijalankan dengan baik, proses deteksi berlangsung sesuai kebutuhan, dan hasil klasifikasi dapat ditampilkan secara cepat, sehingga aplikasi layak digunakan sebagai media bantu identifikasi penyakit daun jeruk sekaligus pengelolaan usaha tani sederhana.

5.3.3. Analisis Performa Aplikasi

Berdasarkan hasil pengujian menggunakan *Flutter DevTools*, diperoleh beberapa hasil sebagai berikut.

1. Frame Rate dan Rendering

Pengujian pada tab *Performance* menunjukkan rata-rata *frame rate* sebesar 56 FPS, mendekati nilai ideal 60 FPS untuk pengalaman pengguna yang lancar. Beberapa frame teramati mengalami durasi rendering melebihi 16ms, yang teridentifikasi berkaitan dengan aktivitas *Garbage Collection* pada momen tertentu.

2. Penggunaan CPU

Pengujian pada CPU *Profiler* menunjukkan bahwa proses penjadwalan frame (*Platform Dispatcher._scheduleFrame*) menjadi kontributor waktu eksekusi terbesar selama pengujian berlangsung.

3. Penggunaan Memori

Pengujian pada tab *Memory* menunjukkan penggunaan heap *Dart* yang stabil, tanpa indikasi kebocoran memori (*memory leak*) selama pengujian berlangsung.

4. Kesimpulan Performa Aplikasi

Secara keseluruhan, hasil pengujian performa menunjukkan bahwa aplikasi berjalan responsif dan efisien, dengan sesekali frame menurun.