BAB IV
PERANCANGAN

4.1. Analisis Sistem Terdahulu

Penelitian terdahulu yang akan dijadikan perbandingan dari jurnal yang berjudul "Implementasi Arsitektur EfficientNetV2 Untuk Klasifikasi Gambar Makanan Tradisional Indonesia" oleh Erin Eka Citra, Dhomas Hatta Fudholi, dan Chandra Kusuma Dewa. Dalam penelitian ini, peneliti menentukan model dari arsitektur EfficientNetV2 untuk mengklasifikasikan gambar makanan tradisional Indonesia berdasarkan akurasi, precision, recall, dan F1-score. Mereka menguji empat varian EfficientNetV2 (S_21k, M_21k, L_21k, dan XL_21k) menggunakan dataset yang dikumpulkan dari Google Images, kamera smartphone, dan gabungan keduanya. Hasilnya menunjukkan bahwa EfficientNetV2_L_21k adalah model terbaik dengan akurasi pengujian tertinggi sebesar 99,4% dan tidak mengalami overfitting. Penelitian ini juga menentukan bahwa pengambilan gambar dari berbagai sudut dan sumber meningkatkan kemampuan model untuk mengenali dan mengklasifikasikan makanan (Citra, Fudhoni, & Dewa, 2023).

4.2. Spesifikasi Kebutuhan Sistem Baru

4.2.1. Spesifikasi Kebutuhan Perangkat Keras

Dalam Penelitian ini menggunakan spesifikasi kebutuhan perangkat keras yang terdiri dari Processor, VGA dan Memory. Kebutuhan perangkat keras yang digunakan sebagi berikut.

No	Perangkat Keras	Kebutuhan Perangkat Keras
1	Processor	Intel Core i5 7200U
2	VGA	Intel UHD Graphics 620
3	Memory	8 GB

Tabel 4.1. Spesifikasi Kebutuhan Perangkat Keras

4.2.2. Spesifikasi Kebutuhan Perangkat Lunak

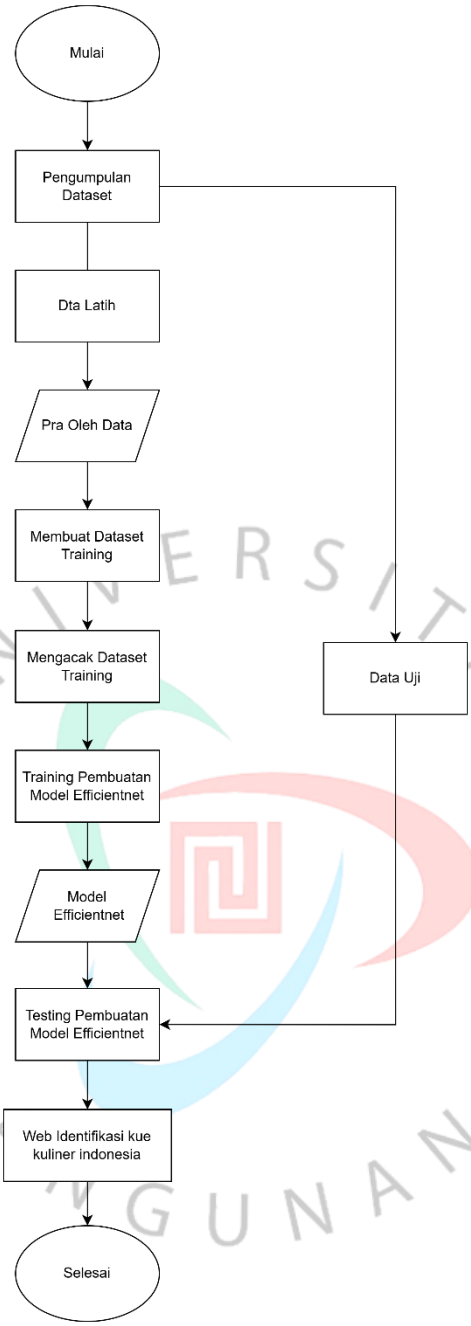
Dalam Penelitian ini menggunakan spesifikasi kebutuhan perangkat lunak yang terdiri dari aplikasi editor dan sistem operasi. Kebutuhan perangkat lunak yang digunakan sebagi berikut.

No	Perangkat Lunak	Kebutuhan Perangkat Lunak
1	*Operation System*	Windows 11
2	*IDE*	Visual Studio Code
3	*Scripting Language*	Python
4	*Framework web*	Streamlit

Tabel 4.2. Spesifikasi Kebutuhan Perangkat Lunak

4.3. Perancangan Sistem

Pada tahap ini, perancangan sistem indentifikasi klasifikasi citra kuliner kue indonesia yang digunakan untuk mengenali kuliner kue Indonesia. Berikut penjelasan dan tahapan mengenai tahapan pembuatan aplikasi berbasis web kuliner kue indoesia.



Gambar 4.1. perancangan sistem indentifikasi klasifikasi citra kuliner kue indonesia

4.3.1. Pengumpulan Dataset

Gambar 4.2. Dataset Kuliner Kue Indonesia

Dalam langkah awal yang dilakukan oleh peneliti dalam proses pembuatan aplikasi klasifikasi citra kuliner kue indonesia adalah pengumpulan dataset. Dataset yang dikumpulkan sebagai data latih terdiri dari 940 gambar kuliner kue Indonesia yang terbagi menjadi 10 kelas yaitu 115 gambar lapis legit, 105 gambar pepe, 100 gambar bolu kukus bunga, 100 gambar bika ambon, 100 gambar dadar gulung, 100 gambar kue talam, 100 gambar unkue, 80 gambar kue mangkok, 70 gambar kue lumpur, dan 70 gambar putu ayu. Sedangkan dataset yang dingunakan untuk data uji terdiri 20%. Berikut rincian tablenya.

No	Nama Kue	Jumlah Gambar
1	Bolu Kukus Bunga	100
2	Bika Ambon	100
3	Dadar Gulung	100
4	Lapis Legit	115
5	Kue Lumpur	70
6	Kue Mangkok	80
7	Kue pepe	105
8	Putu Ayu	70
9	Talam	100
10	Unkue	100
Total		**940**

Tabel 4.3. Dataset Kuliner Kue Indonesia

4.3.2. Pra Olah Dataset

Dalam tahap ini peneliti melakukan proses tahapan data latih yang dikumpulkan. Berikut penjelasan pra olah dataset kuliner kue Indonesia.

1. Peneliti menggunakan rasio aspek gambar 4:3 pada setiap gambar yang diambil, kemudian mengubahnya menjadi rasio 1:1 untuk memastikan fokus pada objek utama. Gambar tersebut selanjutnya disesuaikan ukurannya menjadi 224x224 piksel agar seragam.
2. Untuk meningkatkan keragaman dan kualitas dataset pelatihan, peneliti menerapkan augmentasi data menggunakan kelas *ImageDataGenerator* dari Keras. Augmentasi ini bertujuan memperkaya dataset dengan variasi gambar melalui transformasi acak berikut:
   1.) Rotasi Gambar: hingga 10 derajat.
   2.) Zoom Acak: hingga 10%.
   3.) Pergeseran Posisi Horizontal dan Vertikal: masing-masing hingga 20% dari dimensi gambar.
   4.) Pembalikan Horizontal: memungkinkan gambar dibalik secara horizontal.

   Sementara itu, teknik seperti *featurewise centering*, *samplewise centering*, standardisasi, dan *ZCA whitening* tidak diterapkan dalam proses ini.
3. Setelah parameter augmentasi ditentukan, metode fit diterapkan pada objek datagen dengan dataset pelatihan (X_train). Langkah ini dilakukan untuk menghitung statistik yang diperlukan dalam proses augmentasi.

4.3.3. Membuat Dataset Training

```
X=[]
L=[]
IMG_SIZE = 224
BOLU_KUKUS = 'Input/Bolu Kukus Bunga/'
BIKA_AMBON = 'Input/Kue Bika Ambon/'
DADAR_GULUNG = 'Input/Kue Dadar Gulung/'
LAPIS_LEGIT = 'Input/Kue Lapis legit/'
LUMPUR = 'Input/Kue Lumpur/'
MANGKOK = 'Input/Kue Mangkok/'
PEPE = 'Input/Kue Pepe/'
PUTU_AYU = 'Input/Kue Putu ayu/'
TALAM = 'Input/Kue Talam/'
UNKUE = 'Input/Kue Unkue/'

def assign_label(img,cake_type):
    return cake_type
def make_train_data(cake_type,DIR):
    for img in tqdm(os.listdir(DIR)):
        label = assign_label(img, cake_type)
        path = os.path.join(DIR, img)
        img = cv2.imread(path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))

        X.append(np.array(img))
        L.append(str(label))
```

```
make_train_data('Bolu Kukus Bunga',BOLU_KUKUS)
make_train_data('Bika Ambon',BIKA_AMBON)
make_train_data('Dadar Gulung',DADAR_GULUNG)
make_train_data('Lapis Legit',LAPIS_LEGIT)
make_train_data('Lumpur',LUMPUR)
make_train_data('Mangkok', MANGKOK)
make_train_data('Pepe', PEPE)
make_train_data('Putu Ayu', PUTU_AYU)
make_train_data('Talam', TALAM)
make_train_data('Unkue',UNKUE)
print(len(X))
```

```
  0%|          | 0/73 [00:00<?, ?it/s]
100%|██████████| 73/73 [00:10<00:00,  6.83it/s]
100%|██████████| 67/67 [00:09<00:00,  7.11it/s]
100%|██████████| 66/66 [00:08<00:00,  7.62it/s]
100%|██████████| 114/114 [00:14<00:00,  7.83it/s]
100%|██████████| 51/51 [00:06<00:00,  7.36it/s]
100%|██████████| 77/77 [00:10<00:00,  7.34it/s]
100%|██████████| 81/81 [00:10<00:00,  7.54it/s]
100%|██████████| 64/64 [00:09<00:00,  6.81it/s]
100%|██████████| 94/94 [00:13<00:00,  7.15it/s]
100%|██████████| 94/94 [00:17<00:00,  5.48it/s]
781
```

Gambar 4.3. Pembuatan Dataset Training

Dalam tahap ini peneliti membuat dataset inputs untuk melakukan pembuatan dataset training untuk klasifikasi gambar kue kuliner Indonesia. Dataset terdiri dari beberapa kategori, seperti "Bolu Kukus", "Bika Ambon", "Dadar Gulung", dan lainnya, yang masing-masing disimpan dalam direktori tertentu. Gambar diubah ukurannya menjadi 224x224 piksel menggunakan OpenCV, lalu disimpan dalam array X untuk data gambar dan L untuk label kategori, yang ditentukan melalui fungsi assign_label. Fungsi make_train_data digunakan untuk membaca, memproses, dan menyimpan gambar dari setiap kategori, dengan progres bar tqdm menunjukkan jumlah gambar yang telah diproses dan kecepatan pemrosesan. Setelah seluruh data diproses, total dataset berjumlah 781 gambar, yang siap digunakan untuk melatih model klasifikasi.

4.3.4. Mengacak Dataset Training

```
np.random.seed(2023)
random.seed(2023)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
print("Y_train: ", Y_train.shape)
print("Y_test: ", Y_test.shape)
```

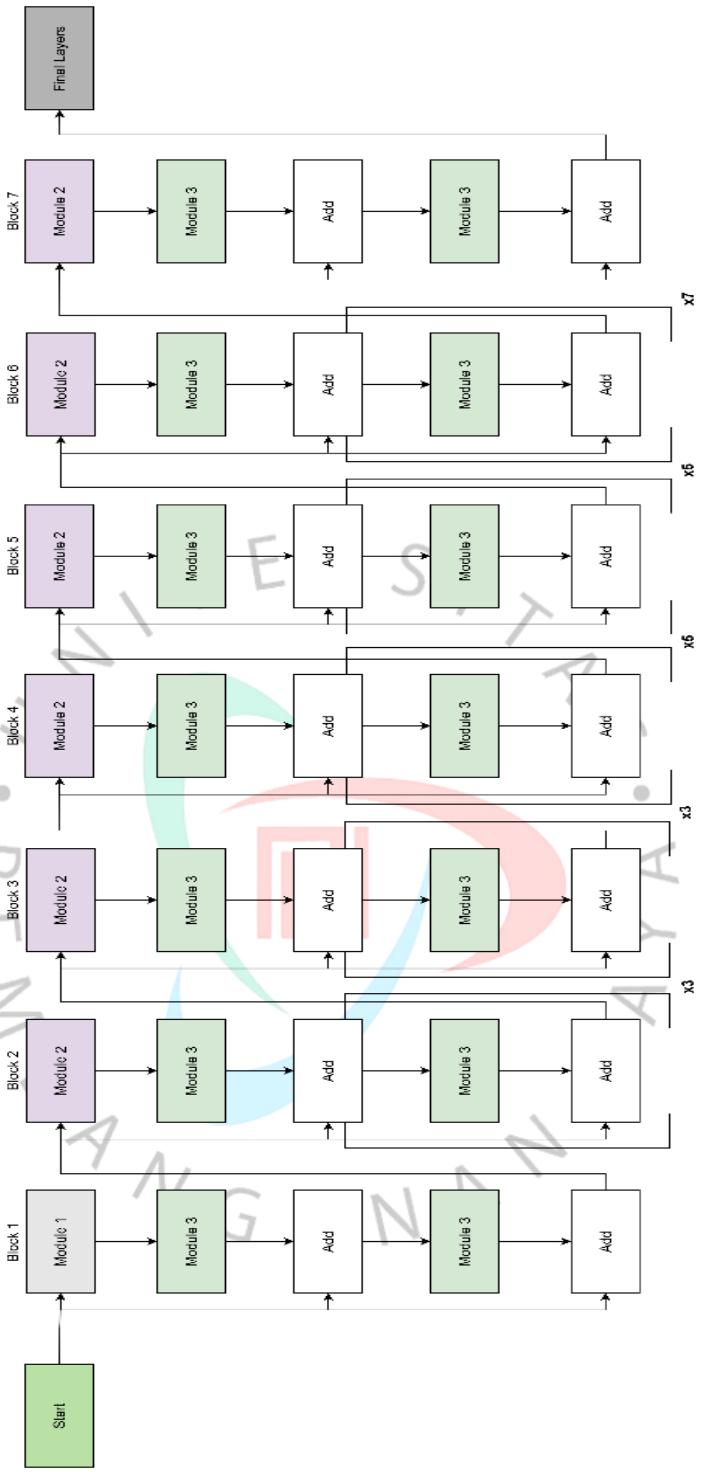
```
X_train:  (624, 224, 224, 3)
X_test:  (157, 224, 224, 3)
Y_train:  (624, 10)
Y_test:  (157, 10)
```

Gambar 4.4. Mengacak Dataset Training

Dalam tahap ini peneliti membagi dataset menjadi data latih dan data uji menggunakan fungsi train_test_split dari *scikit-learn* dengan proporsi 80% untuk data latih (X_train, Y_train) dan 20% untuk data uji (X_test, Y_test). Pengaturan *seed* melalui np.random.seed(2023) dan random.seed(2023) memastikan bahwa pembagian dataset dilakukan secara acak tetapi konsisten setiap kali kode dijalankan. Dataset gambar memiliki dimensi (jumlah sampel, tinggi, lebar, saluran warna), dengan data latih berukuran (624, 224, 224, 3) dan data uji berukuran (157, 224, 224, 3). Sementara itu, labelnya berbentuk (jumlah sampel, jumlah kelas), yaitu (624, 10) untuk data latih dan (157, 10) untuk data uji. Parameter random_state=42 memastikan konsistensi pembagian dataset, dan hasil pembagian diverifikasi melalui output dimensi dari setiap bagian. Proses ini memastikan data uji tetap terpisah untuk mengevaluasi model secara objektif.

4.3.5. Pembuatan Model dengan Arsitektur EffecientNet

Dalam tahap ini peneliti melakukan pengembangan dalam membuat aplikasi identifikasi kuliner kue indonesia menggunakan arsitektur Effecientnet. EfficientNet adalah salah satu arsitektur deep learning yang dikenal karena efisiensinya dalam menghasilkan performa tinggi dengan parameter yang relatif lebih kecil dibandingkan arsitektur lainnya. Model ini dirancang menggunakan varian EfficientNet-B5, yang memiliki keseimbangan optimal antara kedalaman, lebar, dan resolusi jaringan.



Gambar 4.5. Arsitektur EffienceNet B5

Gambar yang ditampilkan menunjukkan struktur arsitektur EfficientNet, di mana jaringan terdiri dari beberapa blok modular. Setiap blok mencakup lapisan seperti Convolution, Batch Normalization, Activation (Swish), serta operasi seperti

Add untuk menyatukan informasi dari berbagai jalur. Model ini secara progresif memperbesar dimensi fitur pada setiap blok, sehingga mampu menangkap informasi kompleks dari gambar kue. Blok pertama menerima input berupa data gambar yang telah diproses sebelumnya (ukuran standar 224x224 piksel). Proses berlanjut melalui beberapa tahap transformasi di dalam blok-blok berikutnya, hingga akhirnya menghasilkan output pada lapisan akhir (*Final Layers*) yang digunakan untuk klasifikasi ke dalam kategori kue yang telah didefinisikan. Arsitektur ini dirancang agar mampu memberikan hasil klasifikasi yang akurat dengan tetap mempertahankan efisiensi komputasi.

Setelah menentukan variable dari *final layers* serta menyiapkan *dataset inputs* maka dapat melakukan pembuatan model dengan arsitektur EfficientNet menggunakan aplikasi IDE Visual studio code. Berikut adalah implementasi kode program dalam pembuatan model aplikasi kuliner kue Indonesia.

```
import os
from tqdm import tqdm
import numpy as np
import pandas as pd
import random
import cv2
from PIL import Image
import matplotlib.pyplot as plt
import seaborn as sns
# Ignore  the warnings
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
```

Gambar 4.6. Potongan Kode Program 1

Potongan kode program ini mengimpor berbagai pustaka yang mendukung pengelolaan dataset, manipulasi data, dan visualisasi. Pustaka seperti os dan cv2 digunakan untuk membaca dan memproses gambar, sedangkan numpy, pandas, dan random membantu manipulasi array dan pengacakan data. Pustaka visualisasi seperti matplotlib dan seaborn mempermudah analisis data, sementara warnings digunakan untuk mengelola peringatan agar tidak mengganggu eksekusi program. Kode ini mempersiapkan seluruh pustaka yang diperlukan untuk mendukung proses pelatihan model..

```
import os
print(os.listdir('Input'))
```

Gambar 4.7. Potongan Kode Program 2

Potongan kode program ini menggunakan pustaka os untuk membaca isi direktori dengan perintah os.listdir('Input'). Perintah ini digunakan untuk menampilkan daftar file dan folder yang terdapat di direktori Input. Langkah ini penting untuk memverifikasi bahwa dataset atau file yang dibutuhkan tersedia di lokasi yang sesuai.

```
import tensorflow as tf

#model selection
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder

#preprocess.
from keras import backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# specifically for cnn
from efficientnet.tfkeras import EfficientNetB5
from keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
```

Gambar 4.8. Potongan Kode Program 3

Potongan kode program ini mengimpor pustaka dan modul yang diperlukan untuk membangun, melatih, dan mengevaluasi model klasifikasi gambar. Modul tensorflow digunakan sebagai inti dari arsitektur jaringan saraf. Pustaka sklearn menyediakan fungsi seperti train_test_split untuk membagi dataset, confusion_matrix untuk mengevaluasi hasil prediksi, dan LabelEncoder untuk mengonversi label kategori. Modul ImageDataGenerator dari Keras digunakan untuk augmentasi data. Selain itu, arsitektur EfficientNet-B5 diimpor untuk membangun model CNN, bersama dengan komponen Keras lainnya seperti Dense, Dropout, dan Sequential untuk mengatur lapisan model. Fungsi to_categorical membantu dalam konversi label ke bentuk one-hot encoding, dan

ReduceLROnPlateau digunakan untuk mengoptimalkan pembelajaran selama pelatihan.

```
import tensorflow as tf
from keras.callbacks import ReduceLROnPlateau

# Mengatur model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),  # Ganti lr dengan learning_rate
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Mengatur batch size dan epoch
batch_size = 64
epochs = 20

# Mengatur callback untuk mengurangi learning rate
red_lr = ReduceLROnPlateau(monitor='val_accuracy', patience=3, verbose=1, factor=0.1)
```

Gambar 4.9. Potongan Kode Program 4

Potongan kode program ini mendefinisikan pengaturan model, pelatihan, dan optimisasi. Model dikompilasi menggunakan optimizer Adam dengan *learning rate* 0.001, *loss function* categorical_crossentropy untuk klasifikasi multi-kelas, dan metrik accuracy untuk memantau performa model. Parameter pelatihan diatur dengan ukuran batch 64 dan jumlah epoch 20. Fungsi ReduceLROnPlateau digunakan untuk menurunkan *learning rate* secara adaptif jika performa validasi tidak meningkat dalam tiga epoch berturut-turut. Pengaturan ini dirancang untuk memastikan pelatihan model berjalan secara efisien dan adaptif terhadap perubahan selama proses pelatihan.

```
# Menggunakan model.fit dengan generator
History = model.fit(datagen.flow(X_train, Y_train, batch_size=batch_size),
                    epochs=epochs,
                    validation_data=(X_test, Y_test),
                    verbose=1,
                    steps_per_epoch=X_train.shape[0] // batch_size)
```

Gambar 4.10. Potongan Kode Program 5

Potongan kode program ini melakukan pelatihan model menggunakan metode model.fit dengan generator data dari ImageDataGenerator. Data pelatihan (X_train, y_train) diproses dalam batch sesuai ukuran batch yang ditentukan, dengan parameter epoch sesuai kebutuhan. Data validasi (X_test, y_test) digunakan untuk memonitor performa model selama pelatihan. Parameter steps_per_epoch dihitung berdasarkan jumlah sampel dalam data pelatihan dibagi ukuran batch, dan

verbose=1 digunakan untuk menampilkan progress selama pelatihan. Kode ini bertujuan melatih model secara efisien dengan memanfaatkan generator data.

```
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train Loss', 'Test Loss'])
plt.show()
```

Gambar 4.11. Potongan Kode Program 6

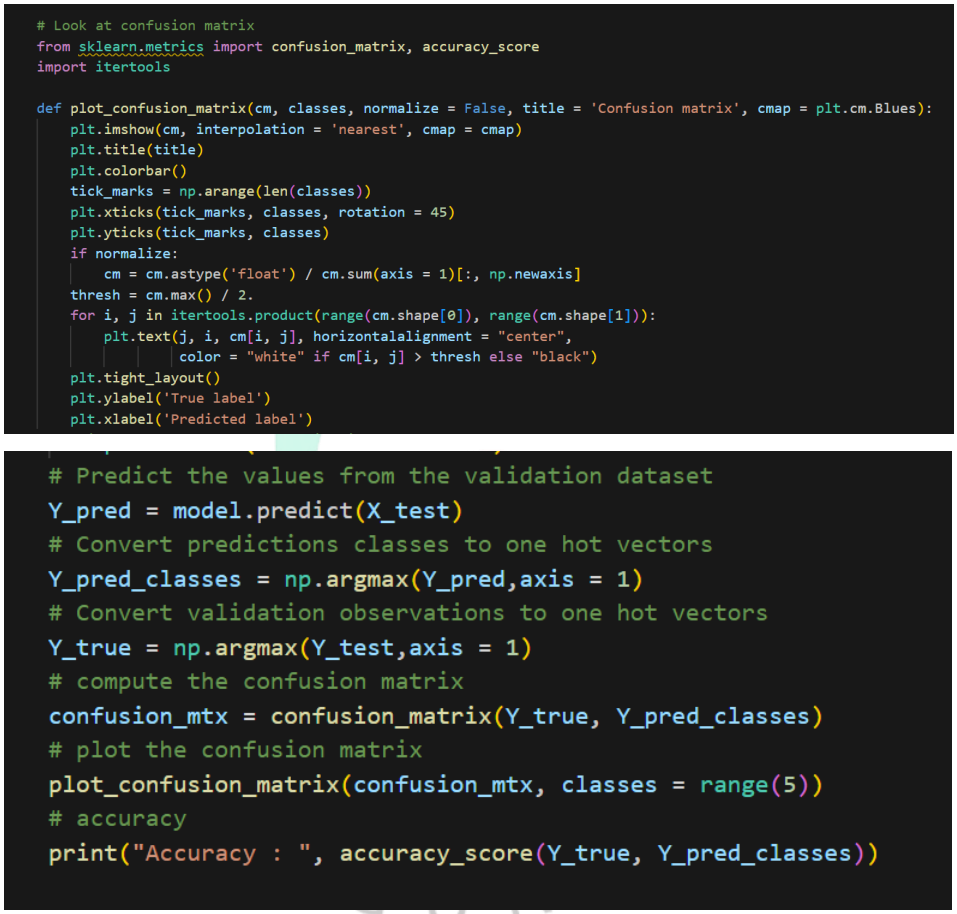
Potongan kode program ini memvisualisasikan performa model selama pelatihan dengan grafik *loss* untuk data latih dan validasi. Fungsi plt.plot digunakan untuk menggambar grafik *loss* (History.history['loss']) dan *validation loss* (History.history['val_loss']). Judul grafik ditentukan dengan plt.title, dan sumbu X serta Y diberi label dengan plt.xlabel dan plt.ylabel. Legenda ditambahkan dengan plt.legend untuk membedakan antara *train loss* dan *test loss*. Terakhir, plt.show digunakan untuk menampilkan grafik. Visualisasi ini membantu memonitor konvergensi model dan mendeteksi potensi overfitting atau underfitting.

```
# Training and Vaidation Accuracy
plt.plot(History.history['accuracy'])
plt.plot(History.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Train Accuracy', 'Test Accuracy'])
plt.show()
```

Gambar 4.12. Potongan Kode Program 7

Potongan kode program ini digunakan untuk memvisualisasikan akurasi model selama pelatihan dan validasi. Fungsi plt.plot menggambar grafik akurasi data pelatihan (History.history['accuracy']) dan akurasi data validasi (History.history['val_accuracy']). Grafik ini diberi judul "Model Accuracy" dengan

menggunakan plt.title, sementara sumbu X dan Y masing-masing diberi label "Epochs" dan "Accuracy" menggunakan plt.xlabel dan plt.ylabel. Fungsi plt.legend menambahkan keterangan untuk membedakan antara *Train Accuracy* dan *Test Accuracy*. Visualisasi ini membantu memonitor performa model dalam mengklasifikasikan data secara akurat selama pelatihan dan validasi. Grafik ini memberikan wawasan tentang potensi overfitting atau underfitting pada model.



```
# Look at confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
import itertools

def plot_confusion_matrix(cm, classes, normalize = False, title = 'Confusion matrix', cmap = plt.cm.Blues):
    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation = 45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis = 1)[:, np.newaxis]
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j], horizontalalignment = "center",
                 color = "white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
# Predict the values from the validation dataset
Y_pred = model.predict(X_test)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(Y_test,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(5))
# accuracy
print("Accuracy : ", accuracy_score(Y_true, Y_pred_classes))
```

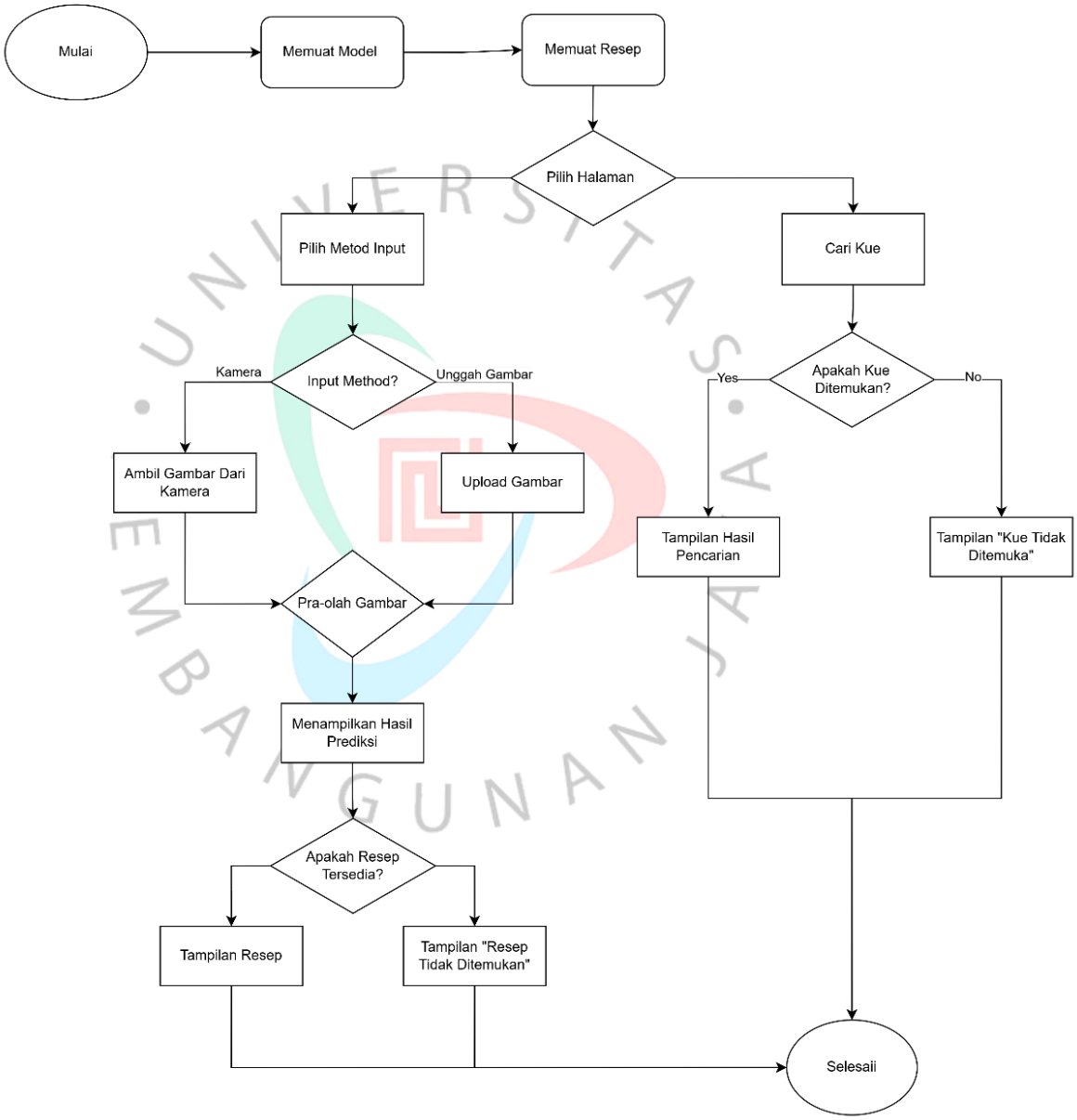
Gambar 4.13. Potongan Kode Program 8

Potongan kode program ini digunakan untuk mengevaluasi performa model menggunakan *confusion matrix* dan menghitung akurasi pada data validasi. Fungsi model.predict digunakan untuk memprediksi label pada data uji (X_test). Prediksi dan label aktual dikonversi menjadi format *one-hot encoding* menggunakan np.argmax. *Confusion matrix* dihitung dengan fungsi confusion_matrix dari sklearn.metrics, dan divisualisasikan menggunakan fungsi plot_confusion_matrix yang dirancang untuk menampilkan hasil prediksi model dalam bentuk matriks dengan jelas. Akurasi model dihitung menggunakan fungsi accuracy_score, yang

kemudian dicetak untuk memberikan informasi seberapa baik model dalam mengklasifikasikan data uji.

4.3.6. Alur Komputasi

Pada tahap alur komputasi ini menggambarkan proses diagram alir.Berikut diagram alir aplikasi identifikasi kuliner kue Indonesia.

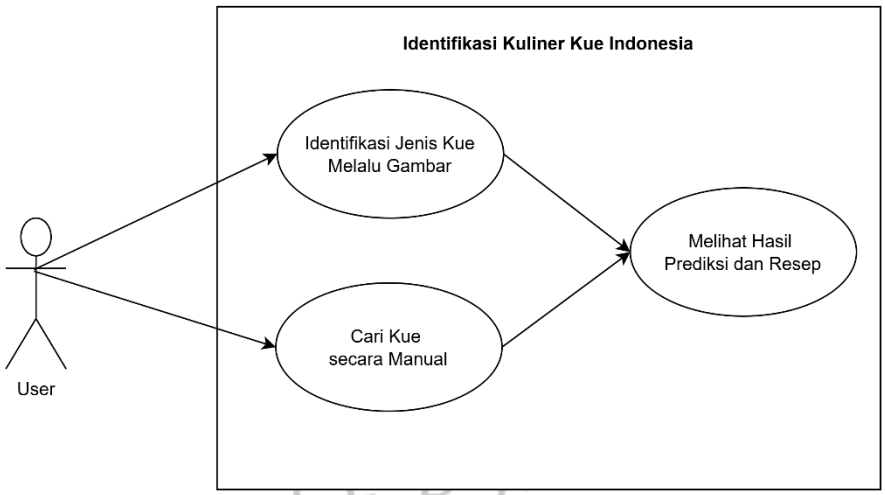


Gambar 4.14. Alur Komputasi Aplikasi Identifikasi Kuliner Kue Indonesia.

Pada gambar diatas memperlihatkan diagram alir aplikasi identifikasi kuliner kue Indonesia. Berikut penjelasan diagram alir tersebut.

1. Proses dimulai dari tahap awal ketika aplikasi diaktifkan.
2. Aplikasi memuat model prediksi yang telah dilatih sebelumnya dan database resep kue. Kedua elemen ini merupakan inti dari sistem untuk mengidentifikasi kue dan menampilkan resepnya.
3. Pengguna diarahkan untuk memilih halaman yang tersedia:
   - Metode Input untuk memprediksi kue menggunakan gambar.
   - Cari Kue untuk mencari informasi tentang kue tertentu.
4. Jika pengguna memilih metode input, mereka dapat memilih antara:
   - Ambil Gambar Dari Kamera: Mengambil gambar kue secara langsung menggunakan kamera.
   - Unggah Gambar: Mengunggah gambar kue dari galeri perangkat.
5. Gambar yang diambil atau diunggah diproses melalui tahap *pra-olah* (pre-processing) agar sesuai untuk diprediksi oleh model.
6. Hasil prediksi ditampilkan berdasarkan gambar yang telah diproses. Model mengidentifikasi jenis kue yang paling mungkin sesuai dengan input.
7. Setelah prediksi kue dilakukan, aplikasi memeriksa apakah resep untuk kue tersebut tersedia:
   - Jika resep tersedia, aplikasi menampilkan resep kue.
   - Jika tidak, pesan "Resep Tidak Ditemukan" akan ditampilkan.
8. Jika pengguna memilih untuk mencari kue, mereka dapat memasukkan nama kue secara manual. Sistem akan memeriksa keberadaan kue tersebut:
   - Jika kue ditemukan, hasil pencarian ditampilkan.
   - Jika tidak, pesan "Kue Tidak Ditemukan" akan ditampilkan.
9. End Proses berakhir setelah pengguna selesai menggunakan salah satu fitur aplikasi.

4.3.7. Use Case Diagram



Gambar 4.15. Use Case Diagram Aplikasi Idetifikasi Kuliner kue Indonesia.

Berdasarkan gambar diatas, dirancang untuk membantu pengguna dalam mengenali jenis kuliner kue Indonesia serta mendapatkan informasi terkait resepnya. Diagram ini menunjukkan tiga fungsi utama yang dapat digunakan oleh User. Pertama, pengguna dapat mengidentifikasi jenis kue dengan mengunggah gambar melalui fitur Identifikasi Jenis Kue Melalui Gambar. Kedua, pengguna dapat mencari kue secara manual menggunakan fitur Cari Kue Secara Manual. Setelah menggunakan salah satu metode tersebut, pengguna dapat mengakses fitur Melihat Hasil Prediksi dan Resep untuk melihat hasil prediksi nama kue yang ditemukan, disertai dengan resep yang relevan. Diagram ini menjelaskan hubungan antara pengguna dengan fungsi-fungsi aplikasi secara sederhana dan jelas, mencerminkan alur interaksi utama dalam sistem.

4.3.8. Skenario Use Case

Berikut terdapat beberapa scenario use case yang digunakan pada aplikasi identifikasi kuliner kue Indonesia.

Tabel 4.4. Skenario Use Case Identifikasi Kuliner Kue Indonesia.

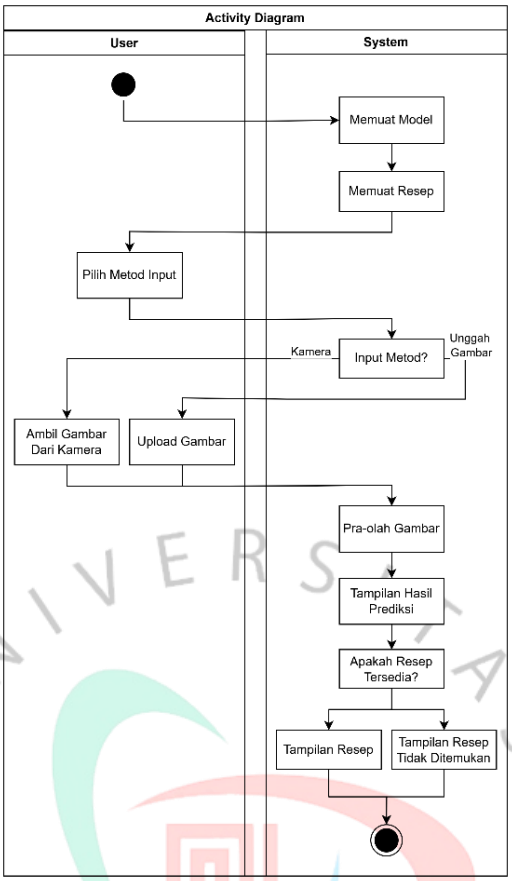
ID	**UC1: Identifikasi Jenis Kue Melalui Gambar**
Nama Use Case	Identifikasi Jenis Kue
Aktor	User

Deskripsi	Mengidentifikasi jenis kue menggunakan gambar yang diunggah atau diambil oleh user melalui kamera, menggunakan model prediksi.
Pre-Condition	1. Aplikasi sudah memuat model prediksi. 2. User membuka aplikasi.
Langkah-Langkah	1. User memilih opsi *Pilih Metode Input*. 2. User memilih untuk *Ambil Gambar* menggunakan kamera atau *Unggah Gambar* dari perangkat. 3. Sistem memproses gambar menggunakan pra-olah data. 4. Sistem menjalankan model prediksi untuk mengidentifikasi jenis kue. 5. Aplikasi menampilkan hasil prediksi kepada user.
Trigger	User mengunggah gambar atau mengambil gambar menggunakan kamera.
Post-Condition	Gambar berhasil diproses dan hasil prediksi jenis kue ditampilkan kepada user.
ID	**UC2: Pencarian Kue Secara Manual**
Nama Use Case	Pencarian Kue
Aktor	User
Deskripsi	User mencari informasi tentang kue dengan memasukkan nama kue secara manual melalui fitur pencarian.
Pre-Condition	1. Aplikasi sudah memuat database kue. 2. User membuka aplikasi.
Langkah-Langkah	1. User memilih opsi *Cari Kue* dari menu utama. 2. User memasukkan nama kue yang ingin dicari. 3. Sistem memproses pencarian dalam database. 4. Jika kue ditemukan, aplikasi menampilkan informasi dan resep kue. 5. Jika tidak ditemukan, aplikasi menampilkan pesan "Kue Tidak Ditemukan".
Trigger	User memasukkan nama kue untuk mencari informasi.

Post-Condition	Informasi tentang kue yang dicari ditampilkan jika tersedia; jika tidak, pesan *Kue Tidak Ditemukan* ditampilkan.
ID	**UC3: Menampilkan Hasil Prediksi dan Resep**
Nama Use Case	Menampilkan Hasil Prediksi dan Resep
Aktor	User, Sistem Prediksi Kue
Deskripsi	Sistem menampilkan hasil prediksi jenis kue berdasarkan gambar input dari user, serta menampilkan resep jika tersedia.
Pre-Condition	1. Gambar sudah diproses melalui pra-olah data. 2. Model prediksi sudah dimuat. 3. Database resep tersedia di sistem.
Langkah-Langkah	1. Sistem menjalankan model prediksi berdasarkan input gambar dari user. 2. Sistem mengidentifikasi jenis kue yang paling sesuai. 3. Sistem memeriksa ketersediaan resep untuk jenis kue tersebut. 4. Jika resep tersedia, sistem menampilkan hasil prediksi dan resep. 5. Jika resep tidak tersedia, sistem menampilkan hasil prediksi dan pesan "Resep Tidak Ditemukan".
Trigger	Gambar telah diproses dan hasil prediksi selesai dihitung.
Post-Condition	Hasil prediksi jenis kue dan resep (jika tersedia) ditampilkan kepada user.

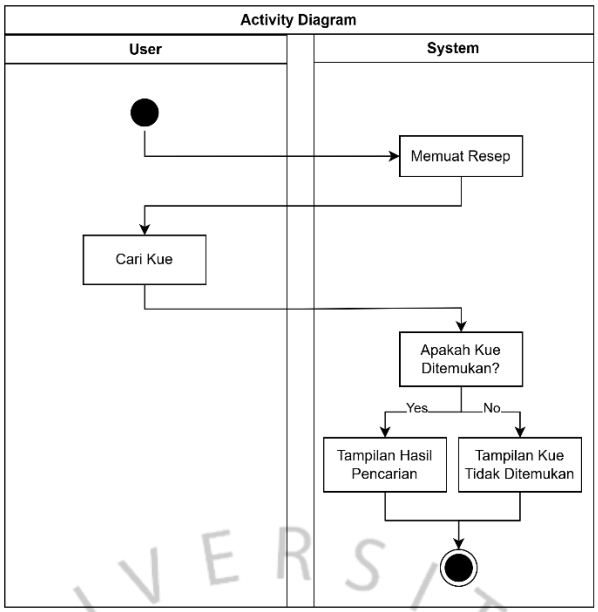
4.3.9. Activity Diagram

Activity diagram adalah gambaran alur aktivitas sistem yang akan dibangun yang berisikan urutan proses dari aplikasi yang akan dirancang oleh peneliti. Berikut adalah activity diagram aplikasi identifikasi kuliner kue indonesia



Gambar 4.16. Activity Diagram Kue Melalui Input Gambar.

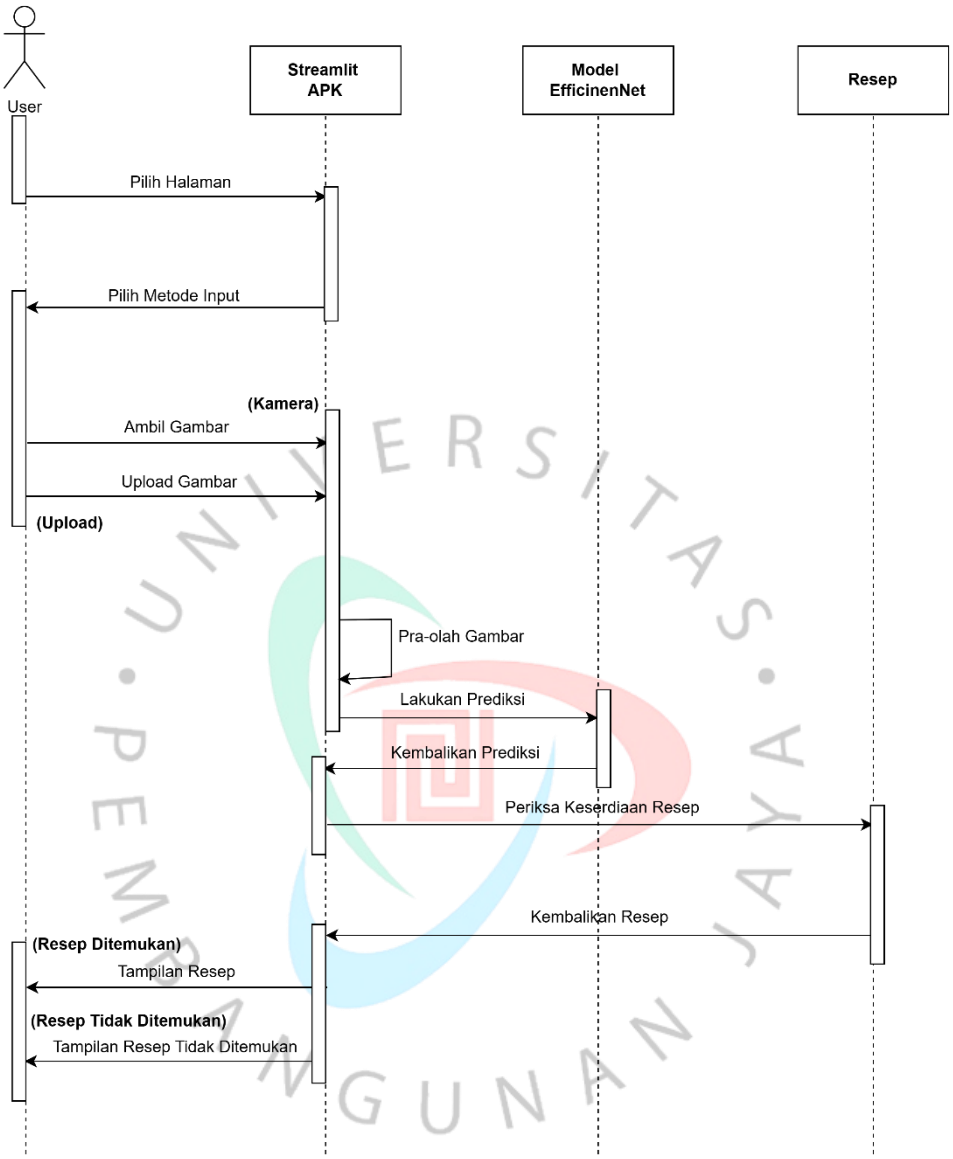
Gambar activity diagram identifikasi kue melalui input gambar untuk memulai tahap identifikasi kue dimulai dari pengguna memilih metode input, yaitu mengambil gambar dari kamera atau mengunggah gambar. Setelah itu, sistem memproses gambar melalui tahap pra-olah untuk memastikan data sesuai model. Model kemudian memprediksi jenis kue berdasarkan input, dan hasil prediksi ditampilkan kepada pengguna. Jika pengguna membutuhkan resep, sistem akan memeriksa ketersediaan resep dan menampilkannya jika tersedia. Jika tidak, pesan "Resep Tidak Ditemukan" akan ditampilkan.



Gambar 4.17. Activity Diagram Pencarian Kue Secara Manual.

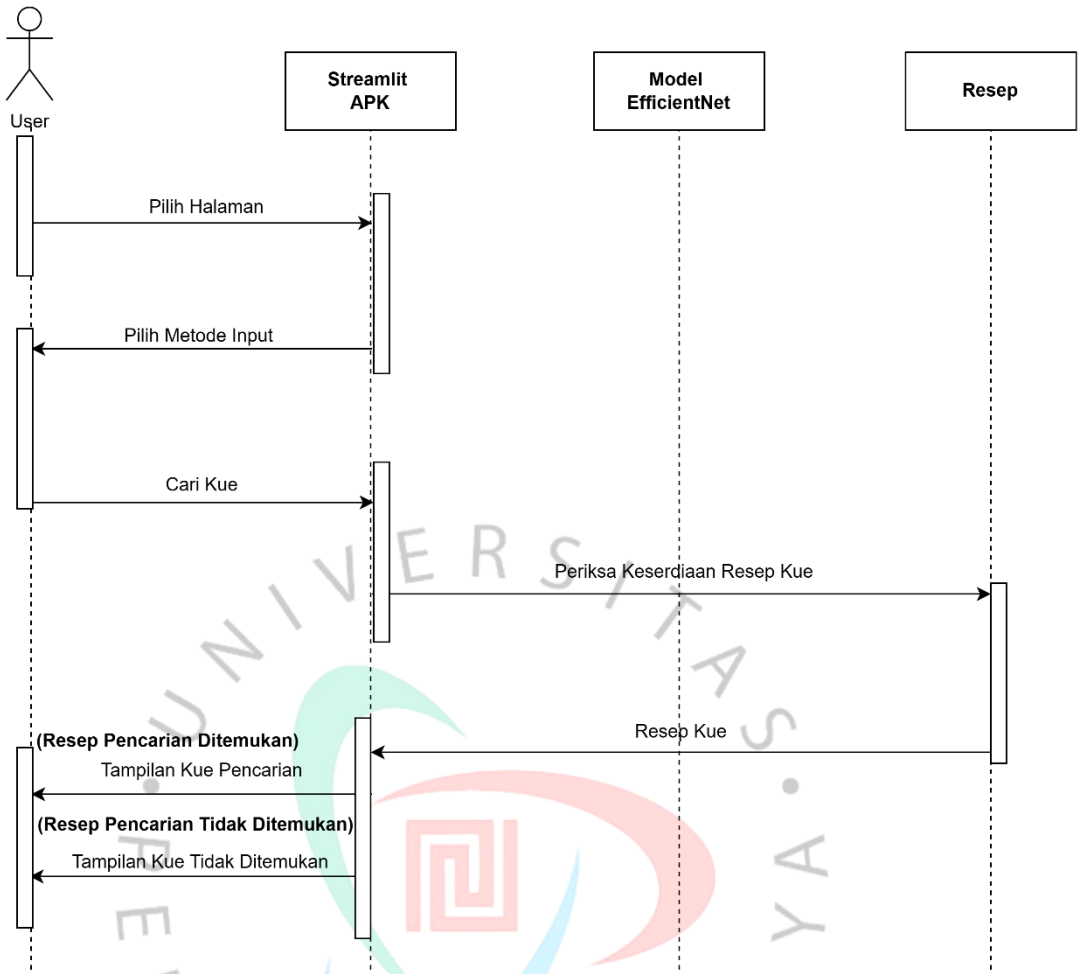
Gambar activity diagram pencarian kue secara manual di mana pengguna memasukkan nama kue melalui fitur *Cari Kue*. Sistem memuat database resep untuk memeriksa keberadaan kue. Jika kue ditemukan, sistem menampilkan informasi dan resepnya. Namun, jika tidak ditemukan, sistem akan menampilkan pesan "Kue Tidak Ditemukan." Diagram ini mempermudah pengguna untuk mencari kue tanpa menggunakan input gambar.

4.3.10. Sequence Diagram



Gambar 4.18. Sequence Diagram Proses Identifikasi Kue Melalui Input Gambar.

Diagram ini menunjukkan alur komunikasi antara Streamlit APK, model prediksi, dan database resep. Proses dimulai dari pengguna yang mengunggah atau mengambil gambar melalui Streamlit APK. Gambar tersebut diproses dan dikirimkan ke model prediksi untuk diidentifikasi jenis kuenya. Setelah prediksi selesai, hasilnya dikembalikan ke aplikasi. Jika pengguna membutuhkan resep, sistem akan mengakses database resep untuk memeriksa ketersediaan dan menampilkannya jika tersedia.



Gambar 4.19 . Sequence Diagram Pencarian Kue Secara Manual.

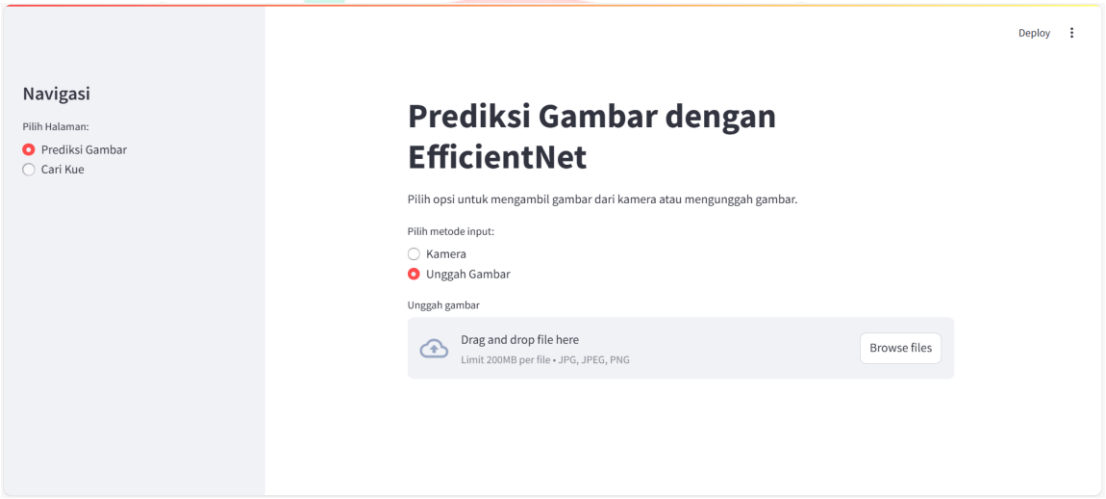
Diagram ini menggambarkan alur komunikasi saat pengguna mencari kue berdasarkan nama melalui Streamlit APK. Setelah pengguna memasukkan nama kue, aplikasi memeriksa database resep untuk mencocokkan data. Jika kue ditemukan, informasi dan resep ditampilkan kepada pengguna. Namun, jika tidak ditemukan, aplikasi menampilkan pesan bahwa kue tidak ditemukan dalam database. Diagram ini menunjukkan proses yang sederhana namun efektif untuk pencarian kue.

4.3.11. Perancangan Antar Muka

Perancangan antar muka adalah proses mendesain dan mengembangkan tampilan yang akan dibuat. Berikut desain antarmuka dari aplikasi identidikasi kuliner kue Indonesia.

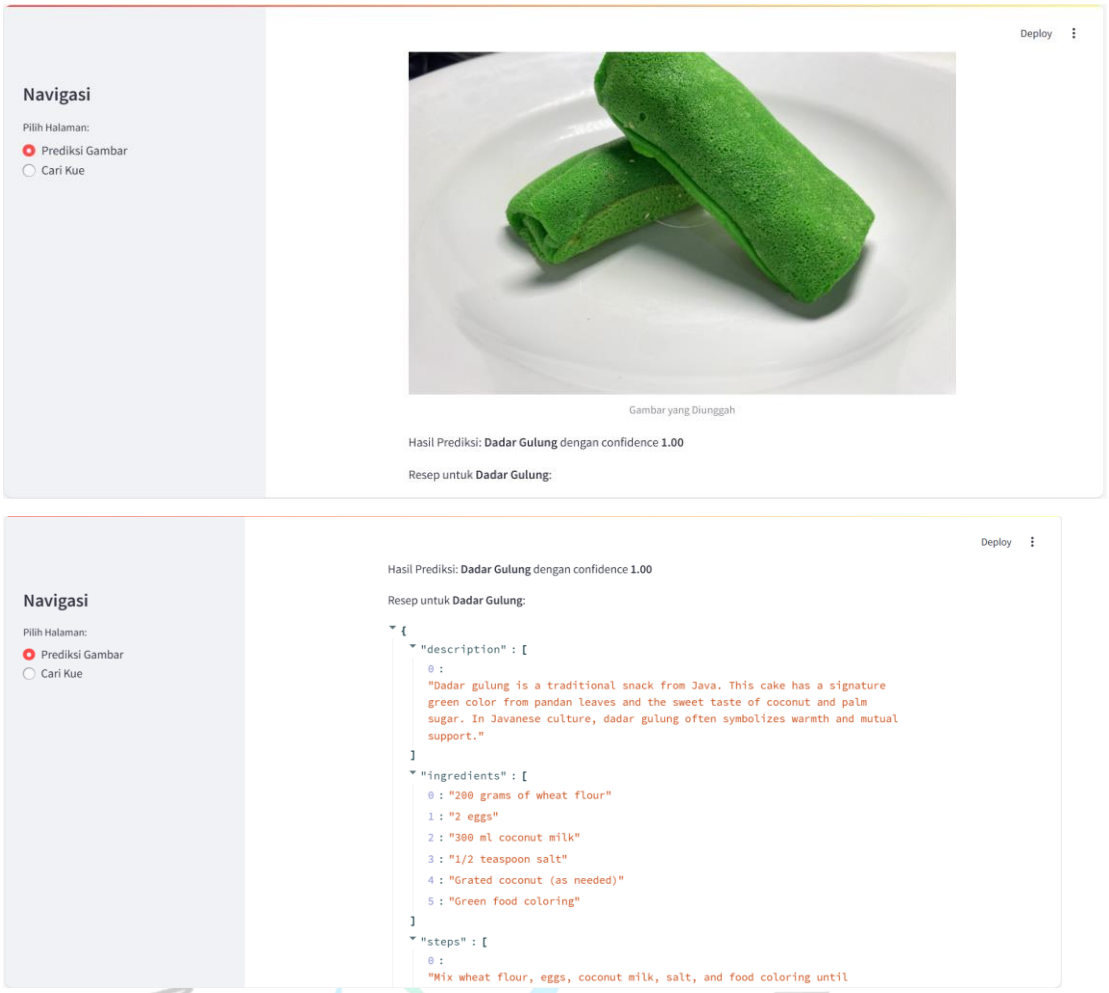
Gambar 4.20. Tampilan Awal Aplikasi Prediksi Gambar Dari Kamera

Halaman ini adalah tampilan awal fitur *Prediksi Gambar dengan EfficientNet*. Pengguna diberikan opsi untuk memilih metode input, yaitu melalui kamera atau unggah gambar.



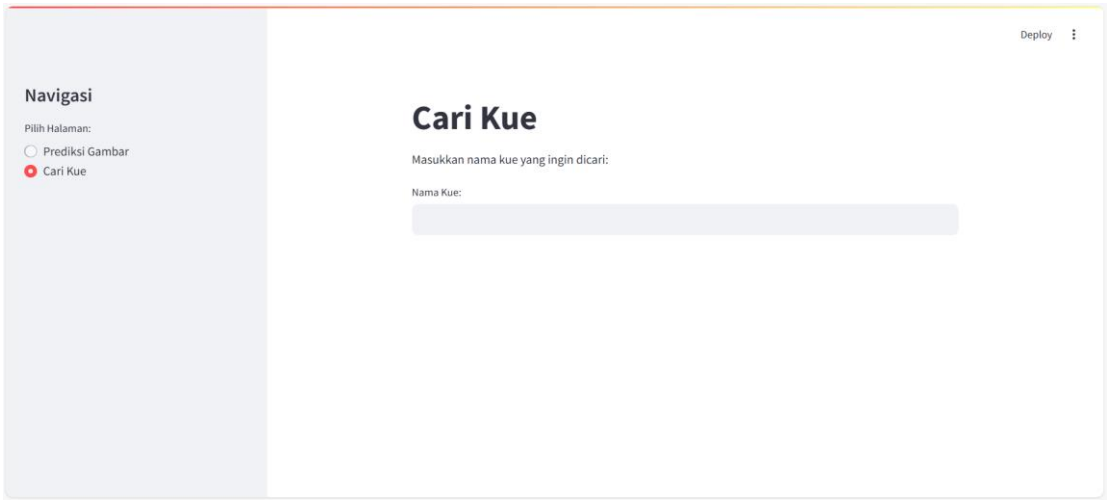
Gambar 4.21. Tampilan Opsi Metode Unggah Gambar

Halaman ini menunjukkan fitur *Prediksi Gambar dengan EfficientNet* saat pengguna memilih metode *Unggah Gambar*.



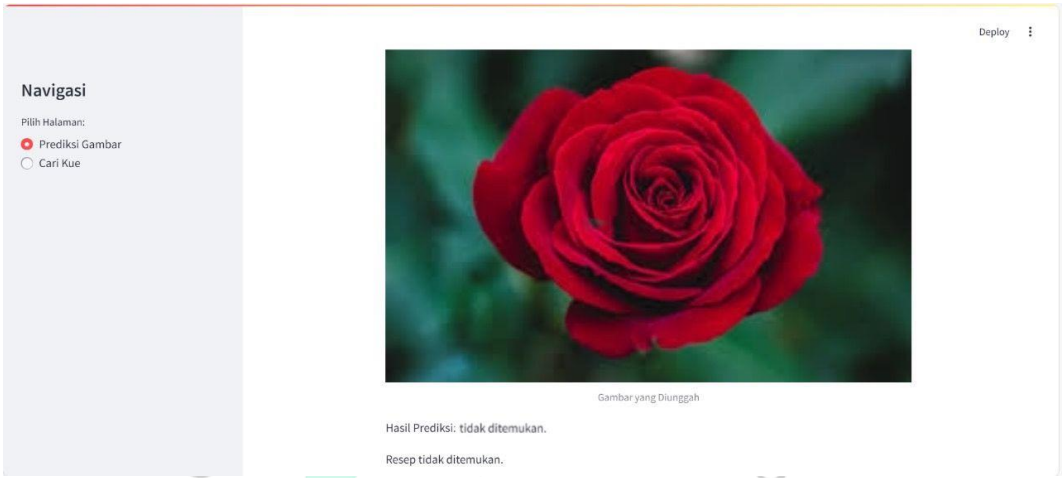
Gambar 4.22. Tampilan Hasil Identifikasi Kuliner Kue Indonesia.

Aplikasi memproses gambar tersebut untuk memprediksi jenis kue dan menampilkan hasilnya. Selain itu, aplikasi juga memberikan resep kue berdasarkan hasil prediksi, mencakup bahan dan langkah-langkah pembuatan, yang disajikan dalam format JSON untuk kejelasan informasi.



Gambar 4.23. Tampilan Halaman Cari Kue

Halaman ini adalah tampilan fitur "Cari Kue" di aplikasi. Pengguna diberikan kolom input untuk memasukkan nama kue yang ingin dicari. Desain minimalis dengan navigasi di sisi kiri memastikan kemudahan bagi pengguna dalam menjelajahi fitur aplikasi.



Gambar 4.24. Tampilan Hasil Identifikasi Jika Memasukan Bukan Jenis Kue

Dibagian ini terdapat gambar mawar merah yang diunggah oleh pengguna, namun sistem tidak dapat memberikan hasil prediksi maupun resep, seperti yang ditunjukkan oleh teks "Hasil Prediksi: tidak ditemukan" dan "Resep tidak ditemukan." Hal ini terjadi karena aplikasi dirancang khusus untuk mengenali gambar kue atau kuliner Indonesia, sehingga model prediksi yang digunakan hanya dapat mengidentifikasi makanan dan tidak mampu mengenali objek di luar kategori tersebut, seperti bunga. Akibatnya, saat pengguna mengunggah gambar mawar, sistem tidak dapat mencocokkannya dengan data yang ada dan menampilkan hasil "tidak ditemukan."

4.3.12. Perancang Pengujian

Dalam proses perancangan aplikasi, tahap pengujian menjadi langkah penting untuk memastikan apakah aplikasi yang dikembangkan telah sesuai dengan rancangan sebelumnya. Pada pengujian ini, peneliti menggunakan pengujian Black Box dan White Box.

4.3.12.1. Perancangan Pengujian Black Box

Pengujian black box dirancang untuk menguji fungsionalitas aplikasi berdasarkan spesifikasi yang telah ditentukan tanpa memperhatikan struktur internal kode program. Dalam aplikasi identifikasi kuliner kue Indonesia, pengujian ini dilakukan untuk memastikan bahwa fitur-fitur utama, seperti input gambar melalui kamera atau unggahan, proses prediksi jenis kue, dan penampilan resep, berfungsi dengan baik. Fokus pengujian mencakup validasi input, keakuratan prediksi, dan respon sistem terhadap berbagai skenario penggunaan, termasuk masukan yang tidak valid.

4.3.12.2. Perancangan Pengujian White Box

Pengujian white box dirancang untuk memeriksa logika internal, jalur kode, dan struktur algoritma aplikasi. Pada aplikasi ini, pengujian white box dilakukan untuk memastikan bahwa semua fungsi, seperti preprocessing gambar, pemrosesan model EfficientNet, dan pencarian resep di database, bekerja sesuai dengan desain. Pengujian ini mencakup evaluasi jalur kontrol, penanganan kesalahan, dan optimalisasi algoritma agar performa sistem tetap efisien dalam memproses data dan memberikan hasil yang akurat.