BAB III
METODE PENELITIAN

3.1 Metode Pengumpulan Data

Pada tahap pengumpulan data yang dilakukan untuk mendapatkan informasi terkait tujuan penelitian ini menggunakan beberapa metode yaitu:

a. Wawancara

   Wawancara dilakukan kepada petani kopi Bpk. Ali Faturrohman di Desa Harapan Jaya, Sinar 2, Kec. Way Rate Pesawaran, untuk memperoleh informasi terkait permasalahan yang dihadapi, serta data-data berupa gambar daun kopi yang terindikasi terkena penyakit. Mewawancarai petani untuk tujuan penelitian ini yaitu membuat aplikasi pendeteksi penyakit pada tanaman kopi sehingga petani kopi dapat melakukan penanganan yang lebih cepat dan efisien, mengatasi gejala awal yang terjadi pada tanaman kopi melalui aplikasi pendeteksi penyakit untuk tanamannya.

b. Observasi

   Observasi dilakukan dengan mengamati daun-daun yang terkena penyakit untuk mengetahui penyakit apa yang terjadi pada tanaman kopi tersebut sehingga dapat dicocokkan dengan data-data penyakit pada tanaman kopi tipe arabika.

c. Data

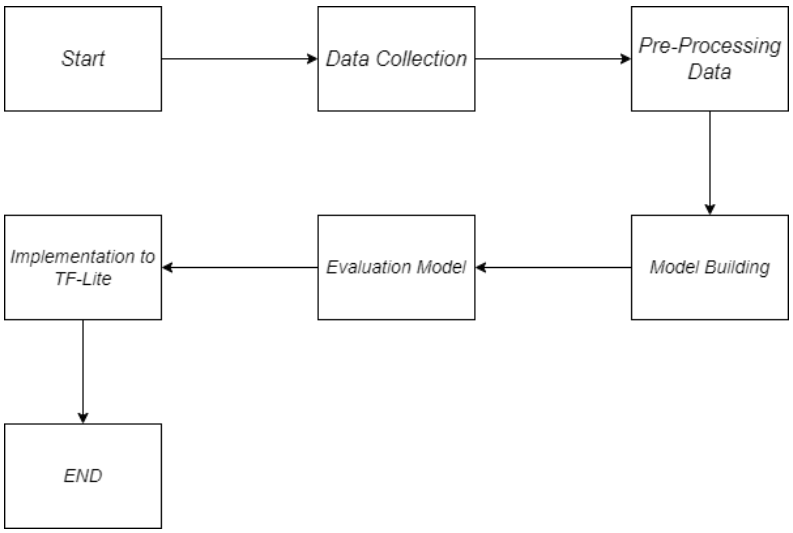
   Pengumpulan Data-data penyakit pada tanaman kopi dilakukan dengan mencari beberapa dataset terkait penyakit tanaman kopi melalui Kaggle dan mengambil beberapa sample pada tempat penelitian yang akan diolah pada proses train data di penelitian ini.

d. Studi Pustaka

   Studi Pustaka mengacu pada beberapa referensi jurnal, artikel ilmiah dan makalah terdahulu yang masih relevan dengan penelitian ini.

3.2 Metode CNN (*Convolutional Neural Network*)

Metode Convolutional Neural Network memiliki beberapa alur dalam pengerjaannya yaitu sebagai berikut:



Gambar 3.1 Alur CNN

3.2.1 *Data Collection*

Pada penelitian ini data yang dikumpulkan menggunakan dataset berupa gambar penyakit pada daun kopi yang diperoleh dari repository Kaggle dan beberapa data sampel gambar penyakit daun kopi yang diambil dari tempat penelitian. Dataset yang digunakan berupa gambar yang berjumlah 3768 gambar yang terdiri dari 167 red spider mite, 719 miner, 484 phoma, 1207 leaf rust, dan 1191 healty. Dan berikut beberapa sampel yang diambil ditempat penelitian:

Gambar 3.2 *Leaf Rust*

Gambar 3.3 *Healty*

3.2.2 *Pre-Processing Data*

Sebelum pelatihan model, kumpulan data akan diproses ke dalam format tertentu untuk meningkatkan kinerja model. Selanjutnya, data gambar dengan nilai piksel antara 0-255 akan dinormalisasi ke dalam rentang nilai 0-1 untuk memudahkan proses pembelajaran dan memudahkan model untuk konvergensi.

```
from keras.preprocessing.image import ImageDataGenerator
# Set the image size and batch size
image_size = (224, 224)
batch_size = 32
# Create an ImageDataGenerator object with data augmentation options for image preprocessing
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
#train data
train_data = data_generator.flow_from_directory(train_path,
                                                target_size = (224, 224),
                                                batch_size = 16)
#test data generator
test_data = data_generator.flow_from_directory(test_path,
                                               target_size = (224,224),
                                                batch_size = 16)
```

Gambar 3.4 Contoh *Pre-processing Data*

3.2.3 *Model Building*

Arsitektur model deep neural network memerlukan lapisan yang ideal untuk mencapai titik nilai accuracy tinggi dengan nilai loss rendah. Saat model berada di antara overfitting dan underfitting itulah yang disebut ideal (good fit), dan juga di antara overcapacity dan undercapacity.

Penelitian ini mengusulkan convolutional neural network untuk arsitektur deep neural network yang akan digunakan.

```
[ ] #CNN MODEL
    from keras.models import Sequential
    from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization

    model=Sequential()
    model.add(Conv2D(32,(3,3),activation='relu',input_shape=(224,224,3)))
    model.add(MaxPooling2D(2,2))
    model.add(BatchNormalization())
    model.add(Conv2D(32,(3,3),activation='relu'))
    model.add(MaxPooling2D(2,2))
    model.add(BatchNormalization())
    model.add(Conv2D(64,(3,3),activation='relu'))
    model.add(MaxPooling2D(2,2))
    model.add(Conv2D(64,(3,3),activation='relu'))
    model.add(MaxPooling2D(2,2))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(5,activation='softmax'))
```

Gambar 3.5 Contoh Model *building CNN*

3.2.4 *Evaluation Model*

Mengevaluasi kinerja model pada dataset uji untuk menilai tingkat akurasi dan kinerja lainnya. Setelah model dilatih, model akan memprediksi dataset testing untuk memperoleh nilai akhir dari total gambar yang berhasil diprediksi.

```
#Evaluation Model

#evaluasi data test
model.evaluate(test_data)

#evaluasi data train
model.evaluate(train_data)
```

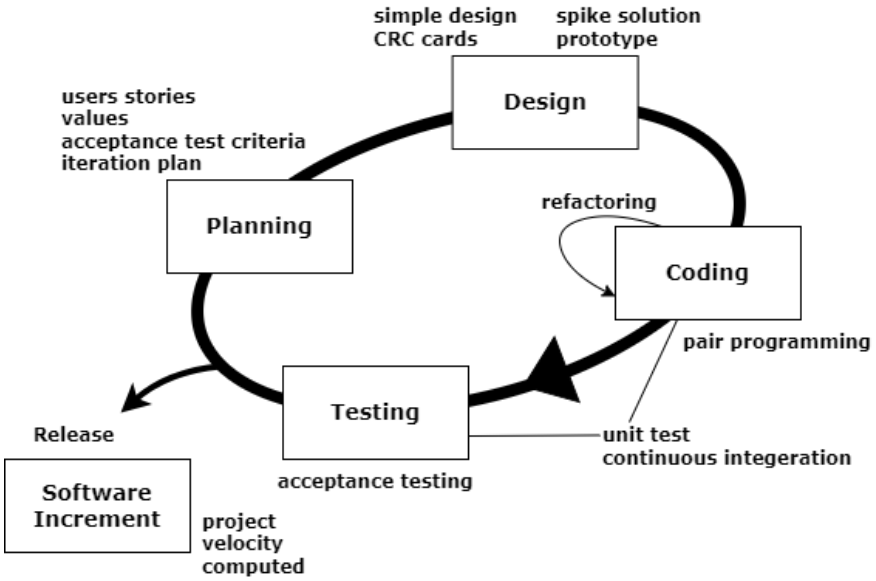
Gambar 3.6 Contoh *Evaluation Code*

3.2.5 Implementasi ke *TF-Lite*

Pada proses ini ketika dataset sudah dites, di train, hingga divalidasi langkah selanjutnya yaitu menyimpan hasil dari dataset yang sudah dilatih kedalam bentuk format Tf-Lite untuk digunakan dalam aplikasi android.

3.3 Metode Pengembangan Sistem *Extreme Programming*

Extreme Programming memiliki beberapa tahapan yang mana meliputi *planning, design, coding,* dan *testing*.



Gambar 3.7 Alur *Extreme Programming*

1 *Planning/*Perencanaan

Proses ini dimulai dengan memahami konteks aplikasi dengan mendefinisikan keluaran, termasuk fitur, fungsi, dan alur pengembangan.

2 *Design/*Perancangan

Tahap perancangan sederhana untuk membuat desain kasar aplikasi dengan alat desain Figma dan pemetaan kelas-kelas yang akan digunakan pada diagram UML.

3 *Coding/*Pengkodean

Komponen pengembangan utama XP adalah pari programming. Penggunaan Android Studio untuk pembuatan Aplikasi Deteksi penyakit dengan bahasa pemrograman *Kotlin*.

4 *Testing/*Pengujian

Tahap ini berkonsentrasi pada pengujian fitur aplikasi sehingga sesuai dengan prosedur bisnis dan tidak mengandung kesalahan.

3.4 Perancangan Sistem

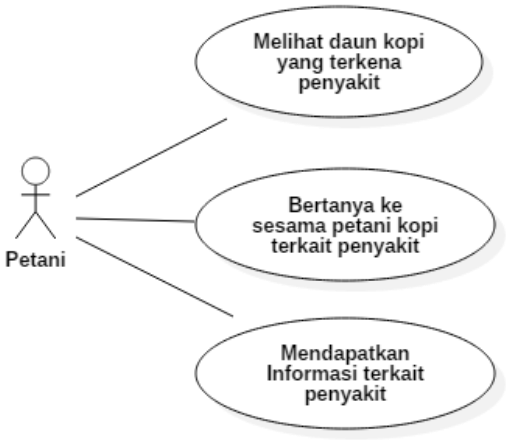
3.4.1 *Use Case Diagram*

Berikut merupakan deskripsi *Use Case Diagram* Berjalan.

Tabel 3.1 Deskripsi *Use Case Diagram* Berjalan

No	Aktor	Deskripsi
1	Petani	Petani Kopi adalah orang yang secara langsung terjun kelapangan yang bertugas untuk mengawasi dan merawat tanaman kopi. Petani mengecek tanaman kopi yang terkena penyakit melalui daun, petani bertanya ke sesame petani kopi yang lain ketika tidak tau apa penyakit tersebut, petani mendapatkan informasi penyakit (tetapi bisa jadi penyakit itu benar bisa juga salah).

Berikut merupakan bentuk *Use Case Diagram* Berjalan.



Gambar 3.8 *Use Case Diagram* Berjalan

Berikut merupakan deskripsi *Use Case Diagram* Diusulkan pada sistem aplikasi pendeteksi penyakit pada daun tanaman kopi.

Tabel 3.2 Deskripsi *Use Case Diagram* Diusulkan

No.	Aktor	Deskripsi
1	Petani	Petani mempunyai hak untuk mengakses aplikasi deteksi penyakit, melihat halaman *page home*, mens*can* daun di halaman *page scan*, melihat hasil berupa informasi penyakit dan cara penanganan, menyimpan hasil informasi penyakit, dan melihat halaman *about app.*

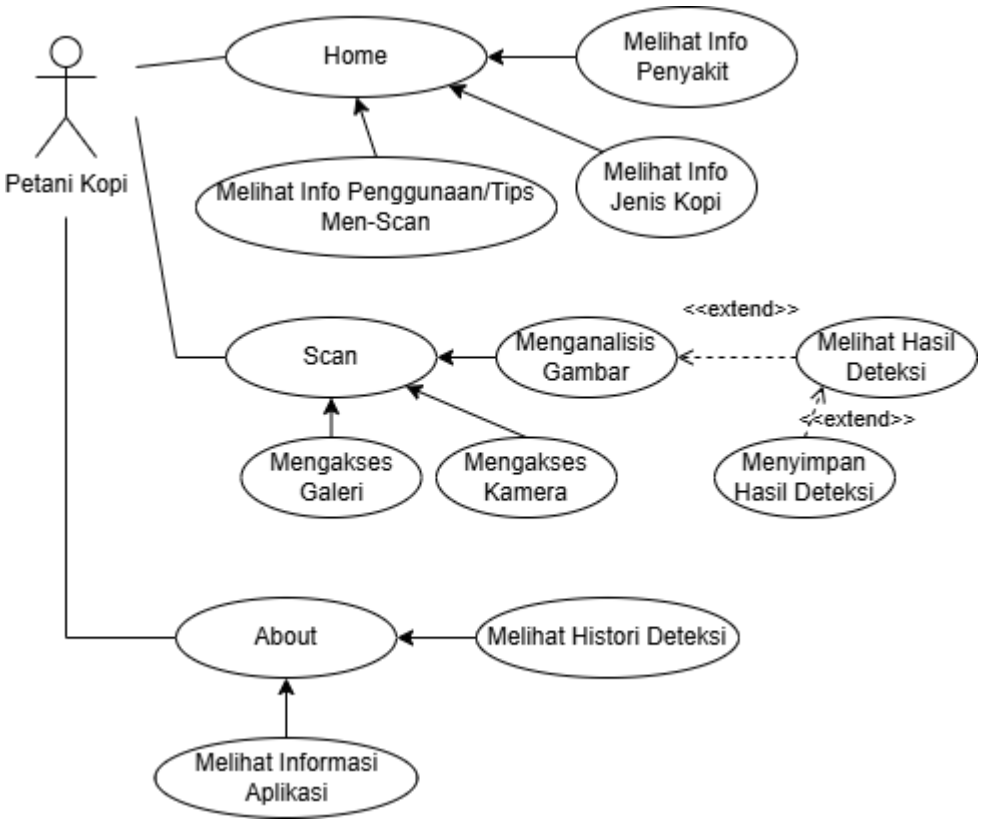
Berikut ini merupakan pendefinisian *Use Case*, yaitu sebagai berikut:

Tabel 3.3 Deskripsi *UseCase*

No	*Use Case*	Deskripsi
1.	Masuk Aplikasi	Masuk aplikasi merupakan proses awal untuk mengakses aplikasi deteksi penyakit tanaman kopi.
2.	*Home*	Merupakan proses menampilkan *UI* yang ada pada aplikasi deteksi, seperti informasi cara penggunaan aplikasi, informasi penyakit, dan informasi jenis-jenis kopi.
3.	Melihat Tampilan Cara Penggunaan/Tips Menggunakan Scan	Merupakan proses menampilkan informasi terkait cara penggunan/tips men-*scan* pada aplikasi deteksi penyakit tanaman kopi.
4.	Melihat Tampilan Jenis-Jenis Kopi	Merupakan proses menampilkan informasi jenis-jenis tipe kopi serta manfaatnya untuk kesehatan.
5.	Melihat Tampilan Informasi Penyakit	Merupakan proses menampilkan informasi berupa nama penyakit yang dialami tanaman kopi, penyebab, dan penanganannya.
6.	Scan	Merupakan proses menampilkan halaman pada scan di aplikasi.
7.	Mengakses Kamera	Merupakan proses saat aplikasi mengakses kamera untuk mens*can*.

8.	Mengakses Galeri	Merupakan proses saat aplikasi mengakses galeri untuk memilih gambar yang telah disiapkan di galeri.
9.	Menganalisis Gambar	Merupakan proses ketika aplikasi menganalisis gambar untuk menampilkan hasil deteksi penyakit tanaman.
10.	Melihat Hasil Deteksi	Merupakan proses menampilkan hasil dari deteksi yang telah dilakukan.
11.	Menyimpan Hasil Deteksi	Merupakan proses menyimpan hasil deteksi ke dalam database aplikasi.
12.	About	Merupakan proses menampilkan *UI* dari tampilan menu *about*.
13.	Melihat Informasi Aplikasi	Merupakan proses menampilkan informasi dari aplikasi CoffeeHealth.
14.	Melihat Histori Deteksi	Merupakan Proses menampilkan hasil dari deteksi yang telah disimpan.

Berikut merupakan bentuk *Use Case Diagram* Diusulkan pada sistem pendeteksi penyakit.



Gambar 3.9 *Use Case Diagram* Diusulkan Sistem Deteksi Penyakit

Berikut ini merupakan pendefinisian *Use Case Scenario* yaitu sebagai berikut:

Tabel 3.4 *Usecase Scenario* Tampilan *Home*

Aksi Petani	Reaksi Sistem
Scenario Normal	
1. Membuka aplikasi *CoffeeHealth*	
	2. Masuk ke aplikasi *CoffeeHealth*
	3. Menampilkan Menu pada page *Home*
4. Mengklik cara penggunaan aplikasi	
	5. Menampilkan informasi penggunaan aplikasi.
6. Mengklik info jenis kopi	
	7. Menampilkan informasi jenis kopi beserta manfaat dari kopi.
8. Mengklik info penyakit	
	9. Menampilkan informasi jenis penyakit kopi.

Tabel 3.5 *Usecase Scenario* Tampilan *Scan*

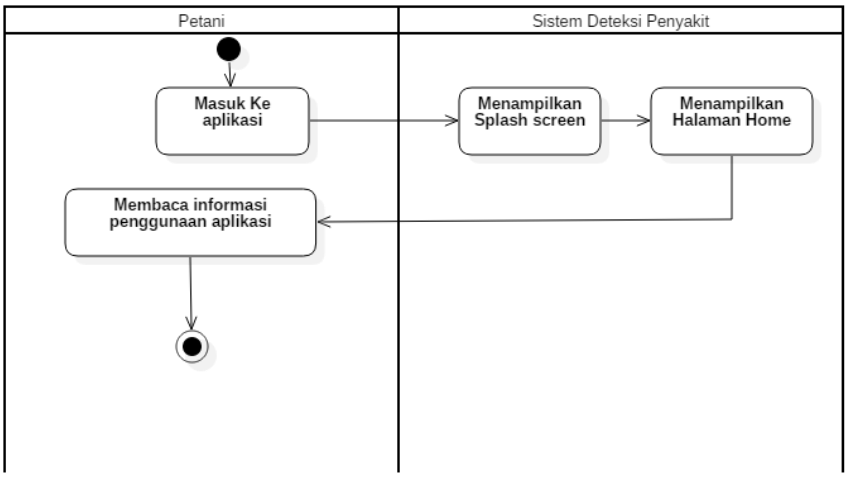
Aksi Petani	Reaksi Sistem
Scenario Normal	
1. Mengklik Menu Scan	
	2. Menampilkan menu *scan*
3. Mengklik tombol kamera	
	4. Akses ke kamera
	5. Mendapatkan hasil dari tangkapan kamera.
	6. Menampilkan hasil tangkapan kamera di halaman *scan.*
7. Mengklik tombol *Analyze*	
	8. Memproses deteksi penyakit kopi
	9. Menampilkan hasil deteksi penyakit.
10. Mengklik tombol *save*	
	11. Menyimpan hasil deteksi
12. Mengklik tombol *Gallery*	
	13. Akses ke Galeri

Tabel 3.6 *Usecase Scenario* Tampilan *About*

Aksi Petani	Reaksi Sistem
Scenario Normal	
1. Mengklik menu *About.*	
	2. Menampilkan menu *about*
3. Mengklik tombol histori	
	4. Menampilkan halaman hasil deteksi yang telah tersimpan
5. Mengklik tombol tentang aplikasi	
	6. Menampilkan informasi tentang aplikasi

3.4.2 *Activity Diagram* Halaman *Home*

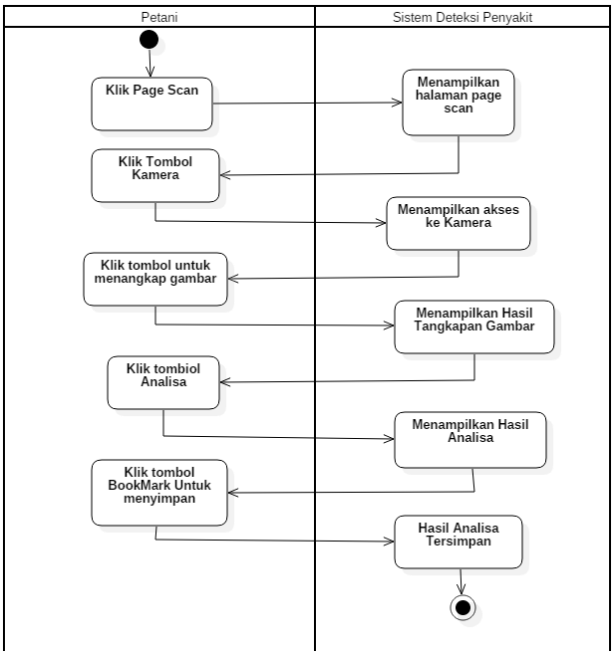
Actor petani pada tahap awal membuka aplikasi dan aplikasi menampilkan halaman utama/*home.*



Gambar 3.10 *Activity Diagram* Halaman Utama

3.4.3 *Activity Diagram* Halaman *Scan*

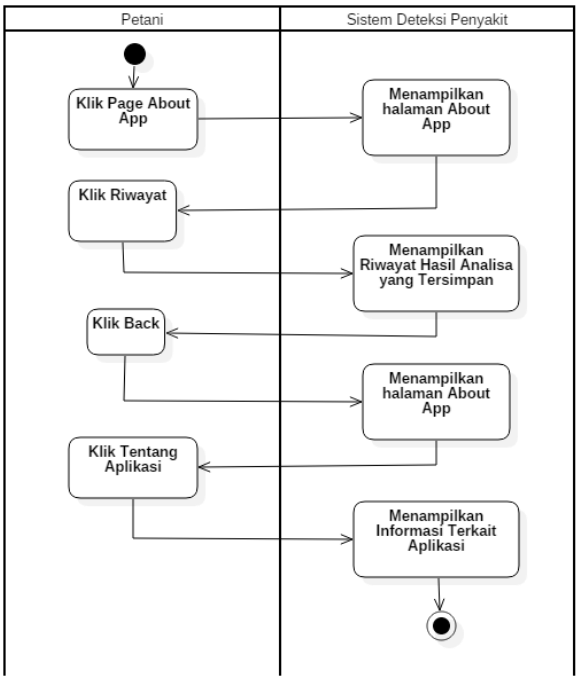
Actor petani mengakses halaman *scan* untuk mengambil gambar daun yang terkena penyakit untuk dideteksi, aplikasi akan menampilkan hasil dari analisa, dan juga petani bisa menyimpan hasil analisa dengan mengklik tombol *bookmark.*



Gambar 3.11 *Activity Diagram* Halaman *Scan*

3.4.4 *Activity Diagram* Halaman *About App*

Actor petani dapat mengakses halaman *About App* untuk melihat informasi seperti riwayat untuk melihat riwayat hasil analisa yang tersimpan, dan Tentang Aplikasi untuk melihat informasi terkait aplikasi deteksi penyakit.

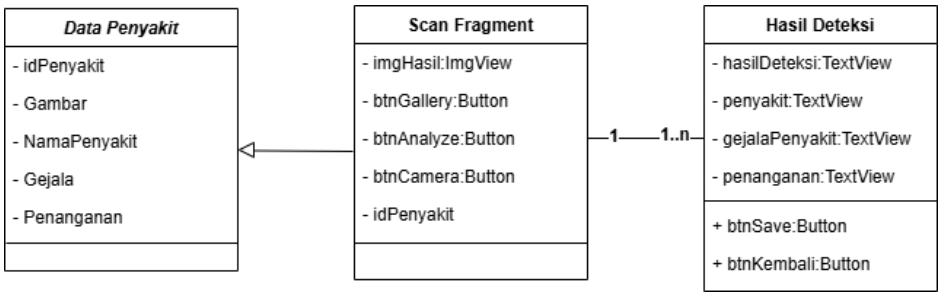


Gambar 3.12 *Activity Diagram* Halaman *About*

3.4.5 *Class Diagram* Sistem *Aplikasi* Deteksi Penyakit

Berikut ini penjelasan *Class Diagram* pada aplikasi deteksi penyakit tanaman kopi.

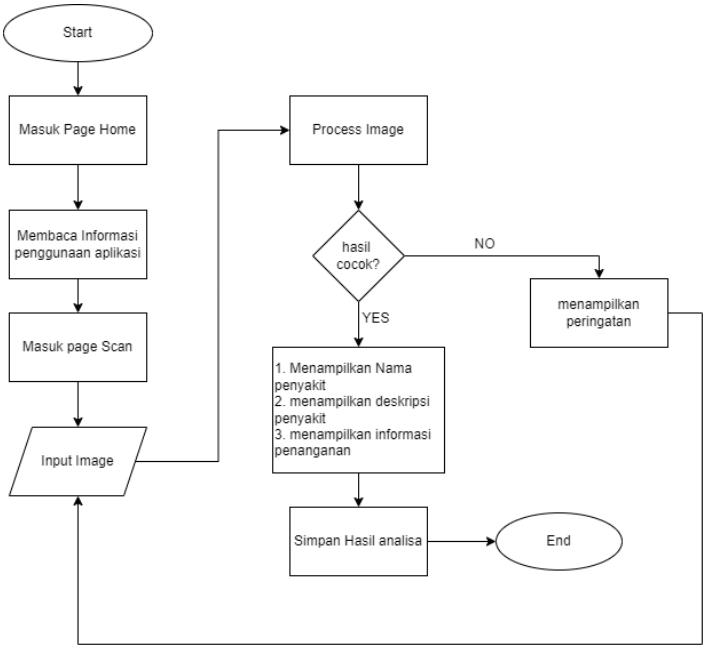
1. *Class* Data Penyakit merupakan sebuah *Parent Class.*
2. *Class* Scan Fragment merupakan *Child Class*, dan memiliki hubungan *One to Many* dengan Hasil Deteksi.



Gambar 3.13 *Class Diagram* Sistem Deteksi Penyakit

3.4.6 *Flowchart* Sistem Deteksi Penyakit

Berikut ini merupakan *flowchart* dari sistem aplikasi pendeteksi penyakit pada tanaman.



Gambar 3.14 *Flowchart* Sistem Aplikasi Deteksi Penyakit

3.5 *Design Aplikasi*

Berikut ini merupakan gambaran mentah desain aplikasi sistem pendeteksi penyakit pada tanaman pada penelitian ini.

3.5.1 Tampilan *Splash Screen*

Merupakan tampilan awal membuka aplikasi

Gambar 3.15 *Splash Screen*

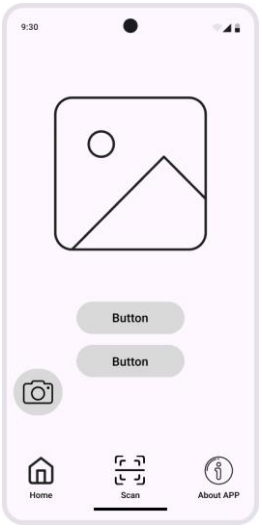
3.5.2 Tampilan Halaman *Home*

Desain tampilan halaman utama

Gambar 3.16 *Page Home*

3.5.3 Tampilan Halaman *Scan*

Desain pada halaman *page scan*



Gambar 3.17 *Page Scan*

3.5.4 Tampilan Halaman Hasil Analisis

Desain halaman hasil analisis penyakit

Gambar 3.18 Tampilan Hasil Analisis

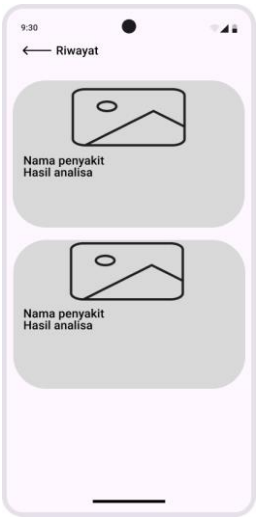
3.5.5 Tampilan Halaman *About App*

Desain pada halaman *About App*

Gambar 3.19 *Page About App*

3.5.6 Tampilan Halaman Riwayat

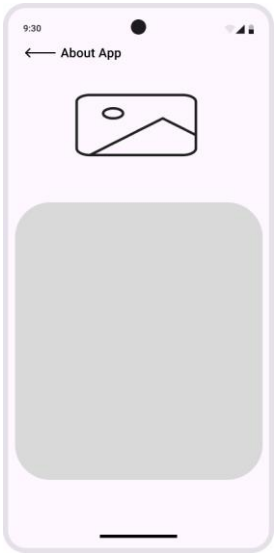
Desain pada halaman riwayat yang tersimpan



Gambar 3.20 Halaman Riwayat

3.5.7 Tampilan Halaman Tentang Aplikasi

Desain tampilan pada halaman Tentang Aplikasi



Gambar 3.21 Halaman Tentang Aplikasi