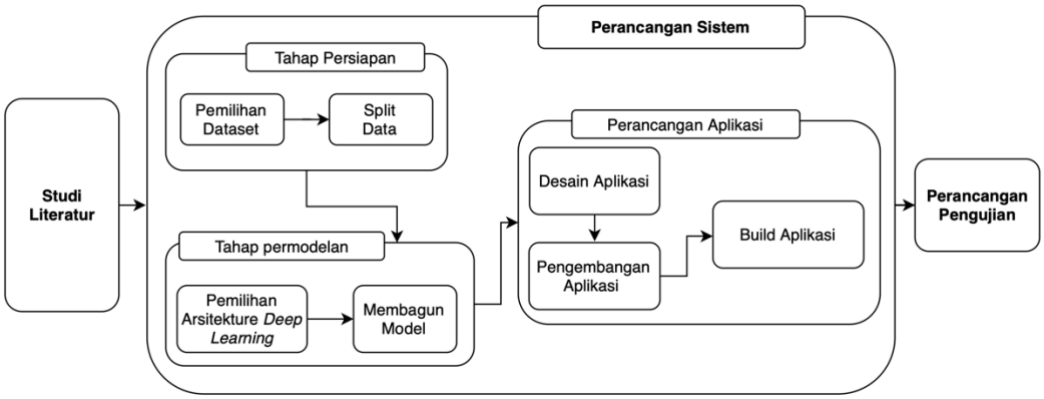
BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Pada penelitian ini, terdapat beberapa tahapan yang dilakukan penulis dalam meneliti serta merancang system degan tujuan untuk menjawab masalah yang telah penulis paparkan sebelumnya. Tahapan-tahapan tersebut meliputi studi literatur, perancangan sistem dan perancangan pengujian. Berikut metode atau alur penelitian yang dilakukan penulis yang dapat dilihat pada Gambar 3.1



Gambar 3.1 Metode atau Alur Penelitian

3.2 Studi Literature

Sebelum masuk ke perancangan modul, penulis mempelajari penelitian-penelitian terdahulu yang berkaitan dengan klasifikasi, *deep learning, dan* model *Convolution Neural Network (CNN)* yang mendukung penelitian ini. Dalam menganalisis penelitian-penelitian terdahulu, penulis dapat menentukan model dan metode yang akan digunakan dalam klasifikasi penyakit tanaman pisang. Penulis juga melakukan pencarian mengenai penerapan-penerapan model yang telah dibuat kedalam aplikasi melalui *website,* ini bertujuan bertujuan agar memberikan referensi dan informasi yang cukup kepada penulis dalam penelitian ini.

3.3 Perancangan Sistem

Perancangan sistem meliputi tahap persiapan, tahap permodelan, dan tahap perancangan aplikasi.

3.3.1 Tahap persiapan

Pada tahap persiapan, langkah pertama adalah pemilihan dataset yang akan digunakan dalam penelitian ini. Setelah dataset dipilih, langkah selanjutnya adalah membagi data tersebut menjadi tiga bagian, yaitu data training, data validasi, dan data uji.

3.3.1.1 Pemilihan Dataset

Dataset yang digunakan dalam penelitian ini berasal dari Harvard Dataverse, yaitu *"The Nelson Mandela African Institution of Science and Technology Bananas Dataset"* [15]. Dataset ini terdiri tiga kategori yaitu folder HEALTHY yang berisi gambar daun pisang yang sehat, folder BLACK SIGATOKA_1 dan BLACK SIGATOKA_2 yang berisi gambar daun dan batang pisang yang terinfeksi Black Sigatoka, serta folder FUSARIUM WILT_1 dan FUSARIUM WILT_2 yang berisi gambar daun dan batang pisang yang terinfeksi Fusarium Wilt Race 1.

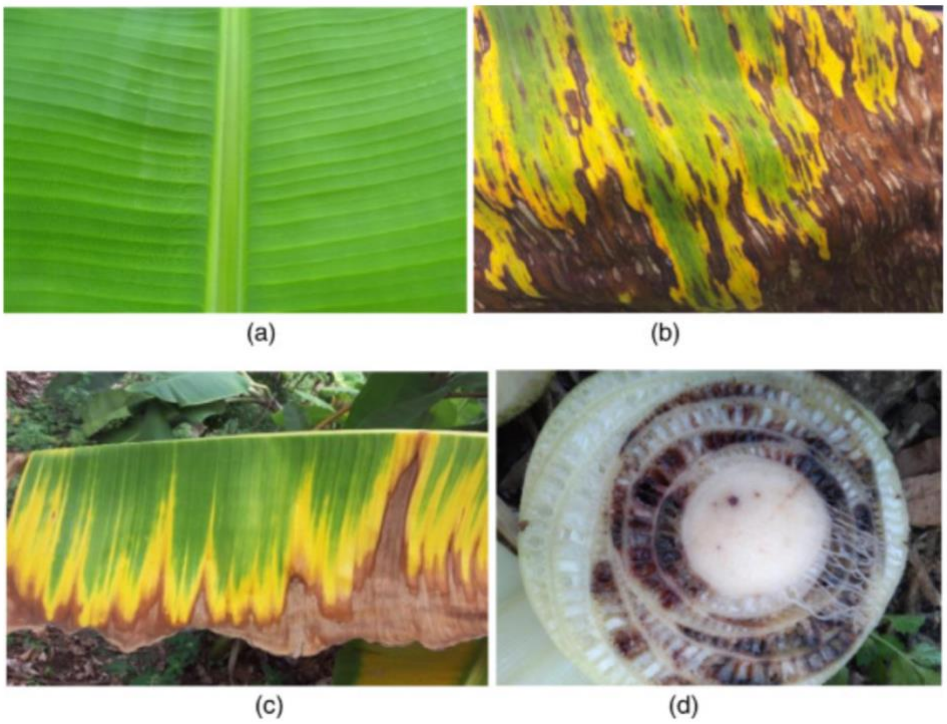
Pemilihan dataset ini mengacu pada penelitian sebelumnya oleh Sophia L. Sanga et al. (2020) [8], yang menggunakan dataset yang sama dalam pengembangan model berbasis *transfer learning*. Kesamaan dataset ini dapat dipastikan, karena peneliti dalam penelitian terdahulu juga terlibat dalam proses pengumpulan data yang digunakan dalam penelitian ini. Selain itu, lokasi pengumpulan dataset dalam penelitian ini juga sama dengan yang digunakan dalam penelitian tersebut. Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi performa model yang dilatih dari awal (tanpa *transfer learning*) dengan dataset yang sama, untuk menentukan apakah pendekatan ini dapat menghasilkan hasil yang lebih baik atau sebaliknya. Rincian lengkap mengenai dataset dan distribusinya disajikan pada Tabel 3.1.

Tabel 3.1 Pemilihan Dataset

Kelas	Jumlah gambar

Layu Fusarium	4697
Sigatoka Hitam	5767
Daun sehat	5628

Berikut merupakan contoh gambar daun sehat, daun yang terkena Sigatoka hitam, serta daun dan batang yang terkena penyakit Fusarium.



Gambar 3.2 Contoh gambar pisang: (a) sehat (b) Black Sigatoka (c) Fusarium Wilt Race 1 pada daun (d) Fusarium Wilt Race 1 pada batang

Gambar citra daun dan batang pisang tersebut dikumpulkan oleh NM-AIST dan IITA di Tanzania Utara menggunakan aplikasi Open Data Kit (ODK) bernama AdSurv pada ponsel Samsung Galaxy 01 (13 MP). Pengambilan gambar dilakukan secara acak di lima wilayah utama (Kagera, Arusha, Dar es Salaam, Kilimanjaro, dan Mbeya) berdasarkan produksi pisang dan prevalensi penyakit. Proses ini melibatkan petani, peneliti, ahli patologi tanaman, petugas penyuluhan dan berlangsung selama enam bulan (Februari–Juli 2021).

3.3.1.2 Pembagian Data

Langkah selanjutnya adalah membagi data menjadi tiga bagian utama, yaitu data training, data validasi, dan data uji, untuk memastikan model dapat dilatih, dievaluasi, dan diuji secara efektif.

- Data Training: Digunakan untuk melatih model agar mengenali pola, hubungan, dan fitur dalam data melalui iterasi dan optimasi.
- Data Validasi: Berfungsi mengevaluasi kinerja model selama pelatihan untuk menyempurnakan hyperparameter, mencegah *overfitting*, dan underfitting.
- Data Uji: Digunakan untuk evaluasi akhir model pada data baru yang tidak pernah terlihat sebelumnya, memberikan gambaran kinerja di dunia nyata.

Pembagian data dilakukan secara manual dengan rasio 70:20:10, di mana 70% digunakan untuk data training, 20% untuk data validasi, dan 10% untuk data uji. Rasio ini didasarkan pada penelitian sebelumnya yang dilakukan oleh Sophia L. Sanga et al. (2020) [8], yang menunjukkan bahwa penggunaan rasio tersebut dapat menghasilkan model dengan akurasi tinggi, bahkan mendekati 100% pada beberapa model. Berikut merupakan pembagian data training, data validasi dan data uji, dapat dilihat pada Tabel 3.2.

Tabel 3.2 Pembagian Data Training, Validasi dan Uji

Dataset Kelas Tanaman Pisang	**Data Training (70%)**	**Data Validasi (20%)**	**Data Uji (10%)**	**Total per Kelas**
Layu Fusarium	3288	939	470	4697
Sigatoka Hitam	4037	1153	577	5767

Daun Sehat	3939	1126	563	5628

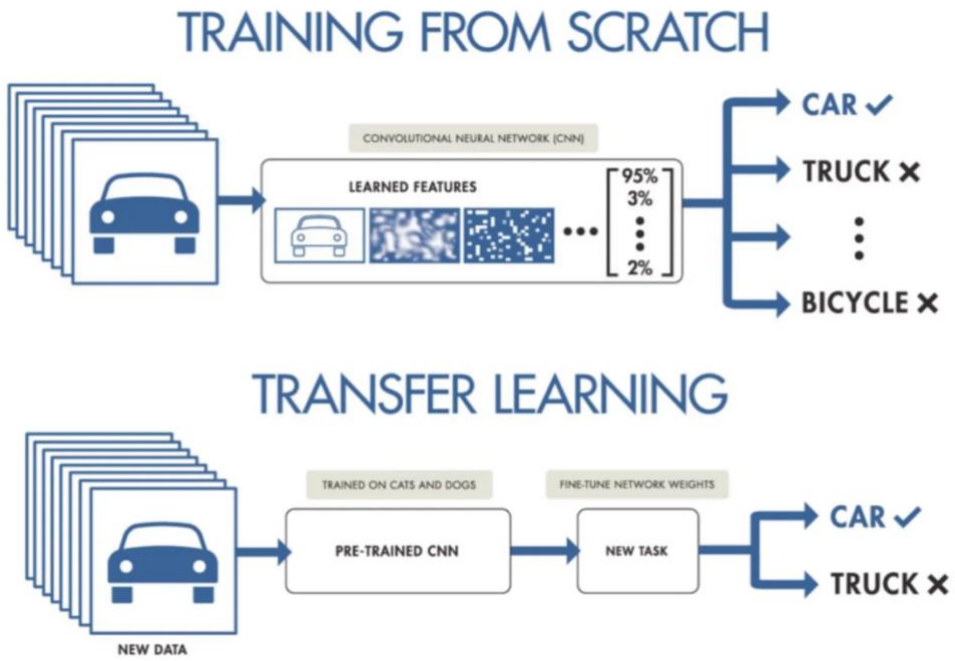
Setelah pembagian dataset selesai, langkah selanjutnya adalah melakukan augmentasi data pada data pelatihan. Augmentasi data ini dilakukan mengikuti metode yang digunakan dalam penelitian sebelumnya oleh Sophia L. Sanga et al. (2020) [8]. Penelitian tersebut menerapkan teknik augmentasi data pada data pelatihan dengan berbagai teknik, termasuk horizontal flipping, zoom range (0.2), rotating-right-bottom, rotating-left-bottom, shear range (0.1), cropping, dan rescaling (1/255).

3.3.2 Tahap Permodelan

Pada tahap pemodelan, penulis memulai dengan pemilihan arsitektur *deep learning* yang akan digunakan, kemudian dilanjutkan dengan proses pelatihan dan validasi model.

3.3.2.1 Pemilihan Arsitektur *Deep learning*

Dalam membangun model dari awal (*training from scratch*), terdapat berbagai jenis arsitektur *deep learning* yang dapat digunakan, seperti *Convolutional Neural Network (CNN), Recurrent Neural Network (RNN)*, serta arsitektur lainnya. Model yang dilatih dari awal tidak memanfaatkan bobot yang telah tersedia dari model *pre-trained*, melainkan memulai pelatihan dengan parameter yang diinisialisasi secara acak. Gambar 3.3 menunjukkan visualisasi perbandingan arsitektur model transfer learning, yang menggunakan bobot dari model yang telah dilatih sebelumnya, dengan model *training from scratch* yang sepenuhnya dimulai dari nol.



Gambar 3.3 Visualisasi Model *Transfer Learning* dan Model *Train from Scratch*

Pada penelitian ini, penulis memilih untuk menggunakan model *CNN*. *CNN* dipilih karena keunggulannya dalam mengenali pola visual secara efisien melalui proses konvolusi. Selain itu, *CNN* dirancang untuk mengekstraksi fitur-fitur lokal pada gambar, seperti tepi, tekstur, dan pola-pola kompleks lainnya, melalui lapisan-lapisan konvolusi dan pooling. Tidak seperti arsitektur lain, seperti *RNN* yang lebih cocok untuk data berurutan, *CNN* sangat ideal untuk tugas berbasis gambar. Oleh karena itu, *CNN* menjadi pilihan yang optimal untuk tugas pengenalan dan klasifikasi gambar.

3.3.2.2 Membangun Model

Model *CNN* dibangun dengan beberapa lapisan utama, yaitu lapisan konvolusi, pooling, normalisasi (batch normalization), dropout, dan lapisan fully connected.

Table 3.3 Lapisan Utama Model *CNN*

Lapisan	Deskripsi

Lapisan Konvolusi	-
Conv2D	Digunakan untuk mengekstraksi fitur dari gambar.
BatchNormalization	Membantu stabilisasi proses pelatihan dengan menormalisasi output dari lapisan sebelumnya.
MaxPooling2D	Mengurangi dimensi fitur untuk menghindari *overfitting* dan mengurangi jumlah parameter.
Dropout	Digunakan untuk mencegah *overfitting* dengan mengabaikan sebagian neuron secara acak.
Lapisan Fully Connected	-
Dense (256 neuron)	Berfungsi untuk menggabungkan semua fitur dari lapisan sebelumnya.
Dense (3 neuron, softmax)	Lapisan output untuk memprediksi tiga kelas.

Setiap lapisan-lapisan ini memiliki peran penting dalam proses ekstraksi fitur dari data masukan dan klasifikasi akhir. Penjelasan lebih rinci mengenai fungsi dan konfigurasi dari masing-masing lapisan dapat dilihat pada Tabel 3.4.

Table 3.4 Konfigurasi Lapisan Model *CNN*

lapisan	Parameter	Deskripsi
Input Layer	Dimensi input: (128, 128, 3)	Lapisan input untuk menerima gambar berukuran 128x128x3.

Conv2D	32 filter, kernel 3x3, aktivasi ReLU	Lapisan konvolusi pertama untuk mengekstraksi fitur awal dari gambar.
BatchNormalization	-	Menormalisasi output lapisan sebelumnya agar pelatihan lebih stabil.
MaxPooling2D	Pool size: 2x2	Mengurangi dimensi spasial fitur.
Dropout	Dropout rate: 25%	Mengurangi *overfitting* dengan mengabaikan 25% neuron secara acak.
Conv2D	64 filter, kernel 3x3, aktivasi ReLU	Lapisan konvolusi kedua untuk mengekstraksi fitur lebih kompleks.
BatchNormalization	-	Menormalisasi output lapisan sebelumnya.
MaxPooling2D	Pool size: 2x2	Mengurangi dimensi spasial fitur lebih lanjut.
Dropout	Dropout rate: 30%	Mengurangi *overfitting* dengan mengabaikan 30% neuron secara acak.
Conv2D	128 filter, kernel 3x3, aktivasi ReLU	Lapisan konvolusi ketiga untuk fitur tingkat lebih tinggi.
BatchNormalization	-	Menormalisasi output lapisan sebelumnya.
MaxPooling2D	Pool size: 2x2	Mengurangi dimensi fitur lebih lanjut.
Dropout	Dropout rate: 40%	Mengurangi *overfitting* dengan mengabaikan 40% neuron secara acak.

Conv2D	256 filter, kernel 3x3, aktivasi ReLU	Lapisan konvolusi keempat untuk fitur paling kompleks.
BatchNormalization	-	Menormalisasi output lapisan sebelumnya.
MaxPooling2D	Pool size: 2x2	Mengurangi dimensi spasial lebih jauh lagi.
Dropout	Dropout rate: 40%	Mengurangi *overfitting* dengan mengabaikan 40% neuron secara acak.
Flatten	-	Mengubah matriks hasil konvolusi menjadi vektor satu dimensi untuk diinputkan ke lapisan berikutnya.
Dense	256 neuron, aktivasi ReLU	Lapisan fully connected untuk menggabungkan fitur dari lapisan sebelumnya.
BatchNormalization	-	Menormalisasi output sebelum ke lapisan berikutnya.
Dropout	Dropout rate: 50%	Mengurangi *overfitting* dengan mengabaikan 50% neuron secara acak.
Output Dense Layer	3 neuron, aktivasi softmax	Lapisan output untuk memprediksi kelas target.

Selain arsitektur model, terdapat beberapa hyperparameter yang digunakan untuk mengatur parameter pelatihan, seperti ukuran batch, learning rate, jumlah epoch, dan pengaturan pada lapisan-lapisan tertentu.

Berikut adalah Tabel 8, yang menunjukkan hyperparameter yang diterapkan beserta fungsinya dalam penelitian ini.

Tabel 3.5 Penggunaan Hyperparameter

Hyperparameter	**Value**	**Fungsi dan Alasan**
Optimizer	adam	Berfungsi untuk menentukan nilai minimum lokal dari suatu fungsi, Adam menggabungkan metode *RMSprop* dan *Stochastic Gradient Descent* dengan momentum. Alasan menggunakan optimizer adam didasarkan pada kenerja yang telah ditunjukan pada penelitian terdahulu.
Learning Rate	0.01	Berfungsi untuk menghitung perubahan bobot selama proses pelatihan. semakin tinggi nilai learning rate, pelatihan akan berlangsung lebih cepat, namun akurasi dapat menurun, dan sebaliknya. Alasan penggunaan nilai 0.00001 dalam penelitian ini untuk mendapatkan ketelitian yang lebih baik.
Regularizer L2	0.001	Setiap lapisan konvolusi menggunakan regularisasi L2 untuk mencegah *overfitting* dengan menambahkan penalti terhadap bobot besar. Alasan penggunaan nilai 0.001 dalam penelitian ini didasarkan pada beberapa kali

		percobaan, dimana nilai tersebut menghasilkan model dengan performa yang baik.
Epoch	15	Memproses seluruh dataset melalui pelatihan jaringan saraf dalam satu putaran. Penggunaan 15 epoch dipilih karena dikarenakan percobaan yang dilakukan penulis, hanya menggunakan 11 epoch saja sudah mendapatkan akurasi yang bagus.

Selain itu, penulis juga menambahkan custom callback berupa EarlyStopping. Callback ini digunakan untuk mencegah terjadinya *overfitting* selama proses pelatihan model. Parameter yang digunakan untuk callback ini dapat dilihat pada Tabel 9.

Tabel 3.6 Penggunaan Parameter *EarlyStopping*

Parameter	**Value**	**Fungsi**
Monitor	Val_Los	Untuk memonitor *validation loss* saat proses *training* model
Mode	Min	Menghentikan proses *training* ketika nilai *validation loss* tidak menunjukkan penurunan lagi
Patience	3	Pelatihan akan dihentikan apabila validation loss tidak menunjukkan

		peningkatan atau penurunan selama 3 epoch berturut-turut

3.3.3 Perancangan Aplikasi

Pada tahap perancangan aplikasi, proses dimulai dengan desain aplikasi, diikuti dengan pengembangan aplikasi, dan diakhiri dengan pembuatan file instalasi (build aplikasi)

3.3.3.1 Desain Aplikasi

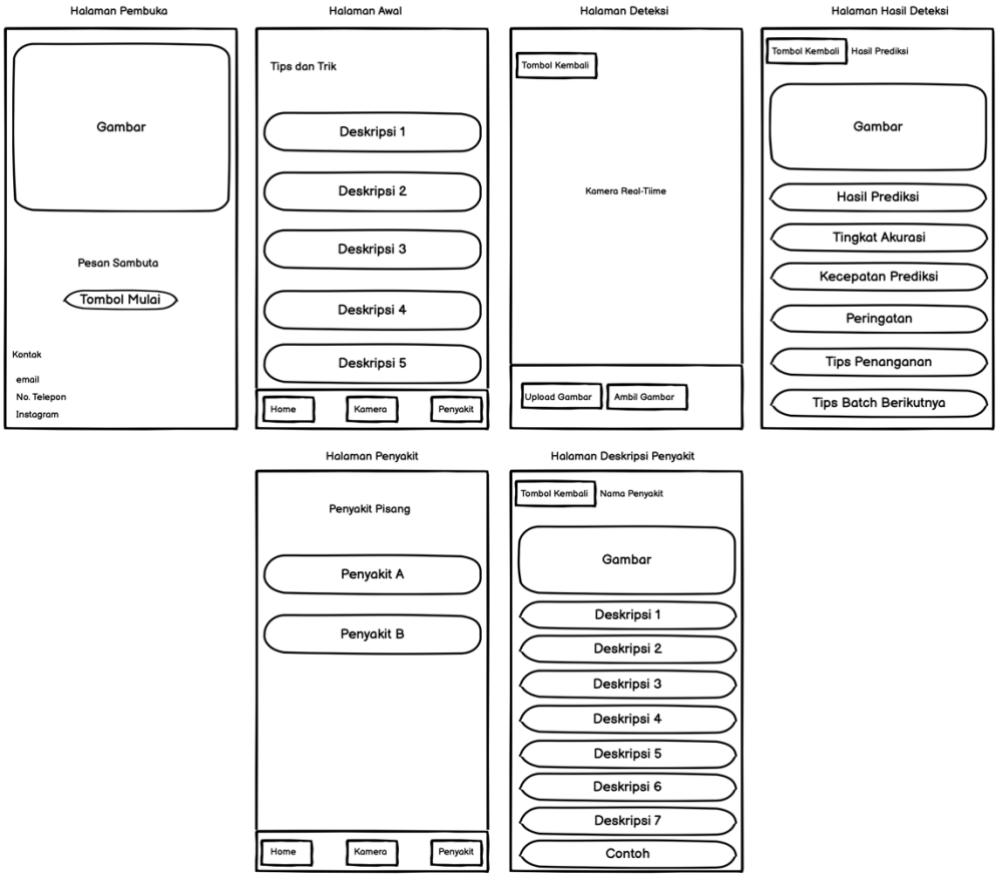
Aplikasi ini akan terdiri dari beberapa tampilan utama, yaitu Home, Deteksi (Kamera), dan penyakit. Setiap tampilan dirancang untuk mendukung fungsi aplikasi secara optimal, sehingga akan memudahkan pengguna dalam menggunakan aplikasi. Berikut merupakan penjelasan dari tampilan aplikasi yang akan dibuat oleh penulis dalam penelitian ini dapat dilihat pada Tabel 10.

Tabel 3.7 Tampilan Aplikasi

Tampilan Aplikasi	Tampilan	Deskripsi
Halaman Pembuka	*Splash Screen*	Merupakan tampilan awal yang muncul saat pengguna pertama kali membuka aplikasi dan saat pengguna mengklik "Mulai Sekarang" akan ke halaman selanjutnya
Halaman Awal (Home)	Tips dan Trik Menanam Pisang	Saat masuk, pengguna aplikasi akan langsung ditampilkan tips dan trik menanam pisang, hal ini bertujuan agar petani memahami teknik penanaman yang tepat sejak awal.

Halaman Deteksi (kamera)	Ambil Gambar	Pengguna aplikasi dapat mengambil gambar tanaman pisang melalui kamera ponsel secara langsung
	Upload Gambar	Pengguna aplikasi dapat mengunggah gambar tanaman pisang dari galeri ponsel
	Hasil Deteksi	Setelah mengambil atau mengunggah gambar, pengguna akan melihat hasil deteksi dari aplikasi
Halaman Penyakit	Kumpulan Penyakit tanaman pisang	Pengguna aplikasi dapat melihat berbagai penyakit yang menyerang tanaman pisang
	Deskripsi Penyakit Pisang	Pengguna aplikasi dapat melihat lebih detail terkait penyakti tanaman pisang

Untuk perancangan tampilan aplikasi, penulis akan merancang sesuai user interface yang telah penulis buat seperti pada Gambar 3.4.



Gambar 3.4 Desain Aplikasi

3.3.3.2 Pengembangan Aplikasi

Setelah membuat desain aplikasi, tahap berikutnya adalah pengembangan aplikasi tersebut. Pada tahap ini, peneliti menggunakan *framework open-source flutter* yang dikembangkan dan didukung oleh google. Flutter dipilih karena memungkinkan membangun aplikasi lintas platform dengan tampilan yang responsif dan performa yang tinggi. Selain itu Framework ini juga memungkinkan developer mengembangkan aplikasi untuk berbagai sistem operasi, seperti Android dan iOS, menggunakan satu basis kode yang sama sehingga, lebih efisien dalam hal waktu dan biaya pengembangan. Flutter menggunakan bahasa pemrograman sumber terbuka Dart yang dirancang untuk pengembangan aplikasi yang cepat dan modern [16].

Selain itu penulis juga menggunakan emulator dalam pengujian aplikasi. emulator merupakan perangkat virtual yang dirancang untuk menjalankan program atau aplikasi dari system host. Dalam penelitian ini peneliti menggunakan emulator android, yang berfungsi untuk menirukan perangkat android secara virtual. Dengan emulator ini, aplikasi dapat dijalankan dan diuji pada system host tanpa memerlukan perangkat android fisik [17].

3.3.3.3 Build Aplikasi

Setelah semua tahap pengembangan selesai, langkah selanjutnya adalah membangun aplikasi dalam format APK. Format APK dipilih karena memungkinkan aplikasi diinstal langsung pada perangkat Android tanpa biaya tambahan, sehingga mempermudah pengujian di smartphone. Berbeda dengan format App Bundle, yang memerlukan pengunggahan ke Play Store dan biaya untuk proses tersebut. Berikut adalah langkah-langkah yang digunakan untuk membangun aplikasi dalam format APK, dapat dilihat pada Tabel 3.7.

Tabel 3.7 Build Aplikasi

Langkah	Penjelasan
Memastikan Aplikasi di Emulator	Mengidentifikasi dan memperbaiki bug sebelum proses pembangunan aplikasi, memastikan aplikasi berfungsi dengan baik di emulator.
Memeriksa Kesesuaian Versi yang Digunakan	Memastikan ndkVersion, compileSdk, jvmTarget, distributionUrl, dan targetSdk sesuai untuk mencegah error selama proses pembangunan aplikasi.
Menambahkan File proguard-rules.pro	File ini digunakan untuk mengoptimalkan dan melindungi aplikasi selama proses pembangunan.

	File ini juga membantu mengurangi ukuran aplikasi dan meningkatkan keamanan.
Membuat Keystore	Keystore digunakan untuk menandatangani aplikasi dan memastikan validitasnya, Untuk membuat keystore, penulis menggunakan perintah berikut: • keytool -genkey -v -keystore ~/sipisang.jks -keyalg RSA -keysize 2048 -validity 10000 -alias sipisang Keystore ini memuat informasi penting, seperti: • KeyAlias: Nama alias kunci • KeyPassword: Kata sandi untuk kunci • StoreFile: Lokasi penyimpanan keystore yang dibuat • StorePassword: Kata sandi untuk keystore
Menempatkan Keystore di File key.properties	Keystore yang dibuat diletakkan pada file key.properties di tingkat proyek untuk digunakan selama proses pembangunan aplikasi.
Membangun Aplikasi dalam Mode Rilis	Langkah terakhir adalah membangun aplikasi dalam mode rilis dengan menggunakan perintah berikut: • flutter build apk --release --verbose

Penulis menamai aplikasi ini SIPISANG, singkatan dari Sistem Informasi Pisang. Nama ini mencerminkan tujuan utama aplikasi, yaitu membantu petani pisang dengan memberikan informasi serta mendeteksi penyakit tanaman pisang. Selain itu, nama ini dipilih karena sederhana, mudah diingat, dan relevan dengan fungsi aplikasi.

3.4 Perancangan Pengujian

Pada tahap akhir ini, peneliti akan melakukan analisis mendalam terhadap performa model dan aplikasi. Evaluasi ini mencakup beberapa aspek, seperti evaluasi metrik pada model dan evaluasi kinerja pada aplikasi.

3.4.1 Evaluasi Metrik Pada Model

Evaluasi metrik pada model memiliki peran yang sangat penting dalam menganalisis performa model, termasuk pengukuran kinerja, pengoptimalan model, dan perbandingan hasil antara model, sehingga penulis dapat menentukan model yang paling sesuai untuk diterapkan pada aplikasi. Berikut merupakan evaluasi metrik yang penulis gunakan dalam penelitian ini:

- Penggunaan grafik pembelajaran atau performa untuk mengevaluasi model, baik dalam pelatihan maupun validasi. Tujuan utama dari grafik ini adalah untuk memantau kestabilan model serta kemampuannya dalam melakukan generalisasi, sehingga dapat diketahui apakah model mengalami *overfitting* dan underfitting. Dengan cara ini, penulis dapat menilai efektivitas model dan menentukan apakah perlu dilakukan penyesuaian pada parameter pelatihan atau tidak.
- Classification Report meruapakan kumpulan metrik evaluasi yang mencakup akurasi, presisi, recall, F1-score, dan support, yang dihitung berdasarkan jumlah sampel di setiap kelas. Metrik-metrik ini memberikan bobot yang berbeda untuk setiap kelas, bergantung pada proporsi sampel yang ada di setiap kelas.
- Confusion Matrix, berisi nilai-nilai seperti true positive, true negative, false positive, dan false negative. Matriks ini berguna untuk menganalisis tingkat

kesalahan model serta kemampuan model dalam membedakan berbagai kelas secara akurat. Berikut rumus dari akurasi, presisi, recall, dan F-1 score.

$$\text{Akurasi} = (TP + TN) / (TP + FP + FN + TN)$$

$$\text{Presisi} = (TP) / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

$$\text{F-1 Score} = (2 * Recall * Precision) / (Recall + Precision)$$

3.4.2 Evaluasi kinerja pada aplikasi

Pada penelitian ini, penulis mengikuti pendekatan serupa dengan yang dilakukan oleh Sophia L. Sanga et al [8], di mana model yang telah dilatih kemudian diimplementasikan ke dalam sebuah aplikasi. Proses implementasi ini bertujuan untuk memastikan bahwa model tidak hanya efektif pada data uji, tetapi juga dapat bekerja dengan baik saat diintegrasikan dalam sebuah sistem aplikasi.