

**PENGEMBANGAN APLIKASI MOBILE PENDETEKSI PENYAKIT  
PADA TANAMAN CABAI DENGAN MENGGUNAKAN  
TEKNOLOGI CLARIFAI**

(Studi Kasus: Balai Pengkajian Teknologi Pertanian, Kecamatan  
Karangploso, Kota Malang)

**SKRIPSI**

Untuk memenuhi sebagai persyaratan  
Memperoleh gelar Sarjana Komputer

Disusun oleh:  
Insan Nurzaman Bangga Adi Pratama  
NIM: 165150200111033



TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
MALANG  
2019

## DAFTAR ISI

DAFTAR ISI .....	ii
DAFTAR TABEL.....	vii
DAFTAR GAMBAR .....	ix
Daftar lampiran .....	1
BAB 1 PENDAHULUAN.....	2
1.1 Latar Belakang.....	2
1.2 Rumusan Masalah.....	3
1.3 Tujuan .....	3
1.4 Manfaat.....	3
1.5 Batasan Masalah .....	3
1.6 Sistematika Pembahasan .....	4
BAB 2 LANDASAN KEPUSTAKAAN .....	5
2.1 Kajian Pustaka .....	5
2.2 Penyakit pada Tanaman Cabai.....	6
2.2.1 Penyakit Layu Fusarium ( <i>Fusarium oxysporum f. Sp</i> ) .....	6
2.2.2 Penyakit Layu Bakteri Ralstonia ( <i>Ralstonia solanacearum</i> ) .....	6
2.2.3 Penyakit Busuk Buah Antraknosa ( <i>Collectotrichom gloeosporioides</i> ) .....	7
2.2.4 Penyakit Virus Kuning ( <i>Gemini virus</i> ) .....	8
2.2.5 Penyakit Bercak Daun ( <i>Cercospora sp.</i> ) .....	9
2.3 Model Pengembangan Perangkat Lunak .....	10
2.3.1 Prototyping.....	10
2.4 Android .....	12
2.5 Clarifai .....	13
2.6 Use Case Diagram .....	13
2.7 Sequence Diagram .....	14
2.8 Class Diagram .....	15
2.9 Pengujian .....	15
2.9.1 Pengujian Fungsional .....	15

2.9.2 Pengujian Akurasi .....	16
2.9.3 Pengujian <i>Compatibility</i> .....	16
2.9.4 Pengujian <i>Usability</i> .....	16
BAB 3 METODOLOGI .....	17
3.1 Studi Literatur .....	18
3.2 Pengumpulan Data .....	18
3.3 Analisis Kebutuhan .....	18
3.4 Prototyping .....	18
3.5 Evaluasi <i>Prototype</i> .....	19
3.6 Perancangan .....	19
3.7 Implementasi .....	19
3.8 Pengujian .....	19
3.9 Pengambilan Kesimpulan dan Saran .....	19
BAB 4 analisis kebutuhan dan perancangan .....	20
4.1 Analisis Kebutuhan .....	20
4.1.1 Hasil Interview .....	20
4.1.2 Hasil Analisis Kebutuhan Perangkat Lunak .....	20
4.1.3 Gambaran Umum Sistem .....	21
4.2 Identifikasi Aktor .....	22
4.3 Kebutuhan Fungsional Sistem .....	22
4.4 Kebutuhan Non Fungsional Sistem .....	23
4.5 Pemodelan Kebutuhan .....	23
4.5.1 <i>Use case Diagram</i> .....	24
4.5.2 Pemodelan <i>Use case Scenario</i> .....	25
BAB 5 PERANCANGAN .....	31
5.1 Kebutuhan Arsitektur Sistem .....	31
5.2 Perancangan <i>Activity Diagram</i> .....	32
5.2.1 <i>Activity Diagram</i> Mendapatkan Gambar .....	32
5.2.2 <i>Activity Diagram</i> Mendeteksi Penyakit .....	35
5.2.3 <i>Activity Diagram</i> Mengetahui informasi penyakit dan pengendalian penyakit .....	37

5.2.4 <i>Activity Diagram</i> Mengetahui Riwayat Gambar yang Telah Dideteksi.....	37
5.2.5 <i>Activity Diagram</i> Menghapus Riwayat Hasil Analisis .....	39
5.3 Perancangan <i>Sequence Diagram</i> .....	41
5.3.1 <i>Sequence Diagram</i> Mendapatkan Gambar .....	41
5.3.2 <i>Sequence Diagram</i> Mendeteksi Penyakit .....	42
5.3.3 <i>Sequence Diagram</i> Mengetahui Informasi penyakit dan pengendalian Penyakit.....	44
5.3.4 <i>Sequence Diagram</i> Mengetahui Riwayat Gambar yang Telah Dideteksi.....	45
5.3.5 <i>Sequence Diagram</i> Menghapus Riwayat Hasil Deteksi.....	46
5.4 Perancangan <i>Class Diagram</i> .....	47
5.5 Perancangan Basis Data .....	52
5.5.1 Perancangan ERD .....	52
5.5.2 Perancangan Tabel .....	52
5.6 Perancangan Antarmuka Pengguna (Wireframe) .....	53
5.6.1 Perancangan Antarmuka SplashScreen .....	53
5.6.2 Perancangan Antarmuka Home .....	54
5.6.3 Perancangan Antarmuka <i>Detail</i> .....	55
5.6.4 Perancangan Antarmuka <i>Snap</i> .....	56
5.6.5 Perancangan Antarmuka Camera .....	57
5.6.6 Perancangan Antarmuka <i>Cropping Image</i> .....	58
5.6.7 Perancangan Antarmuka <i>Analyze</i> .....	59
5.6.8 Perancangan Antarmuka <i>Result</i> .....	60
5.6.9 Perancangan Antarmuka <i>About Apps</i> .....	61
5.7 Perancangan Antarmuka Pengguna (Wireframe) iterasi 1.....	61
5.7.1 Wireframe perbaikan tab informasi pestisida .....	62
5.7.2 Wireframe perbaikan mengganti menu <i>about apps</i> .....	63
5.7.3 Wireframe perbaikan memindahkan menu <i>about apps</i> pada halaman home .....	63
5.8 Perancangan Antarmuka Pengguna (Wireframe) iterasi 2.....	64
5.8.1 Wireframe perbaikan halaman menu home dan halaman <i>about apps</i> .....	65

5.8.2 Wireframe perbaikan pada halaman riwayat penambahan tombol remove .....	66
5.8.3 Wireframe penambahan Screen Intro.....	66
5.9 Perancangan Algoritme .....	69
5.9.1 Perancangan komponen mengetahui informasi penyakit dan pengendalian penyakit.....	69
5.9.2 Perancangan komponen mendeteksi penyakit .....	70
5.9.3 Perancangan komponen mengetahui riwayat gambar yang telah dideteksi .....	71
BAB 6 IMPLEMENTASI .....	72
6.1 Spesifikasi Sistem .....	72
6.2 Spesifikasi Perangkat Keras .....	72
6.3 Spesifikasi Perangkat Lunak.....	73
6.4 Batasan-batasan Implementasi .....	73
6.5 Implementasi Basis Data.....	74
6.6 Implementasi Clarifai .....	75
6.6.1 Define .....	75
6.6.2 Train .....	75
6.6.3 Recognize .....	76
6.7 Implementasi Algoritme .....	77
6.7.1 Implementasi algoritme mengetahui informasi penyakit dan pengendalian penyakit.....	77
6.7.2 Implementasi algoritme mendeteksi penyakit .....	78
6.7.3 Implementasi algoritme mendapatkan riwayat deteksi.....	80
6.8 Implementasi User Interface .....	81
6.8.1 Implementasi user interface splash screen.....	81
6.8.2 Implementasi user interface intro.....	82
6.8.3 Implementasi user interface mengetahui informasi penyakit dan pengendalian penyakit.....	84
6.8.4 Implementasi user interface mendapatkan gambar.....	86
6.8.5 Implementasi user interface mendeteksi penyakit.....	88
6.8.6 Implementasi user interface mengetahui riwayat hasil deteksi.....	90

6.8.7 Implementasi <i>user interface</i> mendapatkan menghapus riwayat hasil deteksi.....	91
6.8.8 Implementasi <i>user interface about apps</i> .....	93
BAB 7 PENGUJIAN .....	94
DAFTAR REFERENSI .....	95

## DAFTAR TABEL

Tabel 2.1 Daftar simbol pada use case diagram .....	13
Tabel 2.2 Daftar simbol <i>sequence diagram</i> .....	14
Tabel 2.3 Daftar simbol <i>class diagram</i> .....	15
Tabel 4.1 Hasil <i>User Interview</i> .....	20
Tabel 4.2 Daftar Kebutuhan Fungsionalitas Aplikasi DiSnap .....	21
Tabel 4.3 Aktor Sistem .....	22
Tabel 4.4 Kebutuhan Fungsional Sistem .....	22
Tabel 4.5 Kebutuhan Non Fungsionalitas .....	23
Tabel 4.6 Skenario <i>Use case</i> Mendapatkan Gambar .....	26
Tabel 4.7 Skenario <i>Use case</i> Mendeteksi Penyakit.....	27
Tabel 4.8 Skenario <i>Use case</i> Mengetahui Informasi Penyakit dan Pengendalian Penyakit.....	28
Tabel 4.9 Skenario <i>Use case</i> Mengetahui Riwayat Gambar yang Telah Dideteksi	29
Tabel 4.10 Skenario <i>Use case</i> menghapus riwayat hasil deteksi .....	30
Tabel 5.1 Rancangan tabel basis data DiSnap.....	52
Tabel 5.2 Temuan masalah wireframe DiSnap iterasi 1 .....	62
Tabel 5.3 Temuan masalah <i>wireframe</i> DiSnap iterasi 2 .....	64
Tabel 5.4 Algoritme perancangan komponen mengetahui informasi penyakit dan pengendalian penyakit.....	70
Tabel 5.5 Algoritme perancangan komponen mendeteksi penyakit.....	70
Tabel 5.6 Algoritme perancangan komponen mengetahui riwayat gambar yang telah dideteksi.....	71
Tabel 6.1 Spesifikasi perangkat keras komputer .....	72
Tabel 6.2 Spesifikasi perangkat keras <i>smartphone mobile</i> .....	72
Tabel 6.3 Spesifikasi perangkat lunak komputer .....	73
Tabel 6.4 Spesifikasi perangkat lunak <i>smartphone mobile</i> .....	73
Tabel 6.5 Implementasi tabel tHistory.....	74
Tabel 6.6 <i>Source code</i> method <code>getData()</code> Class <code>DiseaseJSONFileDataSource</code> .....	77
Tabel 6.7 Penjelasan <i>source code</i> method <code>getDiseaseFromJSONFile ()</code> .....	78
Tabel 6.8 <i>Source code</i> method <code>predictImage()</code> .....	79

Tabel 6.9 Penjelasan source code method predictImage() Class : <i>DiseaseRemoteDataSource</i> .....	80
Tabel 6.10 Source code method getDiseaseAnalysisResultFromDB() .....	80
Tabel 6.11 Penjelasan source code method getDiseaseAnalysisResultFromDB () Class : HistoryFragmentPresenter .....	81



## DAFTAR GAMBAR

Gambar 2.1 Layu Fusarium .....	6
Gambar 2.2 Layu Bakteri Ralstonia .....	7
Gambar 2.3 Busuk Buah Antraknosa .....	8
Gambar 2.4 Penyakit Virus Kuning.....	9
Gambar 2.5 Bercak Daun .....	10
Gambar 2.6 Model Prototype .....	11
Gambar 2.7 Lifecycle Android .....	13
Gambar 3.1 Alur Metodologi Penelitian .....	18
Gambar 4.1 <i>Use case Diagram</i> .....	24
Gambar 5.1 Arsitektur Sistem.....	32
Gambar 5.2 <i>Activity Diagram</i> Mendapatkan Gambar .....	34
<b>Gambar 5.3 <i>Activity Diagram</i> Mendeteksi Penyakit.....</b>	<b>36</b>
Gambar 5.4 <i>Activity Diagram</i> Mengetahui informasi penyakit dan pengendalian penyakit.....	37
Gambar 5.5 <i>Activity diagram</i> Mengetahui Riwayat Gambar yang Telah Dideteksi .....	38
Gambar 5.6 <i>Activity Diagram</i> Menghapus Riwayat Hasil Analisis .....	40
Gambar 5.7 <i>Sequence diagram</i> mendapatkan gambar.....	42
Gambar 5.8 <i>Sequence diagram</i> Mendeteksi penyakit.....	44
Gambar 5.9 <i>Sequence diagram</i> Mengetahui Informasi penyakit dan pengendalian Penyakit.....	45
Gambar 5.10 <i>Sequence diagram</i> Mengetahui riwayat gambar yang telah dideteksi .....	46
Gambar 5.11 <i>Sequence diagram</i> Menghapus riwayat hasil deteksi.....	47
Gambar 5.12 <i>Class Diagram</i> DiSnap .....	48
Gambar 5.13 <i>Class Diagram Package View</i> .....	49
Gambar 5.14 <i>Class Diagram Package Presenter</i> .....	50
Gambar 5.15 <i>Class Diagram Package Model</i> .....	51
Gambar 5.16 Entity Relationship Diagram DiSnap .....	52
Gambar 5.17 Wireframe SplashScreen .....	54
Gambar 5.18 <i>Wireframe Home</i> .....	54

Gambar 5.19 <i>Wireframe Detail</i> .....	55
Gambar 5.20 <i>Wireframe snap(bottomsheet)</i> .....	56
Gambar 5.21 <i>Wireframe Camera</i> .....	57
Gambar 5.22 <i>Wireframe Cropping Image</i> .....	58
Gambar 5.23 <i>Wireframe Analyze</i> .....	59
Gambar 5.24 <i>Wireframe Result</i> .....	60
Gambar 5.25 <i>Wireframe About Apps</i> .....	61
Gambar 5.26 <i>Wireframe perbaikan tab informasi penyakit</i> .....	62
Gambar 5.27 <i>Wireframe menu riwayat</i> .....	63
Gambar 5.28 <i>Wireframe menu home</i> .....	64
Gambar 5.29 <i>Wireframe halaman about apps</i> .....	65
Gambar 5.30 <i>Wireframe halaman riwayat</i> .....	66
Gambar 5.31 <i>Wireframe intro home</i> .....	67
Gambar 5.32 <i>Wireframe intro Snap</i> .....	68
Gambar 5.33 <i>Wireframe intro history</i> .....	69
Gambar 6.1 <i>Implementasi tahap define</i> .....	75
Gambar 6.2 <i>Implementasi tahap train</i> .....	76
Gambar 6.3 <i>Prose mengupload gambar</i> .....	76
Gambar 6.4 <i>Implementasi tahap recognize</i> .....	77
Gambar 6.5 <i>Implementasi user interface splash screen</i> .....	82
Gambar 6.6 <i>Implementasi user interface intro disease information</i> .....	83
Gambar 6.7 <i>Implementasi user interface intro snap</i> .....	83
Gambar 6.8 <i>Implementasi user interface intro history</i> .....	84
Gambar 6.9 <i>Implementasi user interface home</i> .....	85
Gambar 6.10 <i>Implementasi user interface detail informasi penyakit pada cabai</i> .....	85
Gambar 6.11 <i>Implementasi user interface show bottom dialog</i> .....	86
Gambar 6.12 <i>Implementasi user interface mengambil gambar melalui galery</i> ...	87
Gambar 6.13 <i>Implementasi user interface cropping image</i> .....	88
Gambar 6.14 <i>Implementasi user interface hasil dari cropping image</i> .....	88
Gambar 6.15 <i>Implementasi user interface proses mendeteksi penyakit pada tanaman cabai melalui gambar daun cabai</i> .....	89
Gambar 6.16 <i>Implementasi user interface hasil dari deteksi gambar</i> .....	90

Gambar 6.17 Implementasi menu riwayat .....	90
Gambar 6.18 Implementasi detail riwayat .....	91
Gambar 6.19 Implementasi <i>user interface dialog remove history</i> .....	92
Gambar 6.20 Implementasi <i>user interface remove success</i> .....	93
Gambar 6.21 Impelentasi <i>user interface about apps</i> .....	93

## DAFTAR LAMPIRAN

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Di negara-negara tropis seperti Indonesia, tanaman cabai menjadi salah satu tanaman komersil yang banyak dibudidayakan dan memiliki nilai jual tinggi sehingga menguntungkan bagi petani. Tidak hanya dijual di pasaran, cabai juga biasa dijadikan sebagai bahan baku industri sehingga membuka kesempatan peluang kerja bagi masyarakat (Setiadi, 2004). Berdasarkan publikasi yang dikeluarkan oleh Badan Pusat Statistik tahun 2018 tentang Distribusi Perdagangan Komoditas Cabai Merah Indonesia Tahun 2018 menyebutkan bahwa produksi cabai besar di Jawa Timur pada tahun 2017 mencapai 100.977 ton. Sedangkan tingkat konsumsi cabai merah masyarakat Jawa Timur mencapai 3.532 ton perkapita pertahun (Malahayati, 2018).

Kebutuhan akan cabai tiap tahun nya meningkat, akan tetapi produktifitas cabai di Indonesia masih belum dapat memenuhi kebutuhan cabai masyarakat Indonesia dikarenakan produktifitas cabai yang masih fluktuatif yang disebabkan mutu benih, kualitas tanah yang kurang baik kondisi lingkungan, cuaca, penyakit dan hama yang menurunkan hasil panen ataupun menyebabkan gagal produksi (Warisno dan Dahana, 2010). Salah satu kendala yang sering dijumpai yaitu kurangnya pengetahuan para petani dalam mengenali jenis penyakit dan hama yang menyerang tanaman pada cabai (Purwanto, 2015). Sehingga kurang ada penanganan yang tepat sesuai kondisi tanaman.

Dibantu dengan adanya teknologi internet dan perangkat bergerak yang sedang berkembang pada saat ini permasalahan untuk mengetahui penyakit pada tanaman cabai dapat diselesaikan dengan bantuan *Machine Learning* yaitu *Image Classification*. *Machine Learning* merupakan mesin yang banyak digunakan untuk menggantikan atau menirukan perilaku manusia (Ahmad, 2017). Sedangkan *Image Classification* adalah kemampuan mesin untuk mengklasifikasikan sebuah gambar masuk ke dalam kelompok – kelompok tertentu berdasarkan model yang telah dilatih.

Dengan memanfaatkan peluang dan teknologi yang ada, penulis memberikan solusi berupa aplikasi mobile dengan sistem operasi android yang berguna untuk mendeteksi penyakit pada tanaman cabai serta pengendaliannya. Aplikasi ini berfungsi membantu petani untuk mencegah penyebaran penyakit dengan memberikan penanganan yang sesuai dengan kondisi tanaman yang terserang penyakit. Berdasarkan permasalahan yang telah dijabarkan maka penulis memberi judul pada penelitian ini, yaitu “Pengembangan Aplikasi Mobile Pendeteksi Penyakit Pada Tanaman Cabai Dengan Menggunakan Teknologi Clarifai”.

## 1.2 Rumusan Masalah

Berdasarkan latarbelakang permasalahan tersebut, maka penulis dapat merumuskan masalah sebagai berikut:

1. Bagaimana cara menggali kebutuhan pada aplikasi pendeteksi penyakit tanaman cabai dengan menggunakan metode prototyping?
2. Bagaimana cara mengimplementasikan dan mengintegrasikan teknologi clarifai dalam pengembangan aplikasi mobile pendeteksi penyakit pada tanaman cabai berbasis android ?
3. Bagaimana tingkat akurasi dari aplikasi pendekteksi penyakit pada tanaman cabai?

## 1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui kebutuhan fungsional dan non-fungsional dari aplikasi pendeteksi penyakit pada tanaman cabai dengan menggunakan metode prototyping.
2. Mengimplementasikan teknologi clarifai dalam pengembangan aplikasi mobile pendeteksi penyakit pada tanaman cabai berbasis android.
3. Mengetahui tingkat akurasi aplikasi penyakit pada tanaman cabai.

## 1.4 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Membantu menyediakan aplikasi untuk BPTP dalam mempermudah petani mendeteksi penyakit pada tanaman cabai.
2. Dengan adanya aplikasi ini bisa menjadi bahan rujukan ataupun referensi untuk para peneliti dalam pengembangan selanjutnya.

## 1.5 Batasan Masalah

Pengembangan dalam penelitian ini memiliki beberapa batasan masalah, yaitu :

1. Jenis cabai yang diteliti yaitu jenis cabai merah yang ada dalam ruang lingkup BPTP.
2. Fokus dari penelitian ini yaitu dapat mengidentifikasi penyakit pada tanaman cabai di BPTP menggunakan teknologi *Clarifai*.
3. Aplikasi yang dibuat hanya dapat berjalan pada operating system Android dengan *minimal version* Android Lollipop (Android 5.0)
4. Untuk dapat memanfaatkan fitur analisis gambar aplikasi harus terkoneksi dengan internet.

## 1.6 Sistematika Pembahasan

Sistematika penyusunan dokumen skripsi ini dibagi menjadi beberapa bab, yaitu :

### 1. Bab 1 – PENDAHULUAN

Bagian ini menjelaskan tentang latarbelakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika bahasan pada penelitian.

### 2. Bab 2 – LANDASAN KEPUSTAKAAN

Bagian ini menjelaskan tentang uraian dan pembasan tentang teori, konsep, metode dan kajian-kajian yang terkait dengan pengidentifikasian penyakit pada tanaman cabai.

### 3. Bab 3 – METODOLOGI

Bagian ini menjelaskan tentang alur kerja penelitian sebagai proses penyelesaian masalah yang sedang diteliti.

### 4. Bab 4 – ANALISIS KEBUTUHAN

Bagian ini menjelaskan analisis kebutuhan dari *stackholder*. Hasil dari analisis kebutuhan akan dibuat kedalam prototyping yang selanjutnya digunakan untuk mempermudah dalam proses perancangan.

### 5. Bab 5 – PERANCANGAN

Bagian ini menjelaskan tentang perancangan sistem berdasarkan prototyping yang sudah dibuat. Perancangan yang dibuat berupa sequence diagram, class diagram.

### 6. Bab 5 – IMPLEMENTASI

Bagian ini menjelaskan proses implementasi dari hasil rancangan yang sudah dibuat. Implementasi dilakukan menggunakan Android Studio IDE dengan menggunakan bahasa pemograman Java.

### 7. Bab 7 – PENGUJIAN

Bagian ini menjelaskan tentang pengujian pada sistem yang dilakukan oleh peneliti. Pengujian yang akan dilakukan yaitu *Black Box Testing* dan tingkat akurasi aplikasi untuk mendeteksi penyakit pada tanaman cabai.

### 8. Bab 8 – PENUTUP

Bagian ini menjelaskan tentang kesimpulan yang diperoleh dari proses penelitian yang telah dilakukan serta memuat saran untuk dapat dipakai dalam proses pengembangan selanjutnya

## BAB 2 LANDASAN KEPUSTAKAAN

Pada bab landasan kepustakaan berisi beberapa kajian landasan kepustakaan dan teori-teori dasar yang berkaitan dengan penelitian yang sedang diteliti. Kajian pustaka berisi tentang penelitian pendeteksi penyakit pada tanaman jeruk dan penggunaan teknologi Clarifai dalam pemecahan masalah pada sebuah penelitian yang telah dilakukan sebelumnya. Selain itu adapun teori yang akan dijelaskan meliputi penyakit pada tanaman cabai, Android, Clarifai, konsep *Prototyping Model* dan teknik pengujian yang akan dilakukan.

### 2.1 Kajian Pustaka

Dalam penelitian ini terdapat beberapa penelitian sebelumnya yang berkaitan dengan penelitian pengembangan aplikasi pendeteksi penyakit pada tanaman cabai menggunakan teknologi Clarifai, yaitu sebagai berikut:

1. Sistem Pakar Analisa Penyakit Pada Tanaman Cabai Merah Menggunakan Metode Backward Chaining (Nusantara, Pamungkas, Syaifudin, Kusuma, & Fikri, 2017). Pada penelitian ini, terdapat masalah yaitu kurangnya pemahaman petani dalam menanggulangi penyakit pada cabai merah yang diatasi dengan sebuah solusi yaitu membuat sistem informasi berbasis web menggunakan metode backward Chaining untuk membantu para petani dalam menganalisis penyakit pada tanaman cabai. Hasil dari penelitian ini yaitu sebuah produk berbasis web yang dapat membantu para petani untuk mendeteksi penyakit pada tanaman cabai akan tetapi dengan menggunakan backward chaining masih terdapat beberapa kekurangan dalam menentukan pola solusi.
2. Sistem Pakar Deteksi Hama dan Penyakit Pada Tanaman Cabai Dengan Metode Naïve Bayes (Fistrianingtyas & Rahmad, 2015). Pada penelitian ini, terdapat masalah yaitu keterbatasan jumlah pakar atau ahli pertanian tidak dapat mengatasi permasalahan petani cabai yang diatasi dengan membuat sebuah sistem pakar untuk mendeteksi hama dan penyakit pada tanaman cabai menggunakan metode naïve bayes dalam proses identifikasi dengan media web. Hasil dari penelitian ini adalah dapat mengidentifikasi penyakit berdasarkan banyaknya data kejadian yang telah dimasukkan oleh pakar.
3. Rancang Bangun Aplikasi *SmartFoodies* Dengan Memanfaatkan Clarifai Api Untuk *Image Recognition* Berbasis Android (Ryantono, 2017). Pada penelitian ini terdapat permasalahan yaitu kesulitan masyarakat untuk mengenal dan membuat berbagai macam makanan khas nusantara yang harus dilestarikan. Solusi yang diberikan oleh peneliti yang membuat membangun aplikasi *smartfoodies* yaitu aplikasi untuk mempermudah pengguna dalam mengetahui nama bahan dan informasi pada makanan dengan akurat menggunakan teknologi Clarifai untuk melakukan *image recognition* berbasis android. Hasil dari penelitian ini yaitu membantu para pengguna dalam mengetahui tata cara masak, pembuatan resep masakan



berdasarkan pemanfaatan bahan yang ada dan menentukan rekomendasi resep makanan.

## **2.2 Penyakit pada Tanaman Cabai**

### **2.2.1 Penyakit Layu Fusarium (*Fusarium oxysporum f. Sp*)**

Penyakit layu fusarium pada tanaman cabai disebabkan oleh cendawan *Fusarium oxysporum*. Gejala yang dapat terlihat pada tanaman cabai yang terkena penyakit ini yaitu, tanaman mulai mengalami kelayuan dari bawah dan menguning menjalar ke atas ranting muda. Sumber penyakit ini biasanya berasal dari tanah dan sisa tanaman sakit. Adapun pemicu perkembangan penyakit layu fusarium yaitu lahan berpasir, pupuk N(ZA) terlalu tinggi, kurangnya pupuk kandang, tanah kekurangan kalsium dan jumlah nematoda tinggi. Penyakit layu fusarium pada tanaman cabai dapat dilihat pada Gambar 2.1.



**Gambar 2.1 Layu Fusarium**

Sumber : (BPTP Jambi, 2014)

### **2.2.2 Penyakit Layu Bakteri *Ralstonia solanacearum***

Penyebab pada penyakit tanaman cabai ini adalah adalah Bakteri *Pseudomonas solanacearum*. Gejala yang dapat dilihat yaitu pada tanaman tua terjadi daun layu pada bagian bawah tanaman. Sedangkan pada tanaman muda daun layu terjadi pada bagian atas tanaman. Setelah beberapa hari daun yang layu meliputi seluruh bagian pada tanaman, sedangkan warna daun masih tetap hijau terkadang sedikit kekuningan.

Adapun efek lain dari serangan penyakit ini terhadap tanaman cabai yaitu menyebabkan warna buah menjadi kekuningan dan membusuk.

Pemicu perkembangan penyakit Layu Bakteri *Ralstonia* adalah lahan yang terlalu basah, tanah terlalu liat, pupuk N (urea) terlalu tinggi, populasi nematoda tinggi dan tanah yang digunakan untuk menanam cabai sebelumnya digunakan untuk menanam tembakau, terong, tomat ataupun cabai.



**Gambar 2.2 Layu Bakteri *Ralstonia***

Sumber : (BPTP Jambi, 2014)

### **2.2.3 Penyakit Busuk Buah Antraknosa (*Collectotrichom gloeosporioides*)**

Penyakit buah busuk antraknosa pada tanaman cabai disebabkan oleh cendawan *collectotrichom*. Penyakit ini menyerang bagian buah cabai baik buah yang masih muda, maupun buah yang sudah masak. Gejala yang dapat dilihat dari tanaman cabai yang terjangkit penyakit ini yaitu munculnya bercak pada tubuh buah cabai yang agak mengkilap, sedikit berair, berwarna hitam, orange ataupun coklat. Warna hitam yang terlihat pada tubuh buah cabai merupakan struktur dari cendawan (*mikro skelerotia* dan *aservulus*).

Penyakit ini bersumber dari percikan air (termasuk penyemprotan pestisida), hujan angin dan tangan pemetik buah. Adapun pemicu perkembangan penyakit Busuk Buah Antraknosa yaitu benih tidak sehat,

kondisi tajuk terlalu lembab, pupuk N terlalu tinggi dan tanah kekurangan Ca.



**Gambar 2.3 Busuk Buah Antraknosa**

Sumber : (BPTP Jambi, 2014)

#### **2.2.4 Penyakit Virus Kuning (*Gemini virus*)**

Penyebab penyakit virus kuning pada tanaman cabai yaitu *gemini virus*. Gejala yang dapat dilihat dari tanaman cabai yang terkena penyakit virus kuning adalah warna kuning pada daun yang terlihat jelas dan tulang daun berubah menjadi kuning terang dan menebal serta daun yang menggulung ke atas.

Sumber penyakit virus kuning dapat berasal dari gulma atau tanaman sakit lainnya. Penularan penyakit virus kuning salah satunya yaitu melalui kutu kebul. Adapun pemicu perkembangan penyakit virus kuning pada tanaman cabai yaitu tanaman mulai terserang sejak bibit, banyak terjadi di musim kemarau (ketika pembibitan dan penanaman), dan populasi kutu kebul yang tinggi.



**Gambar 2.4 Penyakit Virus Kuning**

Sumber : (BPTP Jambi, 2014)

#### **2.2.5 Penyakit Bercak Daun (*Cercospora sp.*)**

Penyakit bercak daun pada tanaman cabai disebabkan oleh *Cercospora capsici* Heald and Wolf. Bercak daun *cercospora* dapat menimbulkan defoliasi jika serangan terjadi pada daun, sedangkan apabila terjadi pada bunga akan mengakibatkan gugur bunga serta apabila terjadi pada buah maka dapat menimbulkan malformasi pada buah yang mengakibatkan buah menjadi kerdil. Gejala penyakit ini menimbulkan munculnya bercak bulat berwarna coklat pada daun dengan kondisi yang kering serta memiliki ukuran sekitar 1 inci. Bercak yang tua dapat menyebabkan lubang-lubang pada bagian daun.

Kondisi lingkungan yang selalu hujan mendukung perkembangan dan penyebaran daripada penyakit bercak daun. Tanaman yang terserang akan layu dan rontok hal ini mampu menimbulkan kerugian ekonomi bagi para petani. Bahkan dalam kondisi serangan berat tanaman cabai dapat kehilangan hampir semua daunnya dan tentu saja sangat mempengaruhi tanaman cabai dalam menghasilkan buah.



**Gambar 2.5 Bercak Daun**

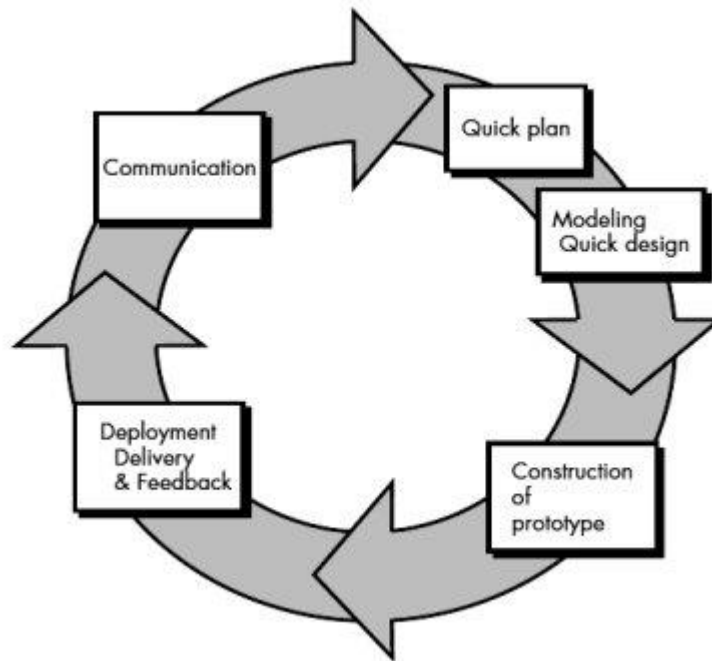
Sumber : (BPTP Jambi, 2014)

## **2.3 Model Pengembangan Perangkat Lunak**

Dalam pengembangan aplikasi penyakit pada tanam cabai ini peneliti menggunakan salah satu model pengembangan perangkat lunak yaitu *prototyping model*. Adapun beberapa model lain yang dapat digunakan dalam proses pengembangan perangkat lunak seperti *Waterfall Model*, *B-Model*, *Incremental Model*, *V-Model*, *Spiral Model*, *Wheel-and-spoke Model*, *Unified Process Model*, *Rapid Application Development (RAD)*, *Agile*, *Extreme Programming(XP)*, *Joint Application Development*, *Lean Development*, dan *Scrum*(Ruparelia, 2010).

### **2.3.1 Prototyping**

Pengembangan aplikasi pendeteksi penyakit pada tanaman cabai ini menggunakan metode *prototyping* dengan pendekatan *evolusioner* karena kebutuhan awal yang belum pasti. Ada beberapa tahapan iterasi pada metode prototyping yaitu *Communication*, *Quick Plan*, *Quick Design*, *Construction of Prototype*, dan *Deployment Delivery Feedback*.



**Gambar 2.6 Model Prototype**

Sumber : (Presman, 2010)

Penjelasan masing-masing tahap pada prototype model adalah sebagai berikut:

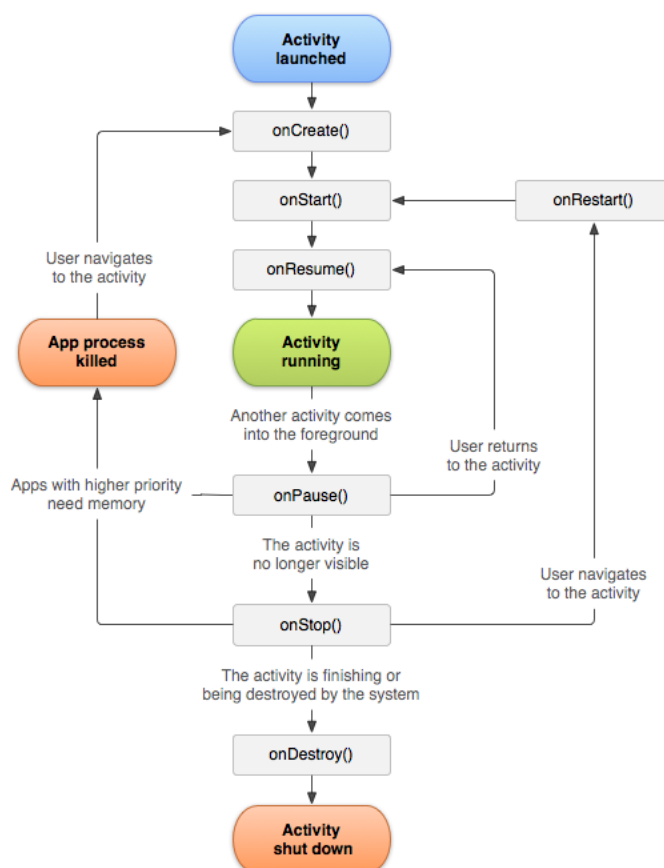
1. *Communication* / Komunikasi  
Perancang perangkat lunak melakukan pertemuan dan melakukan komunikasi dengan cara berdiskusi dengan pemangku kepentingan (*stakeholder*) untuk menentukan kebutuhan-kebutuhan yang ada pada perangkat lunak yang akan dibangun.
2. *Quick Plan* / Perencanaan secara cepat  
Pada tahap ini dilakukan pembuatan prototype secara cepat (*scratching*). Setelah membuat *scratch* dilakukan pemodelan dalam bentuk rancangan cepat.
3. *Modeling Quick Design* / Model Rancangan Cepat  
Pada tahap ini dilakukan pemodelan secara terstruktur dalam bentuk DFD(Data Flow Diagram), ERD(Entity Relationship Diagram) dan Flowchart untuk menggambarkan analisis dan desain sistem.
4. *Constructor of Prototype* / Pembuatan Prototype  
Representasi aspek-aspek perangkat lunak dibutuhkan untuk dapat rancangan cepat yang akan terlihat oleh *end user*. Rancangan cepat digunakan untuk membuat dasar prototype.
5. *Deployment Delivery & Feedback* / Penyerahan dan Memberikan Umpan Balik Terhadap Pengembangan

Prototype hasil dari tahap sebelumnya diserahkan kepada *stackholder* untuk evaluasi dan divalidasi untuk memberikan umpan balik pada *developer* untuk memperbaiki spesifikasi kebutuhan. Iterasi akan terjadi saat *developer* melakukan perbaikan terhadap prototype yang telah dibuat sebelumnya.

## 2.4 Android

Android adalah sistem operasi berbasis kernel Linux yang dirancang oleh Google yang biasa digunakan untuk perangkat seperti *smartphone*, *tablet*, *smartwatch* dan berbagai *smartdevice* lainnya. Android juga memiliki SDK(Software Development Kit) yang membantu dan mempermudah *developer* untuk mengembangkan aplikasi. Android juga memiliki banyak versi yang sangat beraneka ragam seperti: *Cupcake*, *Donut*, *Éclair*, *Froyo*, *Gingerbread*, *Honeycomb*, *Ice Cream Sandwich*, *Jelly Bean*, *Kit Kat*, *Lollipop*, *Marshmallow*, *Nougat*, *Oreo*, *Pie* dan versi android yang baru rilis adalah Android 10 (Developers, 2019)

Pengguna sistem operasi android di Indonesia merupakan pengguna terbanyak daripada pengguna sistem operasi lainnya. Hal ini sesuai dengan informasi dari sebuah situs web penyedia data yaitu StatCounter, pengguna sistem operasi android di Indonesia mencapai 93.69% per Juli 2019.(Statcounter, 2019).



## Gambar 2.7 Lifecycle Android

Sumber (android.developer.com)

Pada Gambar 2.7, merupakan lifecycle dari sistem operasi Android. Lifecycle pada gambar adalah daur hidup dari sistem operasi Android yang dimulai dari state Activity launched kemudian proses pemanggilan method onCreate() dan diakhiri dengan proses pemanggilan method onDestroy() kemudian state Activity shut down. Dengan memanfaatkan lifecycle ini akan membantu mempermudah pengembang untuk mengembangkan aplikasi Android.

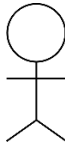
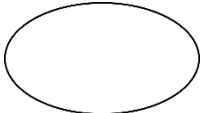
## 2.5 Clarifai

Clarifai adalah perusahaan *Artificial Intelligence* yang bergerak dibidang *Computer Vision* menggunakan *Machine Learning* dan *Neural Network* untuk mengidentifikasi gambar baik berupa photo ataupun video. Clarifai memiliki API yang dapat digunakan dalam mengidentifikasi dan mengklasifikasi citra atau gambar secara custom. Selain itu Clarifai juga menyediakan SDK untuk android maupun ios untuk membantu *developer* mengembangkan aplikasi dengan kemampuan seperti image classification, object detection dan lain-lain.(Clarifai, 2019).

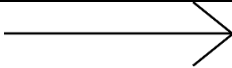
## 2.6 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Berikut merupakan daftar symbol use case diagram yang ditunjukkan pada Tabel 2.1.

**Tabel 2.1 Daftar simbol pada use case diagram**

Simbol	Nama	Keterangan
	<i>Actor</i>	Mendeskripsikan peran pengguna terhadap sistem
	<i>Use case</i>	Gambaran dari fungsional yang ada pada sistem, sehingga memudahkan dalam memahami apa saja yang

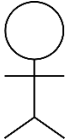


		dapat dilakukan pengguna actor terhadap sistem
	<i>Association</i>	Abstraksi dari penghubung antara aktor dengan use case.


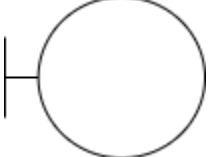
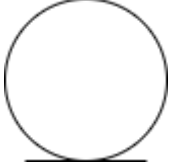


## 2.7 Sequence Diagram

*Sequence diagram* adalah diagram yang menggambarkan interaksi antar objek yang berada di dalam sistem luar dan berinteraksi dengan sistem berupa pesan terhadap waktu berlangsungnya interaksi. Tujuan dari pembuatan sequence diagram adalah agar perancangan pada sistem lebih mudah dipahami dan terarah (Rumbaugh et al., 2004). Daftar simbol *sequence diagram* ditunjukkan pada Tabel 2.2.

**Tabel 2.2 Daftar simbol *sequence diagram***

Simbol	Nama	Keterangan
	<i>Actor</i>	Menggambarkan aktor/pengguna yang ada pada sistem

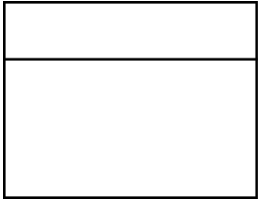
**Tabel 2.2 Daftar simbol *sequence diagram* (lanjutan)**

	<i>Controller</i>	Menggambarkan penghubung antara boundary dengan tabel.
	<i>Boundary</i>	Menggambarkan hubungan kegiatan yang akan dilakukan.
	<i>Entity</i>	Menggambarkan hubungan kegiatan yang akan dilakukan.
	<i>Lifeline</i>	Menggambarkan tempat mulai dan berakhirnya sebuah pesan.
	<i>Line Message</i>	Menggambarkan pengiriman pesan.

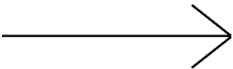
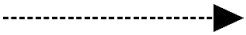
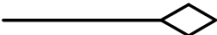
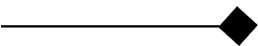
## 2.8 Class Diagram

Class diagram adalah diagram yang dapat membantu untuk menggambarkan struktur dari sebuah sistem dilihat dari cara mendefinisikan kelas-kelas yang ada pada sistem yang akan dibuat (Rumbaugh, Jacobson, & Booch, 2004). Berikut adalah daftar simbol dari *class diagram* yang ditunjukkan pada

**Tabel 2.3 Daftar simbol *class diagram***

Simbol	Nama	Keterangan
	<i>Class</i>	Mendeskripsikan Kelas pada struktur sistem. Terdapat tiga bagian. Bagian atas yaitu nama <i>class</i> . Bagian tengah mendeskripsikan <i>property</i> /atribut <i>class</i> . Bagian bawah yaitu method yang terdapat pada <i>class</i> tersebut.

**Tabel 2.3 Daftar simbol *class diagram* (lanjutan)**

	<i>Association</i>	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity.
	<i>Dependency</i>	Relasi antar kelas dengan makna ketergantungan antar kelas.
	<i>Agregation</i>	Relasi antar kelas dengan makna semua-bagian
	<i>Composition</i>	Menggambarkan relasi komposisi.

## 2.9 Pengujian

### 2.9.1 Pengujian Fungsional

Pengujian fungsional adalah pengujian yang dilakukan untuk memvalidasi keluaran hasil yang diuji sesuai dengan kebutuhan fungsional. Hasil dari keluaran

yaitu berupa *error*, hasil yang diinginkan tidak sesuai atau hasil keluaran sesuai dengan kebutuhan.

### **2.9.2 Pengujian Akurasi**

Pengujian akurasi adalah pengujian yang dilakukan secara langsung terhadap aplikasi untuk mendapatkan nilai akurasi dari sistem dalam mendeteksi penyakit pada tanaman cabai.

### **2.9.3 Pengujian *Compatibility***

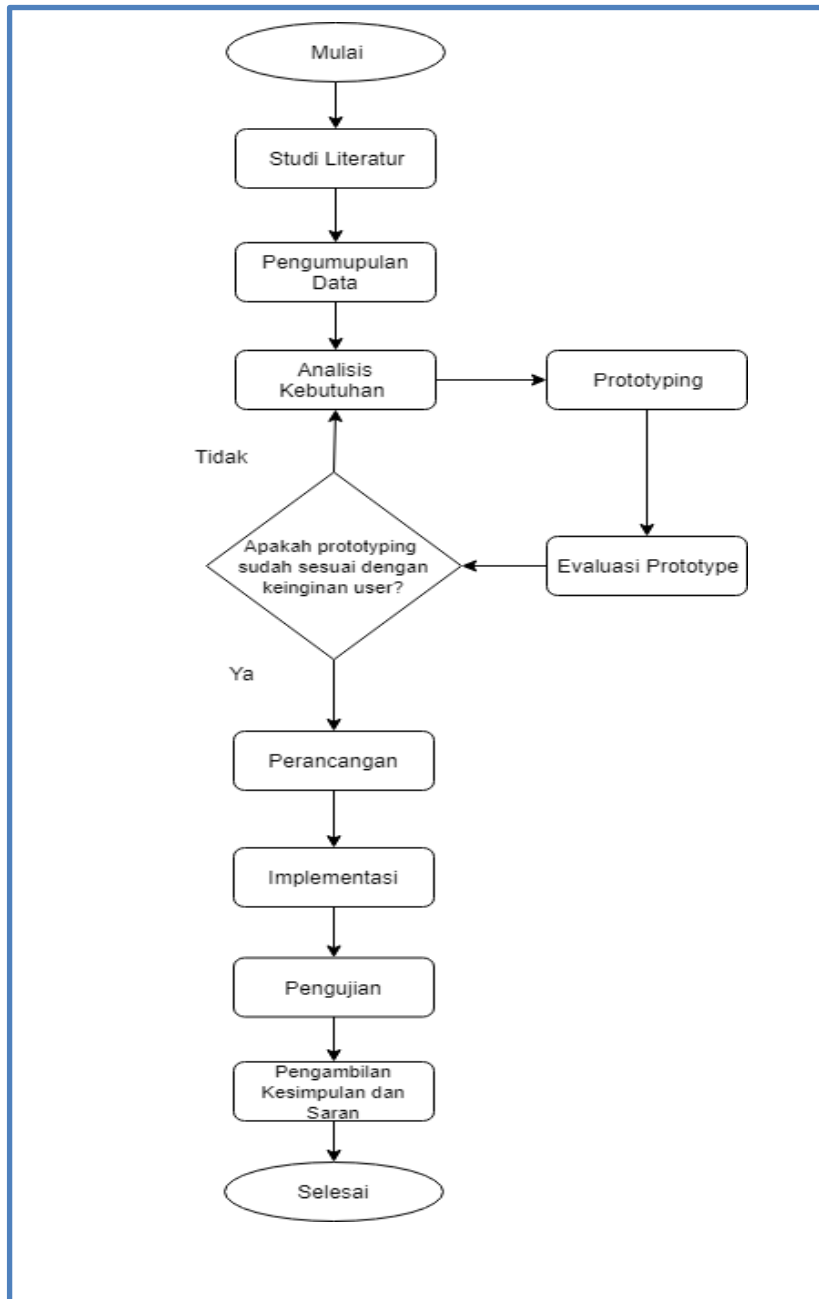
Pengujian *compatibility* adalah pengujian yang dilakukan dengan tujuan untuk mengetahui seberapa kompatibel aplikasi yang dikembangkan oleh peneliti terhadap perangkat yang tersedia.

### **2.9.4 Pengujian *Usability***

Pengujian *usability* yang dilakukan oleh peneliti dilakukan dengan tujuan mengetahui tingkat kepuasan dari pengguna terhadap sistem yang dibangun oleh peneliti.

### BAB 3 METODOLOGI

Peneliti melakukan penelitian dengan jenis penelitian implementatif. Peneliti melakukan implementasi teknologi clarifai kedalam sebuah aplikasi mobile berbasis android untuk mengetahui tingkat akurasi aplikasi dalam mendeteksi penyakit pada tanaman cabai. Selain itu, penelitian ini juga bertujuan untuk membantu para petani yang berada dalam naungan Balai Pengkajian Teknologi Pertanian yang berlokasi di Kecamatan Karangploso, Kota Malang sekaligus menjadi tempat penliti untuk melakukan penelitian.



### **Gambar 3.1 Alur Metodologi Penelitian**

Pada Gambar 3.1 dijelaskan bahwa didalam penelitian ini terdapat 9 tahapan proses pengembangan anatara lain studi literatur, pengumpulan data, analisis kebutuhan, perancangan, prototyping, evaluasi *prototype*, implementasi, pengujian serta pengambilan kesimpulan dan saran.

#### **3.1 Studi Literatur**

Pada tahap ini, peneliti mengumpulkan beberapa literatur bidang ilmu yang digunakan sebagai acuan dan referensi untuk penelitian yang sedang dilakukan yaitu meliputi :

Penyakit dan hama pada tanaman cabai.

*SDLC Prototype*.

1. Pengembangan Aplikasi Android menggunakan bahasa pemograman Java.
2. Pengembangan Aplikasi Android menggunakan Android Studio IDE.
3. Konsep Clarifai API untuk proses identifikasi penyakit pada tanaman cabai.
4. Pengujian fungsional dan pengujian akurasi untuk proses pengujian akurasi.

#### **3.2 Pengumpulan Data**

Dalam mendeskripsikan hal-hal di atas, penulis dapat menyusun subbab-subbab beserta alur logikanya dengan pertimbangan sendiri di bawah supervisi pembimbing, berdasarkan relevansi dengan sifat penelitian dan aspek keterbacaan. Pada tahap ini, dilakukan proses pengumpulan data dari Balai Pengkajian Teknologi Pertanian (BPTP) yang berada di daerah Karangploso, Malang, Jawa Timur.

#### **3.3 Analisis Kebutuhan**

Pada tahap ini dilakukan proses penggalian kebutuhan dengan melakukan wawancara terhadap pakar penyakit dan tanaman cabai untuk menggali kebutuhan perangkat lunak serta melakukan survei secara langsung di Balai Pengkajian Teknologi Pertanian (BPTP) Karangploso, Malang, Jawa Timur. Dari proses menganalisis kebutuhan akan menghasilkan kebutuhan fungsional dan non-fungsional. Pada tahap ini dilakukan pemodelan sistem menggunakan diagram *Unified Modeling Language* (UML), yaitu *use case diagram*, dan *use case scenario*.

#### **3.4 Prototyping**

Setelah selesai membuat rancangan pemodelan dari perangkat lunak maka pada tahap ini dilakukan proses pembuatan prototyping. Prototyping adalah bagian rancangan antar muka dari sebuah sistem yang berhubungan langsung dengan *user*.

### **3.5 Evaluasi *Prototype***

Pada tahap ini, prototype hasil dari tahap sebelumnya diserahkan kepada *stackholder* untuk evaluasi dan divalidasi untuk memberikan umpan balik pada *developer* untuk memperbaiki spesifikasi kebutuhan. Iterasi akan terjadi saat *developer* melakukan perbaikan terhadap prototype yang telah dibuat.

### **3.6 Perancangan**

Tahap ini akan dilakukan setelah proses analisis kebutuhan selesai. Pada tahap ini dilakukan pemodelan sistem menggunakan diagram *Unified Modeling Language* (UML), yaitu *sequence diagram*, *class diagram* dan *activity diagram* serta dilakukan perancangan basis data, perancangan algoritme dan perancangan antarmuka.

### **3.7 Implementasi**

Apabila prototype sudah sesuai dengan kebutuhan user maka iterasi prototyping dihentikan, selanjutnya masuk ke tahapan implementasi. Pada tahap implementasi peneliti melakukan pembangunan program aplikasi dengan memperhatikan hasil rancangan dan hasil prototype sebelumnya. Pada tahap ini pula akan dijelaskan detail penggunaan Clarifai API untuk membangun aplikasi pendeteksi penyakit pada tanaman cabai yang akan diimplementasikan menggunakan Android Studio dengan Bahasa pemograman java.

### **3.8 Pengujian**

Setelah tahap implementasi selesai, pada tahap ini dilakukan dua pengujian yaitu pengujian fungsionalitas dan pengujian persentase akurasi sistem untuk dapat mendeteksi penyakit pada tanaman cabai dengan tepat. Pengujian fungsionalitas berfokus pada hasil keluaran yang diuji dengan tiga kemungkinan yaitu *error*, hasil tidak sesuai, dan hasil yang sesuai.

### **3.9 Pengambilan Kesimpulan dan Saran**

Pada tahap ini, peneliti menarik kesimpulan berdasarkan proses yang telah dilakukan dan berdasarkan rumusan masalah yang telah dijabarkan sebelumnya, kemudian dilanjutkan dengan penulisan saran dari penelitian yang dilakukan berdasarkan kekurangan yang ditemukan dengan tujuan sebagai saran untuk pengembangan selanjutnya.

## BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

Pada bab ini berisikan semua hal terkait proses penggalian kebutuhan yang dilakukan peneliti serta perancangan pada sistem berdasarkan data yang telah didapatkan pada tahap sebelumnya.

### 4.1 Analisis Kebutuhan

#### 4.1.1 Hasil Interview

Berdasarkan tahap yang dilakukan sebelumnya yakni pengumpulan data dari pakar penyakit dan hama pada tanaman cabai maka terdapat beberapa daftar kebutuhan yang dapat diimplementasikan pada aplikasi. Hasil interview dari pengguna dapat disimpulkan menjadi beberapa poin seperti yang ditunjukkan pada Tabel 4.1

**Tabel 4.1 Hasil *User Interview***

No	Hasil <i>User Interview</i>
1	Pengguna biasa mengetahui penyakit dan hama pada tanaman cabai melalui pengalaman.
2	Pengguna merasa kesulitan untuk mengetahui beberapa jenis penyakit tertentu.
3	Pengguna biasa dibantu oleh pakar untuk menentukan pestisida yang tepat yang dapat diberikan kepada tanaman sesuai jenis penyakit pada tanaman cabai yang terserang.

#### 4.1.2 Hasil Analisis Kebutuhan Perangkat Lunak

Pada proses analisis kebutuhan dilakukan proses identifikasi terhadap pengguna yang menggunakan sistem. Selanjutnya dilakukan analisis kebutuhan fungsional dengan metode prototyping yang hasilnya di gambarkan kedalam sebuah tabel kebutuhan fungsional dan *use case diagram*. Selanjutnya dilakukan analisis kebutuhan non-fungsional dengan tujuan untuk mendukung kualitas penggunaan sistem. Analisis kebutuhan ini dilakukan dengan tujuan untuk memudahkan dalam proses perancangan sistem dan memenuhi serta sesuai dengan kebutuha pengguna.

Dalam membuat kebutuhan fungsional diperlukan suatu kode berupa cara penulisan untuk memudahkan dalam proses pengidentifikasian kebutuhan untuk konsistensi terhadap sistem sampai dilakukan proses pengujian sistem. Pada hasil analisis kebutuhan perangkat lunak ini dilakukan pengkodean dengan F\_DS\_XXX . F adalah singkatan dari Fungsional, DS singkatan dari DiSnap yaitu nama aplikasi dari sistem ini, sedangkan XXX merupakan nomor dari kebutuhannya. Dalam membuat kebutuhan fungsional diperlukan suatu kode berupa cara penulisan untuk memudahkan dalam proses pengidentifikasian kebutuhan untuk

konsistensi terhadap sistem sampai dilakukan proses pengujian sistem. Daftar kebutuhan fungsional aplikasi ditunjukkan dalam Tabel 4.2

**Tabel 4.2 Daftar Kebutuhan Fungsionalitas Aplikasi DiSnap**

No	Kode Kebutuhan Fungsional	Nama Kebutuhan Fungsional
1	F-DS-01	Mendapatkan gambar
2	F-DS-02	Mendeteksi penyakit
3	F-DS-03	Mengetahui informasi penyakit dan pengendalian penyakit
4	F-DS-04	Mengetahui riwayat gambar yang telah dideteksi
5	F-DS-05	Menghapus riwayat hasil deteksi

#### **4.1.3 Gambaran Umum Sistem**

Dalam penelitian ini, penulis menganalisis dan membangun sebuah sistem berupa aplikasi *mobile* untuk mendeteksi penyakit pada tanaman cabai. Nama lain dari aplikasi ini yaitu DiSnap. Dibangunnya aplikasi DiSnap bertujuan untuk membantu para petani dalam menganalisis penyakit pada tanaman cabai secara langsung melalui sensor pada kamera ataupun melalui gambar pada galeri yang ada pada *smartphone*. Selain untuk mendeteksi dan menganalisis penyakit melalui kamera ataupun galeri, aplikasi ini juga memberikan rekomendasi penanganan serta saran pestisida yang digunakan kepada tanaman yang terserang penyakit. Untuk melakukan proses pendeteksian penyakit melalui gambar, aplikasi yang dibangun oleh peneliti menggunakan sebuah layanan web service yang telah menyediakan API maupun SDK untuk melakukan proses pengenalan gambar yang sebelumnya telah dilatih untuk mengenali dan mengklasifikasikan gambar yang telah dikostumisasi dengan data gambar pada tanaman cabai yaitu Clarifai.

Ketika pengguna dari aplikasi ini telah mengambil gambar melalui kamera ataupun galeri maka aplikasi akan mendeteksi nama penyakit ataupun hama yang menyerang tanaman, menampilkan tingkat akurasi penyakit, penanganan dan rekomendasi pemberian pestisida sesuai penyakit ataupun hama yang menyerang berdasarkan hasil penelitian yang dilakukan oleh pakar penyakit dan hama pada tanaman cabai. Selain itu aplikasi ini juga memiliki fitur informasi mengenai jenis-jenis penyakit yang pada tanaman cabai beserta cara penanganan dan saran pestisida yang dapat dibaca secara langsung oleh petani untuk menambah pengetahuan para petani. Adapun fitur riwayat yang berguna bagi petani untuk melihat data aktifitas pendeteksian gambar sebelumnya.



## 4.2 Identifikasi Aktor

Aktor adalah seseorang ataupun sebuah sistem diluar sistem utama yang berinteraksi langsung dengan sistem utama untuk melakukan suatu tugas tertentu. Aktor yang ada pada sistem ini ditunjukkan seperti pada Tabel 4.3

**Tabel 4.3 Aktor Sistem**

Aktor	Deskripsi
Pengguna	Pengguna adalah aktor yang menggunakan seluruh fitur pada sistem. Pengguna berinteraksi dengan sistem secara langsung untuk melakukan proses mendeteksi penyakit pada tanaman cabai. Pengguna aplikasi dapat seorang petani, pakar ataupun peneliti tanaman cabai.

## 4.3 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem merupakan suatu kebutuhan yang harus tersedia pada sistem, hal ini termasuk dalam bagaimana sebuah sistem merespon inputan dari pengguna, dapat memberikan informasi ketika sistem dalam kondisi tertentu serta dapat menyelesaikan masalah dalam rumusan masalah sebelumnya (Sommerville, 2011). Pada sistem ini terdapat 5 kebutuhan fungsional yang dijelaskan pada Tabel 4.2

**Tabel 4.4 Kebutuhan Fungsional Sistem**

No	Kode Fungsional	Deskripsi	Use case
1	F-DS-01	Sistem dapat melakukan pengambilan gambar menggunakan kamera ataupun galeri.	Mendapatkan gambar
3	F-DS-02	Sistem dapat menyediakan fungsi untuk mendeteksi penyakit.	Mendeteksi penyakit
4	F-DS-03	Sistem dapat memberikan informasi mengenai penyakit	Mengetahui informasi penyakit

		pada tanaman cabai dan cara pengendalian serta pemberian pestisida berdasarkan penyakit daun pada tanaman cabai yang menyerangnya.	dan pengendalian penyakit
5	F-DS-04	Sistem harus mampu menyediakan informasi tentang riwayat data aktifitas pendeteksian gambar sebelumnya.	Mengetahui riwayat gambar yang telah dideteksi
6	F-DS-05	Sistem dapat menyediakan fungsi untuk menghapus riwayat hasil deteksi	Menghapus riwayat hasil deteksi

#### 4.4 Kebutuhan Non Fungsional Sistem

Kebutuhan non fungsionalitas merupakan kebutuhan yang berfokus dalam membantu jalannya sistem dan perilaku sistem. Kebutuhan fungsionalitas didefinisikan sebagai fungsi yang ditawarkan atau suatu batasan layanan pada sistem (Sommerville, 2011). Pada sistem ini terdapat beberapa kebutuhan non-fungsionalitas seperti *Usability* yaitu kemudahan dalam menggunakan aplikasi dan *Compatibility* yaitu aplikasi hanya mampu berjalan minimal di platform Android dengan *minimal version* Android 21.

**Tabel 4.5 Kebutuhan Non Fungsionalitas**

No	Kode Kebutuhan	Nama	Deskripsi
1	NF-DS-01	<i>Usability</i>	Aplikasi dapat digunakan dengan mudah dan berguna oleh pengguna.
2	NF-DS-02	<i>Compatibility</i>	Aplikasi dapat berjalan sesuai perangkat yang <i>compatible</i> dengan sistem

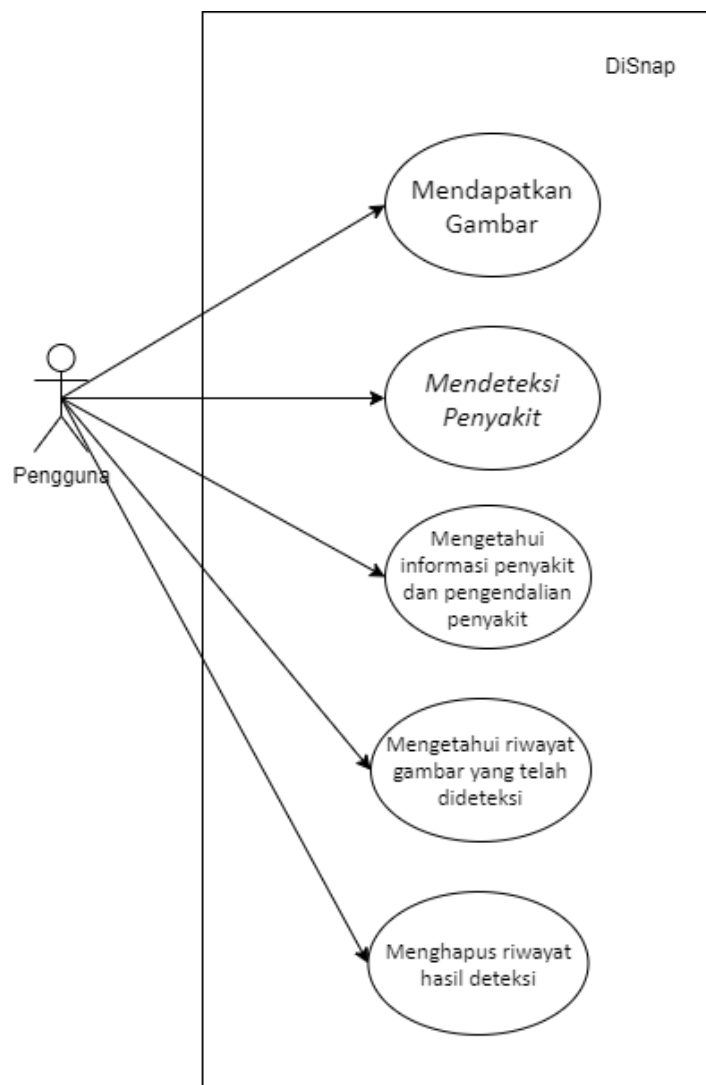
#### 4.5 Pemodelan Kebutuhan

Pemodelan merupakan tahap melakukan pemodelan terhadap kebutuhan yang telah dikumpulkan. Pemodelan ini menggunakan teknik *usecase diagram* dan *usecase scenario*. *Usecase diagram* adalah diagram usecase yang menggambarkan secara ringkas siapa yang menggunakan sistem dan apa saja

yang dapat dilakukan terhadap sistem tersebut. Sedangkan *usecase scenario* adalah jalur jalannya proses dari sisi aktor terhadap sistem.

#### 4.5.1 Use case Diagram

Pemodelan *use case diagram* ditunjukkan pada Tabel 4.3 dimana dalam use case diagram tersebut terdapat satu aktor yaitu aktor pengguna dan lima *use case diagram*. Pada *use case diagram ini*, aktor pengguna dapat mendapatkan gambar, mendeteksi penyakit pada tanaman cabai, mengetahui informasi dan pengendalian penyakit pada tanaman cabai, mengetahui riwayat gambar yang telah dideteksi, dan dapat menghapus riwayat hasil deteksi.



Gambar 4.1 Use case Diagram

#### **4.5.2 Pemodelan *Use case Scenario***

Pemodelan skenario *use case* dapat dibuat setelah dilakukan pembuatan *use case diagram*. Skenario *use case* dilakukan untuk menjelaskan detail proses pada setiap *use case*. Berikut skenario *use case* dari DiSnap seperti yang ditunjukkan pada Tabel 4.6 hingga Tabel 4.9.

#### 4.5.2.1 Skenario *Use case* Mendapatkan Gambar

**Tabel 4.6 Skenario *Use case* Mendapatkan Gambar**

Item	Deskripsi
Kode	F-DS-01
Nama	Mendapatkan Gambar
Aktor	Pengguna
Deskripsi	Pengguna dapat mengambil gambar daun menggunakan kamera ataupun galeri dan melakukan proses <i>image cropping</i>
Pra-Kondisi	Pengguna sudah berada pada halaman utama aplikasi
Tindakan	<ol style="list-style-type: none"> <li>1. Sistem menampilkan halaman utama aplikasi</li> <li>2. Pengguna menekan menu snap pada menu utama aplikasi</li> <li>3. Sistem menampilkan halaman Snap.</li> <li>4. Sistem menampilkan <i>bottom sheet dialog</i> berupa pilihan menu untuk mengambil gambar melalui kamera atau galeri</li> <li>5. Pengguna memilih satu metode pengambilan gambar melalui kamera atau galeri</li> <li>6. Pengguna menekan <i>button next</i></li> <li>7. Sistem menampilkan halaman <i>cropping image</i></li> <li>8. Pengguna melakukan <i>cropping image</i></li> <li>9. Sistem memproses image sesuai ukuran yang telah dipilih pengguna</li> <li>10. Sistem menampilkan gambar hasil <i>cropping image</i> pada halaman Analyze</li> <li>11. Sistem siap untuk mendeteksi gambar</li> </ol>
Post-Kondisi	Pengguna berhasil mendapatkan gambar yang siap untuk dideteksi
Alternatif	<ol style="list-style-type: none"> <li>1. Jika pengguna memilih mengambil gambar melalui kamera maka sistem akan menampilkan <i>camera screen</i></li> <li>2. Jika pengguna memilih mengambil gambar melalui galeri maka sistem akan menampilkan <i>gallery screen</i></li> </ol>

#### 4.5.2.2 Skenario *Use case* Mendeteksi Penyakit

**Tabel 4.7 Skenario *Use case* Mendeteksi Penyakit**

Item	Deskripsi
Kode	F-DS-02
Nama	Me ndeteksi Penyakit
Aktor	Pengguna
Deskripsi	Pengguna melakukan deteksi penyakit pada gambar yang telah didapatkan dari proses sebelumnya untuk memperoleh jenis penyakit, akurasi, dan cara penanganannya.
Pra-Kondisi	Pengguna berada pada halaman Analyze dan telah mendapatkan gambar yang siap untuk dideteksi
Tindakan	<ol style="list-style-type: none"><li>1. Pengguna berada pada halaman Analyze</li><li>2. Sistem menampilkan gambar yang sudah didapatkan pada proses sebelumnya</li><li>3. Pengguna menekan tombol analyze image pada layar</li><li>4. Sistem mendeteksi koneksi internet pada perangkat</li><li>5. Jika perangkat terhubung dengan koneksi internet maka sistem melanjutkan ke proses selanjutnya</li><li>6. Sistem menampilkan progress bar loading dan persentase loading</li><li>7. Sistem mengirim gambar ke imgur hosting image</li><li>8. Setelah mendapatkan url image dari imgur hosting image, url image di dikirm ke Clariafai API untuk dideteksi</li><li>9. Sistem melakukan pendeteksian penyakit pada gambar yang telah dikirim</li><li>10. Apabila berhasil sistem menampilkan pesan “Analisis sukses”</li><li>11. Sistem akan menampilkan hasil deteksi pada halaman Result</li><li>12. Ketika pengguna menekan tombol back/close maka hasil pendeteksian akan disimpan di database</li><li>13. Sistem menampilkan pesan “Hasil deteksi dapat dilihat pada menu riwayat”</li></ol>

Post-Kondisi	Pengguna mendapatkan informasi tentang jenis penyakit, akurasi, cara penanganan serta saran pemberian pestisida.
Alternatif	1. Jika perangkat tidak terhubung dengan koneksi internet maka sistem menampilkan pesan “Periksa koneksi internet”

#### 4.5.2.3 Skenario *Use case* Mengetahui Informasi Penyakit dan Pengendalian Penyakit

**Tabel 4.8 Skenario *Use case* Mengetahui Informasi Penyakit dan Pengendalian Penyakit**

Item	Deskripsi
Kode	F-DS-03
Nama	Mengetahui Informasi Penyakit dan Pengendalian Penyakit
Aktor	Pengguna
Deskripsi	Pengguna memperoleh informasi tentang berbagai penyakit dan pengendalian penyakit pada tanaman cabai
Pra-Kondisi	Pengguna berada pada halaman utama aplikasi
Tindakan	<ol style="list-style-type: none"> <li>1. Pengguna memilih menu <i>home</i></li> <li>2. Sistem menampilkan menu home</li> <li>3. Sistem akan menampilkan berbagai informasi tentang penyakit pada tanaman cabai</li> <li>4. Pengguna memilih salah satu penyakit</li> <li>5. Sistem menampilkan halaman Detail Disease Info</li> <li>6. Sistem menampilkan detail informasi penyakit pada tanaman cabai ke layar</li> </ol>
Post-Kondisi	Pengguna berhasil memperoleh informasi detail penyakit pada tanaman cabai
Alternatif	-

#### 4.5.2.4 Skenario *Use case* Mengetahui Riwayat Gambar yang Telah Dideteksi

**Tabel 4.9 Skenario *Use case* Mengetahui Riwayat Gambar yang Telah Dideteksi**

Item	Deskripsi
Kode	F-DS-04
Nama	Mengetahui riwayat gambar yang telah dideteksi
Aktor	Pengguna
Deskripsi	Pengguna memperoleh informasi tentang riwayat gambar yang telah dideteksi.
Pra-Kondisi	Pengguna berada pada halaman utama aplikasi
Tindakan	<ol style="list-style-type: none"><li>1. Pengguna memilih menu riwayat</li><li>2. Sistem mengambil data riwayat aktifitas deteksi dari database lokal</li><li>3. Jikapada databse terdapat data hasil aktifitas maka sistem mengambil semua data tersebut untuk ditampilkan</li><li>4. Sistem menampilkan menu riwayat</li><li>5. Sistem menampilkan daftar gambar riwayat yang pernah dideteksi</li><li>6. Pengguna memilih salah satu riwayat</li><li>7. Sistem menampilkan detail riwayat hasil deteksi gambar pada tanaman cabai .</li></ol>
Post-Kondisi	Pengguna berhasil memperoleh informasi riwayat penyakit yang pernah dideteksi.
Alternatif	Jika tidak ada data riwayat hasil deteksi pada database maka sistem akan menampilkan informasi bahwa tidak ada riwayat aktifitas deteksi.



#### 4.5.2.5 Skenario *Use case* Menghapus riwayat hasil deteksi

**Tabel 4.10** Skenario *Use case* menghapus riwayat hasil deteksi

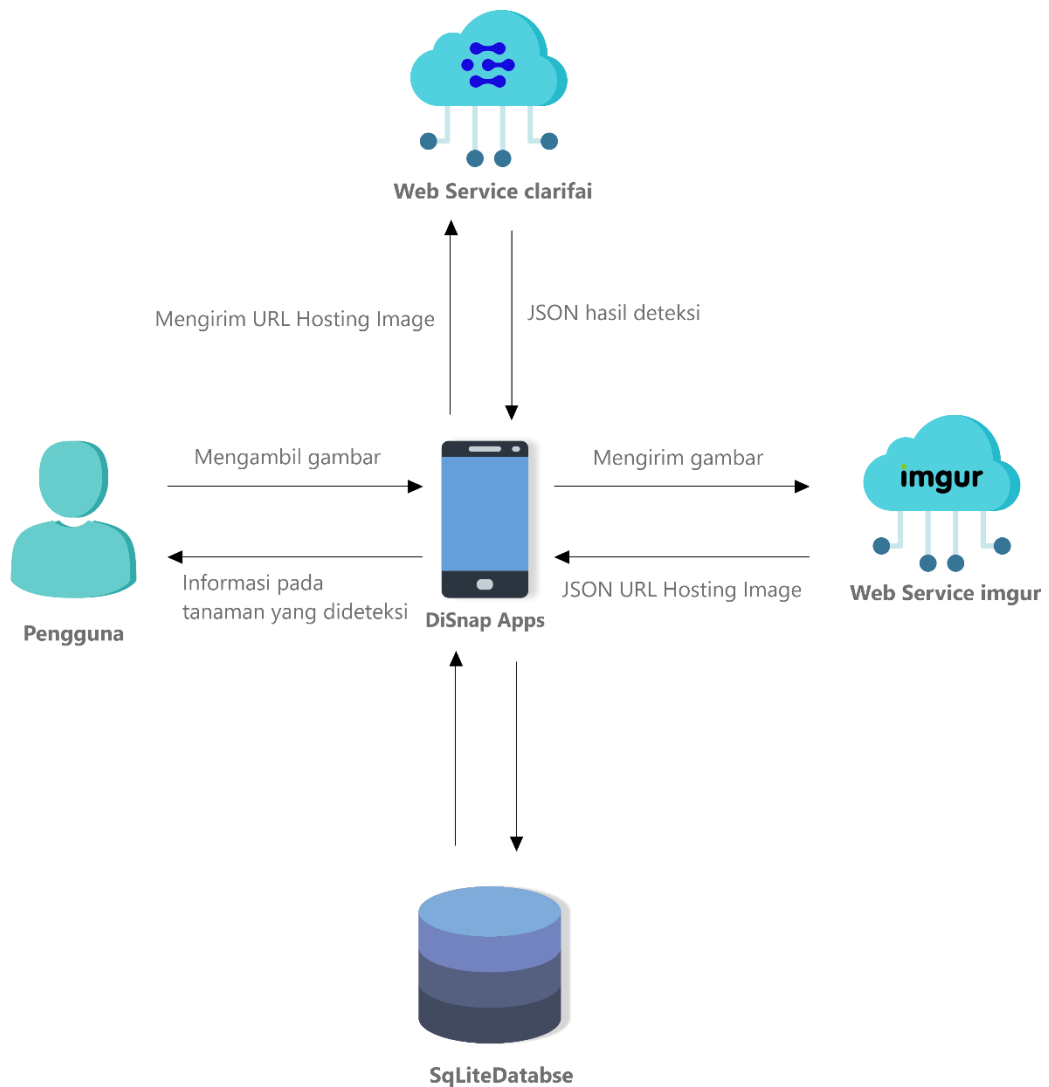
Item	Deskripsi
Kode	F-DS-05
Nama	Menghapus riwayat hasil deteksi
Aktor	Pengguna
Deskripsi	Pengguna dapat menghapus riwayat hasil deteksi pada halaman riwayat
Pra-Kondisi	Pengguna berada pada halaman utama aplikasi
Tindakan	<ol style="list-style-type: none"><li>1. Pengguna memilih menu riwayat</li><li>2. Sistem menampilkan menu riwayat</li><li>3. Sistem menampilkan daftar gambar riwayat yang pernah dideteksi</li><li>4. Pengguna menekan tombol remove pada salah satu riwayat</li><li>5. Sistem menampilkan dialog dengan pesan “Are you sure want to delete this history?”</li><li>6. Pengguna memilih pilihan “Yes”</li><li>7. Sistem menghapus riwayat yang dipilih pengguna</li><li>8. Sistem menampilkan pesan “Remove success”</li><li>9. Sistem mengatur ulang urutan daftar riwayat dengan urutan terbaru</li></ol>
Post-Kondisi	<ol style="list-style-type: none"><li>1. Pengguna berhasil menghapus riwayat yang dipilih</li><li>2. Sistem menampilkan daftar riwayat terbaru</li></ol>
Alternatif	-

## BAB 5 PERANCANGAN

Pada bab ini dilakukan pembahasan mengenai rancangan aplikasi *DiSnap* berbasis Android. Perancangan tersebut terdiri dari perancangan arsitektur sistem, activity diagram, sequence diagram, class diagram, perancangan basis data, perancangan antarmuka pengguna dan perancangan algoritme.

### 5.1 Kebutuhan Arsitektur Sistem

Pada Gambar 5.1 pengguna mengambil gambar menggunakan perangkat *smartphone*. Gambar yang diambil dapat berasal dari kamera ataupun dari galeri yang ada pada *smartphone* pengguna. Selanjutnya setelah dilakukan pengambilan gambar, sistem akan meminta pengguna untuk memotong gambar dengan tujuan memfokuskan gambar terhadap bagian gambar yang ingin dideteksi sebelum akhirnya dilakukan proses pendeteksian. Sebelum gambar dikirimkan ke Clarifai API *Web Service*, gambar dari *smartphone* pengguna harus dilakukan peng-*hosting*-an menggunakan *imgurl web service* dengan tujuan mendapatkan *url image* yang akan dikirimkan ke Clarifai. Hal ini dilakukan karena peneliti menggunakan Clarifai API dengan parameter berupa link *url image*. Setelah proses *image hosting*, maka dilakukan proses pendeteksian penyakit pada tanaman melalui url gambar yang dikirimkan. Setelah proses analisis gambar selesai maka hasilnya akan dikirim kembali ke *smartphone* pengguna dengan format JSON. Data dalam format JSON tersebut kemudian dikelola untuk dioleh menjadi informasi yang akan disampaikan kepada pengguna. Selanjutnya hasil dari aktifitas pendeteksian akan disimpan kedalam lokal database pada *smartphone* pengguna.



**Gambar 5.1 Arsitektur Sistem**

## 5.2 Perancangan *Activity Diagram*

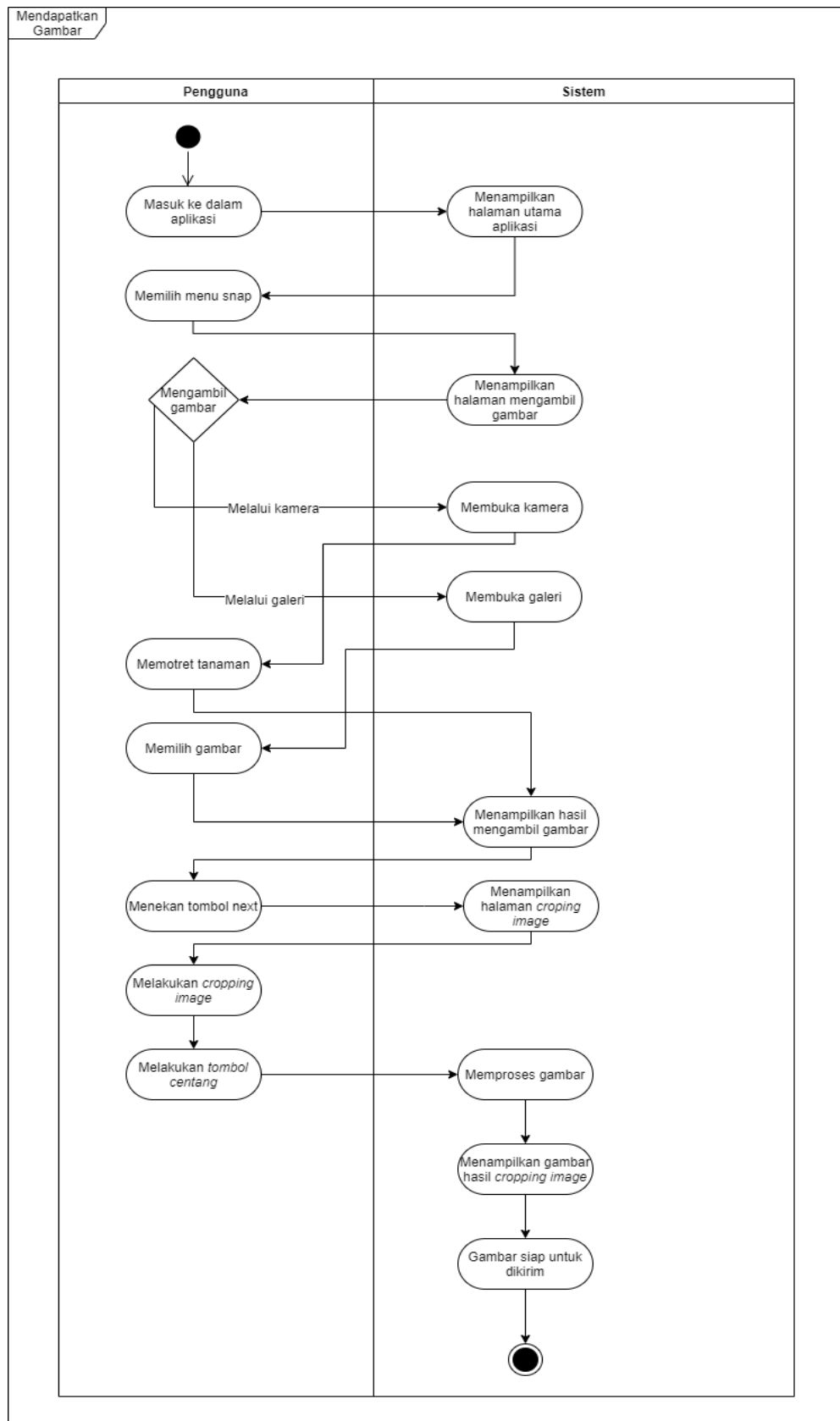
Pada tahap ini dilakukan perancangan *activity diagram* dari use case yang telah dibuat untuk memudahkan dalam mengetahui workflow dari setiap *use case* yang ada. Hasil dari perancangan *activity diagram* dapat dilihat dari xxxxxx.

### 5.2.1 *Activity Diagram* Mendapatkan Gambar

Pada Gambar 5.2 terdapat *activity diagram* dari *use case* mendapatkan gambar. Pada *activity diagram* tersebut pengguna melakukan pengambilan gambar baik itu melalui kamera ataupun galeri. Aktivitas pengguna dimulai dari pengguna berada pada halaman menu utama aplikasi. Selanjutnya pengguna memilih menu *snap* dan sistem akan

menampilkan halaman snap kepada pengguna. Pada halaman snap, pengguna diberikan dua pilihan untuk dapat mengambil gambar melalui kamera ataupun galeri.

Apabila pengguna mengambil gambar melalui kamera maka sistem akan membuka kamera pada aplikasi, sedangkan apabila pengguna memilih mengambil gambar melalui galeri maka sistem akan membuka galeri yang ada pada *smartphone* pengguna. Setelah memilih atau mengambil gambar maka sistem akan menampilkan halaman *image cropping* untuk membantu pengguna memfokuskan gambar yang akan dideteksi. Setelah proses tersebut selesai pengguna menekan tombol centang selanjutnya sistem akan menampilkan halaman analyze yaitu halaman yang menampilkan gambar hasil dari proses sebelumnya dan kemudian siap untuk dilakukan pendeteksian pada gambar.



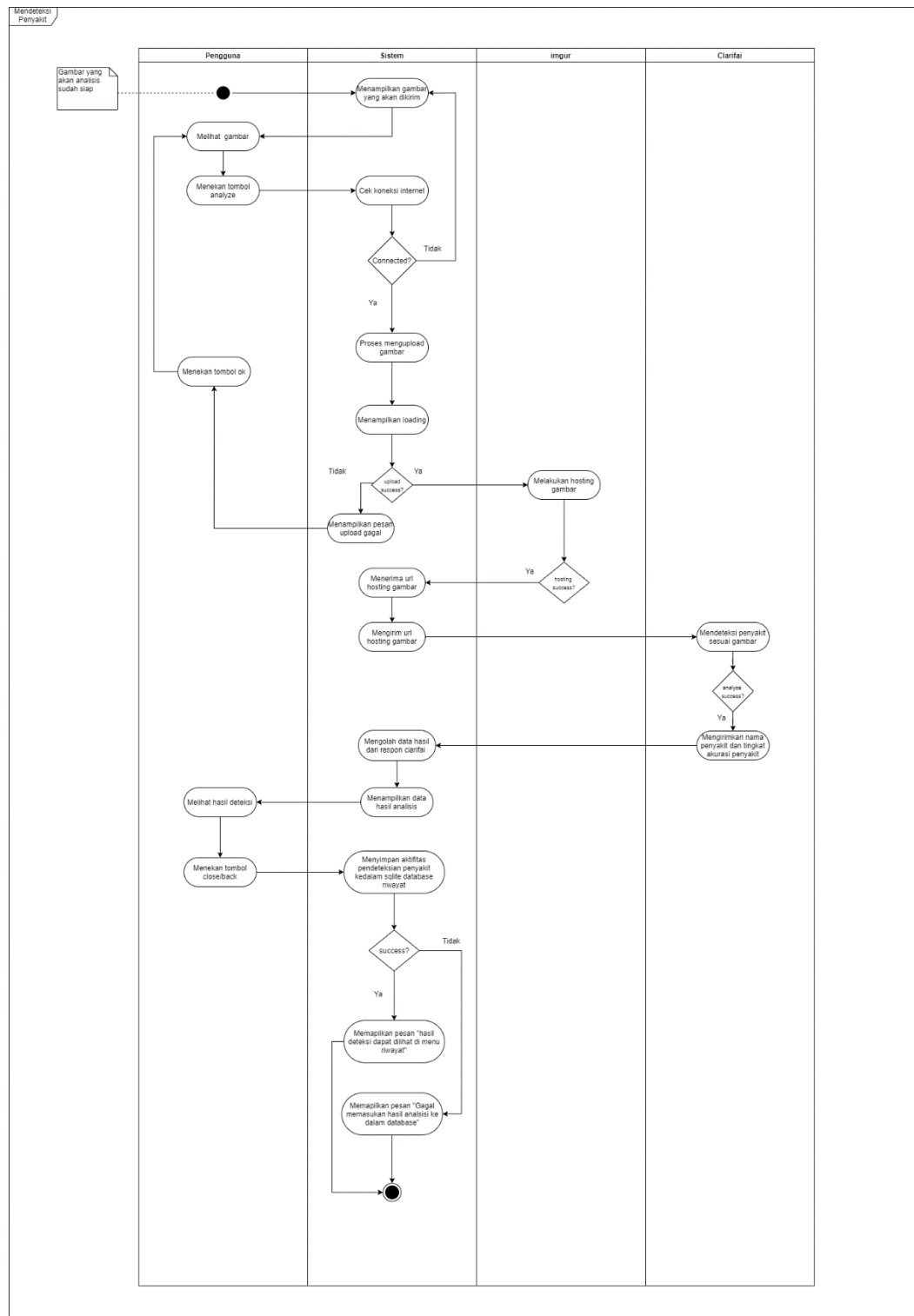
**Gambar 5.2 Activity Diagram Mendapatkan Gambar**

### 5.2.2 Activity Diagram Mendeteksi Penyakit

Pada **Gambar 5.3**, daftar gambar yang akan dikirim oleh pengguna sudah siap untuk dikirim. Selanjutnya pengguna menekan tombol *analyze image*, apabila perangkat pengguna tidak terhubung dengan internet maka sistem akan menampilkan pesan “Periksa koneksi internet anda” dan pengguna masih berada pada halaman *analyze*, apabila terhubung dengan koneksi internet maka terjadi proses upload atau pengunggahan gambar seperti yang terdapat pada **Gambar 5.3**. Sementara dilakukan proses meng-*upload*, maka sistem menampilkan proses *loading* kepada pengguna. Dalam tahapannya gambar terlebih dahulu dilakukan *hosting* di website imgurl dengan tujuan untuk mendapatkan *url image* yang selanjutnya akan digunakan untuk mendeksi gambar yang telah dilakukan *hosting* di website clarifai.

Jika gambar berhasil dikirim ke clarifai maka pada web service clarifai dilakukan pendeteksian penyakit terhadap gambar yang telah di *upload*. Sebelumnya peneliti telah melakukan *training model* terhadap gambar-gambar tanaman cabai yang terkena penyakit. Setelah proses *training model* selesai dilakukan, web service clarifai memberikan sebuah API yang dapat dipakai oleh peneliti.

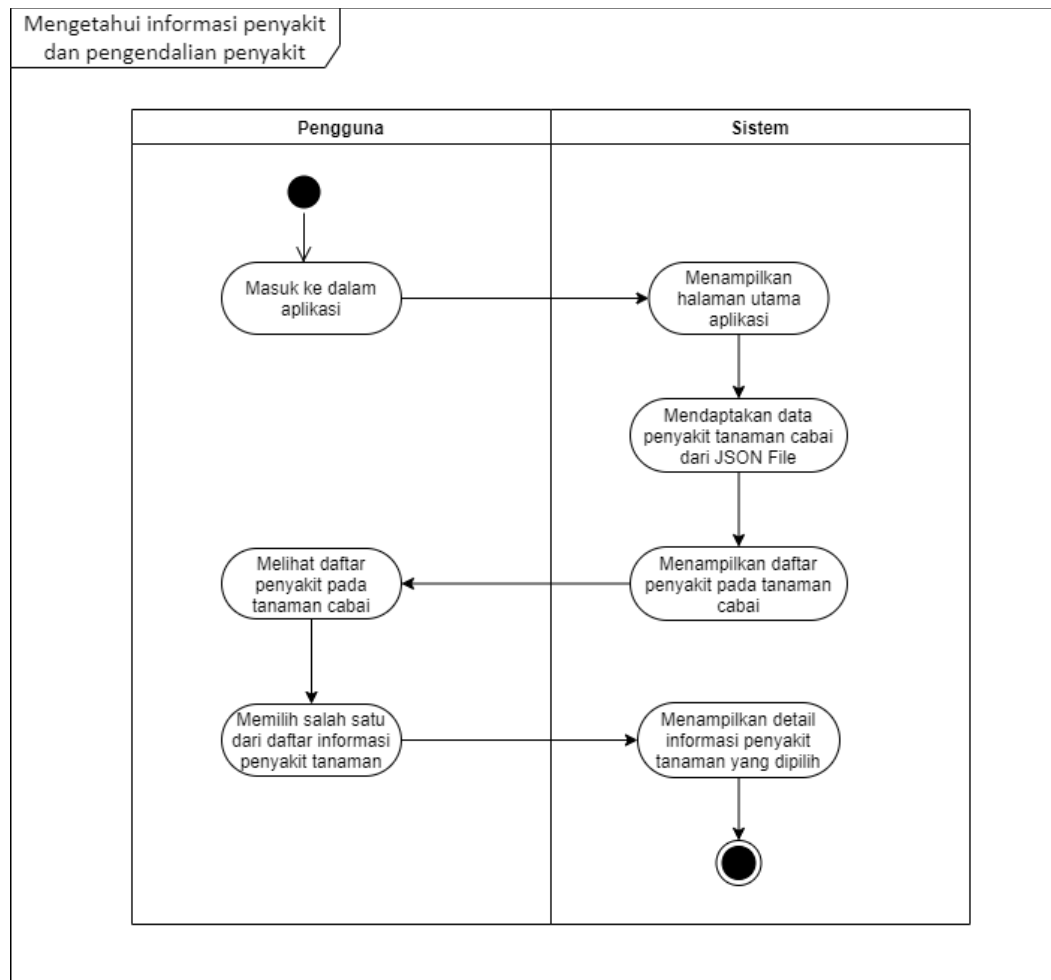
Setelah proses pendeteksian selesai dilakukan maka web service clarifai akan memberikan hasil berupa data nama penyakit dan tingkat akurasi dari gambar dalam format JSON yang kemudian diterima oleh sistem untuk dikelola dan digunakan untuk memanggil data cara pengendalian dan saran pemberian pestisida yang tersimpan di dalam sqlite database sesuai penyakit yang ada. Setelah data siap ditampilkan, maka sistem menampilkan pesan kepada pengguna bahwa proses deteksi berhasil. Kemudian pengguna menekan tombol lanjutkan. Kemudian sistem mulai menampilkan nama penyakit, cara penanganan, dan saran pemberian pestisida terhadap pengguna. Selanjutnya sistem menyimpan riwayat hasil aktifitas mendeteksi penyakit di sqlite database yang nanti akan diakses kembali pada fitur riwayat.



**Gambar 5.3 Activity Diagram Mendeteksi Penyakit**

### 5.2.3 Activity Diagram Mengetahui informasi penyakit dan pengendalian penyakit

Pada Gambar 5.4 pengguna masuk kedalam aplikasi kemudian sistem menampilkan halaman utama aplikasi. Selanjutnya pengguna memilih menu home dan sistem menampilkan halaman home. Kemudian sistem akan mengambil data informasi penyakit dari JSON file yang selanjutnya sistem menampilkan daftar informasi penyakit kepada pengguna. Setelah itu pengguna memilih satu dari daftar informasi penyakit yang ada kemudian sistem menampilkan detail informasi kepada pengguna.



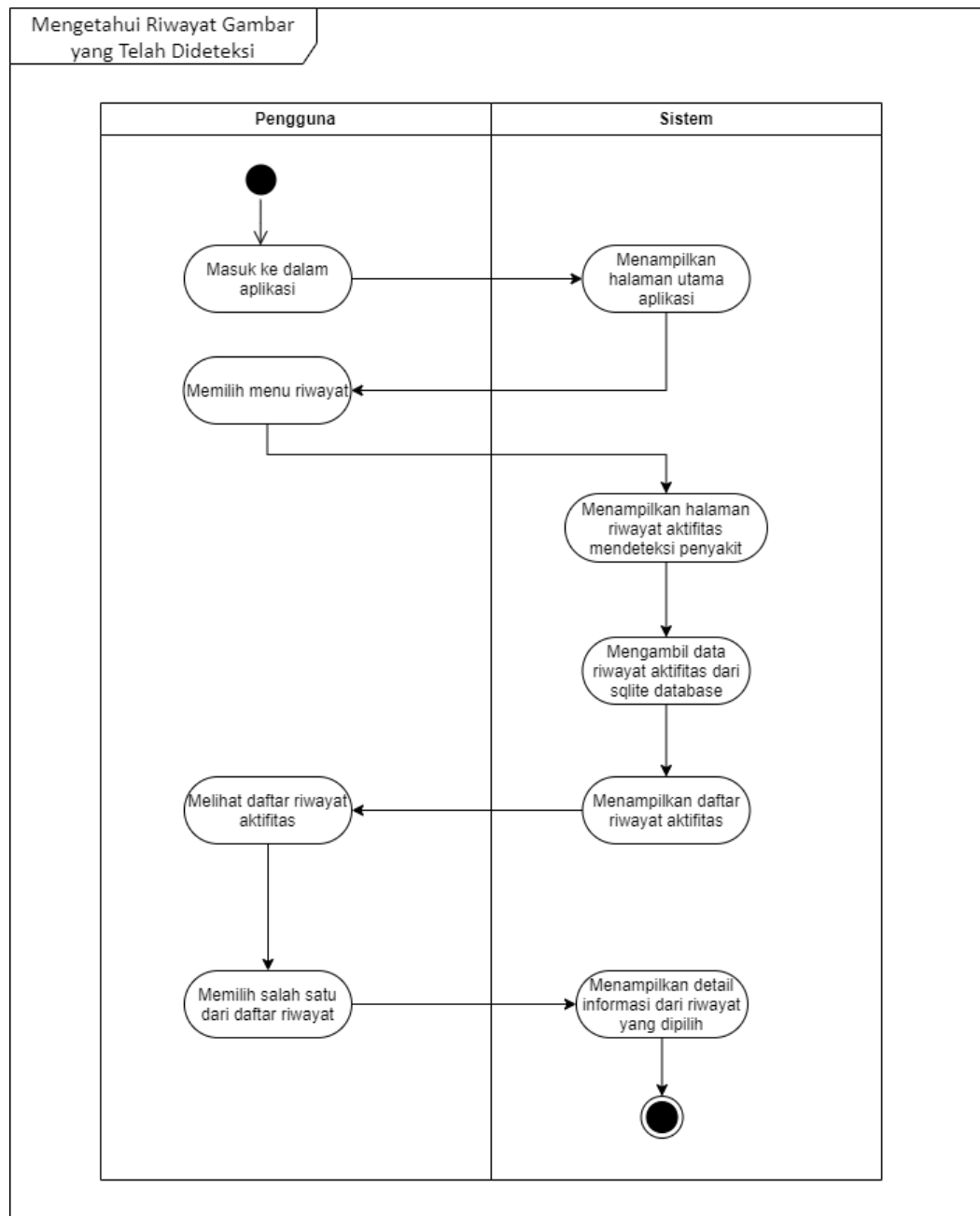
**Gambar 5.4 Activity Diagram Mengetahui informasi penyakit dan pengendalian penyakit**

### 5.2.4 Activity Diagram Mengetahui Riwayat Gambar yang Telah Dideteksi

Pada Gambar 5.5 pengguna masuk kedalam menu utama aplikasi dan sistem menampilkan menu utama dalam aplikasi. Selanjutnya pengguna memilih menu riwayat dan kemudian sistem menampilkan halaman riwayat aktifitas mendeteksi. Sistem mengambil data riwayat aktifitas dari sqlite database. Setelah data diakses, sistem kemudian menampilkan



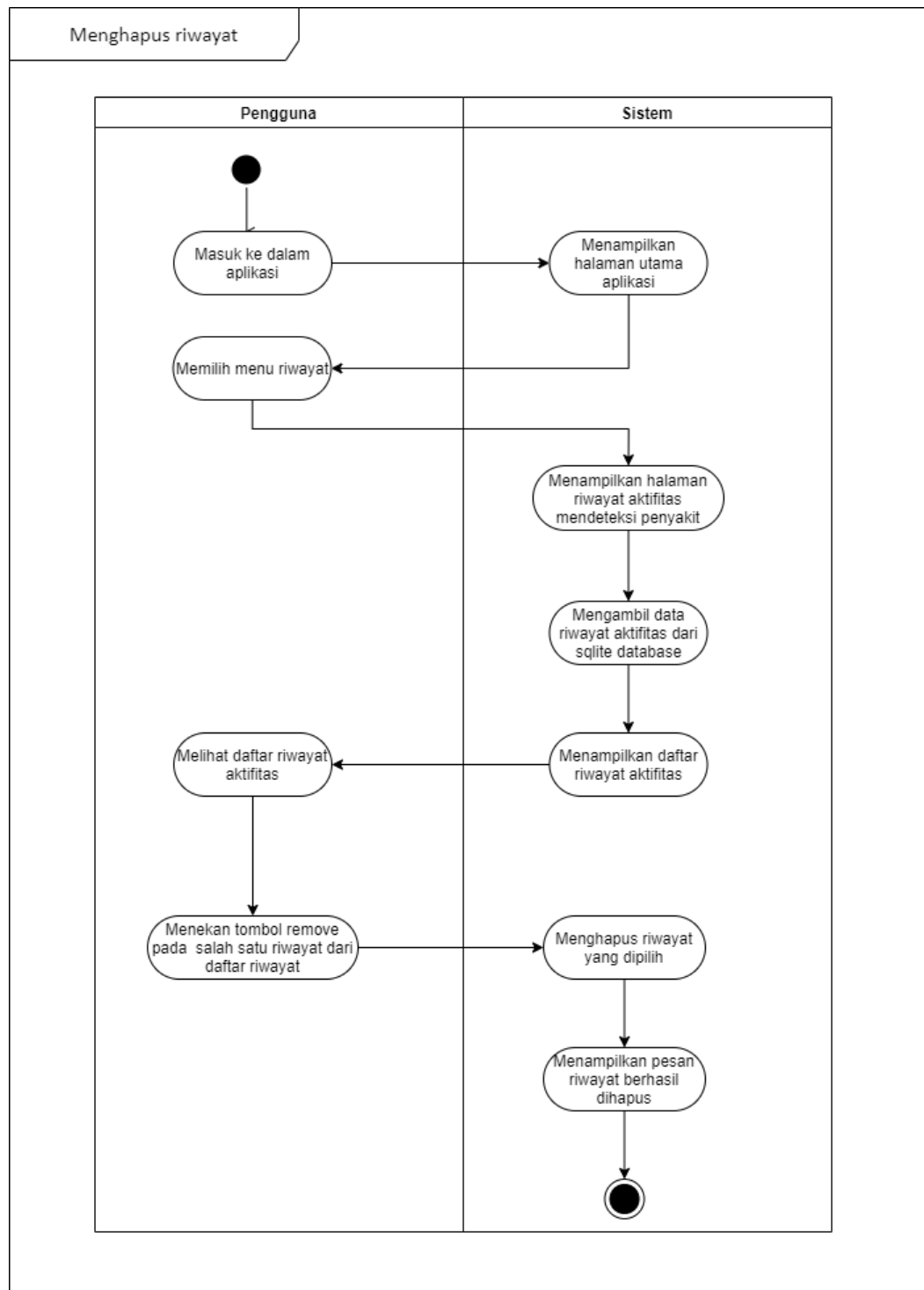
daftar riwayat. Setelah itu pengguna memilih salah satu dari daftar riwayat yang kemudian sistem menampilkan informasi sesuai riwayat yang dipilih.



**Gambar 5.5 Activity diagram Mengetahui Riwayat Gambar yang Telah Dideteksi**

### **5.2.5 Activity Diagram Menghapus Riwayat Hasil Analisis**

Pada Gambar 5.6 pengguna masuk kedalam menu utama aplikasi dan sistem menampilkan menu utama dalam aplikasi. Selanjutnya pengguna memilih menu riwayat dan kemudian sistem menampilkan halaman riwayat aktifitas mendeteksi. Sistem mengambil data riwayat aktifitas dari sqlite database. Setelah data diakses, sistem kemudian menampilkan daftar riwayat. Setelah itu pengguna menekan tombol remove pada salah satu dari daftar riwayat yang kemudian sistem akan menampilkan dialog persetujuan, apabila pengguna menekan pilihan yes maka sistem akan menghapus riwayat aktifitas yang dipilih dan sistem menampilkan pesan "Remove success". Selanjutnya sistem memperbarui daftar riwayat aktifitas pada halaman riwayat.



**Gambar 5.6 Activity Diagram Menghapus Riwayat Hasil Analisis**

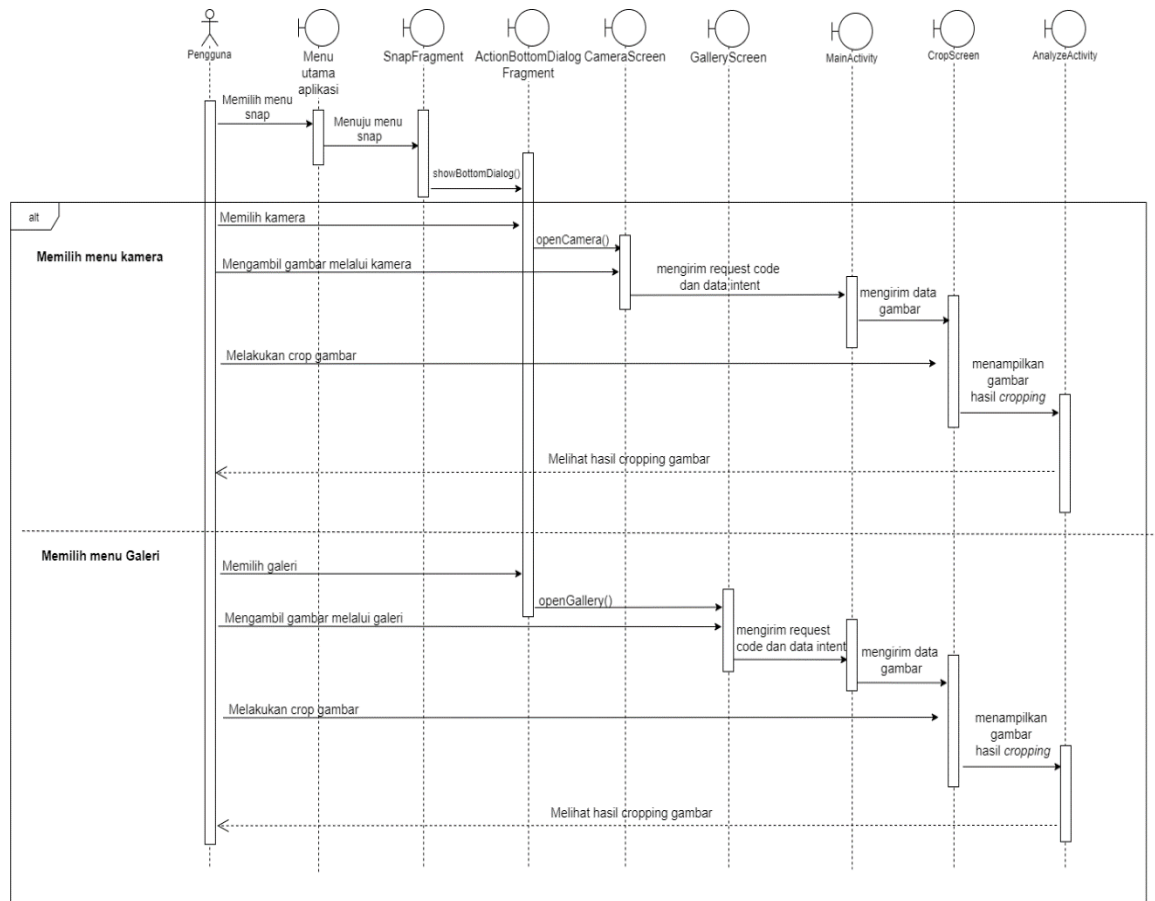
### 5.3 Perancangan *Sequence Diagram*

*Sequence diagram* adalah diagram yang menggambarkan aktivitas dan interaksi antar komponen secara berurutan. Semua komponen yang ada pada *sequence diagram* merupakan hasil dari analisis dan identifikasi dari kebutuhan serta skenario *use case* yang ada pada tahap analisis kebutuhan sebelumnya.

#### 5.3.1 *Sequence Diagram* Mendapatkan Gambar

Pada Gambar 5.7, Gambar 5.7 pengguna memilih menu snap pada MenuUtamaAplikasi selanjutnya sistem menampilkan halaman Snap. Kemudian muncul *ActionBottomDialogFragment* berupa tampilan yang ditampilkan kepada pengguna untuk memilih mengambil gambar menggunakan kamera atau galeri dalam *frame alternative*. Apabila pengguna memilih mengambil gambar menggunakan kamera maka Snap Activity akan memanggil method `openCamera()` untuk membuka `CameraScreen`. Setelah mengambil gambar melalui kamera maka selanjutnya data camera akan dikirim kepada `CropScreen` menggunakan method `startCrop()`. Setelah melakukan *image cropping*, hasil proses tersebut akan dikirimkan ke halaman `AnalyzeActivity`

Apabila pengguna memilih untuk mengambil gambar melalui galeri maka `SnapActivity` akan memanggil method `openGallery()` dan kemudian sistem akan menampilkan `GaleryScreen`. Setelah pengguna mengambil gambar melalui galeri maka pengguna melakukan *cropping image*. Setelah berhasil mengambil gambar dari galeri, pengguna menekan button centang pada `CropImageScreen`. Gambar yang telah melalui proses *image cropping* akan ditampilkan pada halaman `AnalyzeActivity`.



**Gambar 5.7 Sequence diagram mendapatkan gambar**

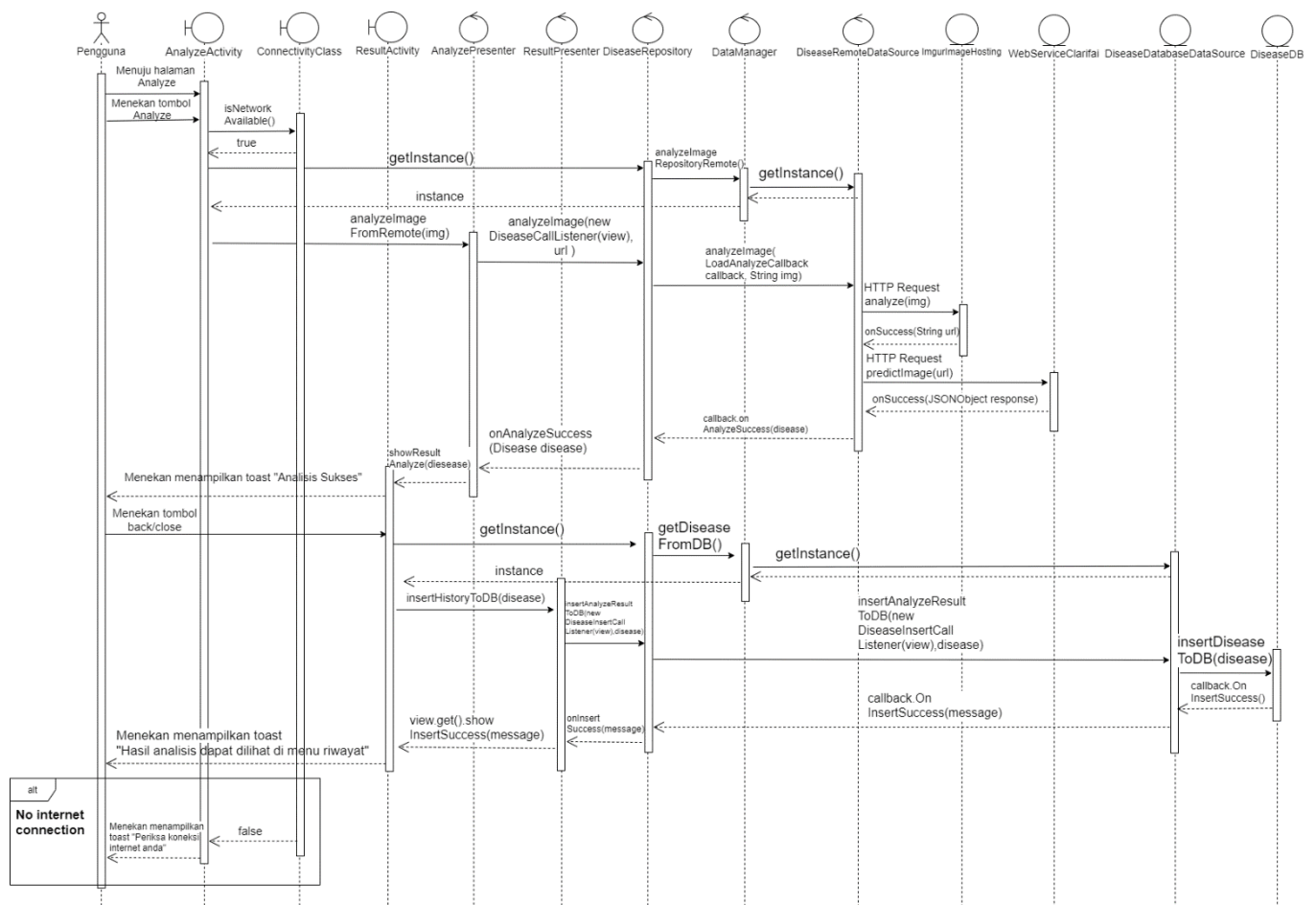
### 5.3.2 Sequence Diagram Mendeteksi Penyakit

Pada Gambar 5.8, pengguna sudah berada pada halaman *AnalyzeActivity*. Selanjutnya pengguna menekan tombol *analyze image*, maka sistem akan melakukan pengecekan koneksi internet pada *smartphone* pengguna dengan memanggil method *isNetworkAvailable()*. Apabila kembalian dari method tersebut adalah *false* maka sistem akan menampilkan pesan “Periksa koneksi internet anda”, apabila *true* maka *AnalyzeActivity* memanggil method *getInstance()* pada *DiseaseRepository* berlanjut ke *DiseaseRemoteDataSource* dan mengembalikan nilai *instance* kepada *AnalyzeActivity*.

Selanjutnya *AnalyzeActivity* memanggil method *analyzeImageFromRemote(img)* dari *AnalyzePresenter* method kemudian memanggil method *analyzeImage(new DiseaseCallListener(view), url)* pada *DiseaseRepository* dilanjutkan dengan peng-hosting-an gambar untuk mendapatkan *url image* dari website *imgurl*. Setelah *url image* didapatkan maka sistem menggunakan *url image* tersebut untuk dikirimkan kepada *WebServiceClarifai* untuk dilakukan pendeteksian penyakit pada gambar.

Setelah proses deteksi berhasil maka sistem akan mendapatkan response berupa JSONObject yang didalamnya terdapat nama penyakit dan tingkat akurasi dari hasil deteksi. Data tersebut diterima oleh DiseaseRemoteDataSource dan dikirimkan kepada DiseaseRepository melalui *callback method* yaitu `callback.onAnalyzeSuccess(disease)`. Setelah itu DiseaseRepository dilanjutkan proses pengiriman data kepada AnalyzePresenter dengan memanggil method `onAnalyzeSuccess(Disease disease)`. Selanjutnya ResultActivity akan menampilkan pesan “Analisis sukses” dan data hasil dari proses deteksi gambar berhasil ditampilkan pada pengguna.

Selanjutnya pengguna menekan tombol *back/close*, terjadi proses penyimpanan hasil deteksi kedalam database sqlite. ResultActivity memanggil method `insertHistorytoDB(disease)` pada ResultPresenter dan akan dikembalikan nilai sukses. Kemudian DiseaseDatabaseDataSource akan memanggil method `callback.onInsertSuccess(message)` pada DiseaseRepository. Kemudian DiseaseRepository akan memanggil method `onInsertSuccess(message)`. Kemudian method `showInsertSuccess(message)` di *override* pada ResultActivity. Maka pesan sistem akan menampilkan pesan kepada pengguna bahwa riwayat aktifitas dapat dilihat kembali pada menu riwayat.

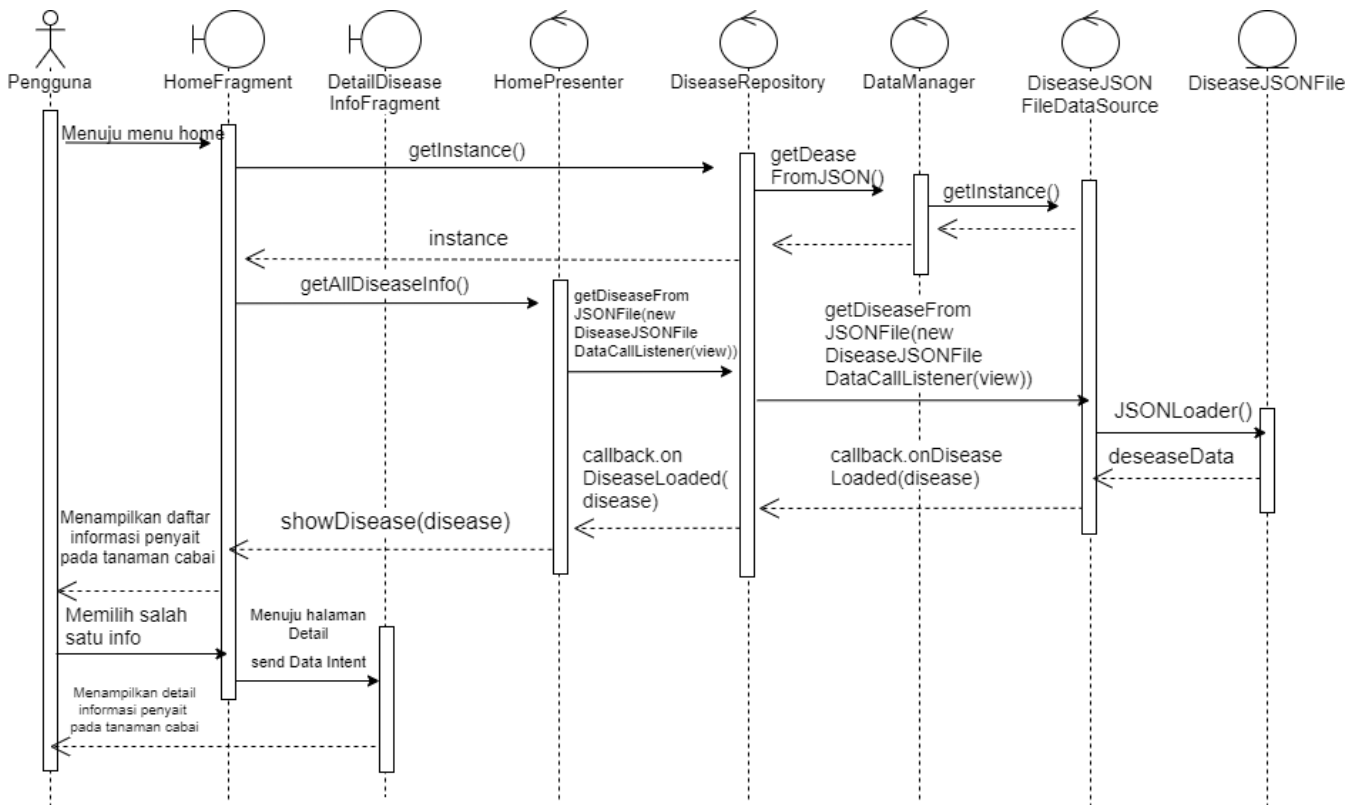


Gambar 5.8 Sequence diagram Mendeteksi penyakit

### 5.3.3 Sequence Diagram Mengetahui Informasi penyakit dan pengendalian Penyakit

Pada Gambar 5.9, Gambar 5.9 pengguna menuju HomeFragment, selanjutnya HomeFragment menginstansiasi *repository* dari DataManager. Dilanjutkan dengan pemanggilan method `getAllDiseaseInfo()` pada HomePresenter. Setelah itu HomePresenter memanggil method `getDiseaseFromJSONFile(new DiseaseJSONFileDataCallListener(view))` pada DiseaseRepository hal tersebut dilanjutkan dengan pemanggilan method dengan nama yang sama pada DiseaseJSONFileDataSource kemudian dilakukan pengambilan data informasi penyakit dari Disease\_JSONFile. Data tersebut akan kembali pada DiseaseRepository, HomePresenter dan akan ditampilkan pada HomeActivity yang kemudian

pengguna melihat daftar gambar penyakit pada tanaman cabai dan apabila di klik salah satu *item*, pengguna melihat detail dari penyakit tersebut.



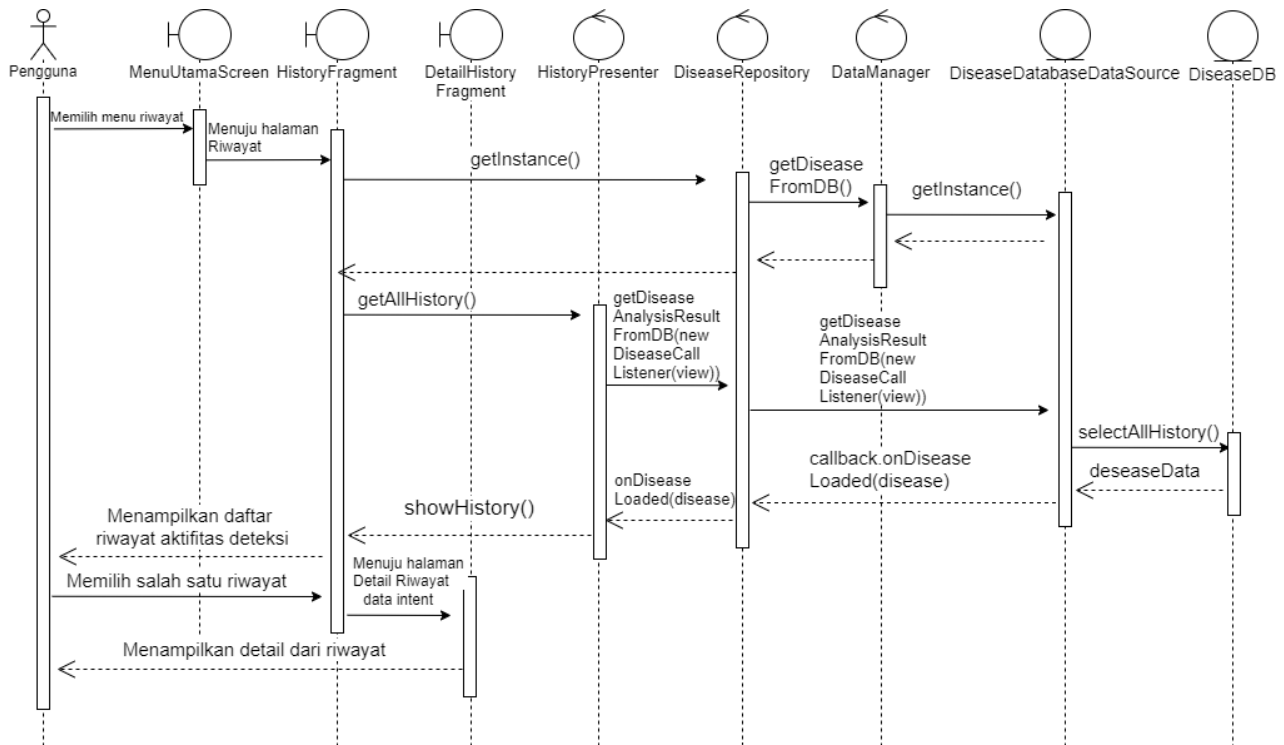
**Gambar 5.9 Sequence diagram Mengetahui Informasi penyakit dan pengendalian Penyakit**

### 5.3.4 Sequence Diagram Mengetahui Riwayat Gambar yang Telah Dideteksi

Pada Gambar 5.10, pengguna memilih menu HistoryFragment pada MenuUtamaScreen yang selanjutnya HistoryFragment memanggil method `getAllhistory()` pada HistoryPresenter yang sebelumnya sudah menginstansiasi repository dari Data Manager dengan melakukan pemanggilan method `getDiseaseFromDB()`. HistoryPresenter memanggil method `getDiseaseAnalyzeResultFromDB(new DiseaseCallListener(view))` pada DiseaseRepository. Pemanggilan method dengan nama yang sama juga dilakukan pada DiseaseDatabaseDataSource dan kemudian mengambil data riwayat aktifitas dari DiseaseDB dengan menggunakan method `selecAllHistory()`. Data yang berhasil diambil akan dikembalikan sesuai gambar dibawah dan pada HistoryFragment, data sampai melalui method



`showHistory(disease)` selanjutnya data diolah sehingga dapat tampil kepada pengguna.



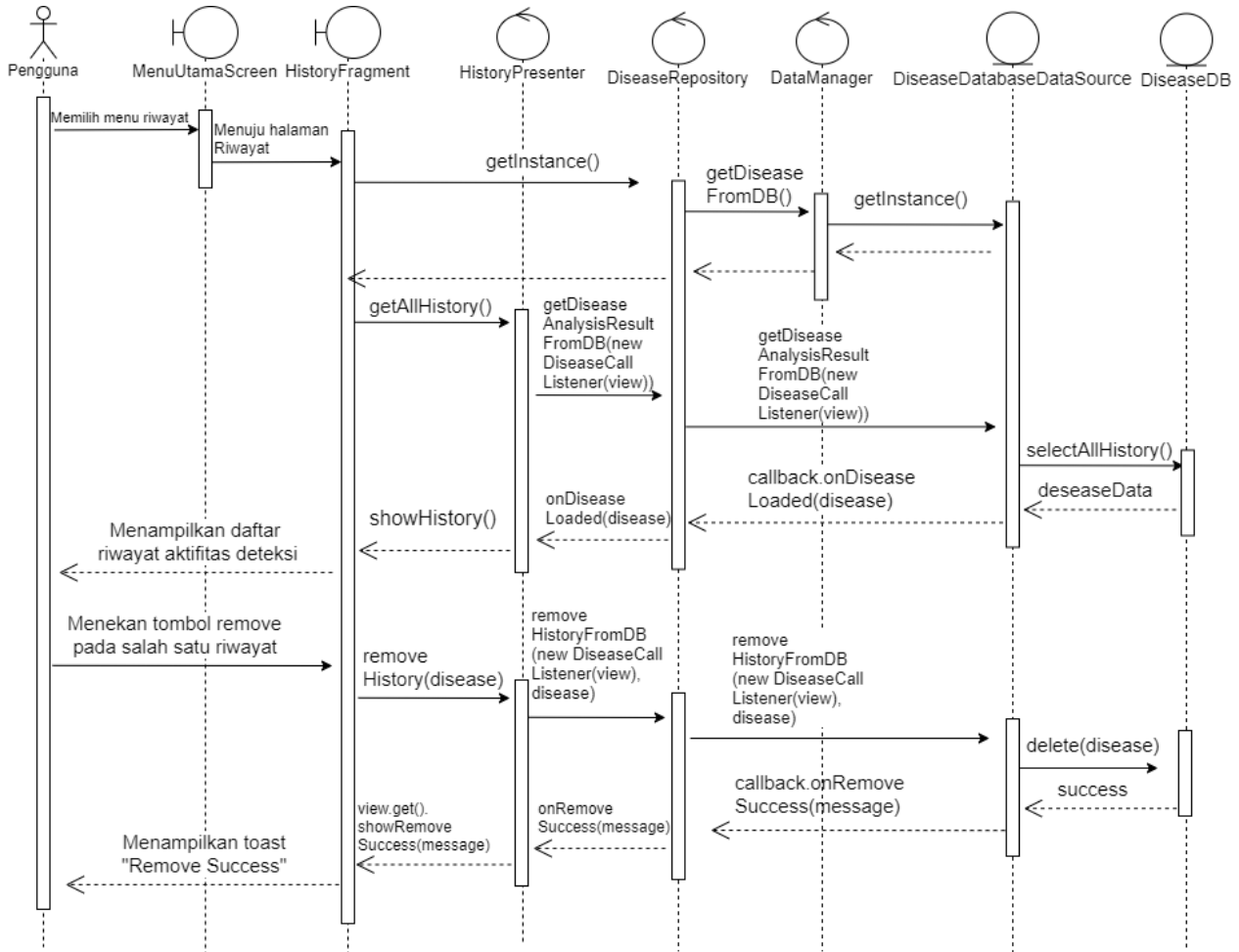
**Gambar 5.10 Sequence diagram Mengetahui riwayat gambar yang telah dideteksi**

### 5.3.5 Sequence Diagram Menghapus Riwayat Hasil Deteksi

Pada Gambar 5.11, pengguna memilih menu HistoryFragment pada MenuUtamaScreen yang selanjutnya HistoryFragment memanggil method `getAllhistory()` pada HistoryPresenter yang sebelumnya sudah menginstansiasi *repository* dari Data Manager dengan melakukan pemanggilan method `getDiseaseFromDB()`. HistoryPresenter memanggil method `getDiseaseAnalyzeResultFromDB(new DiseaseCallListener(view))` pada DiseaseRepository. Pemanggilan method dengan nama yang sama juga dilakukan pada DiseaseDatabaseDataSource dan kemudian mengambil data riwayat aktifitas dari DiseaseDB dengan menggunakan method `selecAllHistory()`. Data yang berhasil diambil akan dikembalikan sesuai gambar dibawah dan pada HistoryFragment, data sampai melalui method `showHistory(disease)` selanjutnya data diolah sehingga dapat tampil kepada pengguna.

Selanjutnya pengguna menekan tombol *remove* pada salah satu item. Maka HistoryFragment memanggil method `removeHistory(disease)` pada

HistoryPresenter. Dilanjutkan pemanggilan method `removeHistoryFromDB(new DiseaseCallListener(view), disease)` pada `DiseaseRepository`. Pemanggilan method dengan nama yang sama dilakukan pada `DiseaseDatabaseDataSource` kemudian mulai menghapus riwayat pada database menggunakan method `delete(disease)`. Dan sistem akan menampilkan pesan kepada pengguna berupa pesan "Remove success".



**Gambar 5.11 Sequence diagram Menghapus riwayat hasil deteksi**

## 5.4 Perancangan Class Diagram

Pada bagian perancangan class diagram dijelaskan struktur dari sistem aplikasi DiSnap. Dikarenakan pada sistem menggunakan arsitektur MVP maka terdapat tiga package utama yaitu, *Model*, *View*, dan *Presenter* yang dimana tiap-tiap package memuat kelas-kelas sesuai fungsinya masing-masing. Kelas-kelas yang satu dengan yang lain saling berhubungan untuk menjalankan fungsionalitas tertentu. Pada bagian ini tidak akan dimasukkan iterasi hanya dimasukkan hasil akhir dari kebutuhan pengguna. Iterasi hanya dilakukan pada pembuatan

```

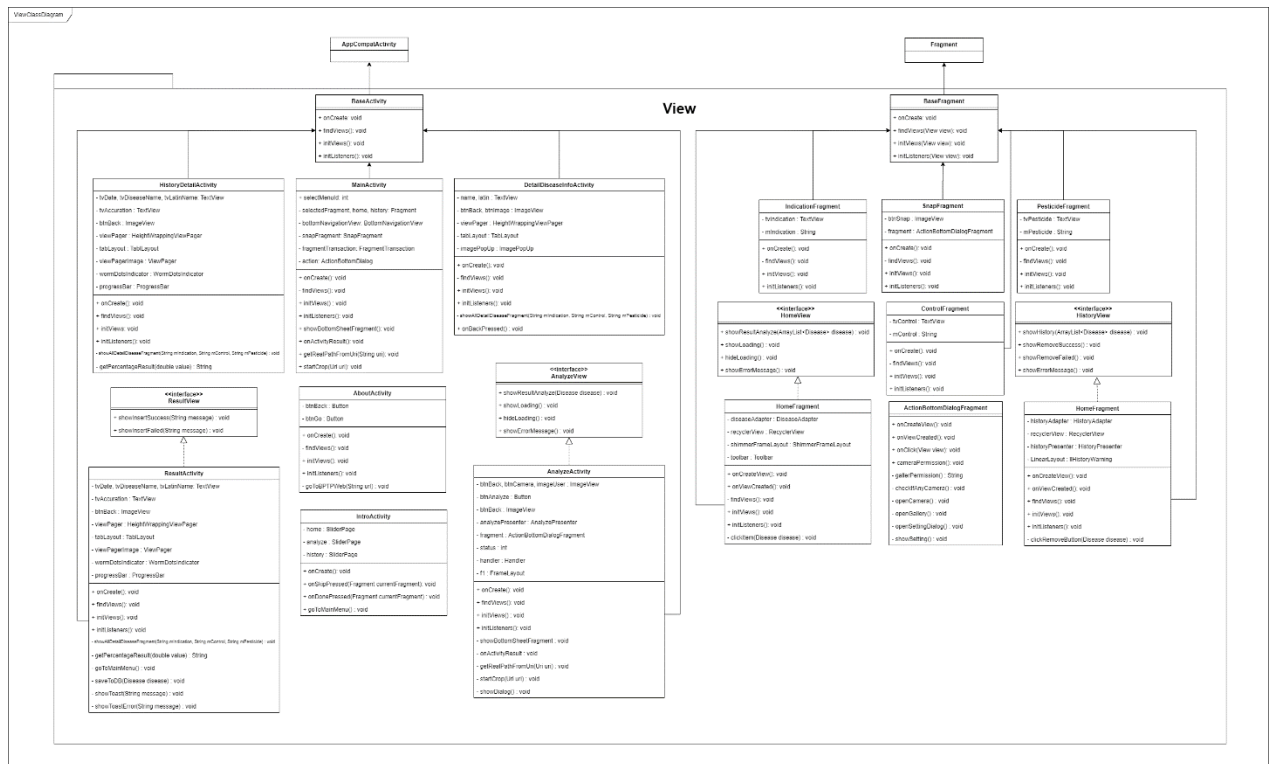
classDiagram
    class AppCompatActivity
    class BaseActivity
    class MainActivity
    class HistoryDetailActivity
    class AboutActivity
    class ResultActivity
    class ResultView
    class DetailDiseaseInfoActivity
    class SplashScreenActivity
    class IntroActivity
    class AnalyzeActivity
    class AnalyzeView
    class ViewPagerAdapterAdapter
    class BaseFragment
    class IndicationFragment
    class SnapFragment
    class PesticideFragment
    class ControlFragment
    class HomeView
    class HomeFragment
    class ActionBottomDialogFragment
    class HistoryFragment
    class HistoryView
    class DiseaseAdapter
    class HistoryAdapter
    class ResultPresenter
    class AnalyzePresenter
    class HomePresenter
    class HistoryPresenter
    class DiseaseRepository
    class DiseaseDataSource
    class DiseaseDatabaseDataSource
    class AppDatabase
    class Remote
    class DiseaseRemoteDataSource
    class DataManager
    class JSONFile
    class DiseaseJSONFileDataSource
    class Disease
    class Connectivity
    class Constant
    class DiskExecutor

    AppCompatActivity --> BaseActivity
    BaseActivity --> MainActivity
    BaseActivity --> HistoryDetailActivity
    BaseActivity --> AboutActivity
    BaseActivity --> ResultActivity
    BaseActivity --> DetailDiseaseInfoActivity
    BaseActivity --> SplashScreenActivity
    BaseActivity --> IntroActivity
    BaseActivity --> AnalyzeActivity
    ResultActivity ..|> ResultView
    AnalyzeActivity ..|> AnalyzeView
    AnalyzeActivity --|> IntroActivity
    ViewPagerAdapterAdapter --> HomeFragment
    ViewPagerAdapterAdapter --> ActionBottomDialogFragment
    ViewPagerAdapterAdapter --> HistoryFragment
    BaseFragment --> IndicationFragment
    BaseFragment --> SnapFragment
    BaseFragment --> PesticideFragment
    BaseFragment --> ControlFragment
    BaseFragment --> HomeView
    BaseFragment --> HomeFragment
    BaseFragment --> ActionBottomDialogFragment
    BaseFragment --> HistoryFragment
    BaseFragment --> HistoryView
    HomeFragment --|> HomeView
    HistoryFragment ..|> HistoryView
    DiseaseAdapter --> HomeFragment
    HistoryAdapter --> HistoryFragment
    ResultPresenter --> ResultActivity
    AnalyzePresenter --> AnalyzeActivity
    HomePresenter --> HomeFragment
    HistoryPresenter --> HistoryFragment
    DiseaseRepository --|> DiseaseDataSource
    DiseaseRepository --|> DiseaseDatabaseDataSource
    DiseaseRepository --|> Remote
    DiseaseRepository --|> DiseaseJSONFileDataSource
    DiseaseDatabaseDataSource --|> AppDatabase
    Remote --|> DiseaseRemoteDataSource
    DataManager --|> JSONFile
    DiseaseJSONFileDataSource --|> JSONFile
    Disease --|> Connectivity
    Disease --|> Constant
    Disease --|> DiskExecutor
  
```

The diagram illustrates the architecture of an Android application, organized into several layers:

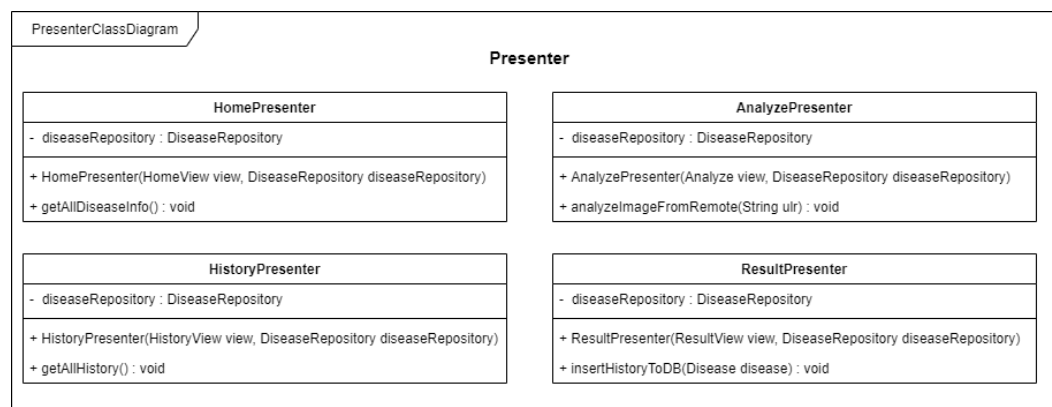
- View Layer:**
  - Activities:** `AppCompatActivity` is the base for `BaseActivity`. `BaseActivity` is the base for `MainActivity`, `HistoryDetailActivity`, `AboutActivity`, `ResultActivity`, `DetailDiseaseInfoActivity`, `SplashScreenActivity`, `IntroActivity`, and `AnalyzeActivity`.
  - Fragments:** `BaseFragment` is the base for `IndicationFragment`, `SnapFragment`, `PesticideFragment`, `ControlFragment`, `HomeView` (which implements `<<interface>> HomeView`), `HomeFragment` (which implements `<<interface>> HomeView`), `ActionBottomDialogFragment`, `HistoryFragment` (which implements `<<interface>> HistoryView`), and `HistoryView` (which implements `<<interface>> HistoryView`).
  - Adapters:** `ViewPagerAdapterAdapter` is associated with `HomeFragment`, `ActionBottomDialogFragment`, and `HistoryFragment`. `DiseaseAdapter` is associated with `HomeFragment`, and `HistoryAdapter` is associated with `HistoryFragment`.
- Presenter Layer:**
  - `ResultPresenter` is associated with `ResultActivity`.
  - `AnalyzePresenter` is associated with `AnalyzeActivity`.
  - `HomePresenter` is associated with `HomeFragment`.
  - `HistoryPresenter` is associated with `HistoryFragment`.
- Repository Layer:**
  - `DiseaseRepository` is the central interface, implemented by `DiseaseDataSource`, `DiseaseDatabaseDataSource` (which implements `<<interface>> DiseaseDataSource`), `Remote` (which implements `<<interface>> DiseaseDataSource`), and `DiseaseJSONFileDataSource` (which implements `<<interface>> DiseaseDataSource`).
  - `DiseaseDatabaseDataSource` is associated with `AppDatabase`.
  - `Remote` is associated with `DiseaseRemoteDataSource`.
  - `DiseaseJSONFileDataSource` is associated with `JSONFile`.
- Model Layer:**
  - `Disease` is the core data model.
- Util Layer:**
  - `Connectivity`, `Constant`, and `DiskExecutor` are utility classes.

**Gambar 5.12 Class Diagram DiSnap**



**Gambar 5.13 Class Diagram Package View**

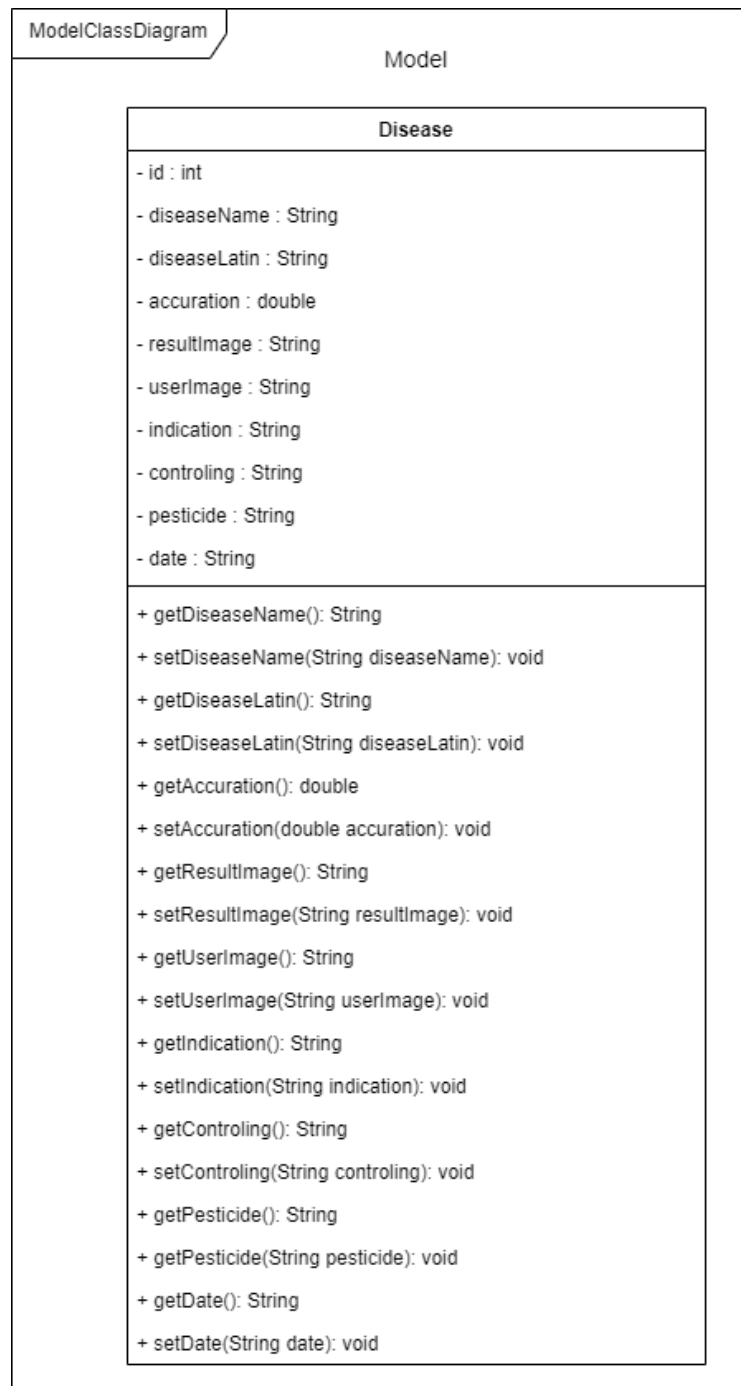
Pada Gambar 5.13, menunjukkan package view yang ada pada sistem, terdapat beberapa kelas seperti MainActivity, ResultActivity, HistoryDetailActivity, DetailDiseaseInfoActivity, AnalyzeActivity yang meng-extends BaseActivity. Adapula BasicFragment, HomeFragment, IndicationFragment, ControllingFragment, PesticideFragment, HistoryFragment yang meng-extends BasicFragment.



### **Gambar 5.14 Class Diagram Package Presenter**

Pada Gambar 5.14, terdapat beberapa *class Presenter* yang berfungsi untuk mengatur mengatur jalur pertukaran data pada sistem yang berhubungan dengan data pada *layer DiseaseRepository*. Ada beberapa class Presenter seperti `HomePresenter`, `AnalyzePresenter`, `HistoryPresenter`, dan `ResultPresenter`.

`HomePresenter` berfungsi untuk mengambil data info penyakit pada `Disease_JSON file` berupa gambar dan nama penyakit untuk ditampilkan pada halaman utama. `AnalyzePresenter` berfungsi melakukan pengiriman data untuk dilakukan pendeteksian penyakit pada gambar melalui `Clai fai API`. `ResultPresenter` berfungsi untuk menyimpan riwayat hasil deteksi kedalam database `sqlite`. `HistoryPresenter` berfungsi untuk melakukan pengambilan data dari database dan melakukan penghapusan data riwayat hasil deteksi pada database.



**Gambar 5.15 Class Diagram Package Model**

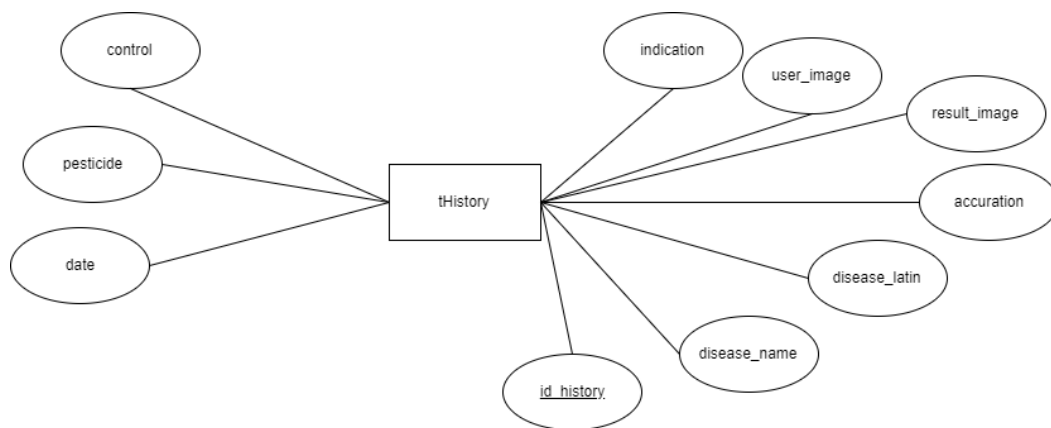
Pada Gambar 5.15, Gambar 5.15 terdapat satu *class model* yaitu Disease. Pada class tersebut terdapat beberapa atribut seperti id : int, diseaseName: String, diseaseLatin: String, accuration: double, resultImage: String, userImage: String, indication: String, controlling : String, pesticide : String, date : String. Kelas ini digunakan untuk membantu menyimpan dan mengolah data pada database serta membantu menampilkan info penyakit kepada pengguna.

## 5.5 Perancangan Basis Data

Pada bagian ini membahas tentang perancangan basis data yang digunakan dalam proses implementasi sistem yang akan dibuat. Perancangan basis data terdiri dari perancangan ERD (*Entity Relationship Diagram*) dan tabel. Pada bagian ini tidak terdapat iterasi dikarenakan fungsionalitas sudah sesuai dengan pengguna. Iterasi terdapat pada perancangan antar muka berdasarkan temuan masalah yang ada.

### 5.5.1 Perancangan ERD

Pada Gambar 5.16 menunjukkan ERD dari aplikasi Disese Snap. Aplikasi ini memiliki satu entitas yaitu entitas tHistory. Entitas tHistory memiliki *primary key* yaitu id\_history. Atribut yang dimiliki oleh entitas tHistory dapat dilihat pada Gambar 5.16.



Gambar 5.16 Entity Relationship Diagram DiSnap

### 5.5.2 Perancangan Tabel

#### 1. Tabel History

Pada Tabel 5.1 berisi rancangan tabel basis data penyakit yang memiliki kolom sebanyak 10 (sepuluh) kolom dan memiliki tipe data yang berbeda; seperti seperti integer, varchar. Panjang dari tipe data tersebut 11 hingga 255.

Nama tabel: tHistory

Nama kelas : Disease

Nama database : DiseaseDB

Pengguna Fungsi: Menyimpan data riwayat aktifitas deteksi penyakit

Tabel 5.1 Rancangan tabel basis data DiSnap

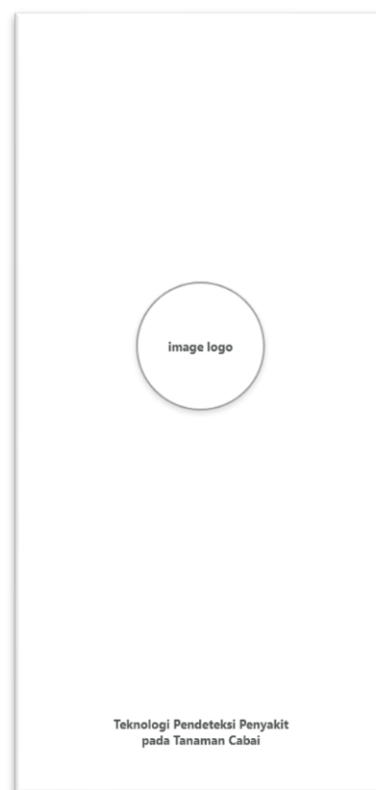
No	Nama Kolom	Tipe Data	Panjang	Keterangan	Auto Generate	Primary Key
1.	id_history	integer	11	NOT NULL	true	tru
2.	disease_name	varchar	255	NULL	false	false

3.	disease_latin	varchar	255	<i>NOT NULL</i>	<i>false</i>	<i>false</i>
4.	accuration	varchar	255	<i>NULL</i>	<i>false</i>	<i>false</i>
5.	result_image	varchar	255	<i>NULL</i>	<i>false</i>	<i>false</i>
6.	user_image	varchar	255	<i>NULL</i>	<i>false</i>	<i>false</i>
7.	indication	varchar	255	<i>NULL</i>	<i>false</i>	<i>false</i>
8.	control	varchar	255	<i>NULL</i>	<i>false</i>	<i>false</i>
9.	pesticide	varchar	255	<i>NULL</i>	<i>false</i>	<i>false</i>
10.	date	varchar	255	<i>NULL</i>	<i>false</i>	<i>false</i>

## 5.6 Perancangan Antarmuka Pengguna (Wireframe)

Perancangan antarmuka yang dilakukan pada aplikasi DiSnap untuk memberikan gambaran kepada peneliti untuk memudahkan dalam mengetahui tata letak dan tampilan pada aplikasi. Peneliti membuat perancangan antarmuka berupa *wireframe* dengan menggunakan Adobe XD. Wireframe yang dibuat ditunjukkan pada gambar dibawah ini mulai dari Gambar 5.17 sampai Gambar 5.25.

### 5.6.1 Perancangan Antarmuka SplashScreen





### Gambar 5.17 Wireframe SplashScreen

Pada Gambar 5.17, merupakan halaman SplashScreen seperti aplikasi pada umumnya ketika pengguna membuka aplikasi maka akan muncul logo aplikasi. Pada aplikasi DiSnap peneliti merancang untuk membuat halaman SplashScreen yang berisikan logo aplikasi dengan posisi di tengah layar serta deskripsi singkat dari aplikasi DiSnap pada bagian bawah aplikasi.

### 5.6.2 Perancangan Antarmuka Home



Gambar 5.18 Wireframe Home

Pada Gambar 5.18 menunjukkan halaman *home*. Pada halaman tersebut terdapat logo dari aplikasi serta nama aplikasi disebelah kanan logo. Terdapat menu navigasi yang disebut dengan bottom navigation yang memiliki 3 menu utama yaitu *home*, *snap*, dan *about apps*. Pada menu home disediakan list berupa informasi penyakit dalam bentuk *rounded radius card*. Ketika salah informasi diklik maka pengguna akan melihat detail informasi pada halaman detail.

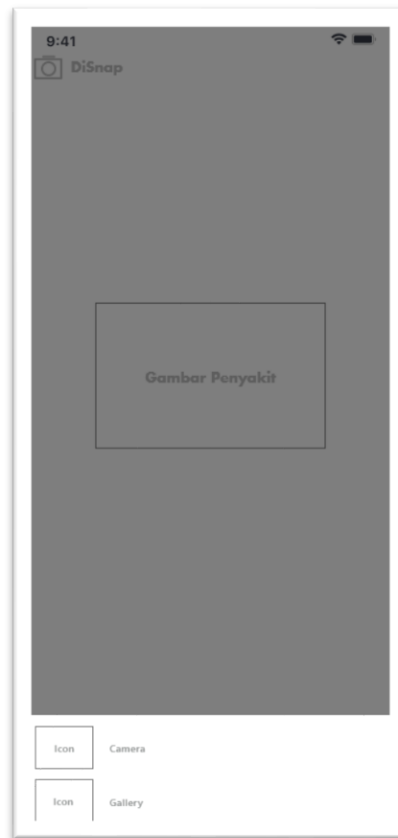
### 5.6.3 Perancangan Antarmuka *Detail*



**Gambar 5.19 Wireframe *Detail***

Pada Gambar 5.19 menunjukkan halaman *detail* pada informasi yang dipilih sebelumnya oleh pengguna pada menu *home*. Pada halaman detail terdapat appbar yang terdiri dari bagian yaitu *icon x letter* untuk membawa pengguna ke halaman *home*. Serta *textview* atau *title bar* dengan nama *Detail*. Pada bagian tengah atas terdapat dua *textview* secara vertical yaitu nama penyakit dan nama latin penyakit. Dibawah text tersebut terdapat *imageview* yaitu contoh gambar tanaman yang terkena penyakit. Dibagian bawah gambar merupakan keterangan berupa gejala dan pengendalian dari penyakit.

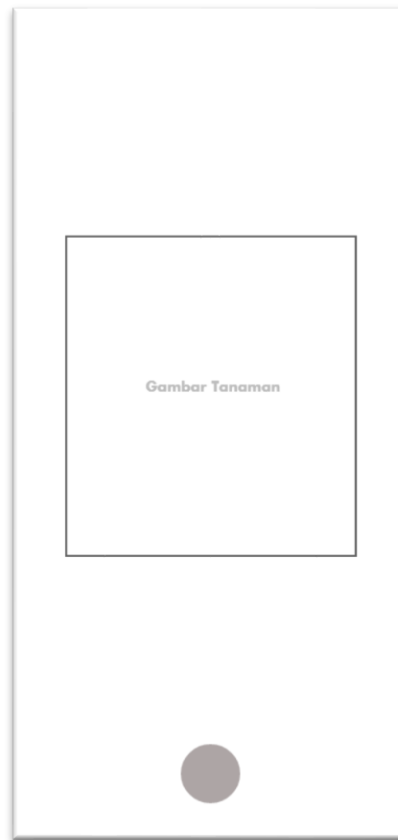
#### 5.6.4 Perancangan Antarmuka *Snap*



**Gambar 5.20 Wireframe snap(bottomsheet)**

Pada Gambar 5.20 menunjukkan halaman snap ketika menu snap pada menu utama bottom navigation view diklik oleh pengguna. Pada halaman ini sistem menampilkan dua pilihan kepada pengguna untuk memilih mendapatkan gambar melalui kamera atau galeri.

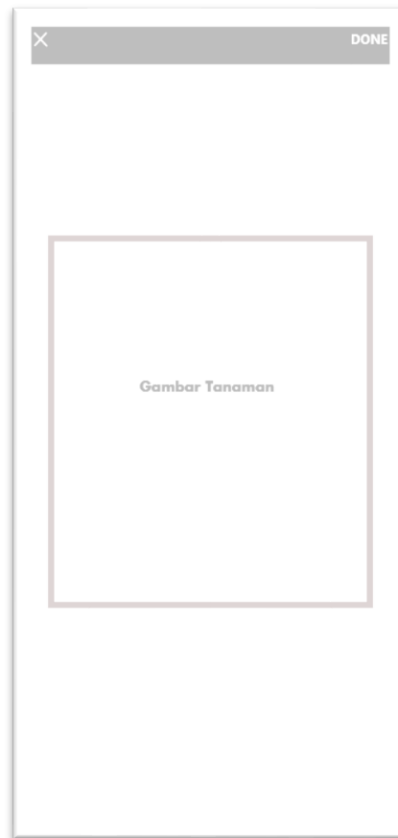
### 5.6.5 Perancangan Antarmuka Camera



**Gambar 5.21 Wireframe Camera**

Pada Gambar 5.21 menunjukkan aktifitas mengambil gambar tanaman menggunakan kamera. Pada aktifitas ini pengguna dapat mengarahkan atau memfokuskan kamera pada kotak fokus yang telah tersedia yang selanjutnya dapat memudahkan aktifitas cropping image pada tahap selanjutnya setelah *button* lingkaran pada kamera ditekan.

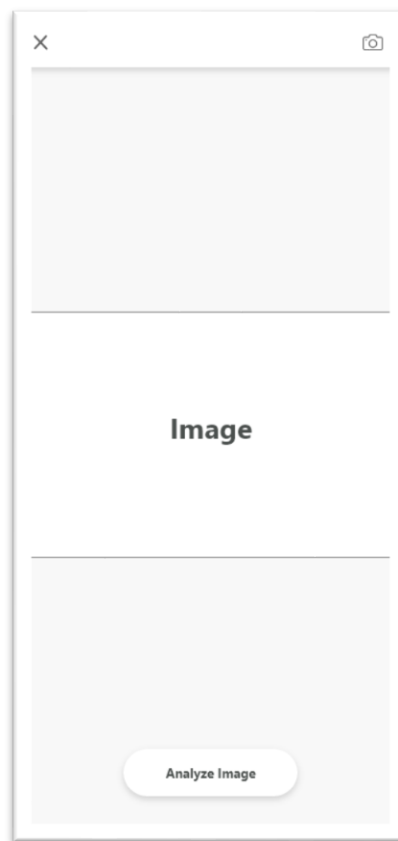
### 5.6.6 Perancangan Antarmuka *Cropping Image*



**Gambar 5.22 Wireframe *Cropping Image***

Pada Gambar 5.22 menunjukkan halaman *cropping image* yaitu aktifitas pengguna untuk memfokuskan bagian tanaman cabai yang ingin dideteksi penyakitnya oleh pengguna. Pada halaman ini pengguna dapat menggerakkan persegi hitam yang menjadi fokus gambar yang ingin di potong.

### 5.6.7 Perancangan Antarmuka *Analyze*



**Gambar 5.23 Wireframe *Analyze***

Pada Gambar 5.23 menunjukkan halaman *Analyze* gambar hasil dari mendapatkan gambar melalui kamera atau galeri yang sudah di lakukan *image cropping*. Pada halaman terdapat *appbar* yang terdiri dari dua bagian yaitu *icon x* (close) yang berguna untuk membatalkan aktifitas dan *button cange image* untuk mengganti gambar yang ada. Selanjutnya ada *button analyze image* yang berfungsi untuk memulai aktifitas analisis gambar dengan meng-*hosting* gambar melalui website *imgurl* kemudian mengirimkan gambar ke Webservice API Clarifai.

### 5.6.8 Perancangan Antarmuka *Result*

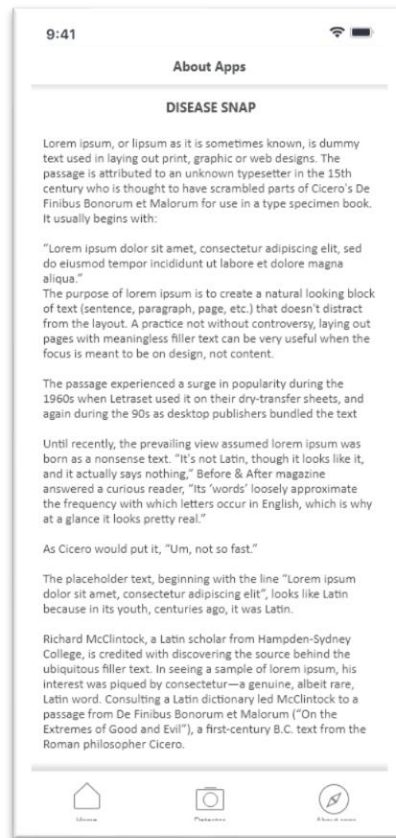


**Gambar 5.24 Wireframe Result**

Pada Gambar 5.24 menunjukkan halaman *result* yaitu halaman ketika hasil analisis gambar dari Webservie API Clarifai dikembalikan maka datanya akan diolah dan akan ditampilkan kepada pengguna. Pada halaman terdapat *AppBar* yang terdiri dari dua bagian yaitu *icon x* yang berguna untuk membatalkan aktifitas dan *icon camera* untuk memulai dari awal aktifitas mendeteksi penyakit. Selanjutnya terdapat *ImageView* dengan bentuk lingkaran yang isinya gambar tanaman yang terkena penyakit sesuai dengan namanya yang sesuai dengan database. Selain terdapat *ImageView* lain yang berbentuk kotak yang berisi salah satu gambar tanaman yang sudah diupload.

Pada bagian bawah terdapat 3 *TextView* yang berisi nama penyakit, nama latin penyakit dan tingkat rata-rata akurasi dari hasil deteksi. Pada halaman ini juga terdapat *Tab* yang memiliki dua bagian yaitu tab yang berisi deskripsi dari penyakit yang ada pada tanaman yang telah dideteksi dan pada tab kedua yaitu berisi tab penanganan yaitu informasi yang ditunjukkan kepada pengguna sebagai panduan dalam menangani penyakit pada tanaman yang telah dideteksi.

### 5.6.9 Perancangan Antarmuka *About Apps*



**Gambar 5.25 Wireframe *About Apps***

Pada Gambar 5.25 menunjukkan halaman *about apps*. Halaman ini memberikan informasi pengguna terhadap aplikasi DiSnap. Pada halaman ini terdapat *AppBar* yang berisi tulisan *About Apps*. Setelah itu dibawah *AppBar* terdapat *textView* dengan tulisan *DiSnap* yang merupakan nama dari aplikasi yang dibuat oleh peneliti. Kemudian terdapat *textView* yang berisi keterangan dari aplikasi ini. Terdapat juga *bottom navigation view* sebagai navigasi utama aplikasi ini.

### 5.7 Perancangan Antarmuka Pengguna (Wireframe) iterasi 1

Pada pengembangan aplikasi DiSnap ini terdapat iterasi dalam hal perancangan antarmuka pengguna. Hasil iterasi diperoleh dari walktrough aplikasi yang dilakukan oleh pengguna. Perancangan antarmuka pengguna iterasi 1 ini akan direpresentasikan oleh wireframe. Evaluasi dilakukan dengan melakukan persentasi dan pengecekan secara langsung oleh expert pakar penyakit tanaman. Berikut daftar temuan masalah yang ditunjukkan pada Tabel 5.2.



**Tabel 5.2 Temuan masalah wireframe DiSnap iterasi 1**

Kode Masalah	Deskripsi Masalah	Saran Perbaikan
M1	Tidak terdapat informasi pemberian pestisida pada tab informasi penyakit	Menambah tab pestisida pada informasi penyakit
M2	Tidak terdapat menu riwayat pada menu utama	Mengganti menu about apps dengan menu riwayat
M3	Peletakan menu about apps kurang sesuai.	memindahkan menu <i>about apps</i> pada toolbar di halaman home

### 5.7.1 Wireframe perbaikan tab informasi pestisida

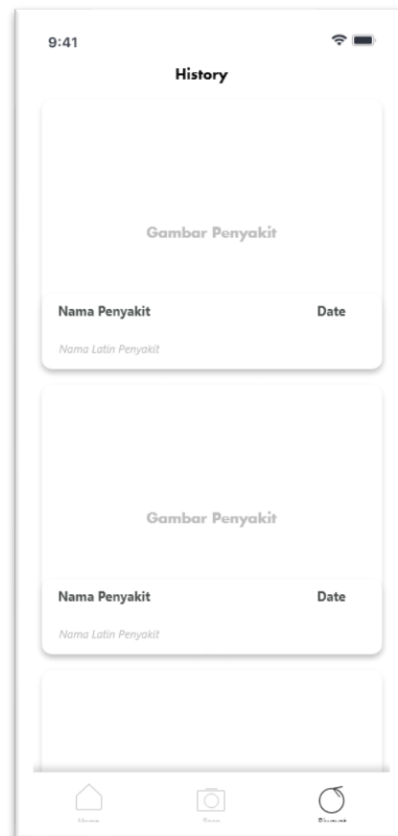
Permasalahan yang muncul adalah tidak adanya informasi mengenai pemberian pestisida yang cukup penting bagi pengguna. Maka dilakukan penambahan tan pestisida pada informasi penyakit. Wireframe tersebut ditunjukkan pada Gambar 5.26.



**Gambar 5.26 Wireframe perbaikan tab informasi penyakit**

### 5.7.2 Wireframe perbaikan mengganti menu *about apps*

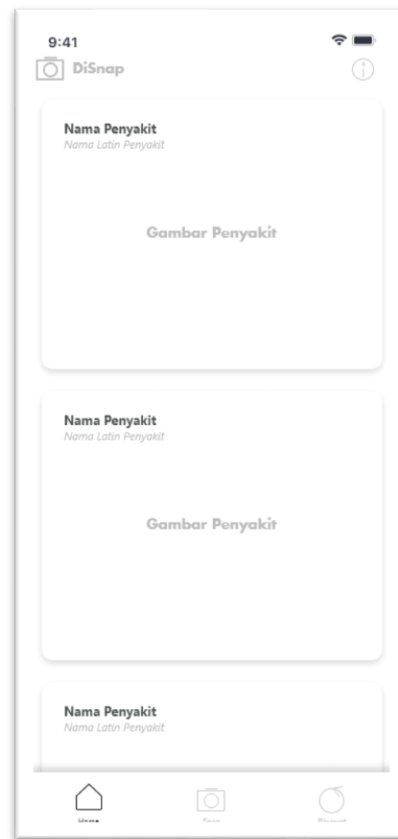
Permasalahan yang ada adalah tidak terdapatnya menu history dan menu about apps kurang menjadi prioritas penggunaan oleh pengguna, maka dapat diganti posisi menu dari about apps menjadi menu riwayat. Hasil dari perbaikan wireframe ditunjukkan pada Gambar 5.27.



Gambar 5.27 Wireframe menu riwayat

### 5.7.3 Wireframe perbaikan memindahkan menu *about apps* pada halaman home

Pada Gambar 5.28, sama seperti menu komponen home sebelumnya, akan terdapat satu perbedaan yaitu peletakan icon *about app* yang terdapat pada *toolbar* pada halaman *home*.



**Gambar 5.28 Wireframe menu home**

## 5.8 Perancangan Antarmuka Pengguna (Wireframe) iterasi 2

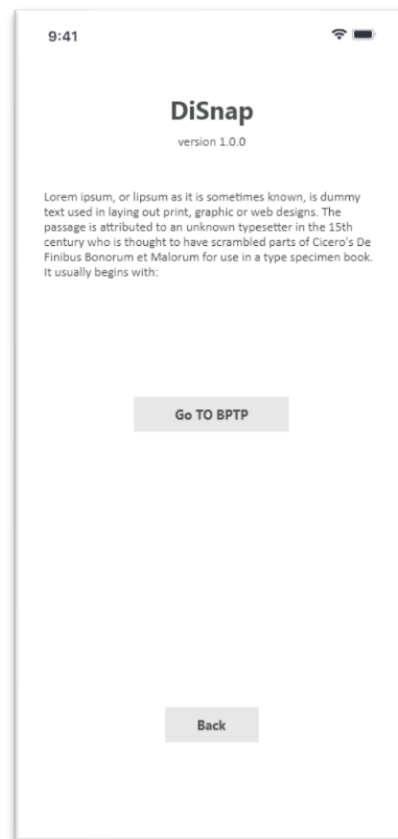
**Tabel 5.3 Temuan masalah *wireframe* DiSnap iterasi 2**

Kode Masalah	Deskripsi Masalah	Saran Perbaikan
M4	Tidak ada link menuju website BPTP Jawa Timur	Menambahkan tombol untuk menuju website BPTP Jawa Timur pada halaman <i>about apps</i>
M5	Tidak terdapat tombol hapus riwayat	Mengganti menu <i>about apps</i> dengan menu <i>history</i>
M6	Tidak terdapat <i>intro</i> aplikasi untuk memudahkan pengguna mengenali aplikasi	Membuat <i>intro</i> penjelasan singkat mengenai fitur pada

		DiSnap di awal menggunakan aplikasi
--	--	-------------------------------------

### 5.8.1 Wireframe perbaikan halaman menu home dan halaman about apps

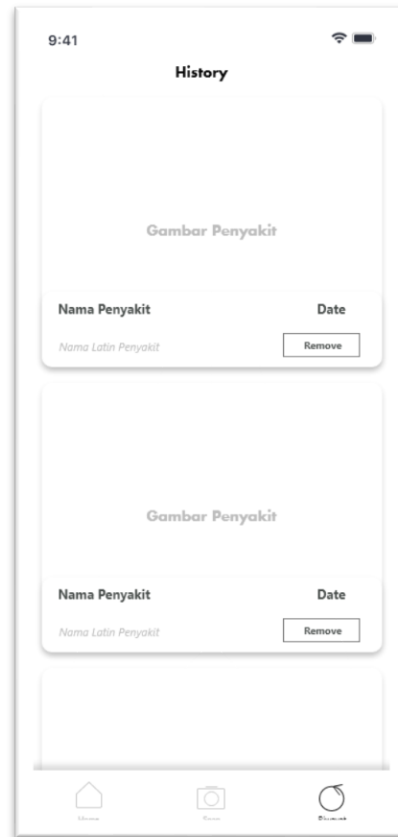
Pada Gambar 5.29 merupakan halaman *about apps*. Pada halaman tersebut terdapat *text view* dengan teks yang bertuliskan DiSnap yaitu nama dari aplikasi yang sedang dikembangkan, “version 1.0.0” yaitu versi dari aplikasi yang sedang dikembangkan selanjutnya ada text berupa deskripsi dari aplikasi DiSnap. Dilanjutkan dengan terdapat dua buah tombol yaitu tombol *Go To BPTP* yang berfungsi untuk membawa pengguna untuk membuka website BPTP Jawa Timur. Selanjutnya ada tombol *back* untuk membawa pengguna kembali ke halaman home.



Gambar 5.29 Wireframe halaman *about apps*

### 5.8.2 Wireframe perbaikan pada halaman riwayat penambahan tombol remove

Pada Gambar 5.30 merupakan halaman riwayat hasil perbaikan dari sebelumnya yaitu dengan menambahkan tombol remove pada setiap item riwayat, sehingga pengguna dapat menghapus riwayat sesuai keinginan pengguna.



**Gambar 5.30 Wireframe halaman riwayat**

### 5.8.3 Wireframe penambahan Screen Intro

Pada Gambar 5.31 merupakan intro yang dibuat untuk memberi tahu pengguna bahwa pada aplikasi ini pengguna dapat melihat dan mengetahui detail informasi penyakit pada tanaman cabai. Pada Gambar 5.32 pengguna diberi tahu bahwa dengan aplikasi DiSnap pengguna dapat melakukan pendeteksian penyakit dengan mendeteksi daun pada tanaman cabai. Sedangkan Gambar 5.33 pengguna diberi tahu bahwa pengguna dapat melihat kembali riwayat aktifitas hasil mendeteksi penyakit pada tanaman cabai.



**Gambar 5.31 Wireframe intro *home***



**Gambar 5.32** Wriframe *intro* Snap



**Gambar 5.33 Wirframe *intro history***

## **5.9 Perancangan Algoritme**

Perancangan algoritme pada aplikasi DiSnap digunakan untuk membantu mendeskripsikan alur logika dari program yang akan dibuat. Pada perancangan algoritme aplikasi DiSnap dipilih 3 fitur utama yaitu mengetahui informasi penyakit dan pengendalian penyakit, mendeteksi penyakit, dan mengetahui riwayat gambar yang telah dideteksi.

### **5.9.1 Perancangan komponen mengetahui informasi penyakit dan pengendalian penyakit**

Pada Tabel 5.4 menunjukkan algoritme dari perancangan komponen mengetahui informasi penyakit dan pengendalian penyakit. Pada algoritme ini pengguna menekan menu home, setelah itu sistem akan melakukan pengambilan data dari Disease\_JSON file melalui *presenter* dan *repository*. Setelah data berhasil didapatkan maka sistem akan mengembalikan data dan menampilkan nya kepada pengguna di menu home. Ketika pengguna menekan salah satu *item*, maka pengguna akan dibawa ke halaman detail dari item yang di pilih.



**Tabel 5.4 Algoritme perancangan komponen mengetahui informasi penyakit dan pengendalian penyakit**

PSEUDOCODE	
1	START
2	Menekan menu home
3	Memanggil presenter
4	Menampilkan loading
5	Mendapatkan data dari JSON file
6	IF(data != null)
7	Mengolah data di UI menggunakan model
8	Menampilkan data melalui recyclerview
9	END IF
10	Menekan salah satu item pada recyclerview
11	Mengirimkan data ke halaman detail melalui bundle
12	Membuat viewpager
13	Menampilkan data gambar, dan informasi penyakit tanaman cabai pada
14	viewpager
15	END

### 5.9.2 Perancangan komponen mendeteksi penyakit

Pada Tabel 5.5 menunjukkan algoritme dari perancangan komponen mendeteksi penyakit. Pada algoritme ini pengguna berada di halaman Analyze. Pada halaman ini gambar hasil dari proses *image cropping* yang akan dikirim sudah siap dan ketika pengguna menekan tombol analize image, sistem akan melakukan pengecekan terhadap koneksi internet. Apabila terdapat koneksi internet maka sistem akan mengirimkan HTTP Request berupa post/upload dan menunggu respon hasil deteksi berupa nama penyakit dan tingkat akurasi yang nanti akan ditampilkan kepada pengguna. Apabila tidak terdapat koneksi internet, maka sistem akan menampilkan pesan tidak ada koneksi internet.

**Tabel 5.5 Algoritme perancangan komponen mendeteksi penyakit**

PSEUDOCODE	
1	START
2	Gambar yang akan di deteksi sudah siap
3	Menekan tombol analize image
4	Memilih salah satu pilian camera atau gallery
5	IF(cek koneksi internet)
6	Mengirimkan gambar ke imgurl
7	IF(sukses)
8	THEN mengirimkan url ke Webservice API Clarifai
9	IF(sukses)
10	THEN Mendapatkan data
11	Mengolah data
12	Menampilkan halaman Result
13	Menampilkan informasi penyakit, akurasi dan detail
14	informasi terhadap penyakit tersebut
15	ENDIF
16	ENDIF
17	ELSE
18	Menampilkan tidak ada koneksi internet
19	ENDIF
20	END

### 5.9.3 Perancangan komponen mengetahui riwayat gambar yang telah dideteksi

Pada Tabel 5.6, pengguna menuju halaman riwayat. Pada halaman riwayat sistem akan memanggil presenter untuk mengambil data dari database melalui repository. Jika data tidak sama dengan 0 maka database akan mengembalikan nilai berupa data dan data selanjutnya akan dilolah pada bagian UI. Apabila ternyata tidak data pada database maka sistem akan menampilkan pesan tidak ada riwayat aktifitas tersimpan pada database .

**Tabel 5.6** Algoritme perancangan komponen mengetahui riwayat gambar yang telah dideteksi

PSEUDOCODE	
1	START
2	Menekan menu riwayat
3	Menuju menu riwayat
4	Memanggil presenter
5	Mengambil semua riwayat aktifitas deteksi
6	IF(data != 0)
7	THEN return data
8	Mengolah data
9	Menampilkan data menggunakan recyclerview
10	ELSE
11	THEN menampilkan pesan tidak ada riwayat hasil deteksi
12	ENDIF
13	IF(viewIsClicked())
14	Menampilkan halaman detail
15	Menampilkan informasi penyakit, akurasi dan detail informasi terhadap
16	penyakit tersebut
	ENDIF
	END

## BAB 6 IMPLEMENTASI

Pada tahap implementasi peneliti mulai melakukan pengembangan aplikasi DiSnap dengan berpedoman pada hasil perancangan yang telah dibuat. Dalam proses model *Prototyping*, maka peneliti sudah memasuki tahap Implementasi. Pembahasan yang terdapat pada implementasi yaitu terdiri dari spesifikasi sistem, batasan implemetasi, implementasi kode program, dan implementasi antarmuka.

### 6.1 Spesifikasi Sistem

Spesifikasi sistem menjelaskan tentang informasi mengenai perangkat keras, perangkat lunak dan sistem operasi yang digunakan oleh tim dalam mengembangkan aplikasi.

### 6.2 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk pengembangan DiSnap ditunjukkan pada Tabel 6.1.

**Tabel 6.1 Spesifikasi perangkat keras komputer**

Nama Komponen	Spesifikasi
<i>System Model</i>	Asus X441U (14-inch, 2017, )
<i>Processor</i>	2 GHz Intel Core I3
<i>Storage</i>	500GB
<i>Memory</i>	12 GB DDR4
Grafis	Intel HD Graphics 520

Adapun spesifikasi dari perangkat keras smartphone mobile yang digunakan untuk proses implementasi dan proses pengujian menggunakan real me 2 dengan sistem operasi android Pie seperti yang ditunjukkan pada Tabel 6.2.

**Tabel 6.2 Spesifikasi perangkat keras *smartphone mobile***

Nama Komponen	Spesifikasi
<i>System Model</i>	Realme 2
<i>Processor</i>	Qualcomm SDM450 Snapdragon 450 (14 nm)
<i>Storage</i>	32 GB
<i>Memory</i>	3 GB
Display	720 x 1520 pixels

### 6.3 Spesifikasi Perangkat Lunak

Dalam mengembangkan aplikasi DiSnap dibutuhkan spesifikasi perangkat lunak yang mendukung proses dari pengembangan aplikasi yang dibangun. Adapun pada Tabel 6.3 menunjukkan spesifikasi dari perangkat lunak komputer, sedangkan pada Tabel 6.4 menunjukkan spesifikasi perangkat lunak dari *smartphone mobile*.

**Tabel 6.3 Spesifikasi perangkat lunak komputer**

Nama Komponen	Spesifikasi
<i>Operating System</i>	Windows 10
<i>Programming Language</i>	Java
<i>IDE (Integrated Development Environment)</i>	Android Studio 3.6.1
Perancangan Diagram	Draw.io
Editor Dokumentasi	Microsoft Word 2013

**Tabel 6.4 Spesifikasi perangkat lunak *smartphone mobile***

Nama Komponen	Spesifikasi
<i>Operating System</i>	Android versi 9.0 (Pie)

### 6.4 Batasan-batasan Implementasi

Pada pengembangannya aplikasi DiSnap memiliki beberapa batasan dalam proses implementasinya. Berikut beberapa batasan implementasi dari aplikasi DiSnap sebagai berikut:

1. Aplikasi DiSnap hanya dapat berjalan pada *smartphone mobile* dengan sistem operasi Android *minimal version* 6 (Marshmallow).
2. Aplikasi ini dikembangkan menggunakan *Integrated Development Environment (IDE)* Android Studio 3.6.1 dengan menggunakan bahasa pemrograman Java.
3. Aplikasi memanfaatkan pihak ketiga yaitu imgurl untuk meng-*hosting* gambar
4. Aplikasi memanfaatkan pihak ketiga yaitu Clarifai untuk proses identifikasi daun tanaman cabai.
5. Aplikasi memanfaatkan library Fast Android Networking versi 1.0.2 sebagai *Rest Client* pada Android
6. Untuk dapat menggunakan fitur mendeteksi penyakit dibutuhkan koneksi internet.

## 6.5 Implementasi Basis Data

Basis data yang digunakan pada aplikasi pendeteksi penyakit pada tanaman cabai menggunakan teknologi clarifai yaitu hanya satu tabel yaitu tabel History. Tabel tHistory memiliki 10 atribut yaitu id, disease\_name, disease\_latin, accuration, result\_image, result\_image, user\_image, indication, control, pesticide, dan date. Pada pengembangan aplikasi DiSnap, peneliti menggunakan library room dari Google untuk membantu mempermudah peneliti dalam mengelola data pada database sqlite. Implementasi basis data aplikasi DiSnap dapat dilihat pada Tabel 6.5.

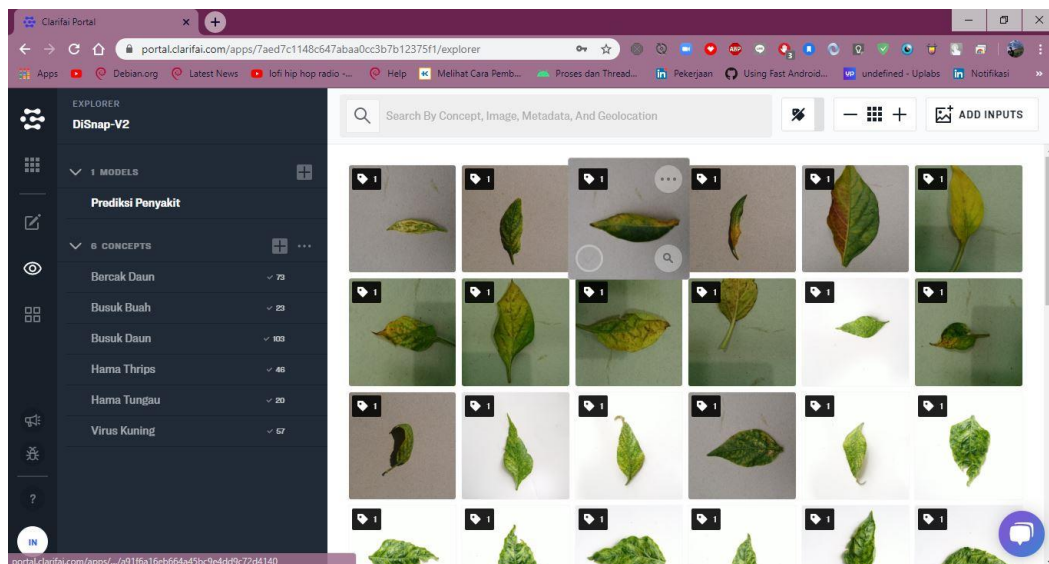
**Tabel 6.5 Implementasi tabel tHistory**

No	Source code
1	<code>@Entity(tableName = "tHistory")</code>
2	<code>public class History implements Serializable{</code>
3	
4	<code>    @PrimaryKey(autoGenerate = true)</code>
5	<code>    private int id;</code>
6	
7	<code>    @ColumnInfo(name = "disease_name")</code>
8	<code>    private String diseaseName;</code>
9	
10	<code>    @ColumnInfo(name = "disease_latin")</code>
11	<code>    private String diseaseLatin;</code>
12	
13	<code>    @ColumnInfo(name = "accuration")</code>
14	<code>    private double accuration;</code>
15	
16	<code>    @ColumnInfo(name = "result_image")</code>
17	<code>    private String resultImage;</code>
18	
19	<code>    @ColumnInfo(name = "user_image")</code>
20	<code>    private String userImage;</code>
21	
22	<code>    @ColumnInfo(name = "indication")</code>
23	<code>    private String indication;</code>
24	
25	<code>    @ColumnInfo(name = "control")</code>
26	<code>    private String controlling;</code>
27	
28	<code>    @ColumnInfo(name = "pesticide")</code>
29	<code>    private String pesticide;</code>
30	
31	<code>    @ColumnInfo(name = "date")</code>
32	<code>    private String date;</code>
	<code>}</code>

## 6.6 Implementasi Clarifai

### 6.6.1 Define

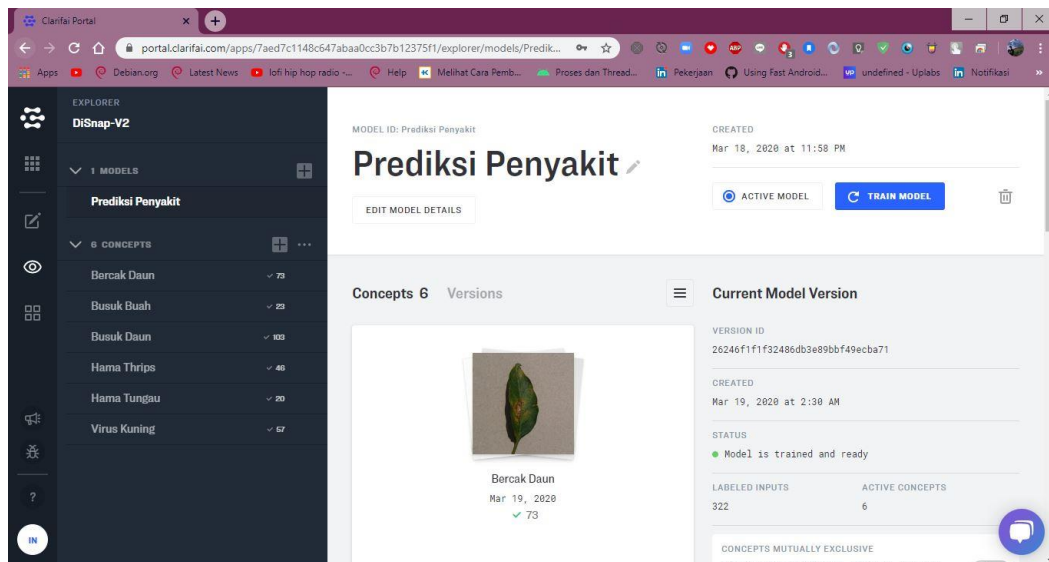
Pada tahap define dilakukan pembuatan *concept* dan *pelabelan* pada data gambar yang sudah di upload. *Concept* adalah label yang akan di sematkan pada setiap gambar. Pada penelitian ini concept merupakan nama penyakit pada tanaman cabai. Terdapat 6 concept yaitu Busuk Buah (23 data), Bercak Daun(73 data), Busuk Daun (103 data), Hama Thrips (46 data), Hama Tungau (20 data), dan Virus Kuning (57 data). Jumlah data pada setiap jenis penyakit adalah hasil pengambilan data yang dilakukan peneliti di BPTP Jawa Timur dibawah pengawasan pakar penyakit tanaman cabai. Implementasi tahap *define* ditunjukkan pada Gambar 6.1 Implementasi tahap *define*.



Gambar 6.1 Implementasi tahap *define*

### 6.6.2 Train

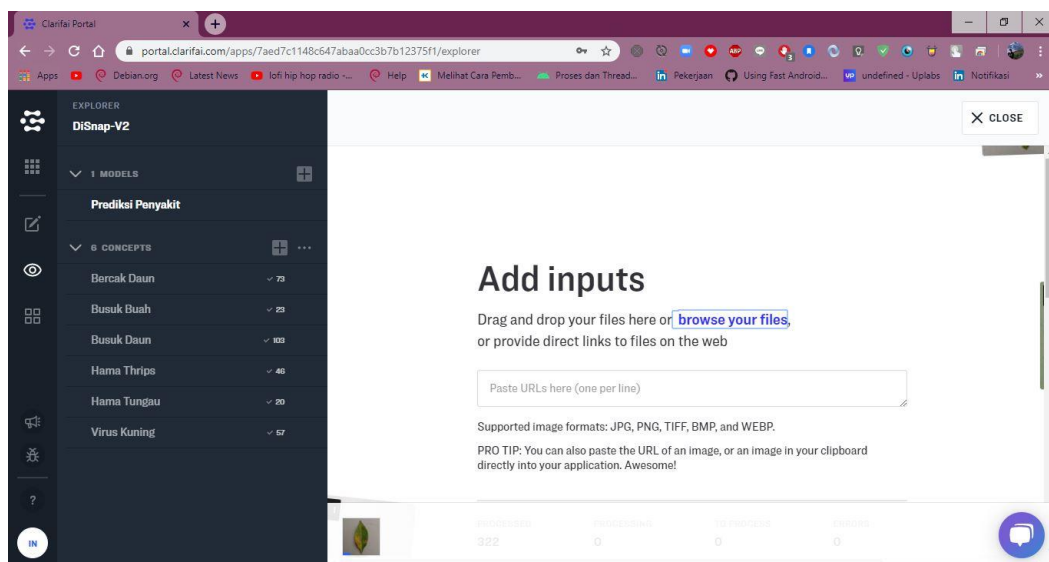
Pada tahap define terdapat tombol biru dengan nama *Train Model* yang berfungsi untuk melatih model dari gambar yang sudah disematkan pada setiap gambar berupa label dengan concept nama penyakit pada tanaman cabai. Setelah tombol *train model*. Maka model id : Prediksi Penyakit sudah dapat dipakai. Implementasi tahap train dapat dilihat pada Gambar 6.2.



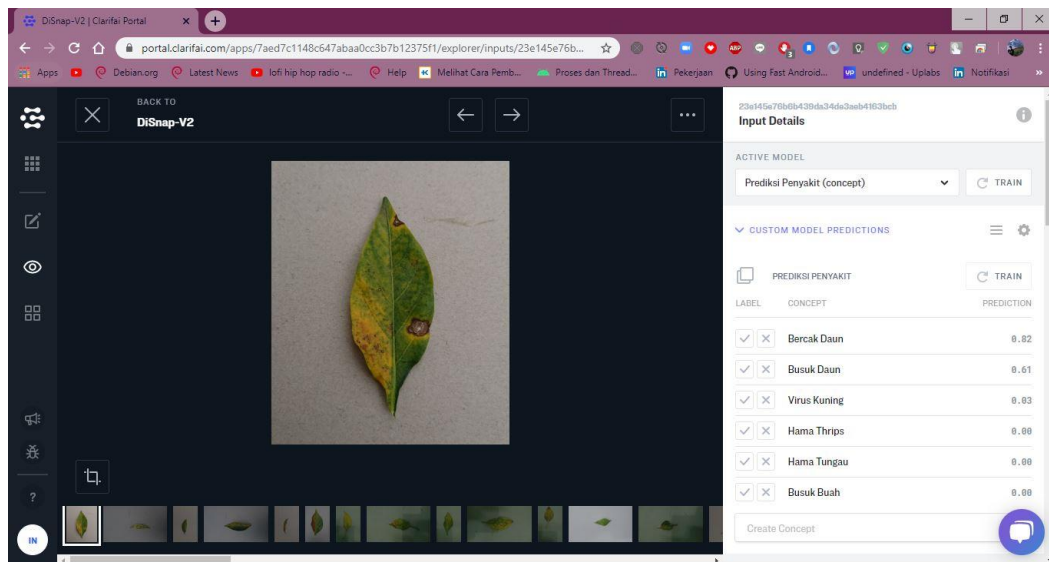
Gambar 6.2 Implementasi tahap *train*

### 6.6.3 Recognize

Pada tahap recognize dilakukan uji coba yaitu dengan mengupload sebuah gambar daun yang terkena penyakit. Maka pada bagian kanan website akan muncul hasil prediksi beserta nilai dari setiap concept. Nilai terbesar adalah nilai yang mendekati satu dan terletak di urutan paling atas. Implementasi tahap recognize dapat dilihat pada Gambar 6.4 dan Gambar 6.4.



Gambar 6.3 Prose mengupload gambar



**Gambar 6.4 Implementasi tahap *recognize***

## 6.7 Implementasi Algoritme

Implementasi algoritme dilakukan berdasarkan hasil perancangan algoritme pada tahap sebelumnya. Algoritme yang akan diimplementasikan merupakan fungsi utama dari aplikasi DiSnap. Fitur tersebut merupakan 3 fitur utama yaitu mengetahui informasi penyakit dan pengendalian penyakit, mendeteksi penyakit, dan mengetahui riwayat gambar yang telah dideteksi.

### 6.7.1 Implementasi algoritme mengetahui informasi penyakit dan pengendalian penyakit

Implementasi kode program pada method `getDiseaseFromJSONFile()` berada pada class `DiseaseJSONFileDataSource`, dimana kode program ini digunakan untuk mengambil data informasi penyakit berupa data json dengan nama `disnap_data.json` pada folder `assets`. Source code dari algoritma mengetahui informasi penyakit dan pengendalian penyakit dapat dilihat pada Tabel 6.6.

Nama Class : `DiseaseJSONFileDataSource`

Nama Method : `getDiseaseFromJSONFile()`

**Tabel 6.6 Source code method `getData()` Class `DiseaseJSONFileDataSource`**

No	Source code
1	<code>public void getDiseaseFromJSONFile(final</code>
2	<code>LoadDiseaseFromJSONFileCallback callback){</code>
3	<code>final ArrayList&lt;Disease&gt; diseases = new ArrayList&lt;&gt;();</code>
4	<code>callback.onShowLoading();</code>
5	<code>JSONLoader.with(App.getContext())</code>
6	<code>.fileName("disnap_data.json")</code>
7	<code>.getAsJSONObject(new JSONObjectLoaderListener() {</code>
8	<code>@Override</code>
9	<code>public void onResponse(JSONObject response) {</code>
10	<code>callback.onHideLoading();</code>



11	try {
12	diseases.addAll(insertData(response, "hama"));
13	diseases.addAll(insertData(response, "penyakit"));
14	callback.onDiseaseLoaded(diseases);
15	Log.d(TAG, "onResponse11: " + diseases.size());
16	} catch (JSONException e) {
17	callback.onHideLoading();
	e.printStackTrace();
18	}
19	}
20	
21	@Override
22	public void onFailure(Exception error) {
23	callback.onHideLoading();
24	}
25	});
26	
27	}

Penjelasan dari source code method `getDiseaseFromJSONFile ()` Class : `DiseaseJSONFileDataSource` ditunjukkan pada Tabel 6.7.

**Tabel 6.7 Penjelasan *source code* method `getDiseaseFromJSONFile ()`**

**Class : `DiseaseJSONFileDataSource`**

Baris	Penjelasan
1	Deklarasi method <code>getDiseaseFromJSONFile</code>
3	Intansiasi objek disease dari <code>ArrayList&lt;Disease&gt;</code>
4	Memanggil method <code>callback.onShowLoading</code>
5-9	Mekanisme menggunakan <code>JSONLoader</code> untuk mengambil data JSON
10-20	Mekanisme mengolah data apabila data berhasil diambil
21-27	Mekanisme ketika data tidak berhasil diambil

### 6.7.2 Implementasi algoritme mendeteksi penyakit

Implementasi algoritme mendeteksi penyakit menggunakan method `analyze image` yang berada pada class `DiseaseRemoteDataSource`. Kode program ini digunakan untuk mengirimkan url yang berisi gambar penyakit yang sebelumnya telah di lakukan *image hosting* menggunakan layanan *imgur*. Setelah mendapatkan url gambar dari layanan *imgur* maka url gambar tersebut dimasukkan kedalam sebuah json objek untuk dijadikan parameter dalam *request* terhadap Clarifai API. Respon dari Clarifai API berupa json objek yang didalamnya terdapat beberapa informasi. Informasi yang diambil adalah informasi berupa nama penyakit dan akurasi dari hasil analisis gambar daun yang dideteksi. Implementasi kode program dari algoritme mendeteksi penyakit dapat dilihat pada Tabel 6.8.

Nama *Class* : `DiseaseRemoteDataSource`

Nama *Method* : `predicImage ()`

**Tabel 6.8 Source code method predictImage()**

No	Source code
1	private void predictImage(final LoadAnalyzeCallback callback,
2	final String url) throws JSONException {
3	callback.onShowLoading();
4	AndroidNetworking.post(Constants.clarifaiAPI)
5	.setPriority(Priority.IMMEDIATE)
6	.addHeaders("Authorization", Constants.authClarifai)
7	.addHeaders("Content-Type", Constants.CONTENT_TYPE)
8	.addJsonObjectBody(this.getBody(url))
9	.build()
10	.getAsJsonObject(new JsonObjectRequestListener() {
11	@Override
12	public void onResponse(JsonObject response) {
13	callback.onHideLoading();
14	try {
15	Date date = Calendar.getInstance().getTime();
16	SimpleDateFormat df = new SimpleDateFormat("dd-MMM-yyyy");
17	String formattedDate = df.format(date);
18	
19	JSONArray jsonArray = response.getJSONArray("outputs");
20	JsonObject a = jsonArray.getJSONObject(0);
21	JsonObject b = a.getJSONObject("data");
22	JSONArray c = b.getJSONArray("concepts");
23	String name = c.getJSONObject(0).getString("name");
24	double value =
25	c.getJSONObject(0).getDouble("value");
26	
27	ArrayList<Disease> diseaseArrayList;
28	Disease disease = new Disease();
29	disease.setDiseaseName(name);
30	if (Rak.grab("ListDiseaseTemp") != null) {
31	diseaseArrayList = Rak.grab("ListDiseaseTemp");
32	for (int i = 0; i < diseaseArrayList.size(); i++) {
33	if
34	(name.equalsIgnoreCase(diseaseArrayList.get(i).getDiseaseName()))
35	{
36	disease.setDiseaseLatin(diseaseArrayList.get(i).getDiseaseLatin()
37	);
38	disease.setAccuration(value);
39	
40	disease.setResultImage(diseaseArrayList.get(i).getUserImage());
41	disease.setUserImage(url);
42	
43	disease.setIndication(diseaseArrayList.get(i).getIndication());
44	
45	disease.setControlling(diseaseArrayList.get(i).getControlling());
46	
47	disease.setPesticide(diseaseArrayList.get(i).getPesticide());
48	
49	disease.setDate(formattedDate);
50	
51	callback.onAnalyzeSuccess(disease);
52	}
53	}
54	}
55	} catch (JSONException e) {
56	callback.onHideLoading();
57	}
58	}
59	
60	@Override
61	public void onError(ANError anError) {
62	callback.onHideLoading();
63	callback.onError();

64	}
65	});
66	}

Penjelasan dari source code method `predictImage()` Class : `DiseaseRemoteDataSource` dapat dilihat pada

**Tabel 6.9 Penjelasan source code method `predictImage()` Class : `DiseaseRemoteDataSource`**

Baris	Penjelasan
1	Deklarasi method void <code>predictImage</code>
3	Pemanggilan method <code>showLoading</code>
4-10	Mekanisme penggunaan Android Fast Networking untuk melakukan pendeteksian pada gambar daun
12-14	Mekanisme apabila data berhasil dianalisis
15-17	Pembuatan nilai <code>date</code>
19-25	Pengambilan data berupa nama penyakit dan tingkat akurasi
27-49	Proses instansiasi objek dengan memberikan informasi pada objek sesuai dengan nama penyakit
51	Pemanggilan method <code>callback.onAnalyzeSuccess(disease)</code>
55-66	Error handling

### 6.7.3 Implementasi algoritme mendapatkan riwayat deteksi

Pada Tabel 6.10, merupakan implementasi dari algoritme mendapatkan riwayat hasil deteksi. Pada source tersebut dapat terlihat bahwa pengambilan data pada database dilakukan pada class `DiseaseDatabaseDataSource`.

Nama Class : `DiseaseDatabaseDataSource`

Nama Method : `getDiseaseAnalysisFromDB()`

**Tabel 6.10 Source code method `getDiseaseAnalysisResultFromDB()`**

No	Source code
1	<code>public void getDiseaseAnalysisResultFromDB(final</code>
2	<code>LoadDiseaseCallback callback) {</code>
3	<code>    Runnable runnable = new Runnable() {</code>
4	<code>        @Override</code>
5	<code>        public void run() {</code>
6	<code>            ArrayList&lt;Disease&gt; diseases = new</code>
7	<code>ArrayList&lt;&gt; (Arrays.asList(AppDatabase.getInstance().disea</code>
8	<code>seDAO().selectAllHistory()));</code>
9	<code>            if (diseases.size() != 0) {</code>
10	<code>                callback.onDiseaseLoaded(diseases);</code>
11	<code>            } else {</code>
12	<code>                callback.onError("You have no story activity yet");</code>
13	<code>            }</code>
14	<code>        }</code>
15	<code>    };</code>
16	<code>    executor.execute(runnable);</code>
17	<code>}</code>

Penjelasan dari source code method `getDiseaseAnalysisResultFromDB()` Class `DiseaseDatabaseDataSource` dapat dilihat pada Tabel 6.11 dibawah ini.

**Tabel 6.11 Penjelasan source code method `getDiseaseAnalysisResultFromDB()` Class : `HistoryFragmentPresenter`**

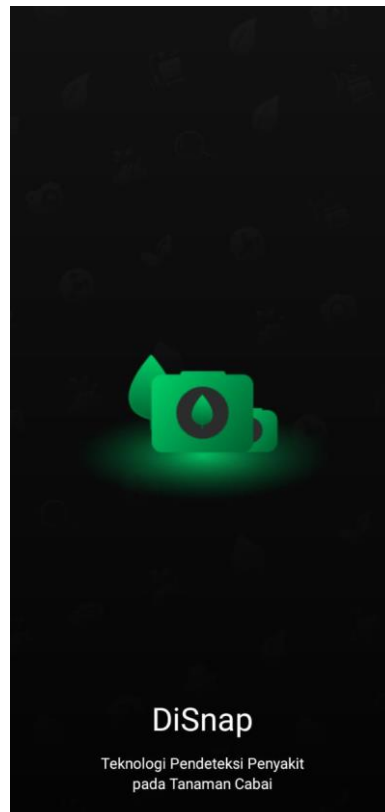
Baris	Penjelasan
1	Deklarasi method <code>getDeaseAnalysisFromResultDB</code>
3	Penggunaan <code>runnable</code> untuk membuat <code>therad</code> baru
6-8	Pemanggilan method <code>selectAllHistory</code>
9-10	Seleksi kondisi jika data tidak sama dengan 0 maka memanggil method <code>callback.onDiseaseLoaded(disease)</code>
11-12	Error handling
16	Eksekusi thread

## 6.8 Implementasi User Interface

Implementasi user interface mengacu pada wireframe yang telah dibuat pada bab perancangan.

### 6.8.1 Implementasi *user interface* splash screen

Pada Gambar 6.5 merupakan halaman *splash screen*, halaman ini akan selalu muncul setiap saat aplikasi dibuka. Pada halaman ini terdapat background berwarna hitam dengan gambar2 *icon* didalamnya. Selain itu terdapat juga logo aplikasi dan *text* tentang deskripsi aplikasi.



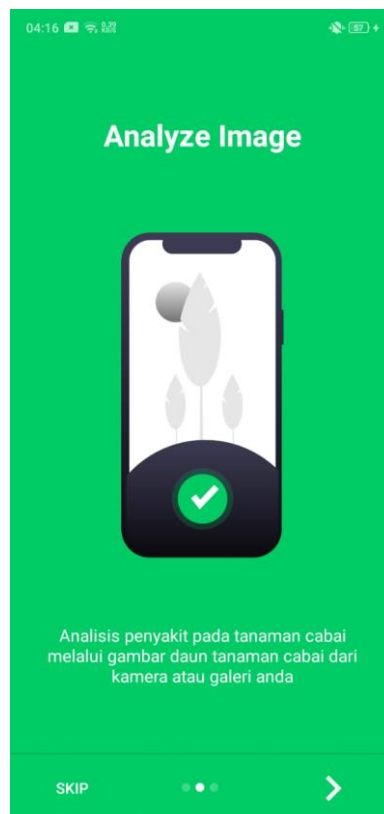
**Gambar 6.5 Implementasi *user interface splash screen***

### **6.8.2 Implementasi *user interface intro***

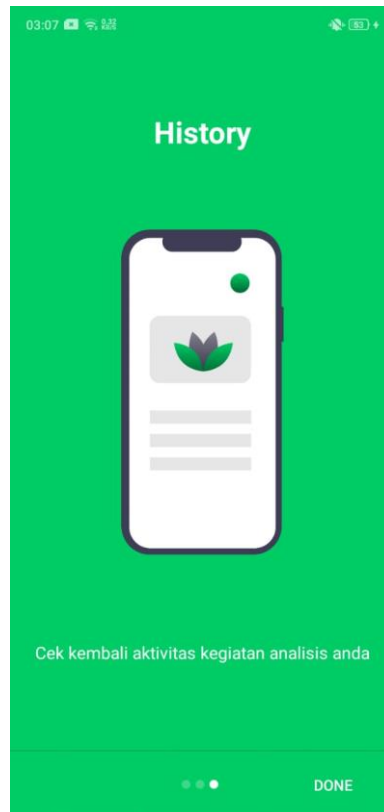
Pada aplikasi DiSnap terdapat halaman intro. Halaman intro adalah halaman pengenalan fitur yang berfungsi untuk memberikan informasi kepada pengguna tentang apa saja fitur yang berada pada aplikasi. Pada Gambar 6.6 menunjukkan halaman *intro disease information* yaitu informasi yang diberitahukan kepada pengguna bahwa aplikasi DiSnap memiliki fitur untuk dapat melihat detail informasi berbagai jenis penyakit pada cabai. Pada Gambar 6.7 menunjukkan halaman *intro snap* yang berfungsi untuk memberikan informasi kepada pengguna bahwa aplikasi DiSnap memiliki fitur untuk dapat mendeteksi penyakit pada tanaman cabai melalui gambar daun pada tanaman cabai. Pada Gambar 6.8 menunjukkan halaman intro history yang berfungsi untuk memberikan informasi kepada pengguna bahwa pengguna dapat kembali melihat riwayat deteksi yang pernah dilakukan.



**Gambar 6.6 Implementasi *user interface intro disease information***



**Gambar 6.7 Implementasi *user interface intro snap***

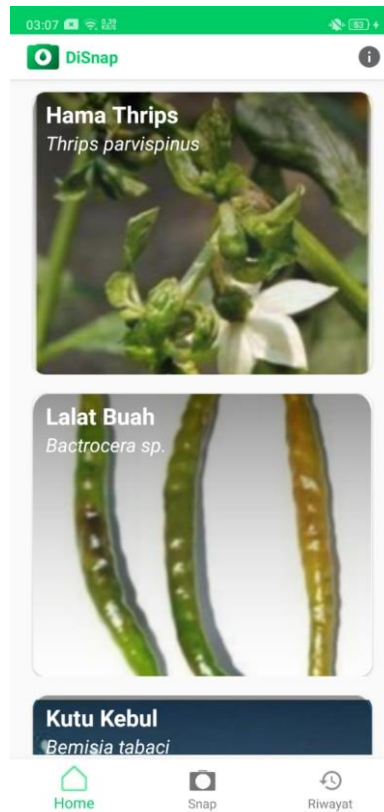


**Gambar 6.8 Implementasi *user interface intro history***

### **6.8.3 Implementasi *user interface* mengetahui informasi penyakit dan pengendalian penyakit**

Di dalam implementasi user interface mengetahui informasi penyakit dan pengendalian penyakit, terdapat app yang berisi logo dan nama aplikasi DiSnap. Selain itu pada implementasi ini juga terdapat *recyclerview* yaitu daftar informasi penyakit tanaman cabai yang berisi gambar daun cabai yang terkena penyakit, nama penyakit, dan nama latin dari penyakit seperti yang ditunjukkan pada Gambar 6.9.

Apabila salah satu dari daftar tersebut di klik maka aplikasi akan menampilkan detail dari penyakit tersebut seperti gejala, pengendalian dan informasi pestisida. Implementasi user interface mengetahui informasi dan pengendalian penyakit seperti yang ditunjukkan dalam Gambar 6.10.



**Gambar 6.9 Implementasi user interface home**



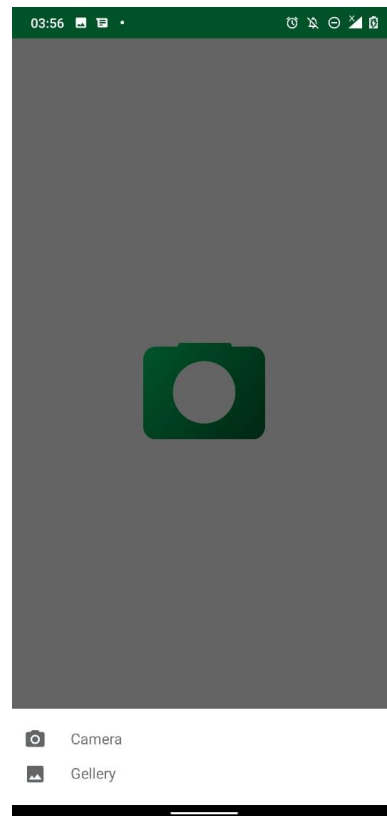
**Gambar 6.10 Implementasi user interface detail informasi penyakit pada cabai**



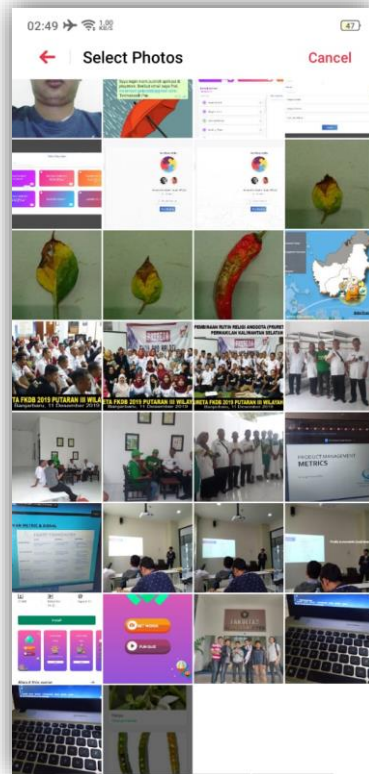
#### 6.8.4 Implementasi *user interface* mendapatkan gambar

Pada Gambar 6.11, menunjukkan pengguna ketika menekan menu snap pada bottom navigation sehingga akan muncul bottomsheets yang merupakan tampilan yang menyajikan dua menu yaitu mendapatkan gambar melalui galeri atau mendapatkan gambar melalui kamera. Pada Gambar 6.12, menampilkan halaman ketika pengguna memilih mendapatkan gambar melalui gallery yang ada pada device pengguna. Pada Gambar 6.13, menunjukkan halaman *cropping image*, yaitu gambar yang dipilih dari gallery ataupun kamera dilakukang pemotongan oleh pengguna.

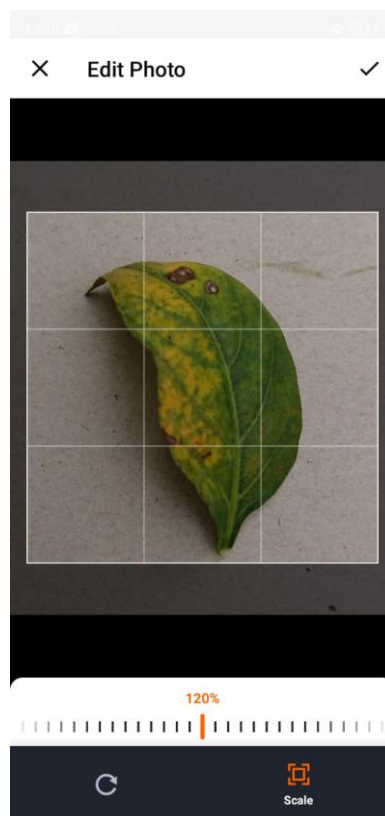
Pada Gambar 6.14, menunjukkan ketika gambar yang sudah selesai dilakukan proses pemotongan dan gambar siap untuk dideteksi atau di analisis dengan menekan *button analyze* yang terletak pada bagian bawah halaman tersebut.



**Gambar 6.11 Implementasi *user interface* show bottom dialog**



**Gambar 6.12 Implementasi user interface mengambil gambar melalui galery**



**Gambar 6.13 Implementasi *user interface cropping image***

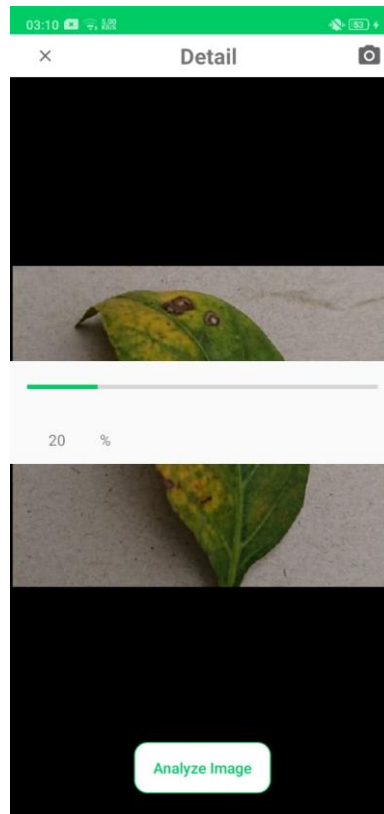


**Gambar 6.14 Implementasi *user interface hasil dari cropping image***

#### **6.8.5 Implementasi *user interface* mendeteksi penyakit**

Pada Gambar 6.15, gambar menunjukkan halaman *Analyze* ketika dengan gambar yang sudah didapatkan dari hasil *cropping image*. Dan tombol *Analyze Image*. Ketika tombol tersebut di tekan maka proses deteksi pun terjadi. Pengguna akan melihat *loading progress bar* pada layar seperti pada Gambar 6.15.

Pada Gambar 6.16, menunjukkan halaman *Result*. Halamann tersebut merupakan halaman hasil dari proses deteksi yang telah dilakukan oleh pengguna. Pada halaman tersebut terdapat gambar yang dikirimkan pengguna, gambar referensi yang cocok dengan gambar yang dideteksi pengguna, nama penyakit, nama latin penyakit, akurasi, bar yang menunjukkan akurasi, serta informasi gejala, pengendalian dan pestisida.



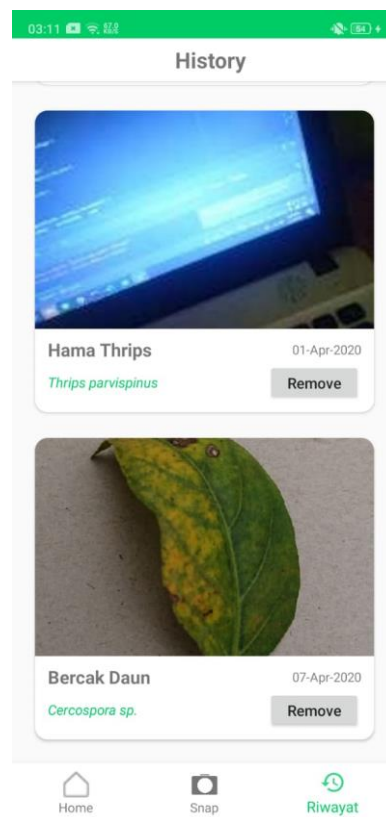
**Gambar 6.15 Implementasi user interface proses mendeteksi penyakit pada tanaman cabai melalui gambar daun cabai**



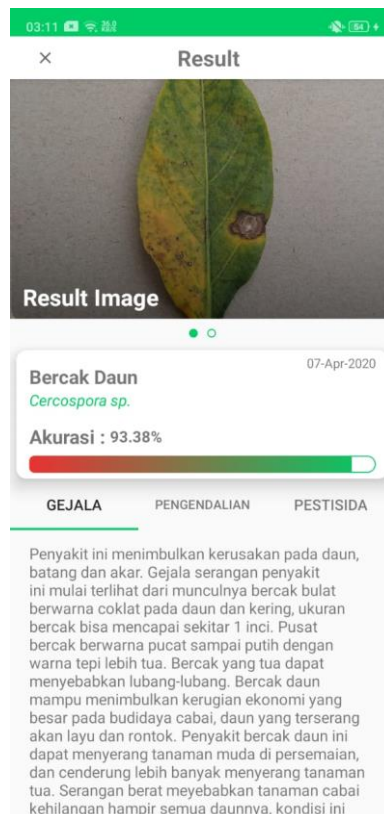
**Gambar 6.16 Implementasi *user interface* hasil dari deteksi gambar**

#### **6.8.6 Implementasi *user interface* mengetahui riwayat hasil deteksi**

Pada Gambar 6.17, menunjukkan halaman daftar dari riwayat aktifitas dari mendeteksi penyakit yang pernah dilakukan oleh pengguna. Pada halaman tersebut terdapat *cardview*, yang berisi photo yang pernah dianalisis oleh pengguna, nama penyakit, nama latin penyakit, tingkat akurasi dan tanggal ketika pengguna melakukan aktifitas menganalisis penyakit pada tanaman cabai. Ketika pengguna melakukan klik atau memilih salah satu dari *cardview* yang ada maka sistem akan membawa pengguna ke halaman `DetailHistoryActivity` dimana pengguna dapat melihat informasi detail dari riwayat deteksi yang berisi photo gambar yang dianalisis pengguna, nama penyakit, nama latin penyakit, akurasi, tanggal deteksi, *accuration bar*, informasi gejala, informasi pengendalian, dan informasi pestisida seperti yang ditunjukkan pada Gambar 6.18.



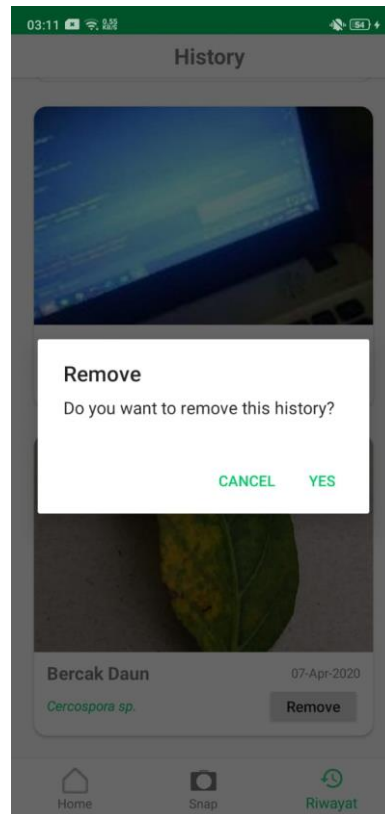
**Gambar 6.17 Implementasi menu riwayat**



**Gambar 6.18 Implementasi detail riwayat**

#### **6.8.7 Implementasi *user interface* mendapatkan menghapus riwayat hasil deteksi**

Pada Gambar 6.19, menunjukkan halaman yang riwayat yang menunjukkan dialog ketika tombol remove ditekan oleh pengguna. Terdapat dua pilihan yang dapat dipilih oleh pengguna yaitu “yes” atau “no”. Apabila pengguna memilih pilihan “no” maka pengguna dialog hilang dan pengguna berada pada halaman riwayat. Akan tetapi apabila pengguna memilih untuk menekan tombol “yes”, maka sistem akan menghapus riwayat yang dipilih oleh pengguna. Setelah itu juga apabila proses menghapus sukses maka akan muncul pesan “Remove success” seperti pada Gambar 6.20 dan *recyclerview* akan otomatis memuat ulang daftar riwayat yang ada apabila masih terdapat data pada database.



**Gambar 6.19 Implementasi *user interface dialog remove history***



**Gambar 6.20 Implementasi *user interface remove success***

### **6.8.8 Implementasi *user interface about apps***

Pada Gambar 6.21, menunjukkan halaman *about apps*. Pada halaman tersebut terdapat nama aplikasi, versi aplikasi, penjelasan singkat aplikasi. Serta dua buah tombol yaitu tombol *GoToBPTP* dan tombol *Back*. Apabila pengguna menekan tombol *GoToBPTP* maka pengguna akan dibawa oleh sistem ke halaman pencarian google untuk dapat mengunjungi website BPTP Jawa Timur. Sedangkan apabila pengguna menekan tombol *Back* maka pengguna akan kembali ke halaman *home*.



**Gambar 6.21 Implementasi *user interface about apps***



## **BAB 7 PENGUJIAN**

## DAFTAR REFERENSI

- Ahmad, A. 2017. Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning. *Yayasan Cahaya Islam, Jurnal Teknologi Indonesia*.
- Akbari, G. W., Hidayat, N. & Santoso, N., 2019. Diagnosis Penyakit Cabai Menggunakan Metode Fuzzy K-Nearest Neighbor (FKNN). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 03(1), 1070-1074.
- Clarifai.2019. About the platform. [Online] Tersedia <https://www.clarifai.com/about> [Diakses 16 Juli 2019].
- Darmawan, D.A. and E. Pasandaran. 2000. Indonesia. In: M. Ali (ed). *Dynamic of vegetable production. distribution and consumption in Asia*. AVRDC Publication 00-498. Shanhua. Tainan: AVRDC. Pp.139-171. <http://www.avrdc.org/pdf/dynamics/Indonesia.pfd>
- Developers, G., 2018. *About the platform*. [Online] Tersedia di: <https://developer.android.com/about/> [Diakses 16 Juli 2019].
- Fitriiningtyas, Y. A., 2015. Sistem Pakar Hama Dan Penyakit Pada Tanaman Cabai Dengan
- Meilin, A. 2014. *Hama dan Penyakit pada Tanaman Cabai Serta Pengendaliannya*. 2014. Jambi : Balai Pengkajian Teknologi Pertanian (BPTP) Jambi.
- Malahayati, Nur., Fadhli, Muhammad. 2018. *Distribusi Perdagangan Komoditas Cabai Merah Indonesia Tahun 2018*. Jakarta: BPS RI.
- Nusantara, D.M., Pamungkas, S. W., Syaifudin, N. R., Kusuma, L. W., Fikri, J. 2017. Sistem Pakar Analisa Penyakit Pada Tanaman Cabai Merah Menggunakan Metode Backward Chaining. *Seminar Nasional Teknologi Informasi dan Multimedia*, 2302 - 3805.
- Pressman, Roger S. 2010. *Software Engineering: A Practitioner's Approach, Seventh Edition*. New York: McGraw-Hill.
- Purwanto, T. & Destiani, D.2015. Pengembangan Sistem Pakar Diagnosis Penyakit Cabai. Garut: Jurnal STT-Garut
- Ryantono, R. P., 2017. Rancang Bangun Aplikasi Smartfoodies Dengan Memanfaatkan Clarifai Api Untuk Image Recognition Berbasis Android.
- Setiadi. 2004. *Bertanam Cabai*. Penebar Swadaya. Jakarta. 12 hlm. Warisno dan Dahana. 2010. *Peluang Usaha dan Budidaya Cabai*. Jakarta: Gramedia Pustaka Utama.
- Setiadi. 2011. *Bertanam Cabai di Lahan Pot*. Jakarta: Penebar Swadaya.
- Statcounter, 2019. *Operating System Market Share in Indonesia*. [Online] Tersedia di : <https://gs.statcounter.com/os-market-share/all/indonesia> [Diakses 26 Agustus 2019].

Warisno dan Dahana. 2010. *Peluang Usaha dan Budidaya Cabai*. Jakarta: Gramedia Pustaka Utama.