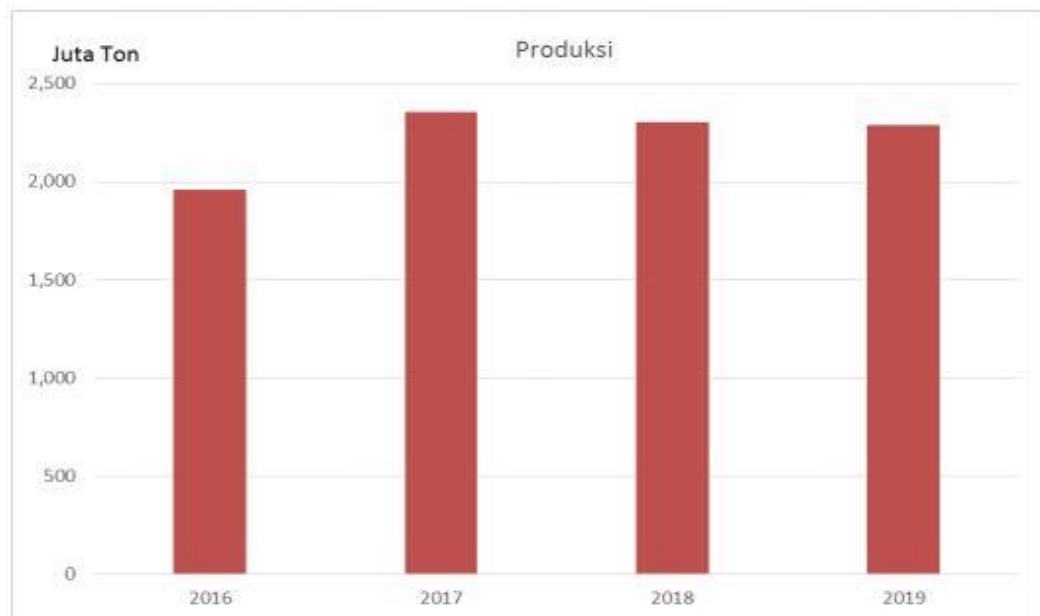


BAB 1 PENDAHULUAN

1.1 Latar Belakang

Indonesia termasuk negara tropis yang menjadikan cabai menjadi salah satu tanaman komersial yang banyak dibudidayakan dan memiliki nilai jual tinggi sehingga menguntungkan bagi petani. Cabai tidak hanya dijual di pasaran, cabai juga dapat memberikan kesempatan peluang kerja bagi masyarakat dengan perannya yaitu sebagai bahan baku dalam perindustrian (Setiadi, 2004). Data dari publikasi 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 per kapita per tahun (Malahayati, 2018).

Untuk data nasional yaitu seluruh wilayah Indonesia, produksi cabai dapat dilihat pada Gambar 1.1. Berdasarkan data Kementerian Pertanian, total produksi cabai pada tahun 2016 sebesar 1,96 juta ton kemudian mengalami peningkatan pada tahun 2017 sebesar 2,35 juta ton. Akan tetapi terjadi penurunan produksi pada tahun 2018 sehingga total produksinya menjadi 2,30 juta ton. Dan prediksi untuk tahun 2019 adalah 2,90 juta ton (Kementerian Pertanian, 2019). Dari data tersebut terlihat bahwa jumlah produksi cabai mengalami peningkatan dan penurunan setiap tahun.

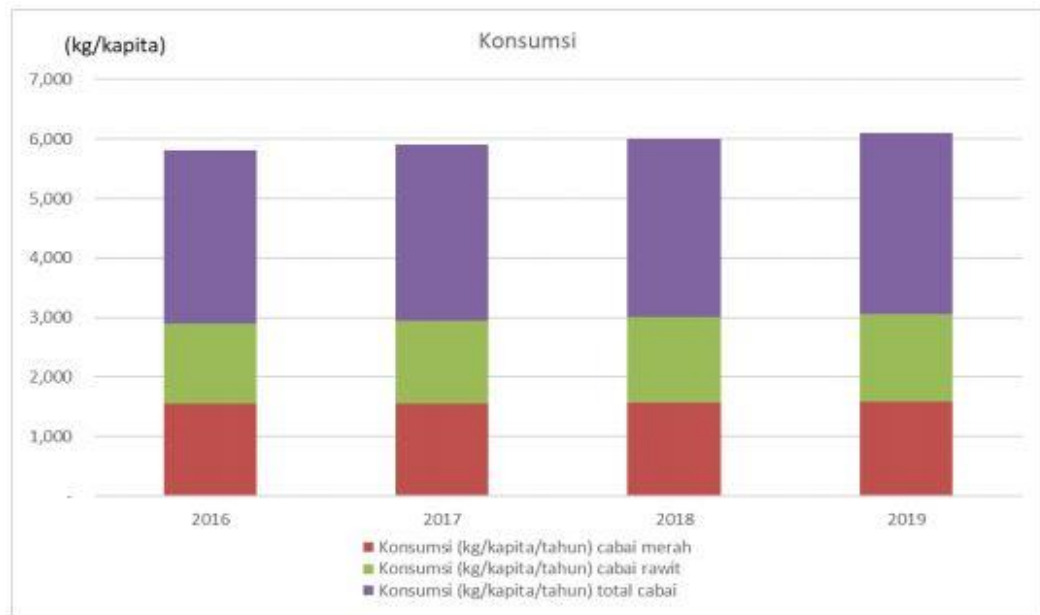


Gambar 1.1 Perkembangan Produksi Cabai Tahun 2016-2019

Sumber : (Kementerian Pertanian, 2019)

Sedangkan untuk data perkembangan konsumsi cabai tahun 2016-2019 dapat dilihat pada Gambar 1.2. Untuk tahun 2016 total konsumsi cabai sebesar 2,90 (kg/kapita), untuk tahun 2017 sebesar 2,95 (kg/kapita), tahun 2018 sebesar 3,00 (kg/kapita) dan untuk tahun 2019 sebesar 3,05 (kg/kapita) (Kementerian

Pertanian, 2019). Dari data tersebut dapat disimpulkan bahwa konsumsi cabai tiap tahun selalu mengalami peningkatan. Tentunya hal tersebut harus merupakan angka yang bagus apabila tingkat produksi cabai dapat memenuhi kebutuhan pasar sehingga harga cabai di pasaran dapat terus stabil.



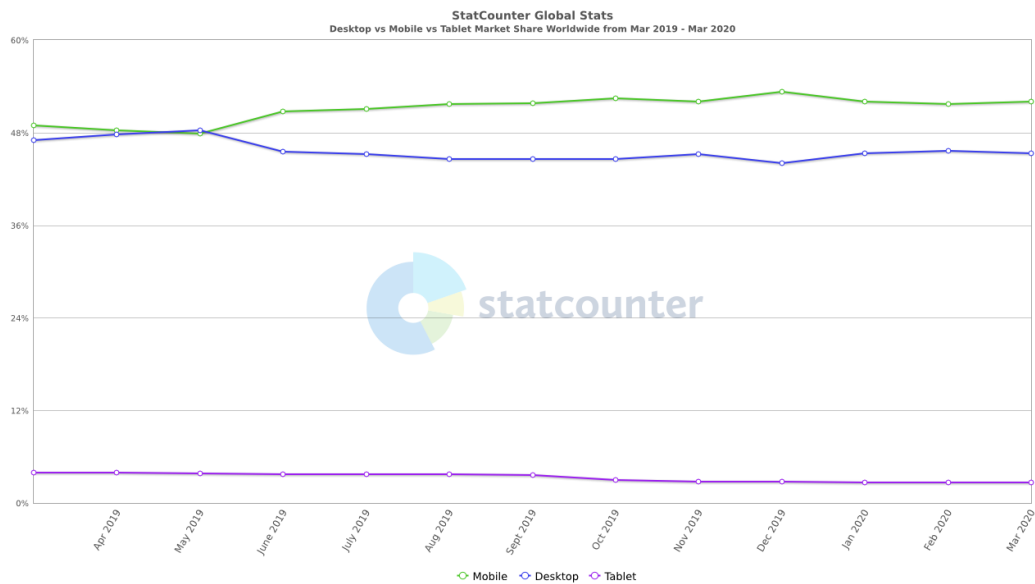
Gambar 1.2 Perkembangan Konsumsi Cabai Tahun 2016-2019

Sumber : (Kementerian Pertanian, 2019)

Produktivitas cabai di Indonesia masih belum dapat dipastikan kestabilannya dalam memenuhi kebutuhan nasional cabai masyarakat Indonesia dikarenakan produktivitas 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 pengetahuan para petani dalam mengenali jenis penyakit dan hama yang menyerang tanaman pada cabai masih kurang (Purwanto, 2015). Sehingga kurang ada penanganan yang tepat sesuai kondisi tanaman. Hal ini terbukti dengan adanya kasus ladang cabai di daerah Magetan bulan Juni-Juli 2019 yang terkena serangan virus. Tanaman cabai yang terkena virus memiliki lahan 0,2 hektar atau sejumlah 3.400 an batang cabai. Menurut Sumarlan, koordinator POPT Kabupaten Magetan menerangkan, salah satu penyebab dari terjadinya penyerangan virus disebabkan oleh cara menggunakan pestisida yang cenderung asal, dikarenakan petani rata-rata belum bisa membedakan antara hama, penyakit, atau fungi(jamur). Hal itu menyebabkan hama penyakit makin kebal (BPTP Jatim, 2019).

Saat ini teknologi sudah berkembang pesat. Hampir setiap masyarakat di Indonesia memiliki setidaknya satu orang satu perangkat baik itu *desktop*, *mobile* maupun *tablet*. Berdasarkan salah satu situs penyedia jasa informasi yaitu statcounter menyebutkan bahwa dari total pengguna perangkat di Indonesia pada bulan Maret 2019 sampai dengan bulan Maret 2020 yang menggunakan

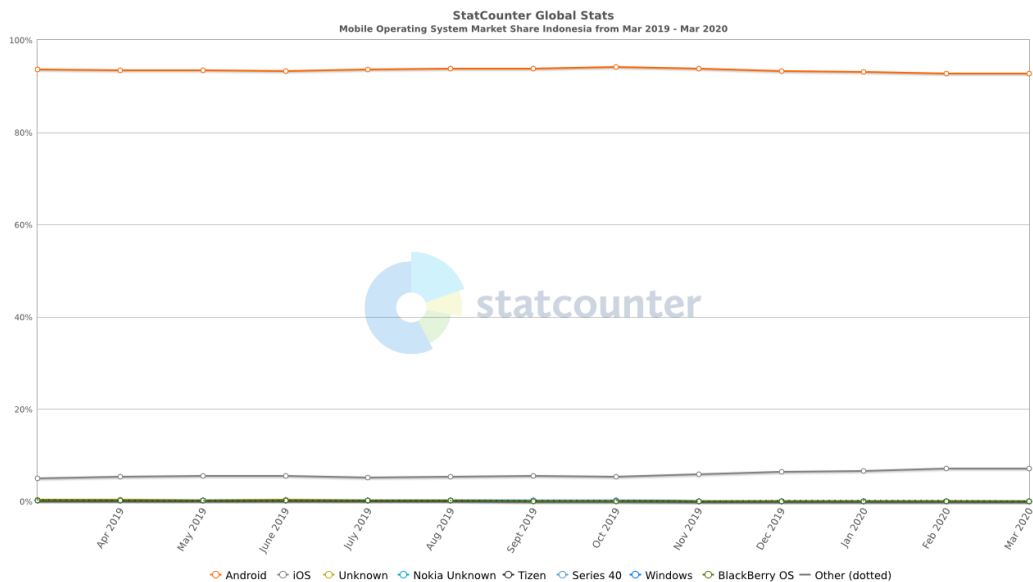
perangkat *mobile* mencapai 54,21%, perangkat *desktop* sebesar 45, 14% dan perangkat *tablet* sebesar 0.65%. Dari data tersebut dapat dilihat bahwa pengguna perangkat mobile merupakan pengguna dengan jumlah terbanyak yaitu 54,21%. Data tersebut dapat dilihat pada Gambar 1.3.



Gambar 1.3 Desktop vs Mobile vs Tablet Market Share di Indonesia pada Bulan Maret 2019 – Maret 2020

Sumber: (Statcounter, 2020)

Dengan banyaknya pengguna perangkat *mobile* di Indonesia dapat memudahkan semua penggunanya dalam menjalankan aktivitas sehari-hari menggunakan aplikasi-aplikasi yang tersedia. Untuk menjalankan perangkat *mobile* dibutuhkan sistem operasi atau yang biasa disebut dengan OS (*Operating System*) yang berfungsi sebagai perantara antara pengguna dan sistem. Saat ini, ada begitu banyak sistem operasi yang berjalan pada perangkat mobile. Adapun sistem operasi yang terkenal seperti sistem operasi Android, dan iOS. Pada Gambar 1.4, menunjukkan grafik pengguna sistem operasi perangkat lunak di Indonesia. Berdasarkan grafik tersebut dapat dilihat bahwa data pengguna perangkat lunak dengan sistem operasi Android sebesar 72,26%, sistem operasi iOS sebesar 27,03%, sistem operasi KaiOS sebesar 0,32%, sistem operasi samsung sebesar 0,16 %, sistem operasi windows sebesar 0,1%, dan *unknown* sebesar 0,06%. Dari data tersebut dapat disimpulkan bahwa pengguna sistem operasi perangkat *mobile* terbesar di Indonesia adalah sistem operasi Android yaitu sebesar 72,26% yang artinya sistem operasi ini mendominasi pasar di Indonesia.



Gambar 1.4 Pangsa Pasar *Mobile Operation System* (OS) di Indonesia pada Bulan Maret 2019 – Maret 2020

Sumber: (Statcounter, 2020)

Selain perkembangan *device* dan sistem operasi perkembangan teknologi lainnya pun ikut berkembang. Salah satu teknologi yang berkembang adalah pada bidang *Machine Learning*. *Machine Learning* merupakan mesin yang banyak digunakan untuk menggantikan atau menirukan perilaku manusia (Ahmad, 2017). Salah satu kemampuan mesin learning adalah dapat melakukan pengolahan citra gambar digital. Salah satu kemampuan pengolahan citra gambar digital adalah *Image Classification*. *Image Classification* adalah kemampuan mesin untuk mengklasifikasikan sebuah gambar masuk ke dalam kelompok – kelompok tertentu berdasarkan model yang telah dilatih.

Clarifai adalah salah satu teknologi *Computer Vision* yang telah memenangkan 5 kali posisi teratas pada kompetisi *Image Classification* di *The ImageNet 2013 Competition*. *Computer Vision* adalah kemampuan untuk mengenali gambar dan video secara otomatis berdasarkan elemen dan pola visual. Clarifai menggunakan *Machine Learning* untuk melakukan pengenalan pada gambar dan video. Clarifai menggunakan Deep Convolutional Neural Network(CNNs), yaitu *subfield* dari *Machine Learning* untuk mengenali objek pada gambar dan video. Clarifai memiliki beberapa fitur yaitu *Image Recognition*, *Video Recognition*, *Automatic tagging*, *Content moderation*, *Visual search*, *Document classification*, *Face verification*, *Aerial surveillance*, *Predictive maintenance*, dan *Demographic analysis* (Clarifai, 2020).

Berdasarkan pemaparan masalah dan pemanfaatan teknologi yang ada, maka penulis mencoba melakukan penelitian terhadap permasalahan tersebut dengan mengembangkan sebuah aplikasi *mobile* dengan sistem operasi android untuk mendeteksi penyakit pada tanaman cabai serta pengendaliannya. Aplikasi ini berfungsi membantu petani dan pakar untuk mencegah penyebaran penyakit dengan memberikan penanganan yang sesuai dengan kondisi tanaman yang

terserang penyakit. Selain itu penulis juga ingin mengetahui tingkat akurasi ketepatan deteksi atau mengenali gambar dari teknologi clarifai. 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”.

Untuk melakukan pengembangan perangkat lunak, diperlukan sebuah metode untuk melakukan perencanaan, pengelolaan dan pengontrolan dari setiap proses yang dikerjakan. Metode pengembangan perangkat lunak pada aplikasi *mobile* pendeteksi penyakit pada tanaman cabai menggunakan metode *Waterfall*. Metode *Waterfall* adalah metode yang pas untuk digunakan dalam pengembangan perangkat lunak dengan semua kebutuhan pada sistem telah didefinisikan di awal, tidak memiliki perubahan fungsional, fungsional tidak terlalu kompleks dan setiap tahapan dilakukan secara berurutan (Pressman, 2010).

Selanjutnya perlu dilakukan pengujian validasi untuk memastikan semua fungsional dapat berfungsi sesuai dengan harapan dan berfungsi dengan baik. Pengujian akurasi dilakukan untuk mendapatkan nilai rata-rata hasil akurasi dari data uji yang ada. Kemudian untuk mengetahui kualitas aplikasi yang dibangun maka perlu dilakukan pengujian *usability* dan *compatibility*. Untuk mengetahui seberapa mudah dan nyaman aplikasi digunakan dengan secara langsung melibatkan calon pengguna aplikasi maka dilakukan pengujian *usability* aplikasi (Henstam, 2018). Untuk dapat mengetahui perangkat lunak yang akan dibangun dapat berjalan sesuai dengan lingkungan seperti perangkat keras dan koneksi maka dilakukan pengujian *compatibility* (Pressman, 2010).

1.2 Rumusan Masalah

Dari latarbelakang permasalahan yang sudah paparkan, maka penulis dapat merumuskan masalah sebagai berikut:

1. Bagaimana hasil analisis kebutuhan fungsional dan non-fungsional pada aplikasi pendeteksi penyakit tanaman cabai?
2. Bagaimana rancangan dan implementasi aplikasi pendeteksi penyakit pada tanaman cabai dalam aplikasi *mobile* dengan mengintegrasikan teknologi clarifai?
3. Bagaimana hasil uji validasi, akurasi, *usability* dan *compatibility* pada aplikasi pendeteksi penyakit pada tanaman cabai?

1.3 Tujuan

Tujuan dari penelitian ini yaitu:

1. Mengetahui hasil analisis kebutuhan fungsional dan non-fungsional dari aplikasi pendeteksi penyakit pada tanaman cabai.

2. Mengetahui rancangan dan implementasi aplikasi pendeteksi penyakit pada tanaman cabai dalam aplikasi *mobile* dengan mengintegrasikan teknologi clarifai.
3. Mengetahui hasil uji dari aplikasi penyakit pada tanaman cabai.

1.4 Manfaat

Manfaat dari penelitian ini yaitu::

1. Manfaat bagi penulis
Penulis dapat mengimplementasikan materi yang telah dipelajari selama kuliah.
2. Manfaat bagi peneliti selanjutnya
Dengan adanya aplikasi ini bisa menjadi bahan rujukan ataupun referensi untuk para peneliti dalam pengembangan selanjutnya.
3. Manfaat untuk pengguna
Membantu menyediakan aplikasi untuk BPTP dalam mempermudah petani mendeteksi penyakit pada tanaman cabai.

1.5 Batasan Masalah

Pengembangan dalam penelitian ini memiliki beberapa batasan masalah, yaitu:

1. Jenis cabai yang diteliti yaitu jenis cabai 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 level SDK Android 23 (Android Marshmallow).
4. Untuk dapat memanfaatkan fitur analisis gambar aplikasi harus ter koneksi dengan internet.
5. Pengujian akurasi adalah pengujian tambahan pada penelitian ini.

1.6 Sistematika Pembahasan

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

1. BAB 1 PENDAHULUAN

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

2. BAB 2 LANDASAN KEPUSTAKAAN

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

3. BAB 3 METODOLOGI

Bab metodologi menjelaskan tahapan dari penelitian yang dilakukan sebagai proses penyelesaian masalah yang sedang diteliti.

4. Bab 4 ANALISIS KEBUTUHAN

Bab analisis kebutuhan memuat tentang analisis kebutuhan dari penggalian kebutuhan untuk mendapatkan kebutuhan dari perangkat lunak yang akan dibangun.

5. BAB 5 PERANCANGAN

Bagian ini menjelaskan perancangan perangkat lunak berdasarkan hasil dari analisis kebutuhan yang telah didapatkan.

6. BAB 6 IMPLEMENTASI

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

7. BAB 7 PENGUJIAN

Bagian ini menjelaskan pengujian pada sistem yang dilakukan oleh peneliti. Pengujian yang dilakukan yaitu pengujian validasi, pengujian akurasi, pengujian *usability*, dan pengujian *compatibility* pada aplikasi pendeteksi penyakit tanaman cabai.

8. BAB 8 PENUTUP

Bab penutup menjabarkan kesimpulan dari penelitian yang diambil mengacu pada proses perancangan, implementasi dan pengujian pada penelitian serta saran yang dapat digunakan oleh peneliti selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi beberapa kajian landasan kepastakaan dan teori-teori dasar yang memiliki kaitan dengan penelitian yang sedang diteliti. Kajian pustaka berisi tentang penelitian pendeteksi penyakit pada tanaman cabai 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 *Waterfall Model* dan teknik pengujian yang akan dilakukan.

2.1 Kajian Pustaka

Dalam penelitian ini, penulis menemukan sejumlah penelitian yang relevan terkait dengan penelitian yang sedang diteliti. Beberapa penelitian tersebut penulis gunakan untuk membantu dalam proses penelitian yang dilakukan. Penelitian-penelitian sebelumnya akan dijabarkan, 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 yaitu 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.

4. *Agile vs waterfall: A Comparative Analysis*. Hasil tinjauan dari penelitian ini yaitu dilakukannya perbandingan untuk mengembangkan *software* lunak dengan menggunakan metode pengembangan *Agile* dan *Waterfall*. Dari penelitian ini, dapat ditarik kesimpulan yaitu untuk mengembangkan perangkat lunak yang berbeda maka harus menggunakan metode yang berbeda pula tergantung tingkat kompleksitas dan daur hidup perangkat lunak yang dibangun. Jika suatu perangkat lunak yang akan dibangun terdapat kebutuhan yang tidak mengalami perubahan maka metode yang paling sesuai yaitu memanfaatkan metode *Waterfall*, namun jika terdapat kemungkinan ada perubahan kebutuhan fungsional saat proses pengerjaan maka metode yang paling sesuai adalah menggunakan metode *Agile* (Kannan, Smita, & Verma, 2014).

2.2 Penyakit pada Tanaman Cabai

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



Gambar 2.1 Layu Fusarium

Sumber: (BPTP Jambi, 2014)

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.

2.2.2 Layu Bakteri Ralstonia (*Ralstonia solanacearum*)

Pada Gambar 2.2, menunjukkan gejala penyakit layu bakteri *ralstonia*. Penyebab pada penyakit tanaman cabai ini adalah adalah Bakteri *Pseudomonas solanacearum*. Pada tanaman tua terjadi daun layu pada tanaman bagian bawah. 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, terung, tomat ataupun cabai.



Gambar 2.2 Layu Bakteri Ralstonia

Sumber: (BPTP Jambi, 2014)

2.2.3 Busuk Buah Antraknosa (*Collectotrichom gloeosporioides*)

Pada Gambar 2.3, menunjukkan gejala busuk buah antraknosa. Penyakit buah busuk antraknosa pada tanaman cabai disebabkan oleh cendawan *collectotrichom*. Busuk buah antraknosa dapat dilihat dari bagian tanaman yang diserang yaitu bagian buah cabai. 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 cokelat. 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 lembap, pupuk N terlalu tinggi dan tanah kekurangan Ca.



Gambar 2.3 Busuk Buah Antraknosa

Sumber: (BPTP Jambi, 2014)

2.2.4 Virus Kuning (*Gemini Virus*)

Pada Gambar 2.4, menunjukkan gejala penyakit virus kuning. 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 bagian daun yaitu tulang nya menjadi berwarna kuning terang dan menebal serta daun bentuk 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 sejak bibit tanaman mulai diserang, bisa terjadi saat musim kemarau (ketika pembibitan dan penanaman), dan populasi kutu kebul yang tinggi.



Gambar 2.4 Virus Kuning

Sumber: (BPTP Jambi, 2014)

2.2.5 Bercak Daun (*Cercospora sp.*)

Pada Gambar 2.5, menunjukkan penyakit bercak daun pada tanaman cabai yang disebabkan oleh *Cercospora capsica*. Bercak daun cercospora dapat menimbulkan defoliiasi jika serangan terjadi pada daun, sedangkan apabila terjadi pada bunga akan mengakibatkan gugur bunga serta apabila terjadi pada buah maka dapat menimbulkan mal formasi pada buah yang mengakibatkan buah menjadi kerdil. Gejala penyakit ini menimbulkan suatu bercak bulat dengan warna coklat pada daun dengan kondisi yang kering serta memiliki ukuran sekitar 1 inci. Bercak yang tua yang terdapat pada daun dapat menimbulkan lubang pada bagian daun.

Lingkungan dengan kondisi dengan tingkat curah yang tinggi dapat mempercepat perkembangan dan penyebaran penyakit bercak daun. Tanaman yang terkena serangan ini 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 Aplikasi Perangkat Bergerak (*Mobile Applications*)

Mobile application adalah *software* (perangkat lunak) atau program yang berjalan dan dapat beroperasi pada perangkat bergerak atau *mobile device*. Aplikasi tersebut berjalan sesuai dengan perintah penggunaannya berdasarkan fitur yang ada pada aplikasi tersebut. *Mobile applications* menjadi segmen baru dalam teknologi informasi yang memberikan dampak besar bagi kehidupan bermasyarakat di segala segment. Aplikasi perangkat bergerak memiliki perkembangan yang sangat cepat karena sifatnya yang mudah dikembangkan, ringan, dapat diunduh, mudah digunakan dan dapat dijalankan pada *smartphone*.

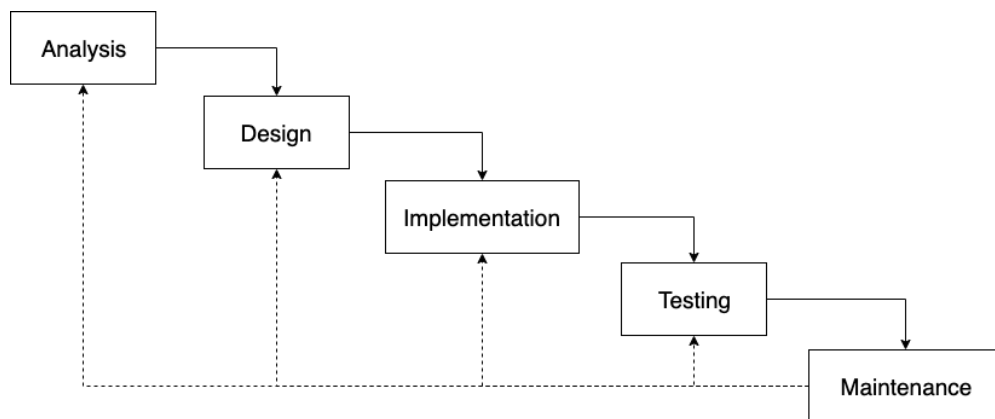
Pengembangan aplikasi perangkat bergerak cukup mudah untuk dikembangkan sehingga banyak developer yang bekerja di bidang ini untuk dapat mengembangkan aplikasi dengan berbagai tujuan seperti komersial, edukasi, hiburan dan lainnya. Adapun market dari aplikasi ini dapat diunduh dari berbagai penyedia aplikasi untuk berbagai sistem operasi. Contoh tempat untuk mengunduh aplikasi pada sistem operasi android adalah *play store*. Dengan adanya platform seperti *play store* dapat memudahkan pada developer untuk memasarkan aplikasi dan juga memudahkan pengguna untuk menemukan dan memanfaatkan aplikasi yang dibutuhkan.

Mobile Application memiliki beberapa karakteristik yang perlu diperhatikan ketika melakukan pengembangan aplikasi perangkat bergerak seperti cara berinteraksi pengguna dan tampilan yang kecil. Adapun kekurangan yang lain yaitu secara umum perangkat bergerak memiliki baterai dan ruang

penyimpanan yang begitu terbatas yang perlu diperhatikan saat melakukan pengembangan (Wasserman, 2010).

2.4 Waterfall

Dalam pengembangan aplikasi mobile pendeteksi penyakit pada tanam cabai ini peneliti menggunakan Waterfall sebagai model pengembangan perangkat. Waterfall adalah metode dari salah satu siklus pengembangan perangkat lunak atau yang biasa di sebut dengan SDLC (*Software Development Life Cycle*). Pada Gambar 2.6, menunjukkan gambar dari metode *waterfall* yang dibagi kedalam fase-fase (tahapan) dari pengembangan yang mirip dengan air terjun. Cara kerja waterfall yaitu secara berurutan dengan lima tahapan yang harus dilalui untuk mengembangkan suatu perangkat lunak (Bassil, 2012).



Gambar 2.6 Siklus Pengembangan *Waterfall*

Sumber: (Bassil, 2012)

Penjelasan masing-masing fase dari siklus pengembangan *waterfall* adalah sebagai berikut:

1. *Analysis* / Analisis
Fase ini merupakan tahapan untuk melakukan penggalan semua kebutuhan sistem sehingga kebutuhan sistem pada perangkat lunak yang akan dikembangkan dapat ter definisikan.
2. *Design* / Desain / Perancangan
Fase ini, merupakan tahapan melakukan perencanaan dalam membangun perangkat lunak yang meliputi desain algoritme, arsitektur, penampilan antar muak serta data model yang akan digunakan.
3. *Implementation* / Implementasi
Fase implementasi merupakan tahapan untuk melakukan realisasi pada perangkat lunak sesuai perancangan sistem yang telah dibuat kedalam kode program.

4. *Testing* / Pengujian

Fase pengujian merupakan tahapan untuk melakukan validasi dan verifikasi pada sistem dengan tujuan memastikan kesesuaian antara kebutuhan dengan perangkat lunak yang dibangun.

5. *Maintenance* / Pemeliharaan

Fase pemeliharaan merupakan tahapan terakhir pada metode waterfall yang dilakukan untuk memperbaiki kesalahan maupun untuk meningkatkan kinerja, menambah fitur, memperbaharui tampilan pada perangkat lunak setelah dilakukan proses peluncuran.

2.5 *Best Practice*

Untuk membuat perangkat lunak baik web ataupun aplikasi pada perangkat bergerak, diperlukan kemudahan dalam pengembangannya yaitu melalui *design architecture* (arsitektur desain). *Design architecture* bertujuan untuk memudahkan pengembang dalam membangun aplikasi, menguji aplikasi dan merawat aplikasi. Ada beberapa *design architecture* dalam pengembangan aplikasi seperti *Model-View-Controller* (MVC), *Model-View-Presenter* (MVP), *Model-View-ViewModel* (MVVM) dan *Model-View-Intent* (MVI). Berikut akan dijelaskan *design architecture* yang digunakan untuk mengembangkan aplikasi pada penelitian ini.

2.5.1 *Model View Presenter (MVP)*

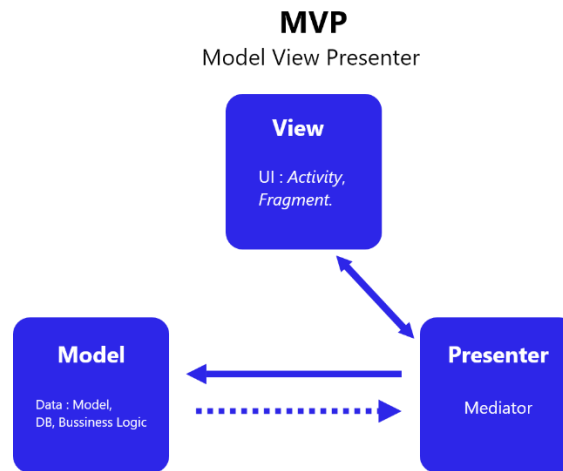
Permasalahan umum untuk pemula dalam mengembangkan perangkat lunak aplikasi adalah sulitnya melakukan *testing* (pengujian) dan *maintenance* (pemeliharaan) dikarenakan kode yang dibuat tidak terstruktur dan terkelompokkan dengan baik. Maka aplikasi yang dibangun harus memiliki arsitektur atau bagian-bagian (*layer*) yang terpisah sesuai dengan perannya masing-masing. Salah satu arsitektur dalam mengembangkan perangkat lunak aplikasi adalah MVP (*Model View Presenter*). Sesuai dengan namanya MVP memiliki 3 bagian, yaitu *Model*, *View* dan *Presenter*. *Model* berfungsi untuk mengurus bagian dari bisnis data, *View* adalah bagian yang menangani semua bagian UI (*User Interface*) dan *Presenter* adalah bagian yang berfungsi untuk menghubungkan antara *View* dan *Model*.

Arsitektur MVP merupakan salah satu arsitektur yang dapat digunakan oleh pengembang yang memiliki keterbatasan dalam kemampuan pemrograman yang terbatas (Ojeda-Guerra, 2015). Adapun karakteristik MVP yaitu sebagai berikut:

1. *Background Service* berfungsi untuk melakukan fungsi dibelakang layar yang terpisah dari *Activity*, *Fragment* atau *View* yang terpisah dari *Lifecycle*.
2. Suatu bagian yang memiliki fungsi, tugas atau peranan yang sama dikelompokkan menjadi lapisan (*layer*) yang sama, sehingga memudahkan dalam membaca, mengembangkan, *testing* dan *maintenance*.

3. Transaksi data dipisahkan dengan *View*.
4. Memudahkan dalam melakukan *Unit Testing* sehingga dapat dilakukan secara otomatis.

MVP dapat mempermudah pengembang dalam melakukan perancangan *Sequence Diagram*, karena setiap *layer* memiliki fungsinya masing-masing. *Model* akan melakukan proses bisnis data, *Presenter* menghubungkan data dari *Model* ke *View* dan *View* digunakan untuk menampilkan data dari *Model* (Wu & Yang, 2009).



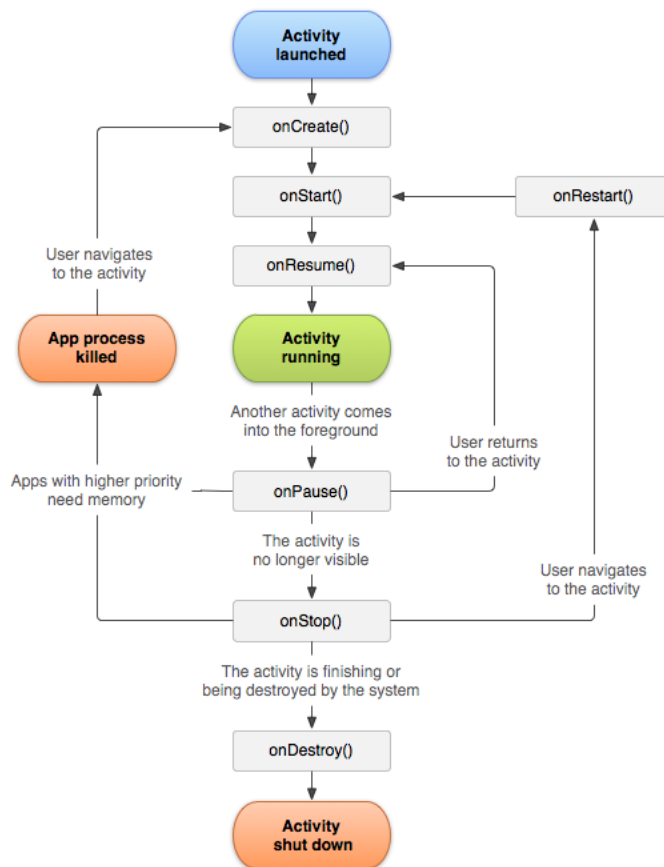
Gambar 2.7 MVP Architecture

Sumber: (Wu & Yang, 2009)

2.6 Android

Android adalah sistem operasi yang memiliki basis yaitu kernel Linux dan dirancang oleh Google serta biasa digunakan untuk *device* seperti *smartphone*, *tablet*, *smartwatch* dan berbagai *smart device* 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 di rilis dan sudah dapat digunakan oleh pengguna di seluruh dunia 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, 2020).



Gambar 2.8 Lifecycle Android

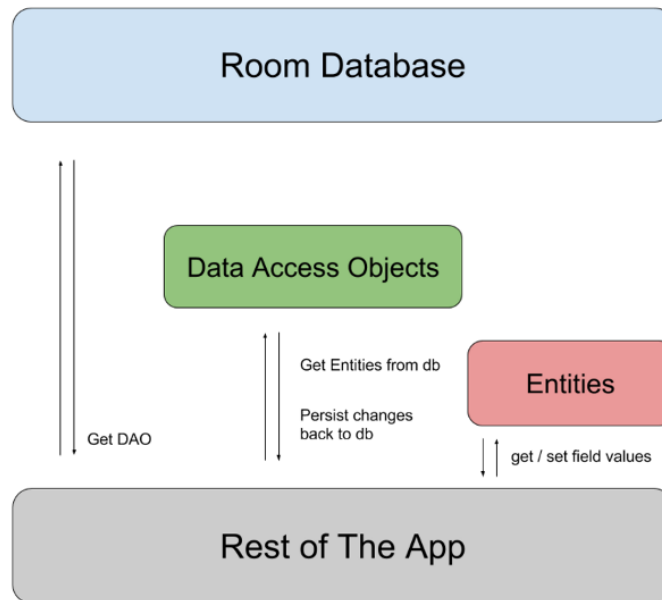
Sumber: (Developers, 2019a)

Gambar 2.8, menunjukkan daur hidup 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.

2.6.1 Room

Room adalah salah satu bagian dari arsitektur komponen yang disediakan oleh google untuk memudahkan para developer dalam mengembangkan aplikasi android. Room membantu menyediakan suatu lapisan (*layer*) yang berada di atas lapisan SQLite yang dapat digunakan untuk mengakses *database* dengan mudah. SQLite adalah penyimpanan lokal yang ada pada sistem android yang membantu memudahkan untuk melakukan proses penyimpanan data secara lokal. Pada Gambar 2.9, menunjukkan komponen dari Room yang terdiri dari 3 bagian yaitu *Room Database*, *Data Access Objects* dan *Entities*. *Room Database* adalah pemegang utama *database* dan juga memiliki fungsi untuk melakukan akses data relasional terhadap aplikasi. *Data Access Objects* yaitu kelas yang berisi method

untuk mengakses *database*. Sedangkan *Entities* merupakan representasi dalam tabel dalam *database*.



Gambar 2.9 Room Component

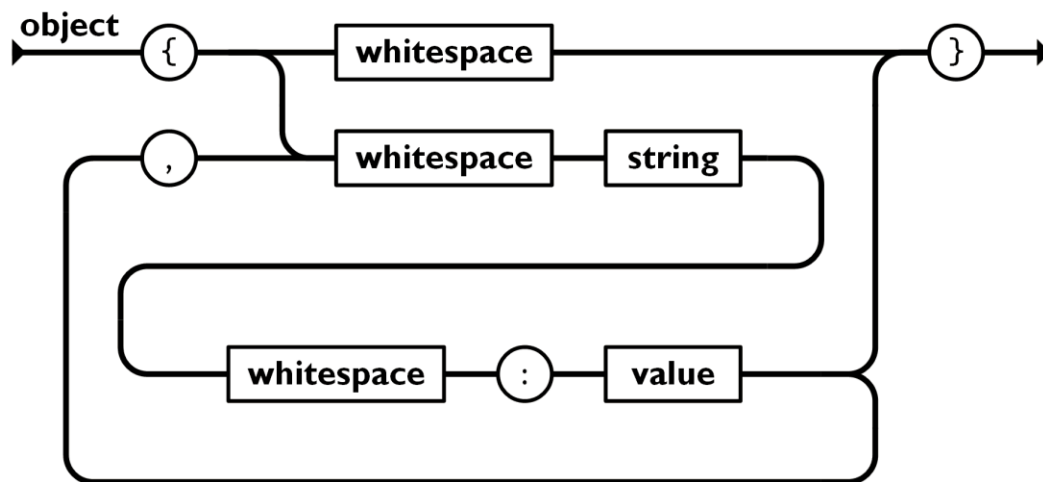
Sumber: (Developers, 2019b)

2.6.2 JSON (*Javascript Object Notation*)

JSON (*Javascript Object Notation*) merupakan format pertukaran data yang dapat mudah untuk dibaca oleh manusia. JSON dibuat berdasarkan pada subset dari Standar Bahasa Pemrograman JavaScript ECMA-262 Edisi 3 - Desember 1999. JSON merupakan struktur data universal yang dapat digunakan oleh berbagai bahasa dalam berbagai bentuk (JSON, 2019).

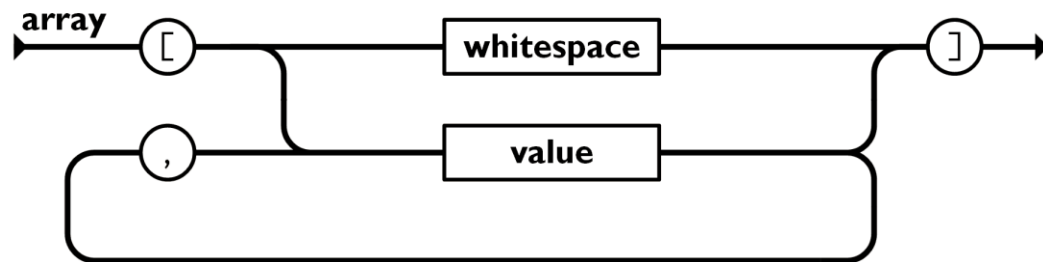
JSON dibangun di atas dua struktur utama yaitu kumpulan pasangan nama/nilai dan nilai yang diurutkan. Untuk pasangan nama/nilai dapat berupa *object*, catatan, *struct*, kamus, *hash table*, *keyed list*, atau *associative array*. Sedangkan Daftar nilai yang diurutkan dapat berupa *array*, *vector*, *list* atau *sequence*. Beberapa contoh bentuk JSON seperti ditunjukkan pada Gambar 2.10 dan Gambar 2.11.

Pada Gambar 2.10, menunjukkan objek pada JSON yang merupakan kumpulan pasangan nilai tidak teratur. Objek dimulai dengan {brace kiri dan berakhir dengan} brace kanan. Setiap nama diikuti oleh: titik dua dan pasangan nama / nilai dipisahkan oleh, koma. Sedangkan pada Gambar 2.11, menunjukkan array pada JSON yang merupakan kumpulan nilai yang diurutkan. Array dimulai dengan [braket kiri dan berakhir dengan] braket kanan. Nilai dipisahkan oleh, koma.



Gambar 2.10 JSONObject

Sumber: (JSON, 2019)



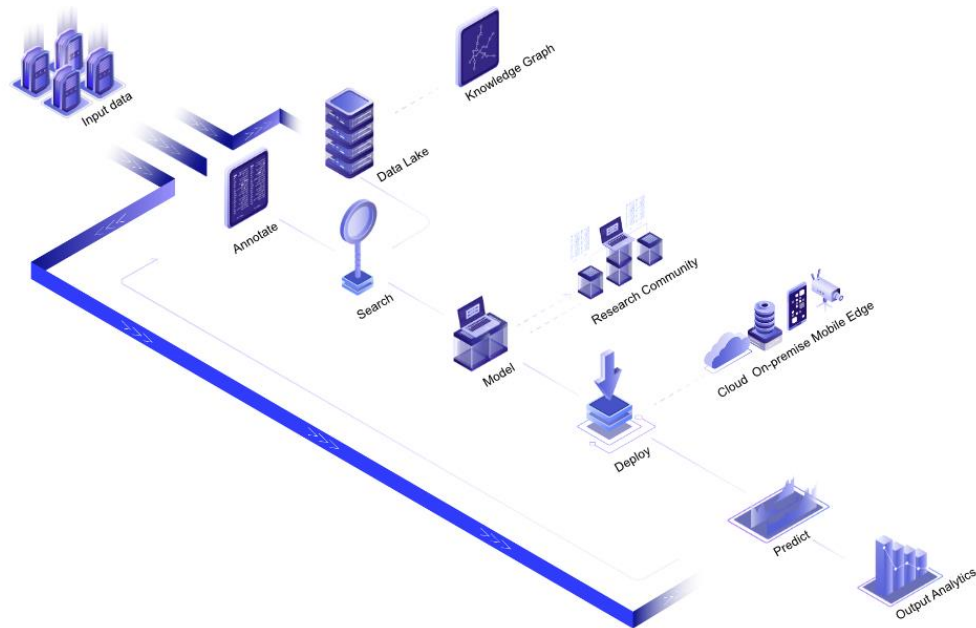
Gambar 2.11 JSONArray

Sumber: (JSON, 2019)

2.7 Clarifai

Clarifai adalah perusahaan *Artificial Intelligence* yang bergerak di bidang *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).

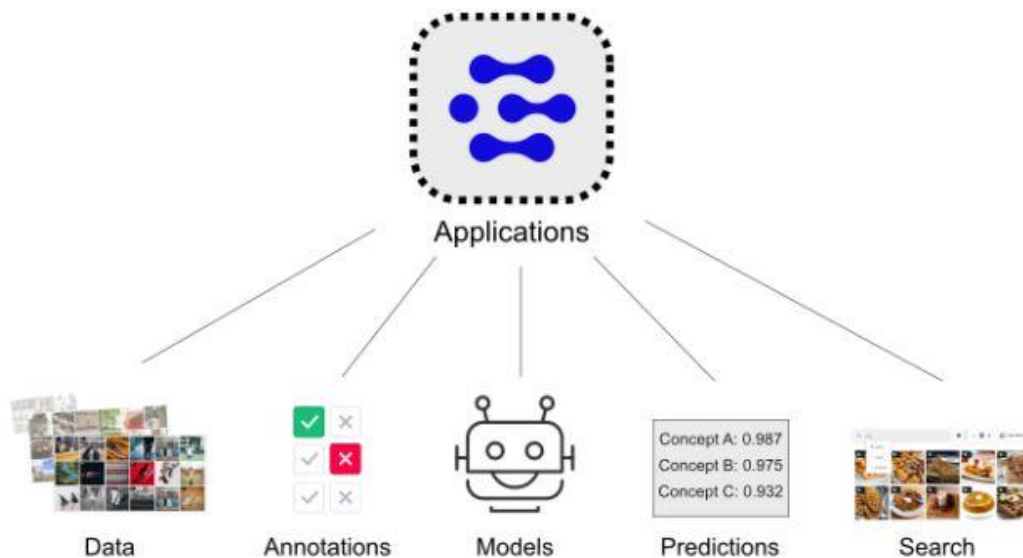
Clarifai menawarkan kemudahan dalam memanfaatkan teknologi *Artificial Intelligence* berbagai keperluan pengguna nya. Clarifai menawarkan *comprehensive set of tools* yang dapat membantu pengguna nya untuk dapat mengelola input mengelola data input, memberikan anotasi untuk pelatihan, membuat model baru, memprediksi, dan mencari data yang telah diolah. Pada Gambar 2.12, merupakan platform diagram dari clarifai yang menunjukkan bagaimana clarifai bekerja.



Gambar 2.12 Clarifai Platform Diagram

Sumber: (Clarifai, 2019)

Untuk dapat menggunakan fitur dari clarifai maka harus membuat *applications* terlebih dahulu dilanjutkan dengan beberapa tahapan seperti memasukkan data baik itu berupa gambar maupun video. Dilanjutkan dengan membuat *concept*, untuk dapat membedakan dalam proses klasifikasi gambar. Kemudian melakukan *annotations* atau yang dapat disebut juga *labeling* yaitu proses untuk mengajari mesin mengenali gambar berdasarkan *concept* yang tersedia dengan cara melabeli setiap input gambar apabila data input berupa gambar. Selanjutnya adalah *training* data dimana *machine learning* yang dimiliki oleh clarifai mulai bekerja. Setelah data berhasil di *training* maka model siap digunakan. Ketika mencoba untuk menginputkan suatu data melalui clarifai API maka, website clarifai akan memberikan respons data dalam bentuk json sesuai model berupa *concept* dengan masing-masing *value* nya. Gambar 2.13, menunjukkan ilustrasi dari clarifai.



Gambar 2.13 Applications pada Clarifai

(Sumber: Clarifai, 2019)

2.8 Imgur

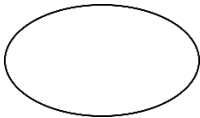
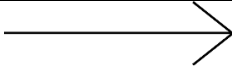
Imgur adalah merupakan salah satu website *hosting gambar* dan situs diskusi komunitas (imgur, 2020). Imgur memiliki fitur seperti media sosial instagram maupun facebook dimana setiap pengguna dapat mencari, mengunggah dan mengomentari gambar yang ada pada situs tersebut. Imgur dapat dimanfaatkan oleh developer untuk membantu melakukan *hosting image* melalui API yang disediakan oleh imgur. API yang digunakan memberikan akses untuk dapat menyimpan gambar di database imgur sehingga pengembang dapat mengakses gambar tersebut melalui URL yang didapatkan dari imgur.

2.9 Use Case Diagram

Use case diagram merupakan gambaran dari fungsi yang di harapkan dari perangkat lunak yang akan dibangun. *Use case diagram* fokus terhadap apa yang dapat dilakukan oleh sebuah sistem. Satu buah use case menggambarkan sebuah interaksi antara aktor dengan sistem terkait apa yang dapat dilakukan oleh sistem. Aktor merupakan entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Tabel 2.1., menunjukkan daftar simbol *use case diagram*.

Tabel 2.1 Daftar simbol pada use case diagram

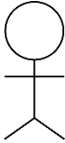

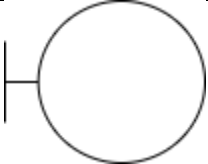
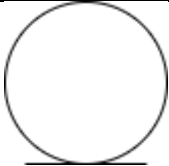

Simbol	Nama	Keterangan
	<i>Actor</i>	Mendeskripsikan peran pengguna terhadap sistem.

	<i>Use case</i>	Gambaran dari fungsional pada sistem, sehingga memudahkan pengembang untuk mengetahui apa yang dapat dilakukan aktor terhadap sistem.
	<i>Association</i>	Abstraksi dari penghubung antara aktor dengan use case.

2.10 Sequence Diagram

Sequence diagram adalah diagram yang menggambarkan hubungan antar objek yang berada di dalam maupun diluar sistem. Tujuan dari pembuatan sequence diagram adalah agar perancangan pada sistem lebih mudah dipahami dan terarah (Rumbaugh et al., 2004). Tabel 2.2, menunjukkan daftar komponen dari *sequence diagram*.

Tabel 2.2 Daftar simbol *sequence diagram*

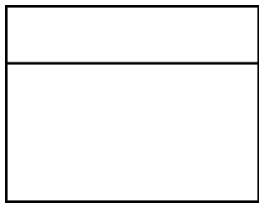
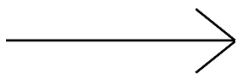
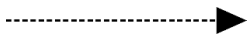

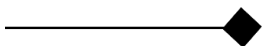
Simbol	Nama	Keterangan
	<i>Actor</i>	Menggambarkan aktor/pengguna yang ada pada sistem.
	<i>Controller</i>	Menggambarkan penghubung antara boundary dengan tabel.
	<i>Boundary</i>	Menggambarkan tepi dari sistem contohnya seperti user interface atau segala sesuatu yang berinteraksi dengan sistem yang lain
	<i>Entity</i>	Menggambarkan komponen yang bertanggung jawab untuk menyimpan informasi atau data..
	<i>Lifeline</i>	Menggambarkan tempat mulai dan berakhirnya sebuah pesan.

	<i>Line Message</i>	Menggambarkan pengiriman pesan.
---	---------------------	---------------------------------

2.11 Class Diagram

Class diagram yaitu gambaran struktur dari sebuah sistem dilihat dari cara mendefinisikan kelas-kelas yang ada pada sistem yang akan dibuat (Rumbaugh, Jacobson, & Booch, 2004). Berikut adalah daftar symbol dari *class diagram* yang ditunjukkan pada Tabel 2.3.

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

2.12 Teknik Pengujian

2.12.1 Pengujian Validasi

Pengujian validasi dipahami sebagai suatu pengujian pada fungsional pada sistem dengan tujuan bahwa fungsional dapat dikatakan valid apabila hasil keluaran sesuai dengan hasil yang diharapkan (Sommerville, 2016). Pengujian validasi dilakukan agar sistem yang sudah dibuat dapat diketahui apakah sudah

dapat berjalan dengan baik atau belum. Pada pengujian ini hasilnya akan berbentuk tabel dengan informasi bahwa kasus uji yang diujikan bernilai valid atau tidak valid.

2.12.2 Pengujian Akurasi

Pengujian akurasi merupakan pengujian yang dilakukan secara langsung terhadap aplikasi untuk mendapatkan nilai akurasi dari sistem dalam mendeteksi penyakit pada tanaman cabai. Pengujian akurasi dilakukan dengan mencoba setiap data uji yang tersedia.

2.12.3 Pengujian *Usability*

Pengujian usability yaitu suatu metode evaluasi untuk mengukur tingkat kemudahan dan kenyamanan penggunaan dan interaksi pengguna dari suatu sistem informasi (Henriyadi & Mulyati, 2016). Dengan menggunakan pengujian *usability*, peneliti dapat mengetahui seberapa mudah suatu perangkat lunak untuk dapat digunakan oleh pengguna.

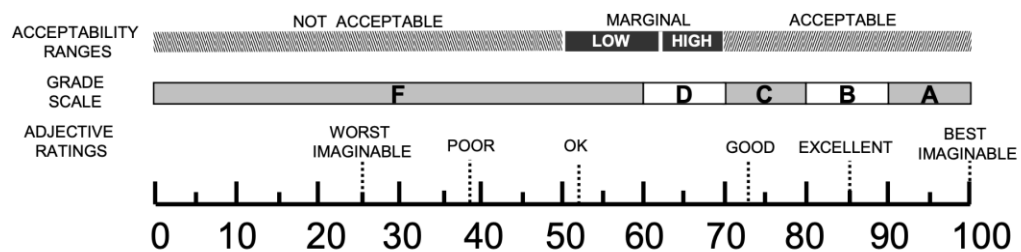
2.12.3.1 SUPR-Qm

Dalam melakukan pengujian *usability*, ada banyak cara yang dapat dilakukan, salah satu cara nya adalah dengan menggunakan instrumen SUPR-Qm (*Standardized User Experience Percentile Rank Questionnaire for Mobile App*). Kuesioner ini memiliki sejumlah 16 (enam belas) pertanyaan yang digunakan untuk melakukan pengujian usability pada *mobile app*. SUPR-QM adalah hasil pengembangan dari kuesioner SUPR-Q yang memiliki 8 (delapan) pertanyaan dalam melakukan pengujian pada *web-based application platform*. Terdapat 4 aspek pertanyaan pada SUPR-Q diantaranya yaitu *usability* atau kegunaan, *credibility* atau kepercayaan, *loyalty* atau kesetiaan, dan *experience* atau pengalaman yang baik (MeasuringU, n.d.).

Pada Gambar 2.14, dapat dilihat daftar pernyataan dari SUPR-Qm yang memiliki sejumlah 16 pernyataan yang dapat digunakan untuk mengukur aplikasi perangkat bergerak dari sisi pengalaman pengguna (Sauro & Zarolia, 2017). Pada pertanyaan SUPR-Qm penilaian dapat dilakukan dengan menggunakan skala Likert yang memiliki suatu nilai rentang dari 1 hingga 5. Setelah mendapatkan nilai dari kuesioner maka nilai tersebut akan di konversikan kedalam kategori usability seperti pada Gambar 2.15. Pada gambar tersebut, dapat dilihat pengelompokan berdasarkan *grade scale*, yaitu *grade F* untuk rentang nilai 0-60, *grade D* untuk rentang nilai 60-70, *grade C* untuk rentang nilai 70-80, *grade B* untuk rentang nilai 80-90, dan *grade A* untuk rentang nilai 90 – 100.

Item	Logit Position	Full Item Wording
CantLiveWo	1.55	I can't live without the app on my phone.
AppBest	1.50	The app is the best app I've ever used.
CantImagineBetter	0.88	I can't imagine a better app than this one.
NeverDelete	0.70	I would never delete the app.
EveryoneHave	0.50	Everyone should have the app.
Discover	0.32	I like discovering new features on the app.
AllEverWant	0.05	The app has all the features and functions you could ever want.
UseFreq	0.03	I like to use the app frequently.
Delightful	-0.04	The app is delightful.
Integrates	-0.04	This app integrates well with the other features of my mobile phone.
DeffFuture	-0.30	I will definitely use this app many times in the future.
FindInfo	-0.59	The design of this app makes it easy for me to find the information I'm looking for.
AppAttractive	-0.71	I find the app to be attractive.
AppMeetsNeeds	-0.79	The app's capabilities meet my requirements.
EasyNav	-1.44	It is easy to navigate within the app.
Easy	-1.63	The app is easy to use.

Gambar 2.14 Daftar Pertanyaan SUPR-Qm



Gambar 2.15 Kategori *Usability* Berdasarkan Nilai

2.12.4 Pengujian *Compatibility*

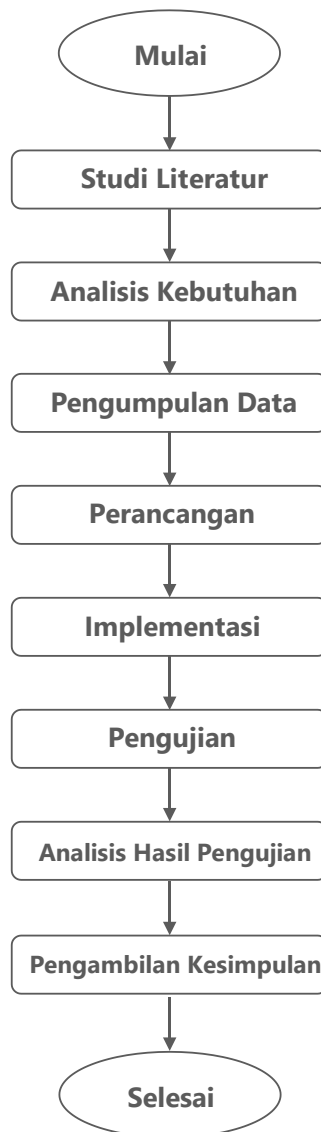
Pengujian *compatibility* bertujuan untuk mengetahui dan memvalidasi lingkungan dan ketergantungan suatu aplikasi (Zhang, Gao, Cheng, & Uehara, 2015). Pengujian *compatibility* dilakukan untuk mengetahui seberapa kompatibel aplikasi yang dikembangkan terhadap perangkat yang tersedia dengan melakukan pengujian terhadap perangkat yang konfigurasi nya berbeda-beda.

2.12.4.1 Firebase Test Lab

Pengujian *compatibility* dibantu dengan instrumen Firebase Test Lab yang memudahkan untuk menjalankan aplikasi pada beberapa perangkat dengan konfigurasi yang berbeda. Firebase Tet Lab memiliki kemampuan untuk menyimulasikan penggunaan aplikasi yang dapat membantu menemukan kegagalan aplikasi yang dijalankan (Firebase, 2019).

BAB 3 METODOLOGI

Untuk melakukan pengembangan sebuah perangkat lunak, maka dibutuhkan suatu metode dengan tujuan agar penelitian dapat dilakukan secara terstruktur. Adapun alur dari penelitian ini ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

Gambar 3.1, menunjukkan 8 tahapan proses pengembangan pada penelitian ini, yaitu studi literatur, analisis kebutuhan, pengumpulan data, perancangan, implementasi, pengujian, analisis hasil pengujian, dan pengambilan kesimpulan.

3.1 Studi Literatur

Peneliti mengumpulkan beberapa literatur pada bidang ilmu yang terkait sebagai acuan dan referensi untuk penelitian yang sedang dilakukan. Beberapa literatur yang tersebut meliputi:

1. Penyakit dan hama pada tanaman cabai.
2. *SDLC Waterfall*
3. *Model View Presenter (MVP)*
4. Android
5. Clarifai
6. *Use case Diagram*
7. *Sequence Diagram*
8. *Class Diagram*
9. Pengujian Validasi
10. Pengujian *Usability*
11. *Standardized User Experience Percentile Rank Questionnaire for Mobile App (SUPR-QM)*.
12. Pengujian *Compatibility*

3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mendapatkan kebutuhan fungsional dan non fungsional dari perangkat lunak. Proses penggalan kebutuhan dilakukan dengan teknik wawancara dan observasi secara langsung di daerah lahan yang di naungi oleh Balai Pengkajian Teknologi Pertanian (BPTP) Karangploso, Malang, Jawa Timur. Wawancara dilakukan secara langsung terhadap pakar hama dan penyakit pada tanaman cabai. 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*, *use case scenario* dan *activity diagram*.

3.3 Pengumpulan Data

Pada proses pengumpulan data dilakukan pengambilan data dari Balai Pengkajian Teknologi Pertanian (BPTP) Jawa Timur, yang berada di daerah Karangploso, Malang, Jawa Timur. Data yang dikumpulkan berupa data latih dan data uji untuk masing-masing penyakit yang ada di lapangan.

3.4 Perancangan

Tahap perancangan akan dilakukan setelah proses analisis kebutuhan diselesaikan. Di tahap perancangan akan dilakukan pemodelan sistem menggunakan diagram *Unified Modeling Language (UML)*, yaitu *sequence*

diagram, dan *class diagram* serta dilakukan perancangan basis data, perancangan algoritme dan perancangan antarmuka.

3.5 Implementasi

Tahapan implementasi dilakukan dengan penulisan kode program sesuai dengan rancangan yang dibuat. Implementasi dilakukan berdasarkan *sequence diagram* dan *class diagram* sesuai pada tahap melakukan perancangan. Pada tahap ini pula akan dijelaskan detail penggunaan clarifai API untuk membangun aplikasi pendeteksi penyakit pada tanaman cabai menggunakan Android Studio IDE dan java sebagai bahasa pemrogramannya.

3.6 Pengujian

Setelah selesai melakukan implementasi, maka selanjutnya dilakukan tahap pengujian yang terdiri dari empat pengujian yaitu pengujian validasi, pengujian akurasi, pengujian *usability* dan pengujian *compatibility*. Pengujian validasi dilakukan untuk mengetahui seluruh fitur yang ada pada aplikasi apakah sudah dapat berjalan dengan baik atau tidak. Pengujian akurasi dilakukan untuk mendapatkan persentase rata-rata akurasi dari hasil deteksi aplikasi untuk mengetahui nama penyakit pada daun yang dideteksi. Pada pengujian ini akan didapatkan berapa persen akurasi aplikasi dalam mengenali gambar berdasarkan data latih yang telah di *training* oleh clarifai dan data uji yang akan diujikan pada aplikasi yang sudah terintegrasi dengan clarifai. Pengujian *usability* dilakukan dengan menggunakan metode SUPR-QM (*Standardized User Experience Percentile Rank Questionnaire for Mobile App*). Seperti namanya pengujian ini dilakukan untuk mengukur kualitas pengalaman pengguna dalam menggunakan aplikasi *mobile* (Sauro & Zarolia, 2017). Pada pengujian *usability* responden diberikan sebuah kuesioner kepada pengguna seperti pakar penyakit dan hama pada tanaman cabai dan petani beserta dua responden lain yang merupakan pegawai dari BPTP Jawa Timur. Total jumlah responden adalah 5 orang dikarenakan dengan mengamati jumlah responden sebanyak 5 orang dapat mengamati 85% kesalahan yang ada pada sistem yang dibangun (Sauro & Lewis, 2012). Pengujian *compatibility* dilakukan untuk mengetahui apakah aplikasi dapat berjalan pada perangkat mobile sistem operasi android dengan minimal level SDK 23 (Marshmallow). Android SDK level 23 merupakan level yang disarankan oleh Android Studio ketika pertama kali membuat proyek aplikasi.

3.7 Analisis Hasil Pengujian

Peneliti melakukan analisis berdasarkan pengujian yang telah didapatkan untuk memperoleh hasil yang lebih dalam tentang:

1. Bagaimana hasil pengujian validasi dari aplikasi pendeteksi penyakit pada tanaman cabai?
2. Bagaimana hasil akurasi dari aplikasi pendeteksi penyakit pada tanaman cabai?

3. Bagaimana hasil pengujian *usability* dari aplikasi pendeteksi penyakit pada tanaman cabai?
4. Bagaimana hasil pengujian *compatibility* dari aplikasi pendeteksi penyakit pada tanaman cabai?

3.8 Pengambilan Kesimpulan dan Saran

Pada proses pengambilan kesimpulan dan saran, peneliti menarik kesimpulan dari proses yang telah dilakukan untuk menjawab rumusan masalah yang ada. Selanjutnya dengan penulisan saran terhadap penelitian yang sedang diteliti oleh penulis berdasarkan kekurangan yang ditemukan pada proses pengembangan.

BAB 4 ANALISIS KEBUTUHAN

Pada tahap ini, dilakukan analisis kebutuhan untuk membantu merancang sistem pada tahap selanjutnya. Pada bab analisis kebutuhan dimulai dengan tahap penggalan kebutuhan, identifikasi aktor, spesifikasi kebutuhan dan pemodelan kebutuhan. Kemudian hasil dari bab analisis kebutuhan akan digunakan pada bab selanjutnya yaitu perancangan sistem.

4.1 Penggalan Kebutuhan

4.1.1 Hasil Wawancara

Pada tahap teknik menggali kebutuhan dilakukan dengan wawancara dan observasi langsung lapangan. Wawancara dilakukan terhadap calon pengguna dan paham tentang pengguna yaitu kepada seorang peneliti yang bergerak di bidang pakar hama dan penyakit pada tanaman cabai. Tabel 4.1, menunjukkan kebutuhan yang didapatkan oleh peneliti.

Tabel 4.1 Temuan Kebutuhan

No	Temuan Kebutuhan
1	Belum adanya sistem yang dapat mengetahui hama dan penyakit cabai secara praktis.
2	Belum adanya sistem yang memudahkan petani untuk dapat menambah pengetahuan informasi dalam mengenali hama dan penyakit pada tanaman cabai
3	Belum adanya sistem yang dapat melakukan dokumentasi setelah hasil analisis.

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 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 kebutuhan 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 peng-kodean 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

Pada penelitian ini, penulis menganalisis dan membangun sebuah perangkat lunak berupa aplikasi *mobile* untuk mendeteksi penyakit pada tanaman cabai. Nama lain dari aplikasi ini yaitu DiSnap. Dibangunnya aplikasi DiSnap bertujuan untuk membantu para petani dan pakar penyakit dan hama tanaman cabai 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* *clarifai* yang menyediakan API maupun SDK untuk melakukan proses pengenalan gambar yang sebelumnya telah dilatih untuk mengenali dan mengklasifikasikan gambar.

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 aktivitas 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. Identifikasi aktor yang ada pada sistem ini ditunjukkan seperti pada Tabel 4.3.

Tabel 4.3 Aktor Sistem

Aktor	Deskripsi	Aktor yang relevan
Pengguna	Pengguna adalah aktor yang menggunakan seluruh fitur pada sistem. Pengguna berinteraksi dengan sistem secara langsung untuk melakukan proses mendeteksi penyakit pada tanaman cabai.	Petani cabai, pakar hama dan penyakit tanaman cabai.

4.3 Spesifikasi Kebutuhan

4.3.1 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem adalah kebutuhan yang harus tersedia pada sistem, hal ini termasuk dalam bagaimana sebuah sistem merespons masukan dari pengguna, dapat memberikan informasi ketika sistem dalam kondisi tertentu serta dapat menyelesaikan masalah dalam rumusan masalah sebelumnya (Sommerville, 2011). Sistem yang akan dibangun memiliki 5 kebutuhan fungsional seperti yang ditunjukkan pada Tabel 4.4.

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 pada tanaman cabai dan cara pengendalian serta pemberian pestisida berdasarkan penyakit daun pada tanaman cabai yang menyerangnya.	Mengetahui informasi penyakit dan pengendalian penyakit

5	F-DS-04	Sistem harus mampu menyediakan informasi tentang riwayat data aktivitas 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

4.3.2 Kebutuhan Non Fungsional Sistem

Kebutuhan non fungsionalitas adalah kebutuhan yang memiliki fokus dalam membantu jalannya sistem dan perilaku sistem. Kebutuhan fungsionalitas diartikan sebagai fungsi yang ditawarkan atau suatu batasan layanan pada sistem (Sommerville, 2011). Aplikasi yang akan dikembangkan memiliki beberapa kebutuhan non-fungsionalitas seperti *Usability* yaitu kemudahan dalam menggunakan aplikasi dan *Compatibility* yaitu aplikasi hanya mampu berjalan minimal di platform Android minimal SDK level 23 (Marshmallow).

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

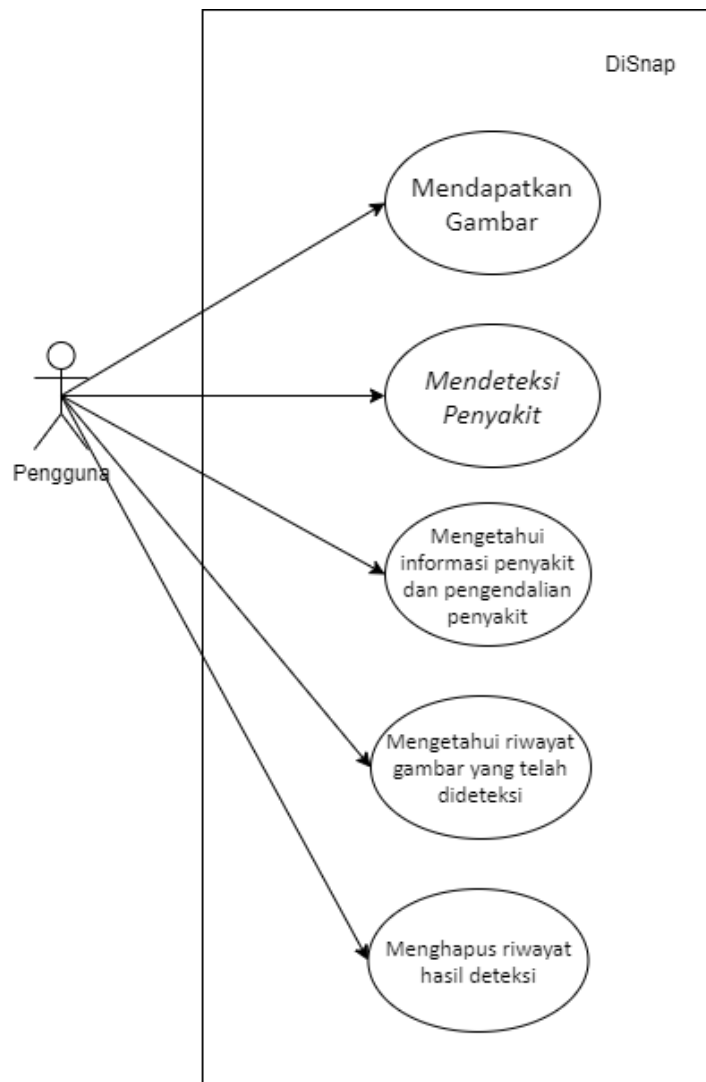
4.4 Pemodelan Kebutuhan

Pemodelan kebutuhan yaitu adalah tahap melakukan pemodelan pada kebutuhan yang telah didapatkan. Pemodelan ini menggunakan teknik *use case diagram* dan *use case scenario*. *Use case diagram* merupakan gambaran ringkas siapa yang menggunakan sistem dan apa saja yang dapat dilakukan terhadap sistem tersebut. Sedangkan *use case scenario* adalah jalur jalannya proses dari sisi aktor terhadap sistem.

4.4.1 *Use case Diagram*

Pemodelan *use case diagram* pada aplikasi yang akan dibangun ditunjukkan pada Gambar 4.1. Di 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

4.4.2 Pemodelan Use case Scenario

Pemodelan skenario *use case* dapat dibuat sesuai dilakukan pembuatan *use case diagram*. Skenario *use case* dibuat untuk menjelaskan detail dari proses pada setiap *use case*. Tabel 4.6 hingga Tabel 4.9, menunjukkan hasil dari skenario *use case* dari sistem yang akan dibangun.

4.4.2.1 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"> 1. Sistem menampilkan halaman utama aplikasi 2. Pengguna menekan menu snap pada menu utama aplikasi 3. Sistem menampilkan halaman Snap. 4. Sistem menampilkan <i>bottom sheet dialog</i> berupa pilihan menu untuk mengambil gambar melalui kamera atau galeri 5. Pengguna memilih satu metode pengambilan gambar melalui kamera atau galeri 6. Pengguna menekan <i>button next</i> 7. Sistem menampilkan halaman <i>cropping image</i> 8. Pengguna melakukan <i>cropping image</i> 9. Sistem memproses image sesuai ukuran yang telah dipilih pengguna 10. Sistem menampilkan gambar hasil <i>cropping image</i> pada halaman Analyze 11. Sistem siap untuk mendeteksi gambar
Post-Kondisi	Pengguna berhasil mendapatkan gambar yang siap untuk dideteksi
Alternatif	<ol style="list-style-type: none"> 1. Jika pengguna memilih mengambil gambar melalui kamera maka sistem akan menampilkan <i>camera screen</i> 2. Jika pengguna memilih mengambil gambar melalui galeri maka sistem akan menampilkan <i>gallery screen</i>

4.4.2.2 Skenario *Use case* Mendeteksi Penyakit

Tabel 4.7 Skenario *Use case* Mendeteksi Penyakit

Item	Deskripsi
Kode	F-DS-02

Nama	Mendeteksi 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"> 1. Pengguna berada pada halaman Analyze 2. Sistem menampilkan gambar yang sudah didapatkan pada proses sebelumnya 3. Pengguna menekan tombol analyze image pada layar 4. Sistem mendeteksi koneksi internet pada perangkat 5. Jika perangkat terhubung dengan koneksi internet maka sistem melanjutkan ke proses selanjutnya 6. Sistem menampilkan progress bar loading dan persentase loading 7. Sistem mengirimkan gambar ke <i>imgur hosting image</i> 8. Setelah mendapatkan <i>URL image</i> dari <i>imgur hosting image</i>, <i>URL image</i> dikirim ke <i>clarifai API</i> untuk dideteksi 9. Sistem melakukan pendeteksian penyakit pada gambar yang telah dikirim 10. Apabila berhasil sistem menampilkan pesan “Analisis sukses” 11. Sistem akan menampilkan hasil deteksi pada halaman Result 12. Ketika pengguna menekan tombol back/close maka hasil pendeteksian akan disimpan di <i>database</i> <p>Sistem menampilkan pesan “Hasil deteksi dapat dilihat pada menu riwayat”</p>
Post-Kondisi	Pengguna berhasil mendapatkan hasil deteksi penyakit pada halaman <i>Result</i>
Alternatif	Jika perangkat tidak terhubung dengan internet maka sistem akan menampilkan pesan “Periksa koneksi internet anda”

4.4.2.3 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"> 1. Pengguna memilih menu <i>home</i> 2. Sistem menampilkan menu home 3. Sistem akan menampilkan berbagai informasi tentang penyakit pada tanaman cabai 4. Pengguna memilih salah satu penyakit 5. Sistem menampilkan halaman Detail Disease Info 6. Sistem menampilkan detail informasi penyakit pada tanaman cabai ke layar
Post-Kondisi	Pengguna berhasil memperoleh informasi detail penyakit pada tanaman cabai
Alternatif	-

4.4.2.4 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">1. Pengguna memilih menu riwayat2. Sistem mengambil data riwayat aktivitas deteksi dari <i>database</i> lokal3. Jika pada <i>database</i> terdapat data hasil aktivitas maka sistem mengambil semua data tersebut untuk ditampilkan4. Sistem menampilkan menu riwayat5. Sistem menampilkan daftar gambar riwayat yang pernah dideteksi6. Pengguna memilih salah satu riwayat7. Sistem menampilkan detail riwayat hasil deteksi gambar pada tanaman cabai.
Post-Kondisi	Pengguna berhasil memperoleh informasi riwayat penyakit yang pernah dideteksi.
Alternatif	Jika tidak ada data riwayat hasil deteksi pada <i>database</i> maka sistem akan menampilkan informasi bahwa tidak ada riwayat aktivitas deteksi.

4.4.2.5 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">1. Pengguna memilih menu riwayat2. Sistem menampilkan menu riwayat3. Sistem menampilkan daftar gambar riwayat yang pernah dideteksi4. Pengguna menekan tombol remove pada salah satu riwayat5. Sistem menampilkan dialog dengan pesan “Apakah anda yakin untuk menghapus riwayat ini?”6. Pengguna memilih pilihan “Ya”7. Sistem menghapus riwayat yang dipilih pengguna8. Sistem menampilkan pesan “Riwayat berhasil dihapus”9. Sistem mengatur ulang urutan daftar riwayat dengan urutan terbaru
Post-Kondisi	<ol style="list-style-type: none">1. Pengguna berhasil menghapus riwayat yang dipilih2. Sistem menampilkan daftar riwayat terbaru
Alternatif	-

4.5 Perancangan *Activity Diagram*

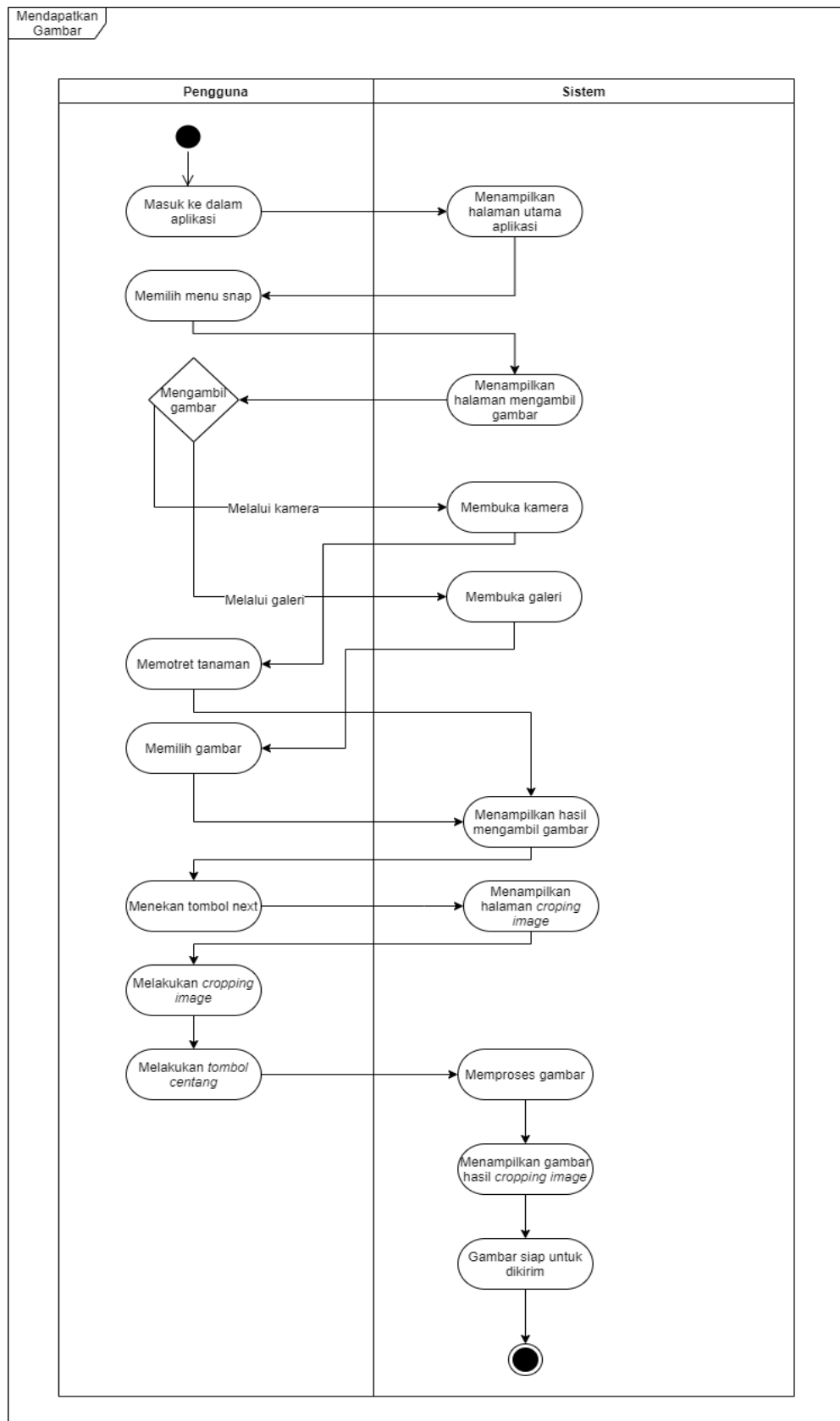
Pada tahap ini dilakukan perancangan *activity diagram* dari use case yang telah dibuat untuk memudahkan dalam mengetahui cara kerja dari setiap *use case* yang ada pada sistem. Gambar 4.2 sampai dengan Gambar 4.6 menunjukkan hasil dari perancangan *activity diagram*.

4.5.1 *Activity Diagram* Mendapatkan Gambar

Gambar 4.2, menunjukkan *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. Kemudian, 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 mengambil gambar melalui kamera maka sistem akan menuju fitur kamera pada aplikasi, apabila memilih mengambil gambar melalui galeri maka sistem akan menuju fitur 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, kemudian sistem akan menuju halaman analyze yaitu halaman yang menampilkan gambar hasil dari proses sebelumnya dan kemudian siap untuk dilakukan pendeteksian pada gambar.



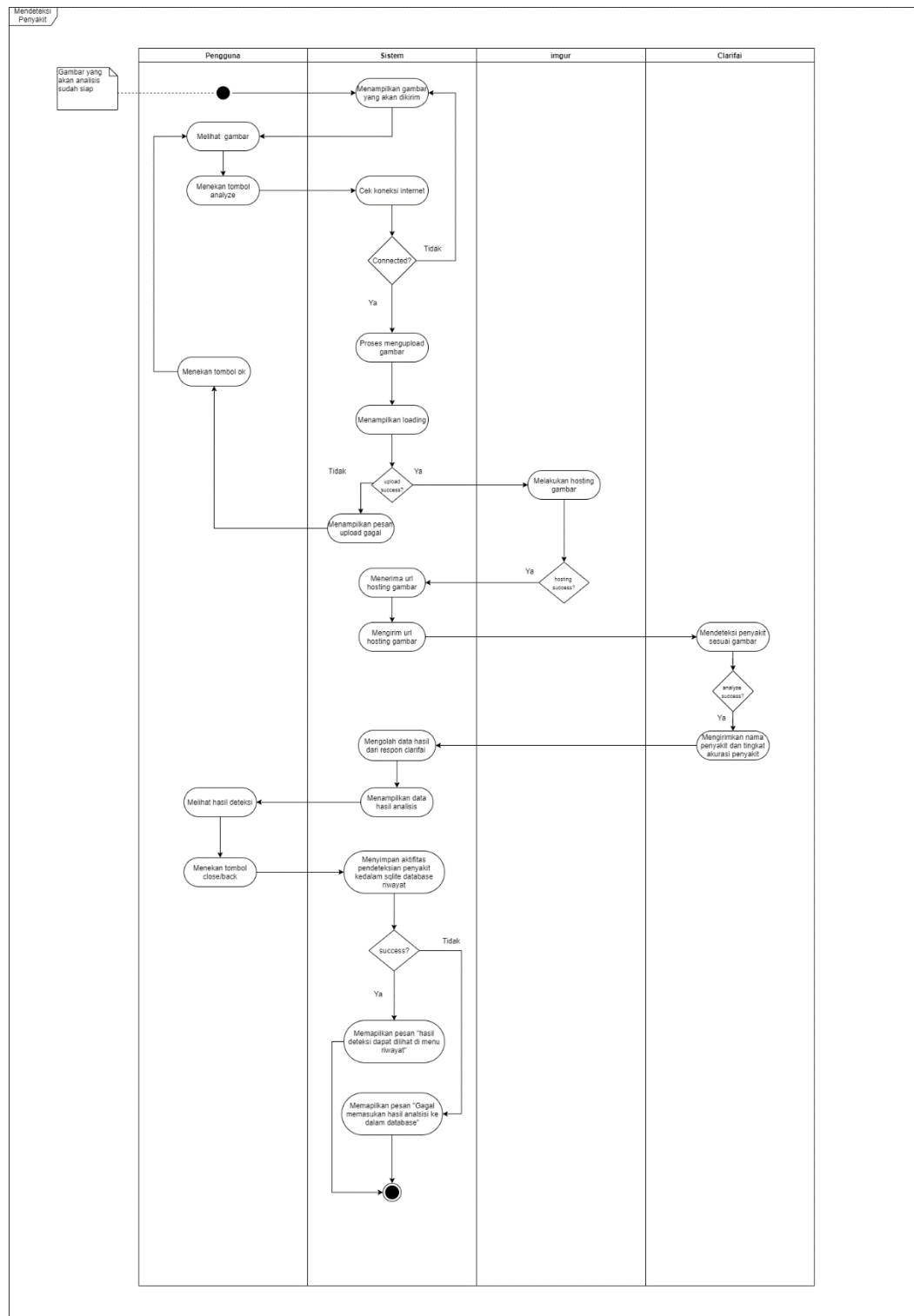
Gambar 4.2 Activity Diagram Mendapatkan Gambar

4.5.2 Activity Diagram Mendeteksi Penyakit

Gambar 4.3, menunjukkan daftar gambar yang akan dikirim oleh pengguna sudah siap untuk dikirim. Selanjutnya pengguna menekan tombol *analyze image*, apabila perangkat memiliki koneksi internet maka sistem akan menampilkan informasi “Periksa koneksi internet anda” dan pengguna masih berada pada halaman *analyze*, apabila terhubung dengan koneksi internet maka terjadi proses *upload* atau unggah gambar seperti yang terdapat pada Gambar 4.3. Sementara dilakukan proses unggah, maka sistem menampilkan proses *loading* kepada pengguna. Dalam tahapannya gambar terlebih dahulu dilakukan *hosting* di website imgur dengan tujuan untuk mendapatkan *URL image* yang selanjutnya akan digunakan untuk mendeteksi 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 memberikan informasi kepada pengguna bahwa proses deteksi berhasil. Selanjutnya, pengguna menekan tombol lanjutkan. Kemudian sistem mulai menampilkan nama penyakit, cara penanganan, dan saran pemberian pestisida terhadap pengguna. Selanjutnya sistem menyimpan riwayat hasil aktivitas mendeteksi penyakit di SQLite *database* yang nanti akan diakses kembali pada fitur riwayat.

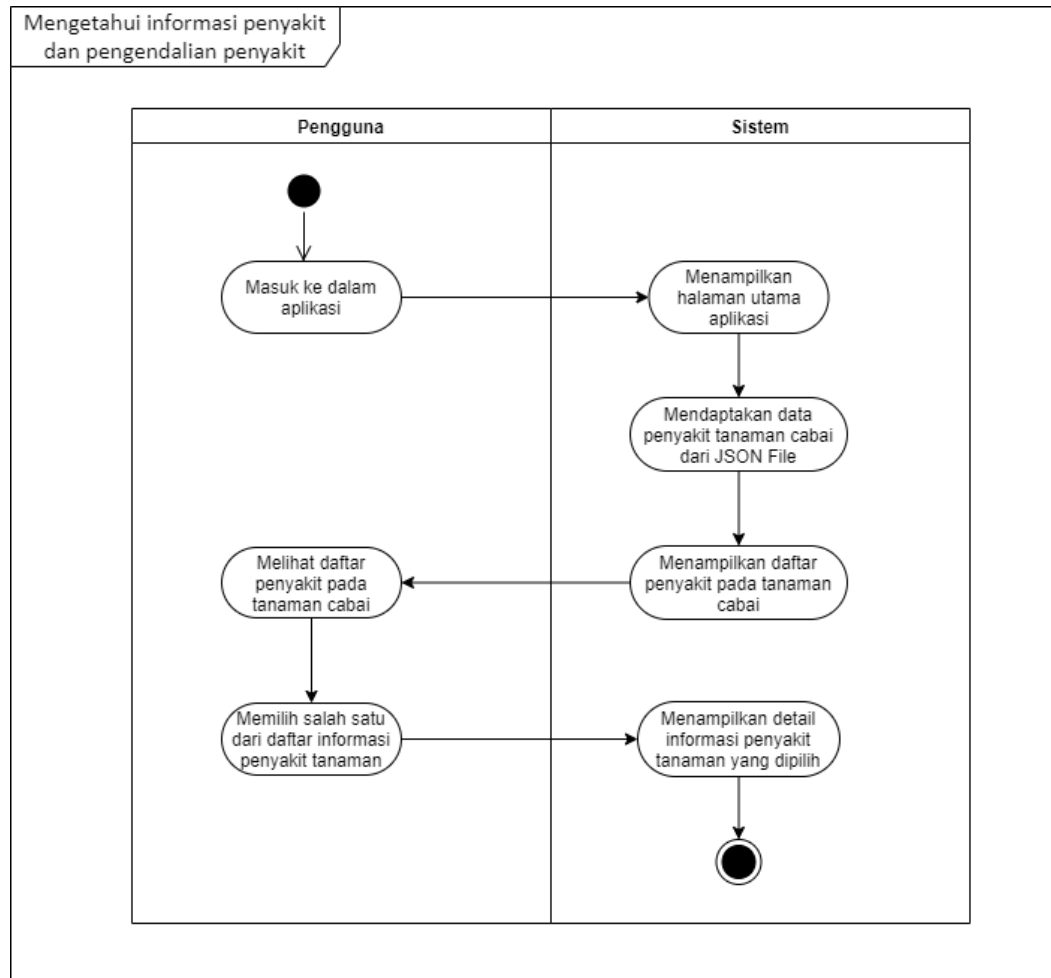


Gambar 4.3 Activity Diagram Mendeteksi Penyakit

4.5.3 Activity Diagram Mengetahui informasi penyakit dan pengendalian penyakit

Gambar 4.4, menunjukkan pengguna masuk kedalam sistem, kemudian sistem menampilkan halaman utama aplikasi. Kemudian pengguna memilih menu

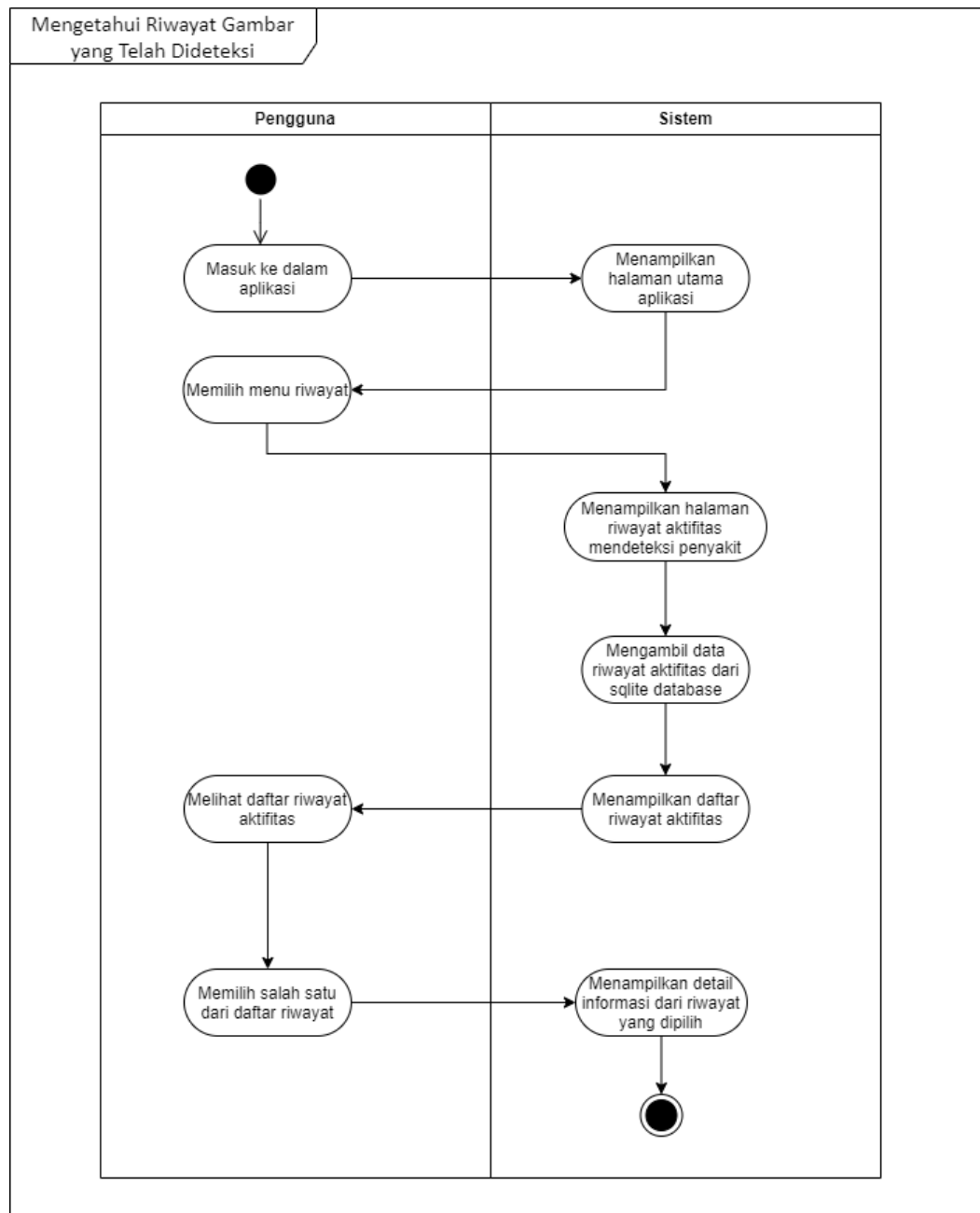
home dan sistem menampilkan halaman home. Kemudian sistem akan mengambil data informasi penyakit dari JSON file yang selanjutnya sistem menampilkan daftar informasi penyakit kepada pengguna. Setelah itu pengguna memilih satu dari daftar informasi penyakit yang ada kemudian sistem menampilkan detail informasi kepada pengguna.



Gambar 4.4 Activity Diagram Mengetahui Informasi Penyakit dan Pengendalian Penyakit

4.5.4 Activity Diagram Mengetahui Riwayat Gambar yang Telah Dideteksi

Gambar 4.5, menunjukkan pengguna masuk kedalam menu utama aplikasi dan sistem menampilkan menu utama dalam aplikasi. Selanjutnya pengguna memilih menu riwayat dan kemudian sistem menampilkan halaman riwayat aktivitas mendeteksi. Sistem mengambil data riwayat aktivitas 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.

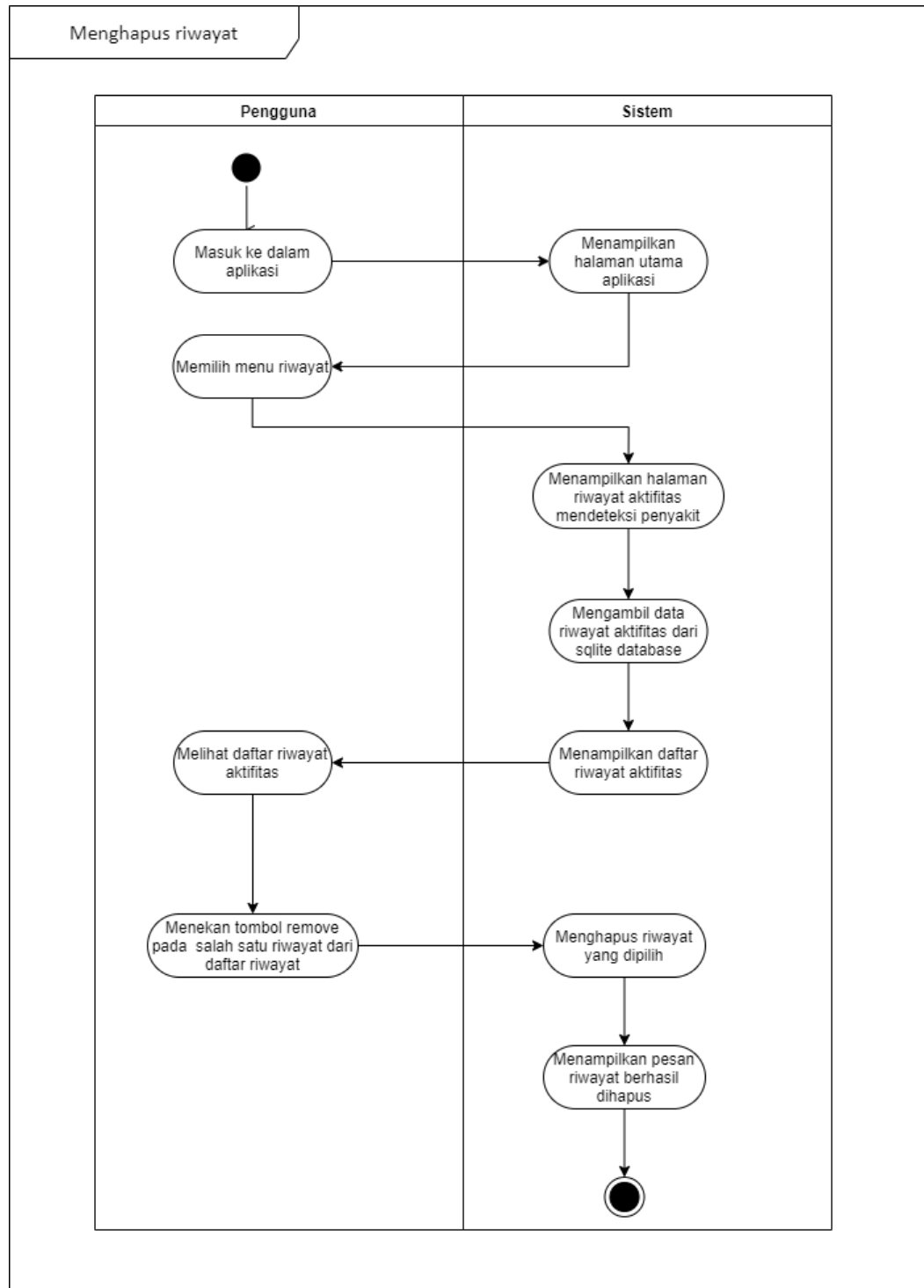


Gambar 4.5 Activity diagram Mengetahui Riwayat Gambar yang Telah Dideteksi

4.5.5 Activity Diagram Menghapus Riwayat Hasil Analisis

Gambar 5.6, menunjukkan pengguna masuk kedalam menu utama aplikasi dan sistem menampilkan menu utama dalam aplikasi. Kemudian pengguna memilih menu riwayat dan kemudian sistem menampilkan halaman riwayat aktivitas mendeteksi. Sistem melakukan pengambilan data riwayat aktivitas 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

aktivitas yang dipilih dan sistem menampilkan pesan “Remove success”. Selanjutnya sistem memperbarui daftar riwayat aktivitas pada halaman riwayat.



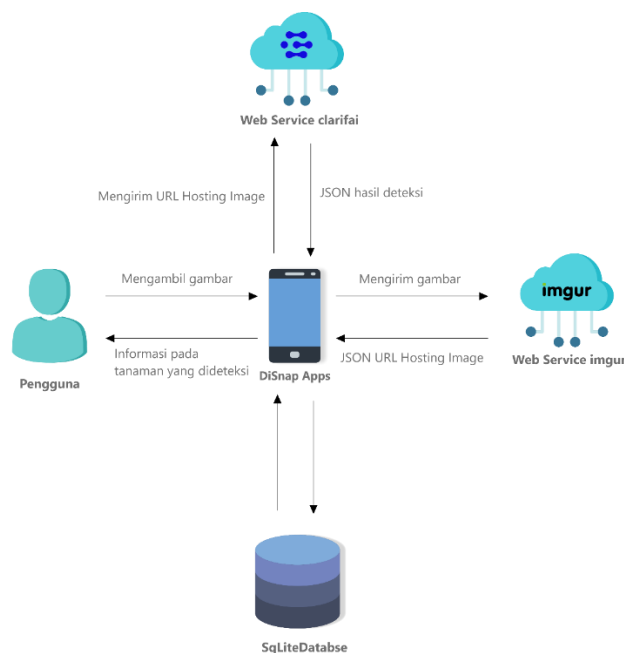
Gambar 4.6 Activity Diagram Menghapus Riwayat Hasil Analisis

BAB 5 PERANCANGAN

Pada tahap ini membahas mengenai rancangan aplikasi DiSnap berbasis android. Perancangan tersebut terdiri dari perancangan arsitektur sistem, sequence diagram, class diagram, perancangan basis data, perancangan antarmuka pengguna (*wireframe*) dan perancangan algoritme.

5.1 Perancangan Arsitektur Sistem

Perancangan ini berguna untuk membantu menjelaskan bagaimana sistem yang akan dibangun bekerja. Gambar 5.1, menunjukkan gambaran umum dari aplikasi DiSnap. Dalam gambar tersebut, 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 *imgur 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 hasil nya akan dikirim kembali ke *smartphone* pengguna dengan format *JSON*. Data dalam format *JSON* tersebut kemudian dikelola untuk diolah menjadi informasi yang akan disampaikan kepada pengguna. Selanjutnya hasil dari aktivitas pendeteksian akan disimpan kedalam lokal *database* pada *smartphone* pengguna.



Gambar 5.1 Arsitektur Sistem

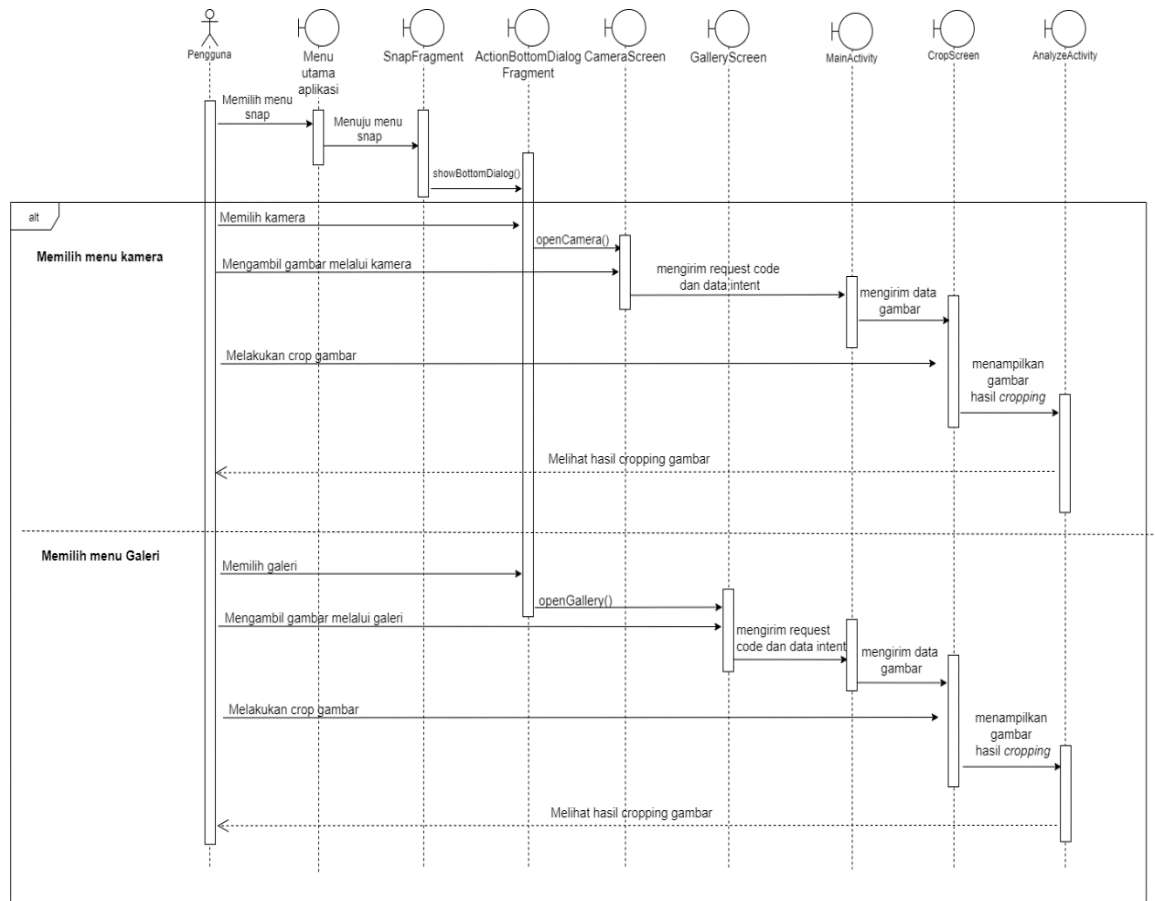
5.2 Perancangan *Sequence Diagram*

Sequence diagram merupakan diagram gambaran 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.

5.2.1 *Sequence Diagram* Mendapatkan Gambar

Gambar 5.2, menunjukkan 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 5.2 Sequence Diagram Mendapatkan Gambar

5.2.2 Sequence Diagram Mendeteksi Penyakit

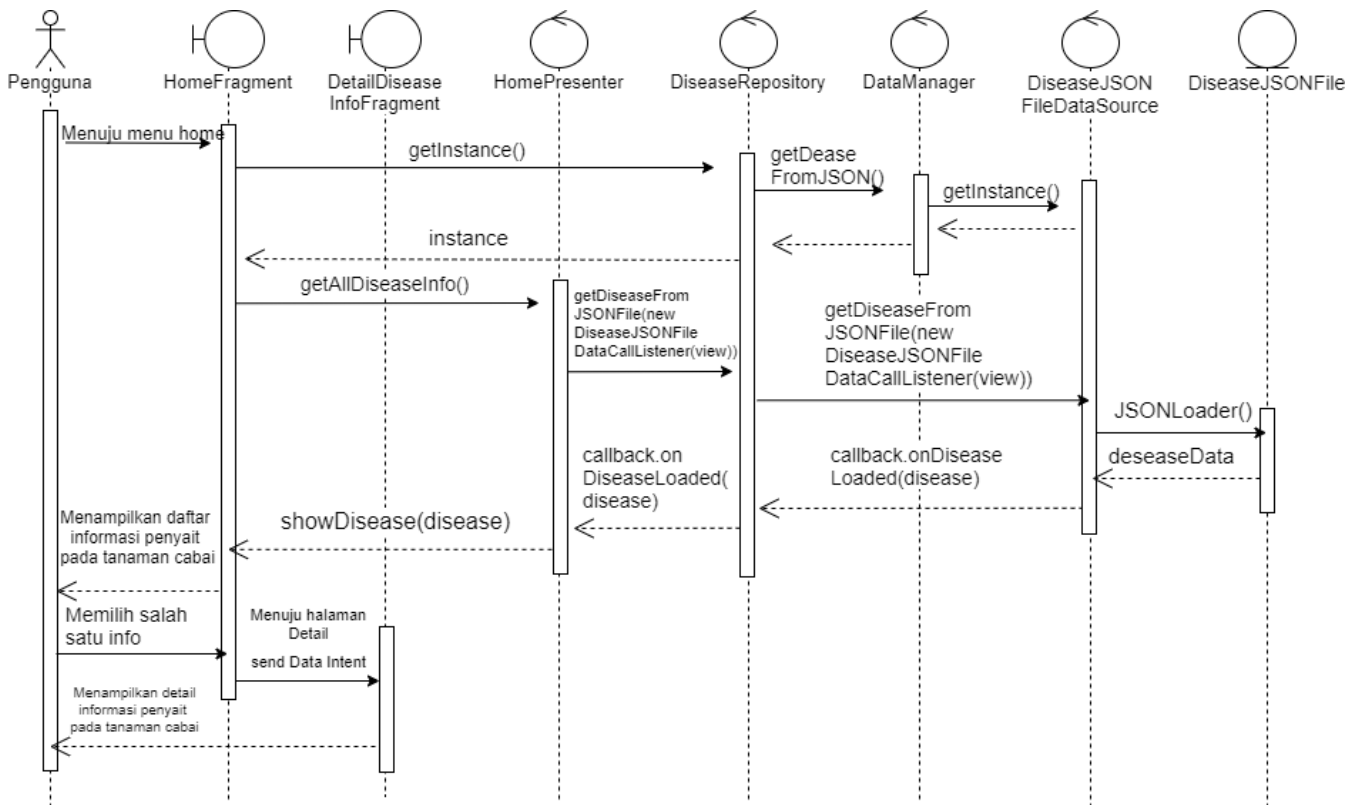
Pada Gambar 5.3, pengguna sudah berada pada halaman *AnalyzeActivity*. Setelah itu, pengguna menekan tombol *analyze image*, maka sistem akan melakukan pengecekan koneksi internet pada *smartphone* pengguna dengan memanggil method *isNetworkAvailable()*. Apabila kembalian dari method tersebut adalah *false* maka memberikan informasi “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 *imgur*. 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 di dalamnya 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 `onInsertSuccess(message)`. Kemudian method `showInsertSuccess(message)` di *override* pada ResultActivity. Maka pesan sistem akan menampilkan pesan kepada pengguna bahwa riwayat aktivitas dapat dilihat kembali pada menu riwayat.

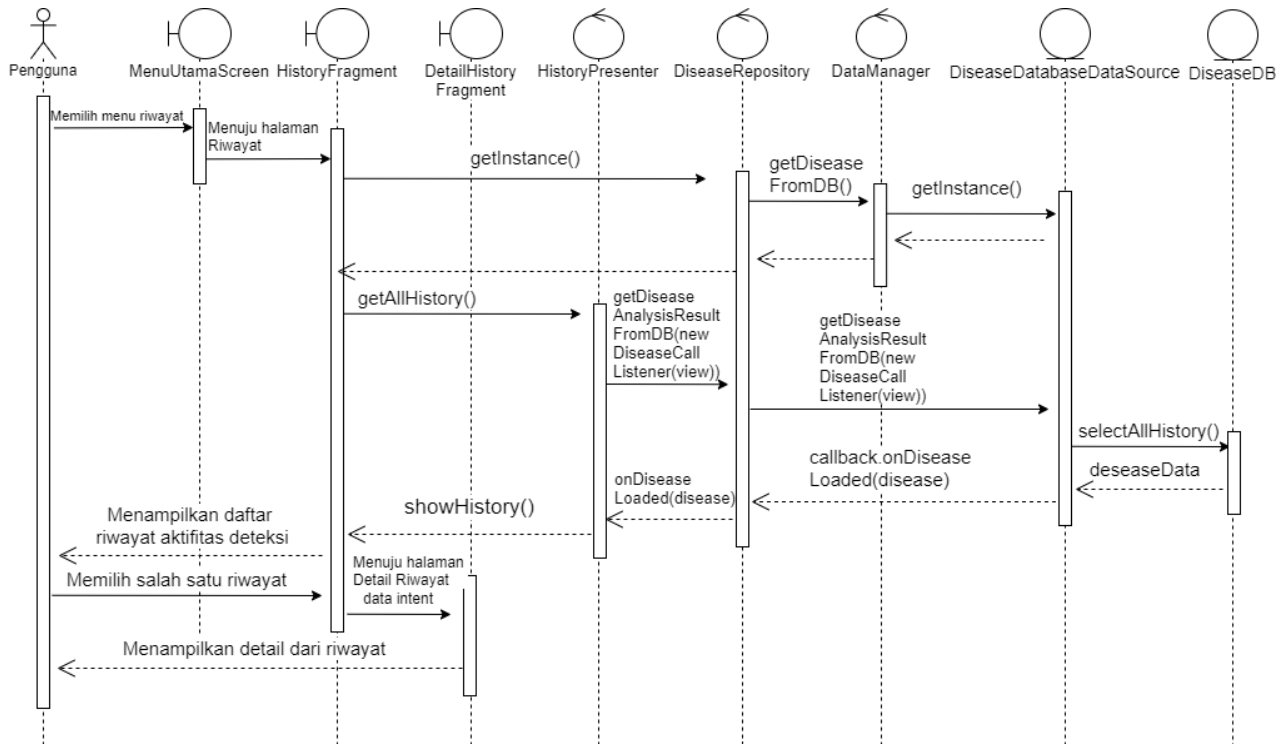
pengguna melihat daftar gambar penyakit pada tanaman cabai dan apabila di klik salah satu *item*, pengguna melihat detail dari penyakit tersebut.



Gambar 5.4 Sequence Diagram Mengetahui Informasi Penyakit dan Pengendalian Penyakit

5.2.4 Sequence Diagram Mengetahui Riwayat Gambar yang Telah Dideteksi

Pada Gambar 5.5, Gambar 5.5 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(newDiseaseCallListener(view))` pada DiseaseRepository. Pemanggilan method dengan nama yang sama juga dilakukan pada DiseaseDatabaseDataSource dan kemudian mengambil data riwayat aktivitas 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.



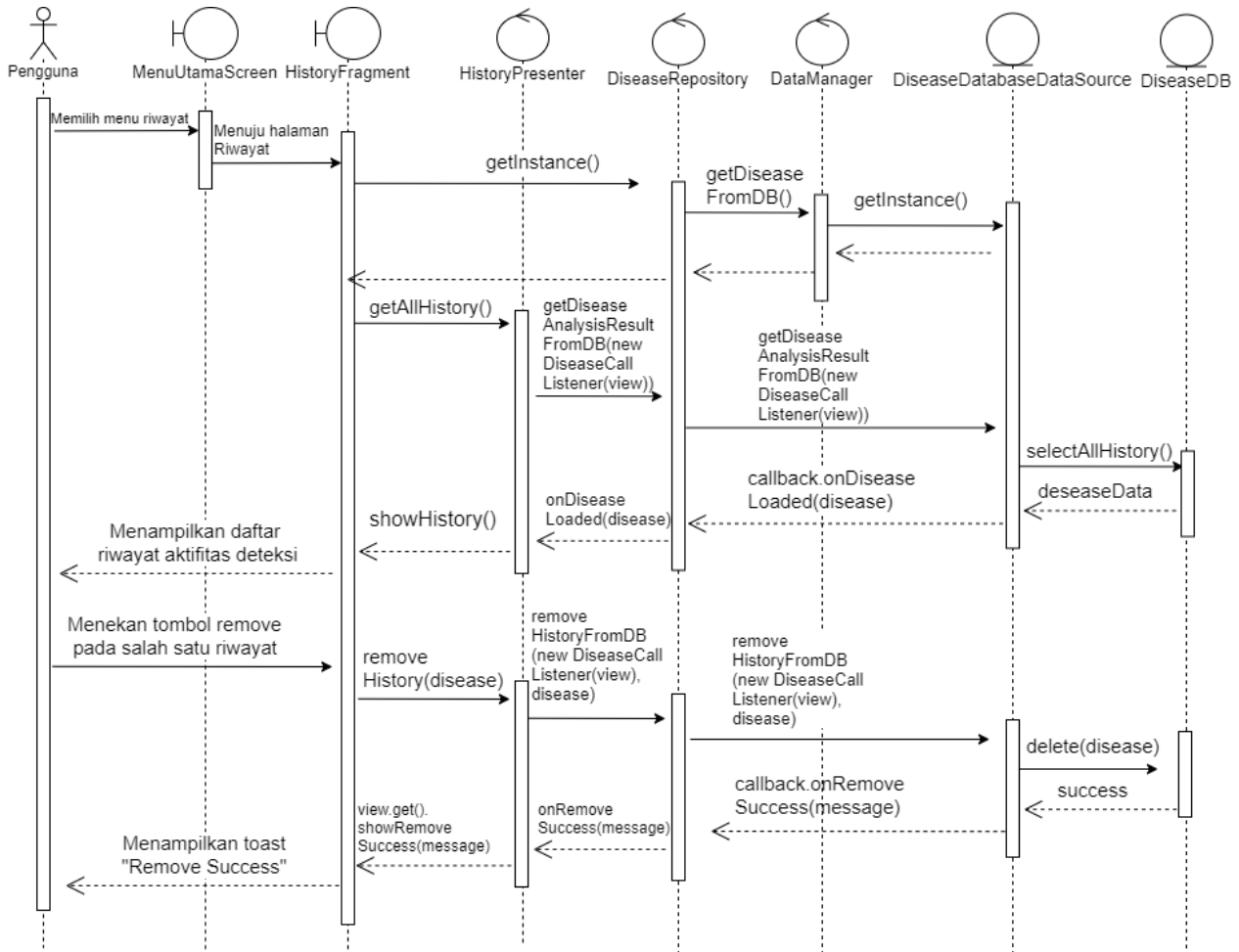
Gambar 5.5 *Sequence Diagram* Mengetahui Riwayat Gambar yang Telah Dideteksi

5.2.5 Sequence Diagram Menghapus Riwayat Hasil Deteksi

Pada Gambar 5.6, 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 `removeHistory(disease)` pada `HistoryPresenter`. Dilanjutkan pemanggilan `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 5.6 Sequence Diagram Menghapus Riwayat Hasil Deteksi

5.3 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. Setiap kelas memiliki hubungan untuk menjalankan fungsionalitas tertentu. Pada bagian ini tidak akan dimasukkan iterasi hanya dimasukkan hasil akhir dari kebutuhan pengguna. Iterasi hanya dilakukan pada pembuatan perancangan antarmuka

```

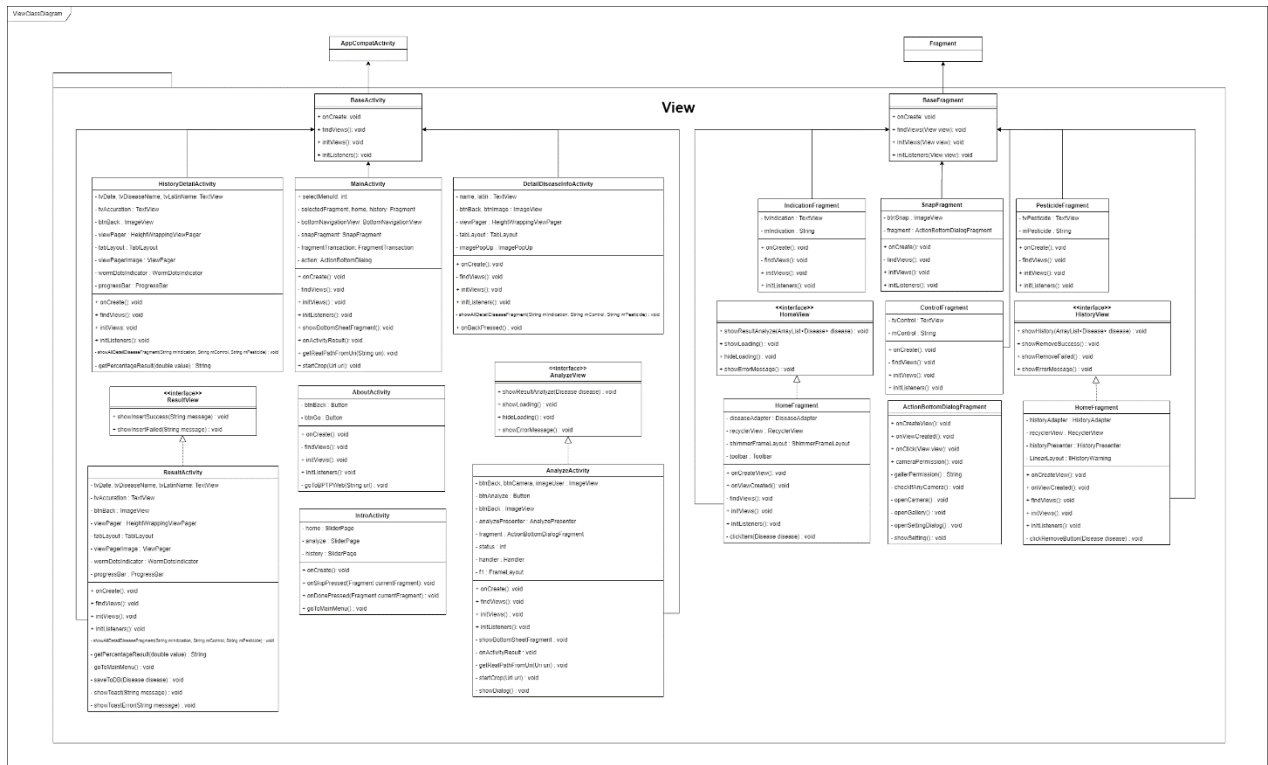
classDiagram
    class AppCompatActivity
    class BaseActivity
    class MainActivity
    class HistoryDetailActivity
    class AboutActivity
    class ResultView["<<interface>>  
ResultView"]
    class ResultActivity
    class DetailDiseaseInfoActivity
    class SplashScreenActivity
    class IntroActivity
    class AnalyzerView["<<interface>>  
AnalyzerView"]
    class AnalyzeActivity
    class ViewPagerAdapterAdapter
    class BaseFragment
    class IndicationFragment
    class SnapFragment
    class PesticideFragment
    class ControlFragment
    class HomeView["<<interface>>  
HomeView"]
    class HomeFragment
    class ActionBottomDialogFragment
    class DiseaseAdapter
    class HistoryAdapter
    class HistoryFragment
    class DiseaseRepository
    class DataManager
    class JSONFile
    class DiseaseJSONFileDataSource
    class Remote
    class DiseaseRemoteDataSource
    class Database
    class DiseaseDAO["<<interface>>  
DiseaseDAO"]
    class DiseaseDatabaseDataSource
    class AppDatabase
    class Connectivity
    class Constant
    class DiskExecutor

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

The diagram illustrates the architecture of a disease management application, organized into several layers:

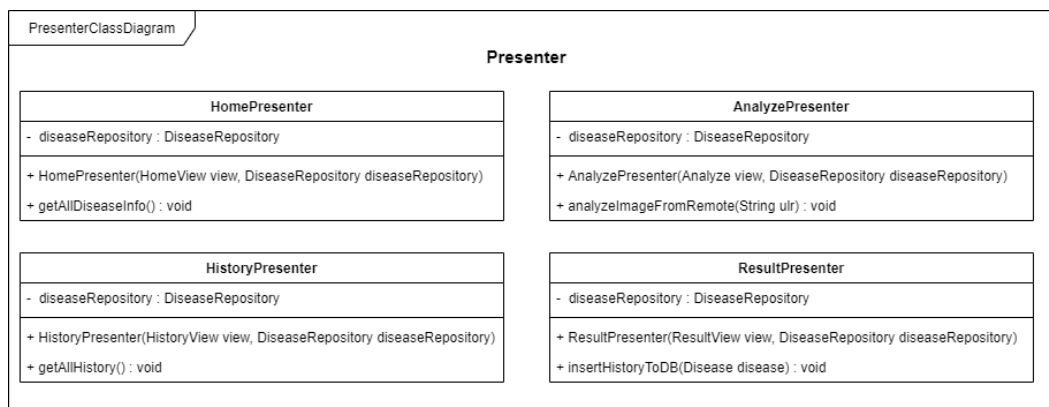
- View Layer:** Contains activities (e.g., `MainActivity`, `HistoryDetailActivity`, `AboutActivity`, `DetailDiseaseInfoActivity`, `SplashScreenActivity`, `IntroActivity`, `AnalyzeActivity`) and fragments (e.g., `IndicationFragment`, `SnapFragment`, `PesticideFragment`, `ControlFragment`, `HomeFragment`, `ActionBottomDialogFragment`, `HistoryFragment`). It also includes interfaces like `<<interface>> ResultView` and `<<interface>> AnalyzerView`.
- Presenter Layer:** Contains presenters such as `ResultPresenter`, `AnalyzePresenter`, `HomePresenter`, and `HistoryPresenter`.
- Repository Layer:** Contains the `DiseaseRepository` interface and its implementations: `DiseaseDatabaseDataSource` (which interacts with the `Database` layer) and `DiseaseRemoteDataSource` (which interacts with the `Remote` layer).
- Model Layer:** Contains the `Disease` entity.
- Util Layer:** Contains utility classes like `Connectivity`, `Constant`, and `DiskExecutor`.

The diagram shows dependencies between these components, indicating how data flows from the Model through the Repository and Presenter layers to the View layer, and how the View layer interacts with the Util layer.



Gambar 5.8 Class Diagram Package View

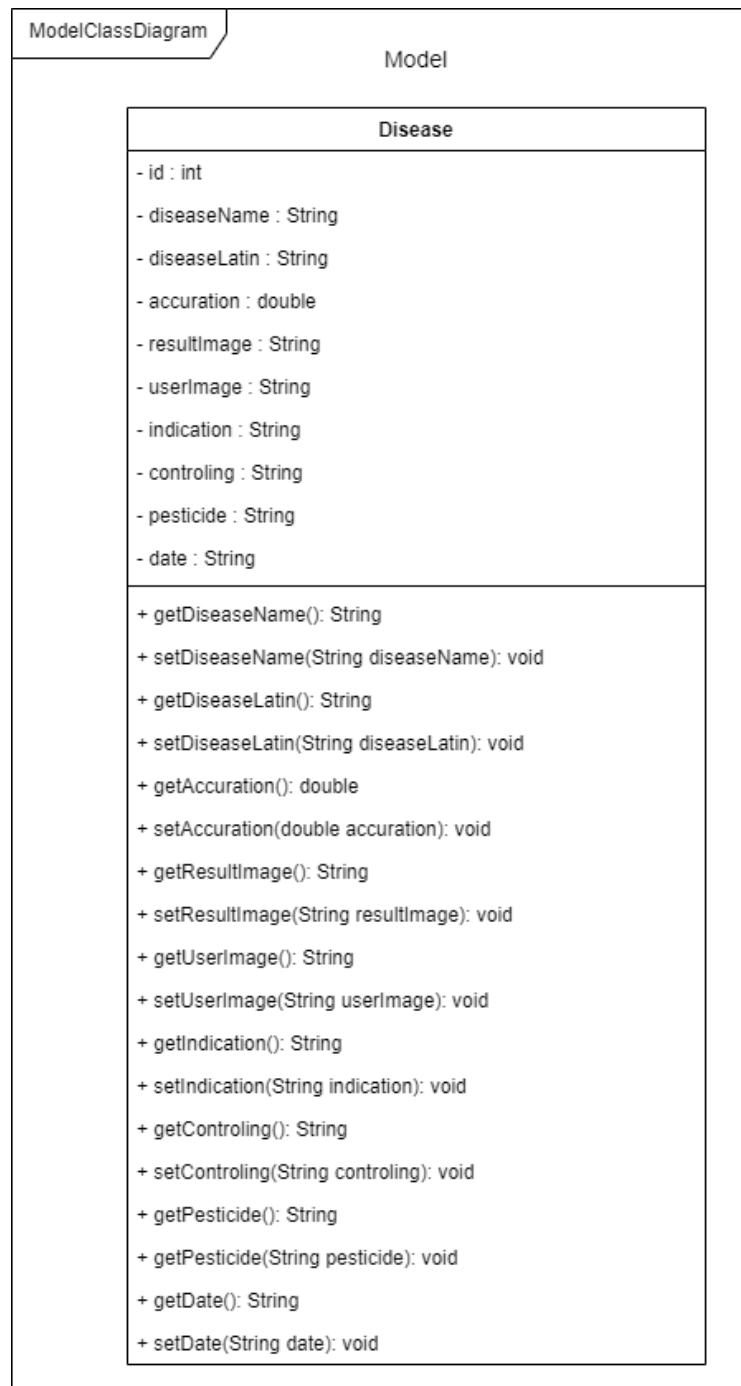
Gambar 5.8, menunjukkan package view yang ada pada sistem, terdapat beberapa kelas seperti MainActivity, ResultActivity, HistoryDetailActivity, DetailDiseaseInfoActivity, AnalyzeActivity yang meng-extends BasicActivity. Ada pula BasicFragment, HomeFragment, IndicationFragment, ControllingFragment, PesticideFragment, HistoryFragment yang meng-extends BasicFragment.



Gambar 5.9 Class Diagram Package Presenter

Pada Gambar 5.9, terdapat beberapa *class Presenter* yang berfungsi untuk 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 *claifai 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 5.10 Class Diagram Package Model

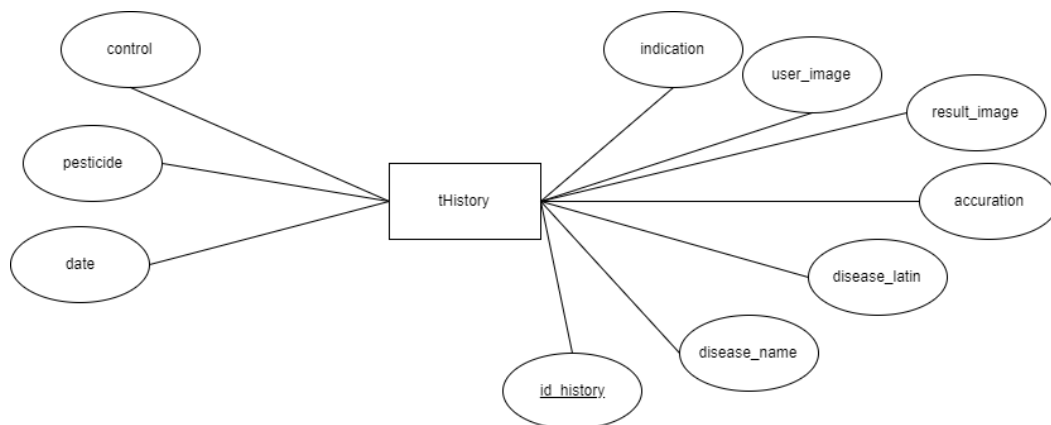
Pada Gambar 5.10, terdapat satu *class model* yaitu Disease. Pada class tersebut terdapat beberapa atribut seperti `id : int`, `diseaseName : String`, `diseaseLatin : String`, `accuration : double`, `resultImage : String`, `userImage : String`, `indication : String`, `controlling : String`, `pesticide : String`, `date : String`. Kelas ini digunakan untuk membantu menyimpan dan mengolah data pada *database* serta membantu menampilkan info penyakit kepada pengguna.

5.4 Perancangan Basis Data

Perancangan basis data membahas tentang basis data yang akan digunakan dalam proses implementasi sistem yang dibuat. Perancangan basis data terdiri dari perancangan ERD (*Entity Relationship Diagram*) dan tabel.

5.4.1 Perancangan ERD

Pada Gambar 5.11 menunjukkan ERD dari aplikasi DiSnap. 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 5.11.



Gambar 5.11 Entity Relationship Diagram DiSnap

5.4.2 Perancangan Tabel

1. Tabel History

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

Nama tabel: tHistory

Nama kelas : Disease

Nama *database* : DiseaseDB

Pengguna Fungsi: Menyimpan data riwayat aktivitas deteksi penyakit

Tabel 5.1 Rancangan Tabel Basis Data DiSnap

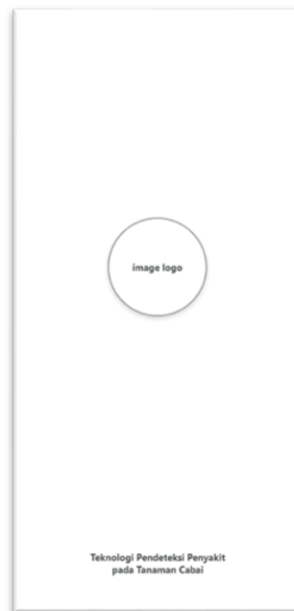
No	Nama Kolom	Tipe Data	Panjang	Keterangan	Auto Generate	Primary Key
1.	id_history	integer	11	NOT NULL	true	true
2.	disease_name	varchar	255	NULL	false	false
3.	disease_latin	varchar	255	NOT NULL	false	false
4.	accuration	varchar	255	NULL	false	false

5.	result_image	varchar	255	NULL	false	false
6.	user_image	varchar	255	NULL	false	false
7.	indication	varchar	255	NULL	false	false
8.	control	varchar	255	NULL	false	false
9.	pesticide	varchar	255	NULL	false	false
10.	date	varchar	255	NULL	false	false

5.5 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 5.12 sampai Gambar 5.24.

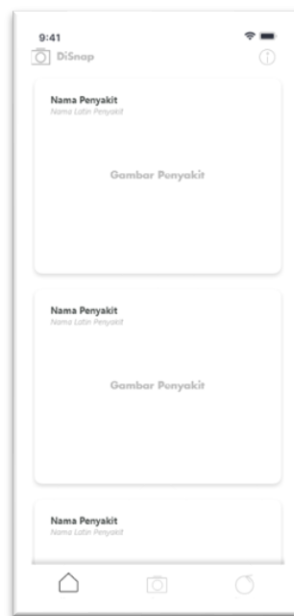
5.5.1 Perancangan Antarmuka Splash Screen



Gambar 5.12 Wireframe Splash Screen

Gambar 5.12, merupakan gambaran halaman Splash Screen seperti aplikasi pada umumnya ketika pengguna membuka aplikasi maka akan muncul logo aplikasi. Pada aplikasi DiSnap peneliti merancang untuk membuat halaman Splash Screen yang berisikan logo aplikasi dengan posisi di tengah layar serta deskripsi singkat dari aplikasi DiSnap pada bagian bawah aplikasi.

5.5.2 Perancangan Antarmuka Home



Gambar 5.13 Wireframe Home

Gambar 5.13, 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.

5.5.3 Perancangan Antarmuka Detail



Gambar 5.14 Wireframe Detail

Gambar 5.14, yaitu adalah gambaran 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. Di bagian bawah gambar merupakan keterangan berupa gejala dan pengendalian dari penyakit.

5.5.4 Perancangan Antarmuka Snap



Gambar 5.15 Wireframe Snap (Bottom sheet)

Pada Gambar 5.15, 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.

5.5.5 Perancangan Antarmuka Camera



Gambar 5.16 Wireframe Camera

Pada Gambar 5.16, menunjukkan aktivitas mengambil gambar tanaman menggunakan kamera. Pada aktivitas ini pengguna dapat mengarahkan atau memfokuskan kamera pada kotak fokus yang telah tersedia yang selanjutnya dapat memudahkan aktivitas cropping image pada tahap selanjutnya setelah *button* lingkaran pada kamera ditekan.

5.5.6 Perancangan Antarmuka *Cropping Image*



Gambar 5.17 Wireframe *Cropping Image*

Pada Gambar 5.17, menunjukkan halaman *cropping image* yaitu aktivitas 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.

5.5.7 Perancangan Antarmuka *Analyze*



Gambar 5.18 Wireframe *Analyze*

Pada Gambar 5.18, 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 analize image* yang berfungsi untuk memulai aktivitas analisis gambar dengan men-*hosting* gambar melalui website imgur kemudian mengirimkan gambar ke Webservice API Clarifai.

5.5.8 Perancangan Antarmuka *Result*



Gambar 5.19 Wireframe Result

Pada Gambar 5.19, menunjukkan halaman *result* yaitu halaman ketika hasil analisis gambar dari Webservice 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 aktivitas dan *icon camera* untuk memulai dari awal aktivitas 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 diunggah.

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.

5.5.9 Perancangan Antarmuka *About Apps*



Gambar 5.20 Wireframe *About Apps*

Wireframe about apps dapat dilihat pada Gambar 5.20. Halaman ini memberikan informasi pengguna terhadap aplikasi DiSnap. Di dalam halaman ini terdapat *appbar* yang bertuliskan *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.

5.5.10 Perancangan Antarmuka Riwayat

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 5.21.



Gambar 5.21 Wireframe Menu Riwayat

5.5.11 Wireframe Screen Intro

Pada Gambar 5.22, 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 5.23, pengguna diberi tahu bahwa dengan aplikasi DiSnap pengguna dapat melakukan pendeteksian penyakit dengan mendeteksi daun pada tanaman cabai. Sedangkan Gambar 5.23, pengguna diberi tahu bahwa pengguna dapat melihat kembali riwayat aktivitas hasil mendeteksi penyakit pada tanaman cabai.



Gambar 5.22 Wireframe Intro Home



Gambar 5.23 Wireframe *Intro Snap*



Gambar 5.24 Wirframe *Intro History*

5.6 Perancangan Algoritme

Perancangan algoritme pada aplikasi DiSnap digunakan untuk membantu mendeskripsikan alur logika terhadap program aplikasi 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.

5.6.1 Perancangan Komponen Mengetahui Informasi Penyakit dan Pengendalian Penyakit

Pada Tabel 5.2 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 memilih salah satu *item*, maka pengguna akan dibawa ke halaman detail dari item yang di pilih.

Tabel 5.2 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

5.6.2 Perancangan komponen mendeteksi penyakit

Pada Tabel 5.3 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 *analyze image*, sistem akan melakukan pengecekan terhadap koneksi internet. Apabila perangkat terhubung dengan internet, maka sistem akan mengirimkan HTTP Request berupa *post/upload* dan menunggu respons hasil deteksi berupa nama penyakit dan tingkat akurasi yang nanti akan ditampilkan kepada pengguna. Apabila tidak perangkat tidak terhubung dengan koneksi internet, maka sistem akan menampilkan tampilan tidak ada koneksi internet.

Tabel 5.3 Algoritme Perancangan Komponen Mendeteksi Penyakit

PSEUDOCODE	
1	START
2	Gambar yang akan di deteksi sudah siap
3	Menekan tombol analyze image
4	Memilih salah satu pilihan camera atau gallery
5	IF(cek koneksi internet)
6	Mengirimkan gambar ke imgur
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

5.6.3 Perancangan Komponen Mengetahui Riwayat Gambar yang Telah Dideteksi

Pada Tabel 5.4, pengguna menuju halaman riwayat. Pada halaman riwayat sistem akan memanggil presenter untuk mendapatkan data dari lokal *database* melalui repository. Apabila data tidak sama dengan 0 maka *database* akan mengembalikan nilai berupa data dan data selanjutnya akan diolah pada bagian UI. Apabila ternyata tidak data pada lokal *database*, maka sistem akan menampilkan pesan tidak ada riwayat aktivitas tersimpan pada *database*.

Tabel 5.4 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 aktivitas 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
17	ENDIF
18	END

BAB 6 IMPLEMENTASI

Pada tahap implementasi, peneliti melakukan pengembangan aplikasi DiSnap dengan berpedoman pada hasil dari proses perancangan yang telah dibuat pada tahap sebelumnya. Dalam proses model *Waterfall*, maka peneliti sudah memasuki tahap Implementasi. Pembahasan yang terdapat pada implementasi yaitu terdiri dari spesifikasi sistem, batasan implementasi, 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 untuk mengembangkan aplikasi.

6.2 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras untuk mengembangkan aplikasi 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 *mobile smartphone* 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 *Mobile Smartphone*

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

Pengembangan aplikasi DiSnap membutuhkan spesifikasi perangkat lunak yang mendukung. Spesifikasi dari perangkat lunak komputer ditunjukkan pada Tabel 6.3, sedangkan pada Tabel 6.4 menunjukkan spesifikasi perangkat lunak dari *mobile smartphone* dalam membantu pengembangan.

Tabel 6.3 Spesifikasi Perangkat Lunak Komputer

Nama Komponen	Spesifikasi
<i>Operating System</i>	Windows 10
<i>Programming Language</i>	Java
IDE (<i>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 sejumlah batasan dalam proses implementasinya. Berikut beberapa batasan implementasi dari aplikasi DiSnap yaitu:

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.
3. Aplikasi dikembangkan dengan menggunakan bahasa pemrograman java
4. Aplikasi memanfaatkan pihak ketiga yaitu imgur untuk *men-hosting* gambar
5. Aplikasi memanfaatkan pihak ketiga yaitu clarifai untuk proses identifikasi daun tanaman cabai.
6. Aplikasi memanfaatkan library Fast Android Networking versi 1.0.2 sebagai *Rest Client* pada Android
7. Untuk dapat menggunakan fitur mendeteksi penyakit dibutuhkan koneksi internet.

6.5 Implementasi Basis Data

Basis data yang digunakan pada aplikasi yang akan dibangun memiliki nama DiseaseDB dengan hanya menggunakan satu tabel yaitu tabel History dengan naman tHistory. 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 libray room dari Google untuk membantu mempermudah peneliti dalam mengelola data pada *database* SQLite yang merupakan database lokal yang digunakan dalam pengembangan. Implementasi basis data aplikasi DiSnap dapat dilihat pada Tabel 6.5.

Tabel 6.5 Implementasi Tabel History

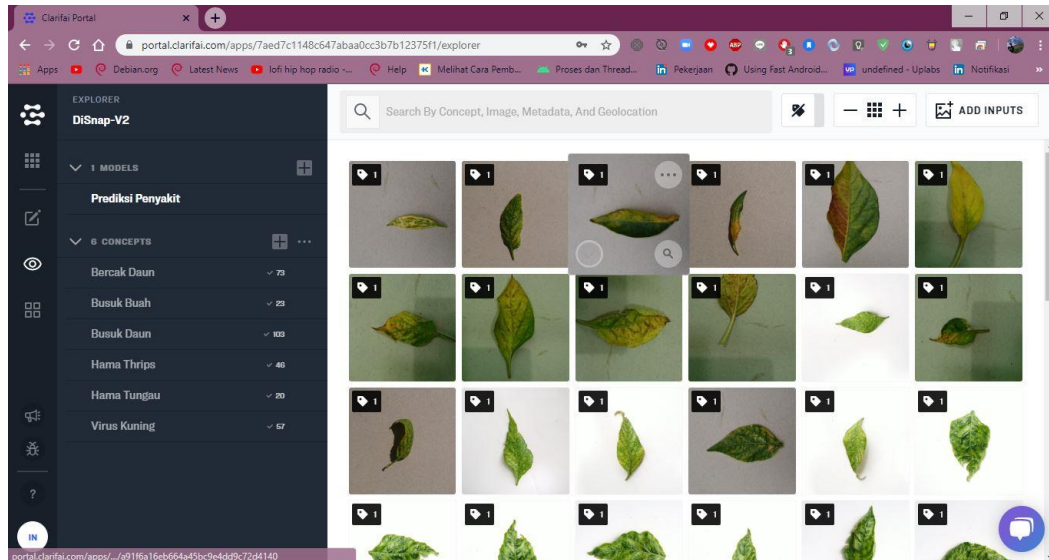
No	Source code
1	<code>@Entity(tableName = "tHistory")</code>
2	<code>public class History implements Serializable{</code>
3	
4	<code> @PrimaryKey(autoGenerate = true)</code>
5	<code> private int id;</code>
6	
7	<code> @ColumnInfo(name = "disease_name")</code>
8	<code> private String diseaseName;</code>
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

Tahap *define* adalah tahap pembuatan *concept* dan pelabelan pada data gambar yang sudah di *upload*. *Concept* adalah label yang akan disematkan 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

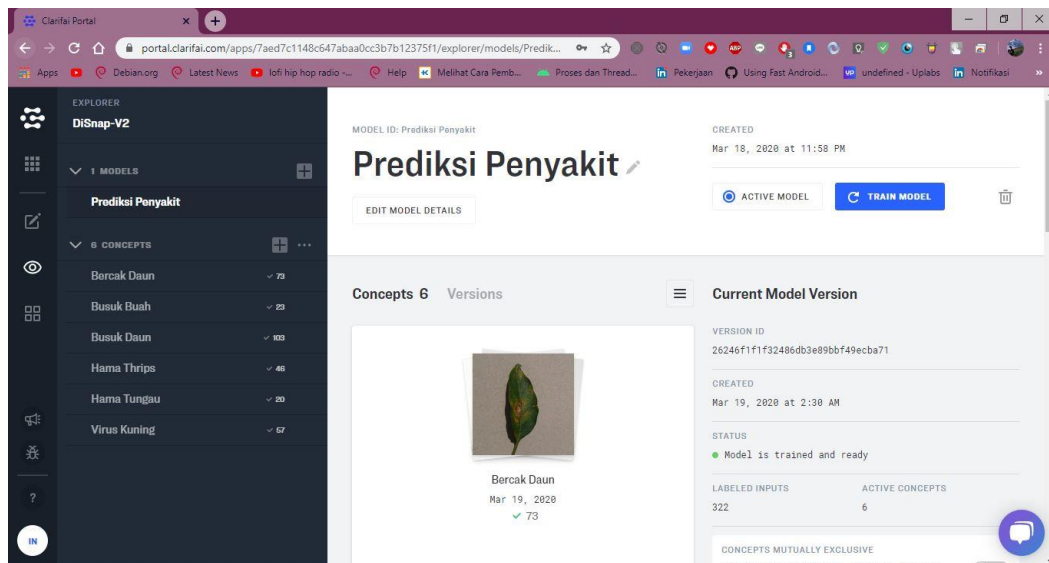
Virus Kuning (57 data). Jumlah data pada setiap jenis penyakit adalah hasil pengambilan data yang dilakukan peneliti di BPTP Jawa Timur dibawah pengawasan pakar penyakit tanaman cabai. Implementasi tahap *define* ditunjukkan pada Gambar 6.1



Gambar 6.1 Implementasi Tahap *Define*

6.6.2 Train

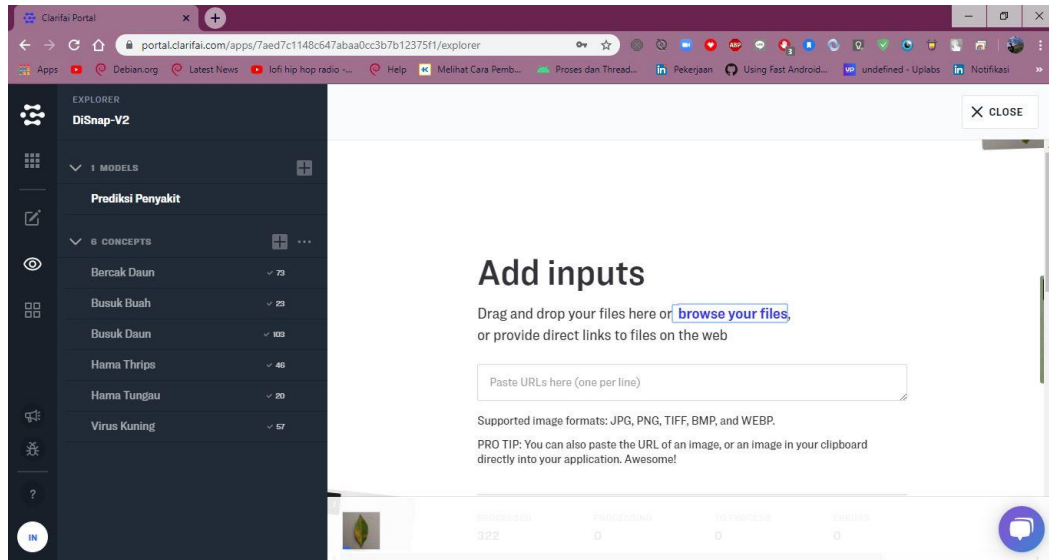
Pada tahap define terdapat tombol biru dengan nama *Train Model* yang berfungsi untuk melatih model dari gambar yang sudah disematkan pada setiap gambar berupa label dengan concept nama penyakit pada tanaman cabai. Setelah tombol *train model*. Maka model id: Prediksi Penyakit sudah dapat dipakai. Implementasi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



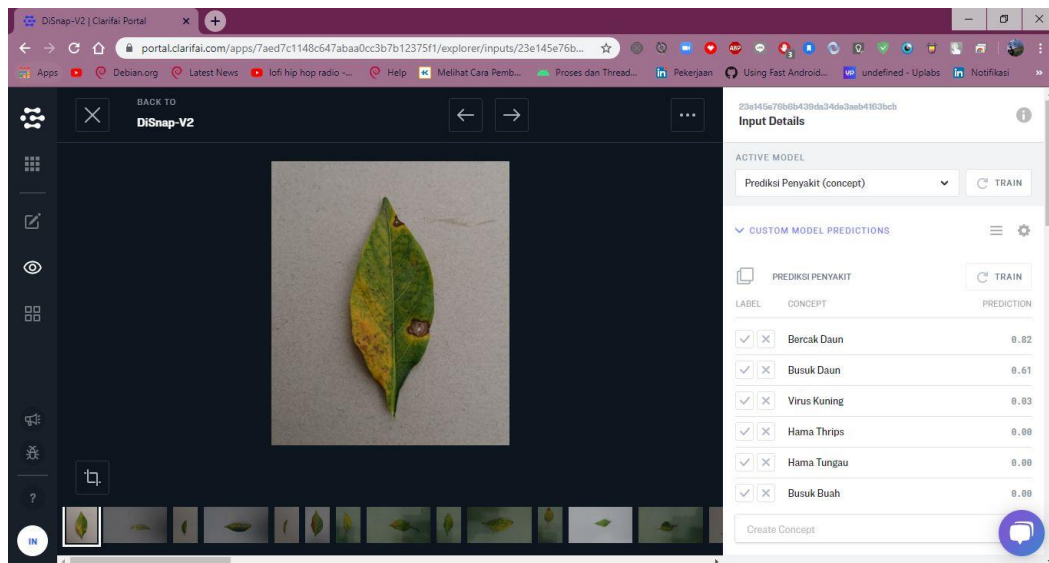
Gambar 6.2 Implementasi Tahap *Train*

6.6.3 Recognize

Pada tahap recognize dilakukan uji coba yaitu dengan mengunggah 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 6.3 dan Gambar 6.4



Gambar 6.3 Proses Mengunggah Gambar



Gambar 6.4 Implementasi Tahap *Recognize*

6.7 Implementasi Algoritme

Implementasi algoritme dilakukan dari hasil perancangan algoritme pada tahap sebelumnya. Algoritme yang akan diimplementasikan merupakan fungsi utama dari aplikasi DiSnap. Fitur tersebut merupakan 3 fitur utama yaitu 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`. Kode program ini digunakan untuk mengambil data informasi penyakit berupa data json dengan nama `disnap_data.json` pada folder `assets`. Source code dari algoritme 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<Disease> diseases = new ArrayList<>();</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	<code>try {</code>
12	<code>diseases.addAll(insertData(response, "hama"));</code>
13	<code>diseases.addAll(insertData(response, "penyakit"));</code>
14	<code>callback.onDiseaseLoaded(diseases);</code>
15	<code>Log.d(TAG, "onResponse1: " + diseases.size());</code>
16	<code>} catch (JSONException e) {</code>
17	<code>callback.onHideLoading();</code>
	<code>e.printStackTrace();</code>
18	<code>}</code>
19	<code>}</code>
20	
21	<code>@Override</code>
22	<code>public void onFailure(Exception error) {</code>
23	<code>callback.onHideLoading();</code>
24	<code>}</code>
25	<code>});</code>
26	
27	<code>}</code>

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<Disease></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 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. Respons dari clarifai API berupa json objek yang di dalamnya 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	<code>private void predictImage(final LoadAnalyzeCallback callback,</code>
2	<code>final String url) throws JSONException {</code>
3	<code> callback.onShowLoading();</code>
4	<code> AndroidNetworking.post(Constants.clarifaiAPI)</code>
5	<code> .setPriority(Priority.IMMEDIATE)</code>
6	<code> .addHeaders("Authorization", Constants.authClarifai)</code>
7	<code> .addHeaders("Content-Type", Constants.CONTENT_TYPE)</code>
8	<code> .addJSONObjectBody(this.getBody(url))</code>
9	<code> .build()</code>
10	<code> .getAsJSONObject(new JSONObjectRequestListener() {</code>
11	<code> @Override</code>
12	<code> public void onResponse(JSONObject response) {</code>
13	<code> callback.onHideLoading();</code>
14	<code> try {</code>
15	<code> Date date = Calendar.getInstance().getTime();</code>
16	<code> SimpleDateFormat df = new SimpleDateFormat("dd-MMM-yyyy");</code>
17	<code> String formattedDate = df.format(date);</code>
18	
19	<code> JSONArray jsonArray = response.getJSONArray("outputs");</code>
20	<code> JSONObject a = jsonArray.getJSONObject(0);</code>

```

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 predictImage
3	Pemanggilan method showLoading
4-10	Mekanisme penggunaan Android Fast Networking untuk melakukan pendeteksian pada gambar daun
12-14	Mekanisme apabila data berhasil dianalisis
15-17	Pembuatan nilai date

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<Disease> diseases = new</code>
7	<code>ArrayList<>(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 `getDiseaseAnalysisResultFormDB()` 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 adalah hasil implementasi dari *wireframe* dan semua bagian yang ada pada perancangan yang telah dibuat sebelumnya menggunakan Android Studio IDE. Hasil dari implementasi *user interface* ini adalah hasil akhir tampilan pada sistem yang menjembatani interaksi antara pengguna dan sistem. *User interface* dibuat sebaik mungkin dengan tampilan dan warna menarik untuk memudahkan pengguna dalam menggunakan aplikasi.

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* di dalamnya. Selain itu terdapat juga logo aplikasi dan *text* tentang deskripsi aplikasi.



Gambar 6.5 Implementasi *User Interface* Splash Screen

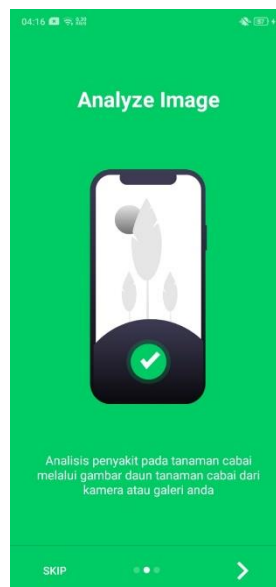
6.8.2 Implementasi *User Interface* Intro

Pada aplikasi DiSnap terdapat halaman intro. Halaman *intro* adalah halaman pengenalan fitur yang memiliki fungsi untuk menyampaikan 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

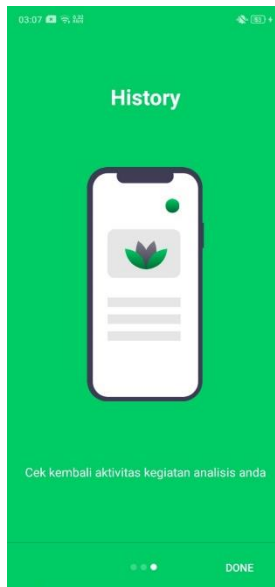
Gambar 6.8 menunjukkan halaman intro history yang berfungsi untuk memberikan informasi kepada pengguna bahwa pengguna dapat kembali melihat riwayat deteksi yang pernah dilakukan.



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

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 *recycler view* 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*

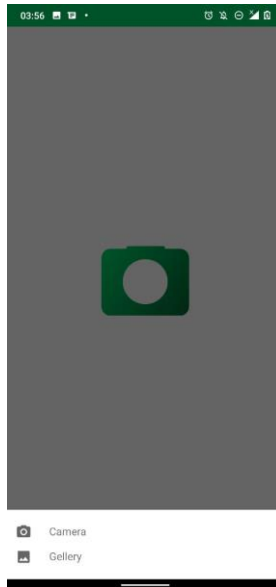


Gambar 6.10 Implementasi User Interface Detail Informasi Penyakit pada Cabai

6.8.4 Implementasi *User Interface Mendapatkan Gambar*

Pada Gambar 6.11, menunjukkan pengguna ketika menekan menu snap pada bottom navigation sehingga akan muncul bottom sheet 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 dilakukan 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 Galeri*



Gambar 6.13 Implementasi *User Interface Cropping Image*



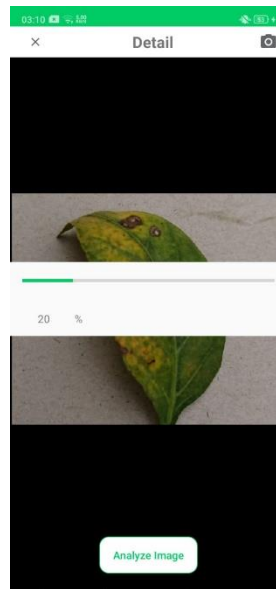
Gambar 6.14 Implementasi *User Interface Hasil dari Cropping Image*

6.8.5 Implementasi *User interface Mendeteksi Penyakit*

Implementasi *user interface* halaman *Analyze* ketika gambar yang sudah didapatkan dari hasil *cropping image* ditunjukkan pada Gambar 6.15. 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.

Implementasi *user interface* halaman *Result* ditunjukkan pada Gambar 6.16. Halaman 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

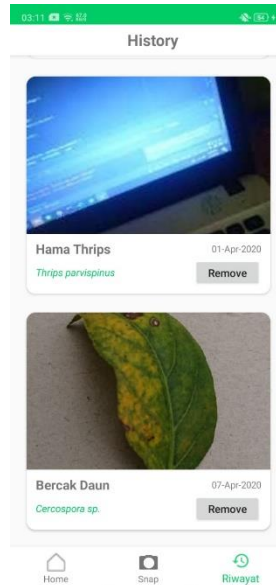


Gambar 6.16 Implementasi *User Interface* Hasil dari Deteksi Gambar

6.8.6 Implementasi *User Interface* Mengetahui Riwayat Hasil Deteksi

Pada Gambar 6.17, menunjukkan halaman daftar dari riwayat aktivitas dari mendeteksi penyakit yang pernah dilakukan oleh pengguna. Pada halaman tersebut terdapat *card view*, yang berisi photo yang pernah dianalisis oleh pengguna, nama penyakit, nama latin penyakit, tingkat akurasi dan tanggal ketika pengguna melakukan aktivitas menganalisis penyakit pada tanaman cabai. Ketika pengguna memilih item dari daftar 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, *accuracy bar*, informasi gejala, informasi pengendalian, dan informasi pestisida seperti yang ditunjukkan pada Gambar 6.18.



Gambar 6.17 Implementasi Menu Riwayat

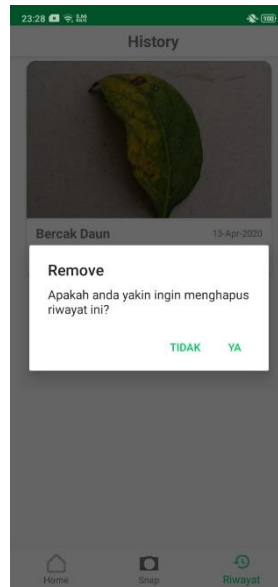


Gambar 6.18 Implementasi Detail Riwayat

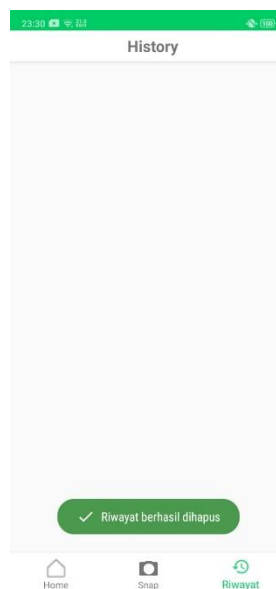
6.8.7 Implementasi *User Interface* Menghapus Riwayat Hasil Deteksi

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 “ya” atau “tidak”. Apabila

pengguna memilih pilihan “tidak” maka pengguna dialog hilang dan pengguna berada pada halaman riwayat. Akan tetapi apabila pengguna memilih untuk menekan tombol “ya”, maka sistem akan menghapus riwayat yang dipilih oleh pengguna. Seketika itu juga apabila proses menghapus sukses maka akan muncul pesan “Riwayat berhasil dihapus” seperti pada Gambar 6.20 dan *recycler view* akan otomatis memuat ulang daftar riwayat yang ada apabila masih terdapat data pada *database*.



Gambar 6.19 Implementasi *User Interface Dialog* Hapus Riwayat



Gambar 6.20 Implementasi *User Interface* Riwayat Berhasil Dihapus

6.8.8 Implementasi *User Interface About Apps*

Implementasi *user interfase about apps* ditunjukkan pada Gambar 6.21. Pada halaman tersebut terdapat nama aplikasi, versi aplikasi, penjelasan singkat aplikasi. Serta dua buah tombol yaitu tombol *GoToBPTP* dan tombol *Back*. Jika pengguna menekan tombol *GoToBPTP* maka pengguna akan dibawah oleh sistem ke halaman pencarian google untuk dapat mengunjungi website BPTP Jawa Timur. Sedangkan jika pengguna memilih tombol *Back*, maka pengguna akan kembali ke halaman *home*.



Gambar 6.21 Implementasi *user interface about apps*

BAB 7 PENGUJIAN

Pada bab pengujian peneliti menguji aplikasi DiSnap yang telah berhasil melalui tahap implementasi. Pengujian yang dilakukan bertujuan untuk mengetahui apakah aplikasi dapat berfungsi sesuai kebutuhan yang telah didefinisikan dan memastikan aplikasi dapat beroperasi sesuai rancangan yang telah dibuat. Pengujian yang dilakukan meliputi pengujian validasi (*blackbox testing*), pengujian akurasi, pengujian kompatibilitas (*compatibility testing*), dan pengujian *usability* (*usability testing*). Selanjutnya pada setiap pengujian akan dibuat hasil analisis.

7.1 Pengujian Validasi (*Blackbox Testing*)

Pengujian validasi bertujuan untuk mengetahui fungsionalitas yang terdapat pada sistem bekerja sesuai dengan kebutuhan fungsionalitas pengguna melalui skenario *use case* yang telah dibuat sebelumnya. Fokus dari pengujian ini adalah memastikan bahwa hasil keluaran pada sistem valid yaitu sesuai dengan harapan pada kasus uji yang telah didefinisikan sebelumnya. Dan validasi dianggap tidak valid apabila hasil keluaran tidak sesuai dengan kebutuhan dan harapan pada proses pendefinisian fungsionalitas.

7.1.1 Pengujian Validasi Mendapatkan Gambar

Hasil pengujian pada fungsionalitas mendapatkan gambar melalui kamera ditunjukkan pada Tabel 7.1. Hasil dari pengujian tersebut adalah valid berdasarkan prosedur yang dijalankan. Ke valid an pengujian dibuktikan dengan berhasilnya mendapatkan gambar melalui kamera sampai pada proses melakukan *image cropping* sehingga gambar siap untuk dideteksi.

Tabel 7.1 Kasus Uji Mendapatkan Gambar Melalui Kamera

Nama Kasus Kuji	Mendapatkan gambar melalui Kamera
Prosedur	<ol style="list-style-type: none">1. Pengguna menekan menu snap pada menu utama aplikasi2. Sistem menampilkan halaman Snap.3. Sistem menampilkan <i>bottom sheet dialog</i> berupa pilihan menu untuk mengambil gambar melalui kamera atau galeri4. Pengguna memilih metode pengambilan gambar melalui kamera5. Pengguna menekan <i>button next</i>6. Sistem menampilkan halaman <i>cropping image</i>7. Pengguna melakukan <i>cropping image</i>8. Pengguna menekan tombol centang

<i>Expected Result</i>	Sistem berhasil mengambil gambar melalui kamera menampilkan gambar hasil <i>cropping image</i> pada halaman <i>Analyze</i>
Result	Sistem berhasil mengambil gambar melalui kamera menampilkan gambar hasil <i>cropping image</i> pada halaman <i>Analyze</i>
Validasi	Valid

Pada Tabel 7.2, menjelaskan hasil dari pengujian mendapatkan gambar melalui galeri. Hasil dari pengujian pada kasus uji tersebut adalah valid berdasarkan prosedur yang dijalankan. Ke valid an pengujian dibuktikan dengan berhasilnya mendapatkan gambar melalui galeri sampai pada proses melakukan *image cropping* sehingga gambar siap untuk dideteksi.

Tabel 7.2 Kasus Uji Mendapatkan Gambar Melalui Galeri

Nama Kasus Kuji	Mendapatkan gambar melalui Kamera
Prosedur	<ol style="list-style-type: none"> 1. Pengguna menekan menu snap pada menu utama aplikasi 2. Sistem menampilkan halaman Snap. 3. Sistem menampilkan <i>bottom sheet dialog</i> berupa pilihan menu untuk mengambil gambar melalui kamera atau galeri 4. Pengguna memilih metode pengambilan gambar melalui kamera 5. Pengguna menekan <i>button next</i> 6. Sistem menampilkan halaman <i>cropping image</i> 7. Pengguna melakukan <i>cropping image</i> 8. Pengguna menekan tombol centang
<i>Expected Result</i>	Sistem berhasil mengambil gambar melalui galeri menampilkan gambar hasil <i>cropping image</i> pada halaman <i>Analyze</i>
Result	Sistem berhasil mengambil gambar melalui galeri menampilkan gambar hasil <i>cropping image</i> pada halaman <i>Analyze</i>
Validasi	Valid

7.1.2 Pengujian Validasi Mendeteksi Penyakit

Pengujian validasi pada fungsionalitas mendeteksi penyakit dilakukan dengan dua pengujian yaitu ketika *device (smartphone)* pengguna tidak tersambung dengan internet dan ketika *device* pengguna tersambung dengan internet. Pada tabel Tabel 7.3, menunjukkan hasil pengujian mendeteksi penyakit dengan *device* terhubung internet. Hasil dari pengujian ini adalah valid. Hal ini dapat ditunjukkan ketika pengguna menekan tombol *analyze image* dan *device* tidak terhubung dengan internet maka sistem menampilkan “Periksa koneksi internet anda”.

Tabel 7.3 Kasus Uji Mendeteksi Penyakit dengan *Device* Tidak Terhubung Internet

Nama Kasus Uji	Mendeteksi penyakit dengan <i>device</i> tidak terhubung dengan internet
Prosedur	1. Pengguna berada pada halaman Analyze 2. Sistem menampilkan gambar yang sudah didapatkan pada proses sebelumnya 3. Pengguna menekan tombol <i>analyze image</i> pada layar
<i>Expected Result</i>	Sistem menampilkan pesan “Periksa koneksi internet anda”
Result	Sistem menampilkan pesan “Periksa koneksi internet anda”
Validasi	Valid

Pada Tabel 7.4, menunjukkan hasil dari pengujian mendeteksi penyakit dengan *device* terhubung internet. Hasil dari pengujian ini adalah valid. Hal ini dapat sesuai dengan *Expected Result* bahwa ketika tombol *analyze image* ditekan, maka sistem akan menampilkan loading, kemudian sistem menampilkan hasil deteksi penyakit tanaman cabai pada halaman Result.

Tabel 7.4 Kasus Uji Mendeteksi Penyakit dengan *Device* Terhubung Internet

Nama Kasus Uji	Mendeteksi penyakit dengan <i>device</i> terhubung internet
Prosedur	1. Pengguna berada pada halaman Analyze 2. Sistem menampilkan gambar yang sudah didapatkan pada proses sebelumnya 3. Pengguna menekan tombol <i>analyze image</i> pada layar
<i>Expected Result</i>	1. Sistem menampilkan <i>loading</i> 2. Sistem menampilkan hasil deteksi penyakit tanaman cabai pada halaman <i>Result</i>
Result	1. Sistem menampilkan loading

	2. Sistem menampilkan hasil deteksi penyakit tanaman cabai pada halaman <i>Result</i>
Validasi	Valid

7.1.3 Pengujian Validasi Mengetahui Informasi Penyakit dan Pengendalian Penyakit

Pada Tabel 7.5 menunjukkan hasil pengujian dari kasus uji mengetahui informasi penyakit dan pengendalian penyakit. Hasil dari kasus uji ini adalah valid, karena berdasarkan prosedur yang dilakukan hasilnya sama dengan hasil yang diharapkan yaitu sistem menampilkan detail informasi penyakit pada tanaman cabai.

Tabel 7.5 Kasus Uji Mengetahui Informasi Penyakit dan Pengendalian Penyakit

Nama Kasus Uji	Mendapatkan gambar melalui Kamera
Prosedur	<ol style="list-style-type: none"> 1. Pengguna memilih menu <i>home</i> 2. Sistem menampilkan menu home 3. Sistem menampilkan berbagai informasi tentang penyakit pada tanaman cabai 4. Pengguna memilih salah satu penyakit
<i>Expected Result</i>	Sistem menampilkan detail informasi penyakit pada tanaman cabai
Result	Sistem menampilkan detail informasi detail penyakit pada tanaman cabai
Validasi	Valid

7.1.4 Pengujian Validasi Mengetahui Riwayat Gambar yang Telah Dideteksi

Pada Tabel 7.6, menunjukkan hasil dari pengujian dengan kasus uji mengetahui riwayat gambar yang telah dideteksi. Mengikuti prosedur kasus uji, hasil dari pengujian validasi mengetahui riwayat yang telah dideteksi adalah valid, karena hasil pengujian sama dengan hasil yang diharapkan yaitu sistem menampilkan data riwayat gambar yang telah dideteksi.

Tabel 7.6 Kasus Uji Mengetahui Riwayat Gambar yang Telah Dideteksi

Nama Kasus Uji	Mendapatkan gambar melalui Kamera
Prosedur	<ol style="list-style-type: none"> 1. Pengguna memilih menu riwayat 2. Sistem menampilkan menu riwayat

<i>Expected Result</i>	Sistem menampilkan data riwayat gambar yang telah dideteksi
Result	Sistem menampilkan data riwayat gambar yang telah dideteksi
Validasi	Valid

Pada Tabel 7.7, menunjukkan hasil dari pengujian kasus uji tidak terdapat data pada menu riwayat. Dengan mengikuti prosedur pada Tabel 7.7, hasil dari pengujian validasi tidak terdapat riwayat hasil deteksi adalah valid, karena hasilnya sama dengan hasil yang diharapkan yaitu ketika tidak ada data riwayat deteksi pada *database*, maka sistem menampilkan pesan “Tidak ada riwayat deteksi”.

Tabel 7.7 Kasus Uji Tidak Terdapat Riwayat Hasil Deteksi

Nama Kasus Uji	Tidak terdapat riwayat hasil deteksi
Prosedur	1. Pengguna memilih menu riwayat 2. Sistem menampilkan menu riwayat
<i>Expected Result</i>	Sistem menampilkan pesan “Tidak ada riwayat deteksi”
Result	Sistem menampilkan pesan “Tidak ada riwayat deteksi”
Validasi	Valid

7.1.5 Pengujian Validasi Menghapus Riwayat Hasil Deteksi

Pada Tabel 7.8, menunjukkan hasil deteksi pengujian pada kasus uji menghapus riwayat hasil deteksi. Hasil dari pengujian validasi menghapus riwayat hasil deteksi adalah valid, karena hasilnya sama dengan hasil yang diharapkan yaitu sistem menampilkan dialog, dilanjutkan dengan pesan “Riwayat berhasil dihapus” apabila tidak

Tabel 7.8 Kasus Uji Menghapus Riwayat Hasil Deteksi

Nama Kasus Uji	Menghapus riwayat hasil deteksi
Prosedur	1. Pengguna memilih menu riwayat 2. Sistem menampilkan menu riwayat 3. Sistem menampilkan daftar gambar riwayat yang pernah dideteksi 4. Pengguna menekan tombol remove pada salah satu riwayat
<i>Expected Result</i>	Sistem menampilkan dialog dengan pesan “Apakah anda yakin ingin menghapus riwayat ini?”. Pengguna menekan

	tombol “Ya”, sistem menampilkan pesan “Riwayat berhasil dihapus”.
Result	Sistem menampilkan dialog dengan pesan “Apakah anda yakin ingin menghapus riwayat ini?”. Pengguna menekan tombol “Ya”, sistem menampilkan pesan “Riwayat berhasil dihapus”.
Validasi	Valid

7.2 Pengujian Akurasi

Tujuan pengujian akurasi yaitu untuk mendapatkan nilai ketepatan aplikasi DiSnap untuk mendeteksi jenis penyakit pada tanaman cabai melalui daun ataupun buah yang terkena penyakit. Pengujian akurasi yang dilakukan terhadap aplikasi DiSnap adalah dengan mencoba mendeteksi gambar/*photo* daun pada tanaman cabai secara langsung dengan mengambil gambar melalui galeri ataupun kamera yang ada pada *smartphone*. Jumlah data yang digunakan untuk data latih dan data uji yaitu menggunakan perbandingan 1: 4 atau 80 % untuk data latih dan 20 % untuk data uji sesuai dengan jumlah masing-masing penyakit saat pengambilan data di lapangan. Adapun perhitungan yang dilakukan oleh peneliti untuk mendapatkan rata-rata nilai akurasi pengujian ini ditunjukkan pada persamaan 6.1.

$$Akurasi = \frac{Total\ Jumlah\ Pendeteksian\ Valid}{Total\ Jumlah\ Data\ Uji} \times 100\%$$

(6.1)

Berdasarkan hasil pengambilan data, peneliti mendapatkan 6 jenis penyakit di lapangan yaitu bercak daun, busuk buah, busuk daun, hama *thrips*, hama tungau, dan virus kuning. Pada Tabel 7.9, menunjukkan jumlah total data latih dan data uji.

Tabel 7.9 Rincian data latih dan data uji pada aplikasi DiSnap

No	Nama Label	Total Data	Data Latih	Data Uji
1.	Bercak Daun	120	96	24
2.	Busuk Buah	50	40	10
3.	Busuk Daun	120	96	24
4.	Hama Thrips	90	72	18
5.	Hama Tungau	50	40	10
6.	Virus Kuning	80	64	16
	Total	510	408	102

7.2.1 Hasil Pengujian Akurasi

Peneliti menguji akurasi aplikasi dengan pengujian secara langsung terhadap data uji pada masing-masing penyakit. Pengujian akan dinyatakan valid apabila hasil deteksi sama dengan target *output*, dan dinyatakan tidak valid apabila hasil deteksi tidak sama dengan target *output*. Berikut hasil pengujian akurasi ditunjukkan pada Tabel 7.10 sampai dengan Tabel 7.15.

Tabel 7.10 Pengujian Terhadap Bercak Daun

No	Citra Input	Target <i>Output</i>	Hasil Deteksi	Status Pendeteksian
1	Data Uji - 1	Bercak Daun	Bercak Daun	Valid
2	Data Uji - 2	Bercak Daun	Bercak Daun	Valid
3	Data Uji - 3	Bercak Daun	Hama Thrips	Tidak Valid
4	Data Uji - 4	Bercak Daun	Bercak Daun	Valid
5	Data Uji - 5	Bercak Daun	Bercak Daun	Valid
6	Data Uji - 6	Bercak Daun	Bercak Daun	Valid
7	Data Uji - 7	Bercak Daun	Bercak Daun	Valid
8	Data Uji - 8	Bercak Daun	Bercak Daun	Valid
9	Data Uji - 9	Bercak Daun	Bercak Daun	Valid
10	Data Uji - 10	Bercak Daun	Hama Thrips	Tidak Valid
11	Data Uji - 11	Bercak Daun	Bercak Daun	Valid
12	Data Uji - 12	Bercak Daun	Bercak Daun	Valid
13	Data Uji - 13	Bercak Daun	Bercak Daun	Valid
14	Data Uji - 14	Bercak Daun	Bercak Daun	Valid
15	Data Uji - 15	Bercak Daun	Hama Thrips	Tidak Valid
16	Data Uji - 16	Bercak Daun	Hama Thrips	Tidak Valid
17	Data Uji - 17	Bercak Daun	Bercak Daun	Valid
18	Data Uji - 18	Bercak Daun	Bercak Daun	Valid
19	Data Uji - 19	Bercak Daun	Bercak Daun	Valid
20	Data Uji - 20	Bercak Daun	Bercak Daun	Valid
21	Data Uji - 21	Bercak Daun	Bercak Daun	Valid
22	Data Uji - 22	Bercak Daun	Bercak Daun	Valid
23	Data Uji - 23	Bercak Daun	Bercak Daun	Valid
24	Data Uji - 24	Bercak Daun	Bercak Daun	Valid
Total Valid				20

Tabel 7.11 Pengujian Terhadap Busuk Buah Antraknosa

No	Citra Input	Target Output	Hasil Deteksi	Status Pendeteksian
1	Data Uji - 1	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid

2	Data Uji - 2	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
3	Data Uji - 3	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
4	Data Uji - 4	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
5	Data Uji - 5	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
6	Data Uji - 6	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
7	Data Uji - 7	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
8	Data Uji - 8	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
9	Data Uji - 9	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
10	Data Uji - 10	Busuk Buah Antraknosa	Busuk Buah Antraknosa	Valid
Total Valid				10

Tabel 7.12 Pengujian Terhadap Busuk Daun

No	Citra Input	Target Output	Hasil Deteksi	Status Pendeteksian
1	Data Uji - 1	Busuk Daun	Busuk Daun	Valid
2	Data Uji - 2	Busuk Daun	Busuk Daun	Valid
3	Data Uji - 3	Busuk Daun	Busuk Daun	Valid
4	Data Uji - 4	Busuk Daun	Busuk Daun	Valid
5	Data Uji - 5	Busuk Daun	Busuk Daun	Valid
6	Data Uji - 6	Busuk Daun	Busuk Daun	Valid
7	Data Uji - 7	Busuk Daun	Bercak Daun	Tidak Valid
8	Data Uji - 8	Busuk Daun	Bercak Daun	Tidak Valid
9	Data Uji - 9	Busuk Daun	Busuk Daun	Valid
10	Data Uji - 10	Busuk Daun	Busuk Daun	Valid
11	Data Uji - 11	Busuk Daun	Busuk Daun	Valid
12	Data Uji - 12	Busuk Daun	Busuk Daun	Valid
13	Data Uji - 13	Busuk Daun	Bercak Daun	Tidak Valid
14	Data Uji - 14	Busuk Daun	Busuk Daun	Valid
15	Data Uji - 15	Busuk Daun	Busuk Daun	Valid
16	Data Uji - 16	Busuk Daun	Busuk Daun	Valid
17	Data Uji - 17	Busuk Daun	Busuk Daun	Valid
18	Data Uji - 18	Busuk Daun	Hama Thrips	Tidak Valid
19	Data Uji - 19	Busuk Daun	Busuk Daun	Valid

20	Data Uji - 20	Busuk Daun	Busuk Daun	Valid
21	Data Uji - 21	Busuk Daun	Busuk Daun	Valid
22	Data Uji - 22	Busuk Daun	Busuk Daun	Valid
23	Data Uji - 23	Busuk Daun	Bercak Daun	Tidak Valid
24	Data Uji - 24	Busuk Daun	Bercak Daun	Tidak Valid
Total Valid				18

Tabel 7.13 Pengujian Terhadap Hama Thrips

No	Citra Input	Target Output	Hasil Deteksi	Status Pendeteksian
1	Data Uji - 1	Hama Thrips	Hama Thrips	Valid
2	Data Uji - 2	Hama Thrips	Busuk Daun	Valid
3	Data Uji - 3	Hama Thrips	Hama Thrips	Valid
4	Data Uji - 4	Hama Thrips	Bercak Daun	Valid
5	Data Uji - 5	Hama Thrips	Hama Thrips	Valid
6	Data Uji - 6	Hama Thrips	Hama Thrips	Valid
7	Data Uji - 7	Hama Thrips	Hama Thrips	Valid
8	Data Uji - 8	Hama Thrips	Hama Thrips	Valid
9	Data Uji - 9	Hama Thrips	Hama Thrips	Valid
10	Data Uji - 10	Hama Thrips	Virus Kuning	Tidak Valid
11	Data Uji - 11	Hama Thrips	Virus Kuning	Tidak Valid
12	Data Uji - 12	Hama Thrips	Hama Thrips	Valid
13	Data Uji - 13	Hama Thrips	Hama Thrips	Valid
14	Data Uji - 14	Hama Thrips	Hama Thrips	Valid
15	Data Uji - 15	Hama Thrips	Bercak Daun	Tidak Valid
16	Data Uji - 16	Hama Thrips	Hama Thrips	Valid
17	Data Uji - 17	Hama Thrips	Bercak Daun	Tidak Valid
18	Data Uji - 18	Hama Thrips	Hama Tungau	Tidak Valid
Total Valid				13

Tabel 7.14 Pengujian Terhadap Hama Tungau

No	Citra Input	Target Output	Hasil Deteksi	Status Pendeteksian
1	Data Uji - 1	Hama Tungau	Hama Tungau	Valid
2	Data Uji - 2	Hama Tungau	Hama Tungau	Valid
3	Data Uji - 3	Hama Tungau	Hama Tungau	Valid
4	Data Uji - 4	Hama Tungau	Hama Tungau	Valid
5	Data Uji - 5	Hama Tungau	Hama Tungau	Valid
6	Data Uji - 6	Hama Tungau	Hama Tungau	Valid
7	Data Uji - 7	Hama Tungau	Hama Tungau	Valid
8	Data Uji - 8	Hama Tungau	Hama Tungau	Valid

9	Data Uji - 9	Hama Tungau	Hama Tungau	Valid
10	Data Uji - 10	Hama Tungau	Hama Tungau	Valid
Total Valid				10

Tabel 7.15 Pengujian Terhadap Virus Kuning

No	Citra Input	Target Output	Hasil Deteksi	Status Pendeteksian
1	Data Uji - 1	Virus Kuning	Virus Kuning	Valid
2	Data Uji - 2	Virus Kuning	Busuk Daun	Tidak Valid
3	Data Uji - 3	Virus Kuning	Busuk Daun	Tidak Valid
4	Data Uji - 4	Virus Kuning	Bercak Daun	Tidak Valid
5	Data Uji - 5	Virus Kuning	Virus Kuning	Valid
6	Data Uji - 6	Virus Kuning	Virus Kuning	Valid
7	Data Uji - 7	Virus Kuning	Virus Kuning	Valid
8	Data Uji - 8	Virus Kuning	Virus Kuning	Valid
9	Data Uji - 9	Virus Kuning	Virus Kuning	Valid
10	Data Uji - 10	Virus Kuning	Virus Kuning	Valid
11	Data Uji - 11	Virus Kuning	Bercak Daun	Tidak Valid
12	Data Uji - 12	Virus Kuning	Virus Kuning	Valid
13	Data Uji - 13	Virus Kuning	Hama Thrips	Tidak Valid
14	Data Uji - 14	Virus Kuning	Busuk Daun	Tidak Valid
Total Valid				8

Tabel 7.16 Hasil Pengujian Data Uji

No	Nama Konsep	Jumlah Valid	Jumlah Tidak Valid
1	Bercak Daun	20	4
2	Busuk Buah Antraknosa	10	0
3	Busuk Daun	18	6
4	Hama Thrips	13	5
5	Hama Tungau	10	0
6	Virus Kuning	8	6
Total		79	21

Pada Tabel 7.16 didapatkan nilai total jumlah valid dari semua proses pengujian. Selanjutnya menghitung nilai akurasi total pada sistem dengan memasukkan nilai pada persamaan 6.1. Dengan memasukkan nilai pada rumus maka didapatkan hasil nilai akurasi sistem yaitu sebesar 70.59 %.

7.3 Pengujian *Usability*

Pengujian usability dilakukan dengan tujuan untuk mengetahui tingkat kegunaan terhadap pengguna dari aplikasi yang telah dikembangkan. Untuk

melakukan pengujian pengguna diberikan *task scenario* kepada pengguna, selanjutnya pengguna akan menjalankan aplikasi sesuai dengan *task scenario* yang telah dibuat. *Task Scenario* harus dijalankan sesuai urutan pada nomor. Task Scenario yang diberikan kepada pengguna dapat dilihat pada Tabel 7.17.

Tabel 7.17 Task Scenario Pengujian Usability

No	Nama Task	Langka- Langkah
1	Mengambil gambar	<ol style="list-style-type: none"> 1. Pengguna diminta untuk menekan tombol snap. 2. Selanjutnya pengguna diminta untuk mengambil gambar daun cabai melalui kamera atau galeri 3. Pengguna melakukan proses <i>image cropping</i> 4. Pengguna menekan tombol centang
2.	Mendeteksi Penyakit	<ol style="list-style-type: none"> 1. Pengguna diminta untuk menekan tombol <i>analyze image</i> pada halaman Analyze setelah melakukan proses <i>image cropping</i>
3.	Mengetahui riwayat gambar hasil deteksi	<ol style="list-style-type: none"> 1. Pengguna diminta memilih menu riwayat pada menu utama. 2. Pengguna memilih salah satu riwayat untuk melihat detail riwayat.
4.	Menghapus riwayat hasil deteksi	<ol style="list-style-type: none"> 1. Pengguna diminta untuk memilih menu riwayat pada menu utama 2. Pengguna diminta untuk menekan tombol <i>remove</i> pada salah satu riwayat 3. Pengguna diminta untuk menekan tombol ya
5.	Mengetahui informasi penyakit dan pengendalian penyakit	<ol style="list-style-type: none"> 1. Pengguna diminta untuk memilih menu <i>home</i> pada halaman utama 2. Pengguna diminta untuk memilih salah satu dari daftar penyakit untuk melihat detail penyakit
6.	Menuju halaman Website BPTP Jawa Timur	<ol style="list-style-type: none"> 1. Pengguna diminta untuk menekan <i>icon</i> informasi pada menu home 2. Pengguna diminta untuk menekan tombol <i>Go To BPTP</i>

Setelah menjalankan *task scenario*, selanjutnya responden memberikan penilaian tentang *usability* (kegunaan) pada aplikasi DiSnap dengan mengisi kuesioner yang diberikan. Pada penelitian ini pengujian *usability* yang dilakukan menggunakan kuesioner *SUPR-Qm* yang memiliki sejumlah 16 pernyataan seperti yang dapat dilihat pada Tabel 7.19. Untuk masing-masing pernyataan memiliki skor skala *likert* yaitu 1 sampai dengan 5 seperti yang ditunjukkan pada Tabel 7.18.

Tabel 7.18 Skor Skala Likert untuk Setiap Pertanyaan

Skor	Keterangan
1	Sangat Tidak Setuju
2	Tidak Setuju
3	Netral
4	Setuju
5	Sangat Setuju

Tabel 7.19 Kuesioner SUPR-Qm

No	Pernyataan	Sangat Tidak Setuju	Tidak Setuju	Netral	Setuju	Sangat Setuju
1.	Aplikasi ini penting untuk saya					
2.	Aplikasi ini merupakan aplikasi pendeteksi penyakit pada tanaman cabai terbaik yang pernah saya gunakan					
3.	Saya tidak tahu apakah ada aplikasi pendeteksi penyakit pada tanaman cabai yang lebih baik dari aplikasi ini					
4.	Saya tidak akan menghapus aplikasi ini dari <i>smartphone</i> saya					
5.	Saya akan menyarankan aplikasi ini kepada teman saya					
6.	Saya suka melakukan eksplorasi terhadap fitur yang ada pada aplikasi ini					
7.	Aplikasi ini memiliki fitur dan fungsi yang saya inginkan pada aplikasi pendeteksi penyakit pada tanaman cabai					
8.	Saya akan sering membuka aplikasi ini untuk membantu					

	saya setiap kali ingin mendeteksi penyakit pada tanaman cabai					
9.	Aplikasi ini menyenangkan					
10.	Saya berpikir bahwa aplikasi ini terintegrasi dengan baik dengan fitur-fitur lain dari ponsel saya (misalnya membuka kamera dan galeri)					
11.	Saya akan menggunakan aplikasi ini ketika ingin mendeteksi penyakit pada tanaman cabai					
12.	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan					
13.	Menurut saya aplikasi ini menarik					
14.	Aplikasi ini sesuai dengan kebutuhan saya					
15.	Melakukan navigasi dalam aplikasi ini mudah bagi saya					
16.	Aplikasi ini mudah digunakan					

Pada Tabel 7.19, merupakan kuesioner SUPR-Qm yang diisi oleh responden, jawaban dari responden diolah untuk mendapatkan nilai kuantitatif terhadap kemudahan dalam penggunaan aplikasi DiSnap. Persamaan untuk mendapatkan nilai kuesioner SUPR-Qm sama dengan persamaan untuk mendapatkan nilai SUPR-Q seperti pada persamaan 6.2. Dari nilai yang didapatkan, hasilnya akan di konversikan kedalam pengelompokan usability.

$$Nilai SUPR - Q = \frac{Jumlah\ Nilai\ Diperoleh}{Jumlah\ Nilai\ Maksimal} \times 100\%$$

(6.2)

7.3.1 Hasil Pengujian *Usability*

Pengujian *usability* pada aplikasi DiSnap dilakukan kepada 5 orang responden dari Balai Pengkajian Teknologi Pertanian (BPTP) Jawa Timur yang merupakan calon pengguna dari aplikasi ini. Berikut 5 orang yang bertindak sebagai responden ditunjukkan pada Tabel 7.20.

Tabel 7.20 Responden Pengujian *Usability*

No	Nama Responden	Pekerjaan
1.	Dwi Setyorini	Peneliti di bidang Tanaman Cabai
2.	Gunawan	Penyuluh
3.	Bambang Pikukuh	ASN
4.	N. Arifin	Petani
5.	Sholeh	Petani

Pada Tabel 7.20, merupakan daftar responden yang membantu peneliti dalam melakukan pengujian *usability* pada aplikasi DiSnap yang terdiri dari 1 orang peneliti, 1 orang ASN, 1 orang penyuluh dan 2 orang petani. Tujuan dilakukannya pengujian *usability* terhadap bermacam-macam pekerjaan adalah agar mendapatkan nilai yang lebih bervariasi dari banyak sudut pandang pengguna yang masih memiliki hubungan.

Selanjutnya, peneliti meminta responden menjalankan dan mengoperasikan aplikasi DiSnap berdasarkan *task scenario* seperti yang dijabarkan pada Tabel 7.17 selanjutnya moderator melakukan pengamatan terhadap responden untuk mengetahui dapat atau tidaknya responden mengikuti dan menyelesaikan *task scenario* yang ada. Hasil dari pengamatan moderator terhadap responden ditunjukkan pada Tabel 7.21.

Tabel 7.21 Task Completion Rate

	User 1	User 2	User 3	User 4	User 5	Completion Rate
Task 1	√	√	√	√	√	100%
Task 2	√	√	√	√	√	100%
Task 3	√	√	√	√	√	100%
Task 4	√	√	√	√	√	100%
Task 5	√	√	√	√	√	100%
Task 6	√	√	√	√	√	100%

Pada Tabel 7.21 dapat ditarik kesimpulan bahwa semua responden dapat menyelesaikan semua *task* dari *task* 1 sampai dengan *task* 7. Hal ini ditunjukkan

dengan kolom *Completion Rate* yang memberikan hasil 100% dari semua *task scenario* yang diberikan.

Selanjutnya setelah responden berhasil melakukan semua *task scenario*, maka responden diminta oleh peneliti untuk dapat mengisi kuesioner SUPR-Qm seperti yang ditunjukkan pada Tabel 7.19. Adapun hasil dari pengisian kuesioner SUPR-Qm ditunjukkan pada Tabel 7.22.

Tabel 7.22 Hasil Pengujian Kuesioner SUPR-Qm

No	Pernyataan	Skor					Total Skor
		1	2	3	4	5	
1	Aplikasi ini penting untuk saya	0	0	0	5	0	20
2	Aplikasi ini merupakan aplikasi pendeteksi penyakit pada tanaman cabai terbaik yang pernah saya gunakan	0	0	1	2	2	21
3	Saya tidak tahu apakah ada aplikasi pendeteksi penyakit pada tanaman cabai yang lebih baik dari aplikasi ini	0	0	1	4	0	19
4	Saya tidak akan menghapus aplikasi ini dari <i>smartphone</i> saya	0	1	2	2	0	16
5	Saya akan menyarankan aplikasi ini kepada teman saya	0	0	0	4	1	21
6	Saya suka melakukan eksplorasi terhadap fitur yang ada pada aplikasi ini	0	1	0	2	2	20
7	Aplikasi ini memiliki fitur dan fungsi yang saya inginkan pada aplikasi pendeteksi penyakit pada tanaman cabai	0	0	0	3	2	22
8	Saya akan sering membuka aplikasi ini untuk membantu saya setiap kali ingin mendeteksi penyakit pada tanaman cabai	0	0	0	4	1	21
9	Aplikasi ini menyenangkan	0	0	1	3	1	20
10	Saya berpikir bahwa aplikasi ini terintegrasi dengan baik dengan fitur-fitur lain dari ponsel saya (misalnya membuka kamera dan galeri)	0	0	0	4	1	21

11	Saya akan menggunakan aplikasi ini ketika ingin mendeteksi penyakit pada tanaman cabai	0	0	0	2	3	23
12	Desain dari aplikasi ini memudahkan saya dalam menemukan informasi yang saya inginkan	0	0	0	3	2	22
13	Menurut saya aplikasi ini menarik	0	0	0	4	1	21
14	Aplikasi ini sesuai dengan kebutuhan saya	0	0	1	2	2	21
15	Melakukan navigasi dalam aplikasi ini mudah bagi saya	0	0	2	1	2	20
16	Aplikasi ini mudah digunakan	0	0	0	2	3	23
Total Akhir							331
Total Nilai Maksimum							400
Nilai SUPR-Qm							82.75 %

Pada Tabel 7.22 merupakan hasil dari pengisian kuesioner SUPR-Qm dan hasil perhitungan dari nilai SUPR-Qm. Berdasarkan tabel tersebut nilai tertinggi ditunjukkan pada pernyataan nomor 11 dan 16 yaitu dengan skor 23. Selain itu pada tabel tersebut dapat dilihat nilai terendah yaitu pernyataan nomor 4 dengan skor 16. Dari tabel tersebut dapat dilihat bahwa total akhir dari berjumlah 331 dan total nilai maksimumnya berjumlah 400. Untuk menghitung nilai SUPR-Qm maka Total akhir dibagi dengan total nilai maksimum dikalikan dengan 100% seperti pada persamaan 6.2, maka didapatkan nilai 82.75%.

7.4 Pengujian *Compatibility*

Pengujian ini dilakukan terhadap aplikasi yang dibangun dengan tujuan mengetahui aplikasi yang sudah dikembangkan dapat berjalan dan beroperasi pada perangkat dengan level SDK Android *Operating System* (OS) yang sudah ditentukan. Pada penelitian aplikasi yang dikembangkan sudah di atur untuk dapat dioperasikan dengan sistem operasi Android minimal level 23 dengan target level 29. Artinya aplikasi hanya akan berjalan pada perangkat dengan minimal level 23.

Dalam melakukan pengujian ini akan dilakukan penginstalan/pengoperasian aplikasi pada perangkat dengan sistem operasi Android di bawah level 23 dan di atas level 23 menggunakan bantuan layanan Test Lab dari Firebase. Hasil dari pengujian *compatibility* menggunakan layanan Test Lab dari Firebase ditunjukkan pada Tabel 7.23.

Tabel 7.23 Pengujian *Compatibility*

No	Level API	Lulus	Tidak Lulus
1.	22		√
2.	21		√
3.	23	√	
4.	24	√	
5.	25	√	
6.	26	√	
7.	27	√	
8.	28	√	
9.	29	√	

7.5 Analisis Hasil Pengujian

Analisis dari hasil pengujian pada aplikasi DiSnap terbagi menjadi 4 bagian yaitu pengujian validasi, pengujian akurasi pengujian usability, dan pengujian *compatibility*.

7.5.1 Analisis Hasil Pengujian Validasi

Hasil uji validasi pada aplikasi DiSnap dapat dilihat pada pada Tabel 7.1 sampai dengan Tabel 7.8. Kasus uji merupakan kasus berdasarkan kebutuhan yang telah telah didefinisikan serta pengembangan nya atau kemungkinan yang ada pada setiap proses. Dari 8 kasus uji yang diujikan, seluruh kasus uji yang pada aplikasi memiliki hasil valid. Dengan begitu aplikasi DiSnap dapat dikatakan berhasil dengan tingkat keberhasilan 100%.

7.5.2 Analisis Hasil Pengujian Akurasi

Hasil uji akurasi mendeteksi gambar daun yang terkena penyakit pada tanaman cabai dengan menggunakan clarifai ditunjukkan pada Tabel 7.16. Dari hasil pengujian yang didapatkan, persentase akurasi dari pengujian data uji sebesar 70.59%. Dilihat dari hasil persentase nya maka keakuratan dari aplikasi ini dapat dikatakan belum terlalu akurat, hal ini disebabkan karena keterbatasan jumlah data latih yang digunakan dalam proses *define* pada website clarifai serta kualitas gambar yang dideteksi.

7.5.3 Analisis Hasil Pengujian *Usability*

Pada pengujian *usability* menggunakan SUPR-Qm, aplikasi DiSnap mendapatkan nilai *usability* SUPR-Qm yaitu 82.75%. Selanjutnya nilai tersebut di konversikan kedalam kategori nilai *usability*, didapatkan kategori nilai B atau

Excellent. Bisa dikatakan bahwa aplikasi ini dapat digunakan dengan baik oleh pengguna di Balai Pengkajian Teknologi Pertanian (BPTP) Jawa Timur.

7.5.4 Analisis Hasil Pengujian *Compatibility*

Pada pengujian *compatibility* yang dilakukan terhadap aplikasi DiSnap dengan menggunakan Firebase Test Lab hasilnya seperti yang ditunjukkan pada Tabel 7.23. Dari pengujian tersebut dapat dibuktikan bahwa pengembangan aplikasi DiSnap sudah sesuai dengan yang dikembangkan oleh peneliti. Aplikasi DiSnap hanya dapat dijalankan pada Android *Operating System* (OS) *SDK level* 23 ke atas, dan tidak dapat berjalan/ beroperasi pada Android *Operating System* (OS) *SDK level* 23 ke bawah.

BAB 8 PENUTUP

Pada bab penutup berisi kesimpulan dan saran terhadap penelitian pengembangan aplikasi *mobile* pendeteksi penyakit pada tanaman cabai menggunakan teknologi clarifai.

8.1 Kesimpulan

Berdasarkan semua proses yang telah diselesaikan pada pengembangan aplikasi *mobile* pendeteksi penyakit pada tanaman cabai menggunakan teknologi clarifai, maka diperoleh 3 kesimpulan, yaitu:

1. Hasil analisis kebutuhan pada pengembangan aplikasi pendeteksi penyakit tanaman cabai didapatkan dengan cara penggalian kebutuhan menggunakan teknik wawancara terhadap seorang pakar penyakit dari Balai Pengkajian Teknologi Pertanian (BPTP) Jawa Timur. Selain itu dilakukan juga observasi ke lapangan yaitu ke lahan pertanian cabai di Malang yang berada di bawah naungan BPTP Jawa Timur. Selanjutnya dilakukan proses identifikasi aktor, kebutuhan sistem dan pemodelan kebutuhan pada sistem. Aplikasi *mobile* pendeteksi penyakit pada tanaman cabai memiliki 5 kebutuhan fungsional dan 2 kebutuhan non fungsional.
2. Perancangan pada aplikasi *mobile* pendeteksi penyakit pada tanaman cabai menghasilkan perancangan arsitektur sistem, *sequence diagram*, *class diagram*, ERD (*Entity Relationship Diagram*), *wireframe* (perancangan antarmuka) dan perancangan algoritme. Implementasi aplikasi pendeteksi penyakit pada tanaman dilakukan pada sistem operasi android dengan memanfaatkan teknologi clarifai untuk melakukan pengenalan penyakit pada tanaman cabai melalui gambar daun cabai yang diambil oleh pengguna menggunakan kamera atau galeri dan di unggah ke imgur secara otomatis. Aplikasi pendeteksi penyakit pada tanaman cabai diimplementasikan dengan bahasa pemrograman Java serta menggunakan *design architecture* MVP (*Model View Presenter*). Integrasi teknologi clarifai pada pengembangan aplikasi *mobile* pendeteksi penyakit pada tanaman cabai dilakukan dengan 3 tahapan yaitu *define* (mengelompokkan), *train* (melatih) dan *recognize* (mengenali). Untuk tahap *define* dilakukan pengelompokan dan pelabelan gambar menjadi 6 label yaitu Bercak Daun, Busuk Buah Antraknosa, Busuk Daun, Hama Thrips, Hama Tungau, dan Virus Kuning. Pada tahap *train* dilakukan pelatihan oleh clarifai untuk dapat mengenali penyakit pada tanaman cabai. Pada tahap *recognize* dilakukan pengenalan pada gambar yang dikirimkan kepada website clarifai yang sebelumnya telah dilakukan *hosting image* pada website imgur menggunakan *web service* API yang disediakan sehingga didapatkan nama penyakit dengan tingkat akurasi dari ketepatan pendeteksian penyakit dalam JSON. Data yang didapatkan dikelola pada perangkat pengguna kemudian ditampilkan kepada pengguna berupa nama penyakit, nama latin penyakit, detail penyakit

serta tingkat akurasi dari ketepatan teknologi clarifai dalam mendeteksi penyakit.

3. Pengujian validasi yang telah dilakukan pada sistem yaitu dengan mengujikan fungsional sistem. Dari pengujian ini menghasilkan bahwa aplikasi pendeteksi penyakit pada tanaman cabai memiliki tingkat 100% pada fungsionalitas dikarenakan semua hasil pengujian validasi bernilai valid. Pada pengujian akurasi mendeteksi penyakit cabai menggunakan teknologi clarifai didapatkan akurasi sebesar 70.59% berdasarkan pengujian pada data uji gambar sebanyak 100 gambar. Akurasi dari aplikasi ini dapat dikatakan belum akurat karena belum semua data uji yang diuji memiliki ketepatan dalam mendeteksi ketepatan nama penyakit yang ada pada gambar karena kurangnya data latih dan kualitas gambar. Dapat dipahami bahwa aplikasi yang dikembangkan dapat memiliki tingkat akurasi yang lebih baik apabila memiliki banyak data latih dan kualitas gambar yang baik. Pada pengujian *usability* didapatkan skor dengan nilai 82.75%. Nilai tersebut termasuk kedalam kategori nilai B atau *Excellent*. Sehingga dapat disimpulkan aplikasi *mobile* pendeteksi penyakit pada tanaman cabai dinilai dapat digunakan dengan baik oleh pengguna. Sedangkan pada pengujian *compatibility*, berdasarkan pengujian menggunakan Firebase Test Lab didapatkan hasil bahwa sistem dapat beroperasi sesuai dengan minimal level SDK Android yang telah ditentukan oleh peneliti yaitu level 23 (Android Marshmallow).

8.2 Saran

Mengacu pada proses dan hasil penelitian, masih terdapat beberapa kekurangan yang ditemukan dalam penelitian ini. Oleh sebab itu, ada beberapa saran yang bisa dilakukan untuk penelitian selanjutnya, yaitu:

1. Perlu adanya penambahan label untuk daun sehat dan penyakit lain pada tanaman cabai.
2. Perlu ada penambahan jumlah data latih dengan kualitas gambar yang baik pada masing-masing label.
3. Perlu ada penambahan fitur pada aplikasi untuk membantu pengguna dalam mengambil gambar sesuai frame yang dianjurkan.
4. Dilakukan studi lebih lanjut dalam cara mendeteksi penyakit pada tanaman cabai seperti dari batang ataupun akarnya.