PENGEMBANGAN APLIKASI MOBILE PENDETEKSI PENYAKIT PADA TANAMAN CABAI DENGAN MENGGUNAKAN XIMILAR CUSTOM IMAGE RECOGNITION

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

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Zain Fikri Hanastyono
NIM: 165150201111083

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PERSETUJUAN

PENGEMBANGAN APLIKASI MOBILE PENDETEKSI PENYAKIT PADA TANAMAN CABAI DENGAN MENGGUNAKAN XIMILAR CUSTOM IMAGE RECOGNITION

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Zain Fikri Hanastyono
NIM: 165150201111083

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Issa Arwani, S.Kom., M.Sc.	Ir. Handoko, M.Sc.
NIP: 19830922 201212 1 003	NIP: 19650308 199903 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 1 Maret 2020

Zain Fikri Hanastyono

NIM: 165150201111083

PRAKATA

Segala puji dan syukur bagi Allah SWT atas limpahan rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi yang berjudul "Pengembangan Aplikasi Mobile Pendeteksi Penyakit pada Tanaman Cabai dengan Menggunakan Ximilar Custom Image Recognition". Dalam kesempatan ini, penulis ingin memberikan ucapan terima kasih kepada seluruh pihak yang telah membantu penulis dalam proses penyusunan skripsi ini, diantaranya:

1. Kedua orang tua penulis, yaitu Bapak Hari Prihatin dan Ibu Sri Nastiti beserta seluruh keluarga besar yang terus memberikan dukungan dan motivasi serta doa yang tidak kunjung putus.
2. Bapak Issa Arwani, S.Kom., M.Sc. sebagai dosen pembimbing I dan Bapak Ir. Handoko, M.Sc. sebagai dosen pembimbing II atas segala bimbingan, masukan, dan saran yang bermanfaat bagi penulis selama proses penelitian skripsi ini.
3. Seluruh Bapak dan Ibu Dosen Fakultas Ilmu Komputer, Universtas Brawijaya atas ketesediannya membimbing dan memberikan ilmu yang bermanfaat kepada penulis.
4. Para Peneliti di Balai Pengkajian Teknologi Pertanian yang telah membantu dalam menyelesaikan penelitian skripsi ini.
5. Seluruh teman dan sahabat penulis yang secara tidak langsung memberikan ilmu serta motivasi dalam penyusunan skripsi ini.
6. Semua pihak yang tidak semuanya bisa dituliskan disini yang terlibat secara langsung maupun tidak langsung dalam penyusunan skripsi ini.

Penulis sadar masih banyak kekurangan dan kesalahan yang terdapat dalam penyusunan skripsi ini baik dari sisi teknis penelitian maupun penyajian materi. Maka dari itu, penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga karya tulis ini bermanfaat dan dapat memberikan manfaat bagi pihak yang membutuhkan.

Malang, 1 Maret 2020

Penulis

zainfikrih@gmail.com

ABSTRAK

Zain Fikri Hanastyono, Pengembangan Aplikasi Mobile Pendeteksi Penyakit pada Tanaman Cabai dengan Menggunakan Ximilar Custom Image Recognition

Pembimbing: Issa Arwani, S.Kom., M.Sc. dan Ir. Handoko, M.Sc.

Tanaman cabai merupakan tanaman sayuran buah yang sangat penting dan memiliki nilai ekonomis yang tinggi. Berdasarkan data dari Direktorat Jenderal Hortikultura Kementrian Pertanian produksi tanaman cabai sendiri mengalami peningkatan dan termasuk komoditas yang berkontribusi tinggi di Indonesia. Meskipun produksi tanaman cabai meningkat namun pada kenyataan di lapangan petani cabai mengalamai beberapa kendala, salah satu kendala tersebut adalah kurangnya pengetahuan mengenai penyakit pada tanaman cabai yang meliputi gejala, cara pengendalian, dan pestisida yang digunakan. Kemajuan teknologi memberikan banyak peluang untuk menyelesaikan permasalahan manusia. Penulis memanfaatkan peluang untuk mengembangkan aplikasi yang dapat membantu menyelesaikan permasalahan petani tanaman cabai mengenai pendeteksian penyakit pada tanaman cabai. Aplikasi yang dikembangkan memanfaatkan teknologi perangkat *mobile* yang ada saat ini. Layanan untuk melakukan pengidentifikasian penyakit tanaman cabai menggunakan Ximilar Custom Image Recognition yang merupakan *web service* pengenalan gambar. Fitur yang ada pada aplikasi disesuaikan dengan kebutuhan pengguna dengan menerapkan metode *prototyping*. Aplikasi dikembangkan pada *platform* Android dengan menggunakan bahasa pemrograman Java dan menggunakan *design pattern* Model-View-ViewModel. Hasil pengujian akurasi aplikasi ini adalah 75,32% yang dapat dibilang belum akurat.

Kata kunci: cabai, penyakit cabai, pendeteksi, Ximilar, Android, *Architecture Components*

ABSTRACT

Zain Fikri Hanastyono, Development of Mobile Application for Disease Detection in Chili Plants Using Ximilar Custom Image Recognition

Supervisors: Issa Arwani, S.Kom., M.Sc. dan Ir. Handoko, M.Sc.

The chili plant is a plant vegetable fruit that is really important and has high economic value. Based on the data from Directorate General Horticulture Agricultural Ministry, chili plant production is increasing and belongs to high commodities contributing in Indonesia. However, the chili plant faces a number of problems, one of the problems is the lack of knowledge about the disease of chili plants, involving the symptoms, controlling technique, and the use of pesticides. As well as advancing technology over time is giving chances to develop an application that can solve human problems. The writer tried to take the chances to develop the application for solving the chili farmer problems to detect chili plant disease. To develop the app, the writer utilizes mobile technology. The service for identifying chili plant disease is using Ximilar Custom Image Recognition as well as a web service for image recognition. The features in this application are adjusted to the user requirements implementing the prototyping method. The application is developed for the Android platform using Java programming language and Model-View-ViewModel design pattern. The result of accuracy testing states that 75,32% haven't accurate.

Keywords: chili, chili disease, detection, Ximilar, Android, Architecture Components

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Cabai (Capsicum annuum L) adalah salah satu tanaman yang penting dalam kehidupan masyarakat Indonesia terutama sebagai bahan tambahan pada makanan. Cabai juga menjadi tanaman yang diperjual belikan dan memiliki nilai jual yang tinggi. Cabai merupakan tanaman hortikultura dan salah satu jenis sayuran buah yang cukup penting di Indonesia karena memiliki protein untuk dikembangkan dan juga memiliki nilai ekonomis yang tinggi (Ralahalu, 2013). Dengan adanya nilai ekonomis tersebut maka pemerintah melakukan beberapa usaha agar nilai dari cabai tersebut tetap stabil dan produksi meningkat. Beberapa upaya pemerintah yaitu upaya peningkatan luas tanam cabai pada musim hujan, pengaturan luas tanam dan produksi cabai pada musim kemarau, stabilisasi harga cabai dan pengembangan kelembagaan kemitraan yang andal dan berkelanjutan (Kementerian Pertanian, 2016).

Berdasarkan data dari Direktorat Jenderal Hortikultura Kementrian Pertanian, tanaman cabai masuk dalam komoditas yang memiliki kontribusi yang besar terhadap produksi sayuran nasional. Secara nasional pada tahun 2014 produksi sayuran mengalami peningkatan sekitar 3,12 persen dibandingkan tahun 2013. Produksi tanaman cabai sendiri mengalami peningkatan sebesar 6.09 persen dan berada pada urutan ke empat komoditas yang memiliki kontribusi yang tinggi seperti yang dijelaskan pada Tabel 1.1. Jumlah produksi dan kontribusi tanaman cabai ini perlu diperhatikan oleh Kementrian Pertanian agar mambantu pemerintah dalam mengambil kebijakan yang dapat membantu meningkatkan produksi tanaman cabai.

Tabel 1.1 Perbandingan Produksi Sayuran Tahun 2014 terhadap 2013 serta Kontribusi Produksi Tahun 2014 terhadap Nasional

No	Komoditas	Perbandingan Produksi Tahun 2014 terhadap 2013 (%)	Kontribusi Produksi Tahun 2014 terhadap Nasional (%)
1	Kol/Kubis	-3,03	12,05
2	Kentang	19,88	11,31
3	Bawang Merah	22,08	10,35
4	Cabai Besar	6,09	9,02
5	Tomat	-7,74	7,69
6	Sayuran lainnya	-0,45	49,59
	Total Sayuran	3.12	-

Sumber: (DitjenHorti (Direktorat Jendral Hortikultura), 2015)

Walaupun produksi tanaman cabai meningkat namun pada kenyataannya para petani di lapangan mendapatkan beberapa kendala, salah satunya adalah pengetahuan mengenai penyakit dan hama yang menyerang pada tanaman cabai.

Kebanyakan tidak mengetahui dengan pasti jenis penyakit dan hama yang sedang mengenai cabai, kebanyakan petani akan menduga-duga penyakit apa yang sedang menyerang tanaman cabai. Ketidaktahuan tersebut akan mengakibatkan petani merugi, karena tanaman mati, harga jual menurun, dan hasil panen yang sedikit. Sehingga dibutuhkan sebuah pengetahuan yang dapat memebantu petani untuk meningkatkan kualitas dalam menanam cabai (Wijaya & Hidayat, 2018).

Perkembangan teknologi memiliki banyak kemajuan, perkembangan ini dapat dimanfaatkan untuk menerapkan berbagai inovasi dengan lebih mudah. Kemajuan teknologi ini juga terjadi pada bidang *machine learning,* dengan memanfaatkan teknologi m*achine learning* ini akan mempermudah manusia dalam memroses data, salah satunya data citra gambar. Ximilar Custom Image Recognition merupakan sebuah *web service* yang menyediakan layanan *mechine learning*. Ximilar Custom Image Recognition mampu menandai banyak label pada satu objek gambar. Dengan Ximilar Custom Image Recognition memungkinkan untuk membuat pengenalan gambar dan kalsifikasi objek yang dapat diimplementasikan pada aplikasi yang terkoneksi internet dengan menggunakan API (Ximilar, 2019a).

Kemudahan mengakses internet saat ini terbantu dengan adanya *smartphone*. Mengakses internet dimanapun dan kapanpun jadi lebih mudah. Dengan kemudahan tersebut mengakibatkan peningkatan jumlah pengguna *smartphone* di Indonesia. Tercatat pengguna *smartphone* di Indonesia tiap tahunnya mengalami peningkatan, seperti dijelaskan di Gambar 1.1 pengguna *smartphone* di Indonesia pada tahun 2018 mencapai 103 juta.

Gambar 1.1 Pengguna *Smartphone* Tahun 2013 – 2018 di Indonesia

Sumber: (Techinasia, 2018)

Berdasarkan pemaparan yang ada maka penulis membuat sebuah aplikasi berbasis *mobile* dengan sistem operasi Android untuk mendeteksi penyakit pada tanaman cabai dengan memanfaatkan Ximilar Custom Image Recognition. Aplikasi tersebut adalah aplikasi untuk membantu petani mendeteksi penyakit tanaman cabai dan memberikan pengetahuan mengenai pengendaliannya. Dengan latar balakang tersebut maka peneliti memberi judul pada penelitian ini yaitu "Pengembangan Aplikasi Mobile Pendeteksi Penyakit Pada Tanaman Cabai Dengan Menggunakan Ximilar Custom Image Recognition".

Demi tercapainya manfaat dan terwujudnya aplikasi pendeteksi penyakit pada tanaman cabai diatas diperlukan penelitian. Dalam penelitian ini menggunakan metode *prototyping* karena mampu memberikan pemodelan awal berupa pemodelan antarmuka dari aplikasi yang berguna untuk menemukan lebih banyak kebutuhan yang akan diimplementasikan dalam aplikasi pendeteksi penyakit pada tanaman cabai (Pressman, 2010). Menemukan kebutuhan yang lebih banyak berpengaruh dalam pengembangan aplikasi ini karena kebutuhan awal pengguna belum pasti. Mengingat pengembangan aplikasi ini dilakukan pada aplikasi perangkat bergerak berbasis Android maka perlu dilakukan beberapa pengujian yang akan menunjang keberhasilan aplikasi ini. Pengujian yang dilakukan meliputi pengujian validasi, pengujian akurasi, pengujian *usability*, dan pengujian *compatibility*. Pengujian *compatibility* diperlukan untuk mengetahui apakah aplikasi berjalan pada perangkat yang ditargetkan mengingat perangkat bergerak berbasis Android memiliki banyak jenis versi sistem operasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang dikaji, penulis dapat merumuskan masalah yang disusun dibawah ini:

1. Bagaimana hasil analisis kebutuhan pada aplikasi pendeteksi penyakit tanaman cabai dengan menggunakan metode *prototyping*?
2. Bagaimana mengintegrasikan layanan Ximilar Custom Image Recognition sebagai pengklasifikasi gambar pada aplikasi pendeteksi penyakit tanaman cabai?
3. Bagaimana hasil pengujian pada aplikasi pendeteksi penyakit tanaman cabai yang meliputi pengujian akurasi, pengujian *usability*, dan pengujian *compatibility*?

1.3 Tujuan

Tujuan dalam penelitian ini adalah sebagai berikut:

1. Mengetahui hasil analisis kebutuhan pada aplikasi pendeteksi penyakit tanaman cabai dengan menggunakan metode *prototyping*.
2. Mengetahui cara mengintegrasikan layanan Ximilar Cutom Image Rcognition sebagai pengklasifikasi gambar pada aplikasi pendeteksi penyakit tanaman cabai.
3. Mengetahui hasil pengujian pada aplikasi pendeteksi penyakit tanaman cabai yang meliputi pengujian akurasi, pengujian *usability*, dan pengujian *compatibility*.

1.4 Manfaat

Manfaat dari penelitian ini ialah:

1. Menyediakan aplikasi yang dapat membantu untuk mengenali jenis penyakit pada tanaman cabai.

2. Dapat menjadi referensi pada pengembangan selanjutnya untuk aplikasi yang serupa.

1.5 Batasan Masalah

Penelitian ini memiliki beberapa batasan dalam pengembangannya, antara lain:

1. Data yang digunakan dalam penelitian hanya terbatas pada data hasil pengumpulan pada Balai Pengkajian Teknologi Pertanian (BPTP) Jawa Timur dengan hasil terdapat 6 jenis penyakit dan hama pada tanaman cabai.
2. Penelitian ini difokuskan pada pengembangan aplikasi perangkat bergerak penyakit pada tanaman cabai dengan menggunakan Ximilar Custom Image Recognition. Tidak membahas mengenai Teknik yang dilakukan pada layanan Ximilar Custom Image Recognition.
3. Dalam pengambilan gambar penyakit pada tanaman cabai menggunakan kamera Sony Xperia XZ Premium.
4. Sistem yang akan dibuat harus terkoneksi dengan internet.

1.6 Sistematika Pembahasan

Penyusunan dokumen skripsi ini dibagai dalam beberapa bab, yang terdiri dari:

BAB 1 PENDAHULUAN

Bagian ini menjelaskan tentang latar belakang, rumusan masalah, tujuan dari penelitian, manfaat dari penelitian, batasan masalah, dan sistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Bagian ini memuat kajian-kajian kepustakaan terkait yang digunakan sebagai referensi dalam melakukan penelitian.

BAB 3 METODOLOGI

Bagian ini memuat alur kerja peneliti dalam proses penyelesaian permasalahan penelitian.

BAB 4 ANALISIS KEBUTUHAN

Bagian ini memuat hal-hal yang berkaitan dengan proses penggalian kebutuhan yang digunakan dalam pengembangan sistem. Dalam bagian ini juga mencakup proses *prototyping* yang bertujuan untuk mendapatkan kebutuhan.

BAB 5 PERANCANGAN

Bagian ini memuat hal-hal yang berkaitan dengan perancangan sistem berdasarkan data yang diperoleh dari tahap sebelumnya.

BAB 6 IMPLEMENTASI

Bagian ini menejalskan proses yang berkaitan dengan implementasi pengembangan sistem yang sesuai dengan pemodelan yang telah dibuat dalam tahap perancangan.

BAB 7 PENGUJIAN

Bagian ini memuat hal-hal yang berkaitan dengan pengujian sistem yang dihasilkan dari proses implementasi yang dilakukan oleh peneliti dengan tujuan untuk melakukan analisis hasil yang sebelumnya telah diperoleh.

BAB 8 PENUTUP

Bagian ini menejelaskan mengenai kesimpulan yang didapat dari penelitian yang dilakukan dan juga terdapat saran untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Kajian pustaka merupakan pembahasan untuk menganalisis penelitian terdahulu yang memiliki kemiripan topik bahasan dengan penelitian yang akan dilakukan penulis. Pada bagian ini juga menjelaskan teori yang berkaitan dengan penelitian ini antara lain Android, Architectur Components, Ximilar Custom Image Recognition, Firebase Cloud Storage, JSON dan juga metode *prototyping*.

2.1 Kajian Pustaka

Pada Tabel 2.1 dijelaskan penelitian mengenai deteksi penyakit pada tanaman cabai yang pernah dilakukan sebelumnya.

Tabel 2.1 Penelitian Berkaitan dengan Kasus

No	Judul	Sumber	Hasil
1.	Sistem Pakar Diagnosa Hama Dan Penyakit Cabai Berbasis Teorema Bayes	(Muslim, 2015)	Penelitian ini mengenai diagnose hama dan penyakit pada tanaman cabai dengan teorema bayes. Berdasarkan penelitian tersebut dihasilkan keakuratan sebesar 100%
2.	Diagnosis Penyakit Cabai Dengan Menggunakan Metode Forward Chaining – Dempster-Shafer	(Wijaya & Hidayat, 2018)	Penelitian ini mengenai diagnose penyakit cabai dengan metode Forwaed Chaining. Berdasarkan penelitian tersebut dihasilkan keakuratan 90% dari 20 kasus uji.
3.	Penerapan Teknik Computer Vision Pada Bidang Fitopatologi Untuk Diteksi Penyakit dan Hama Tanaman Cabai	(Wibowo, 2017)	Penelitian ini mengenai penggunaan *computer* vision untuk deteksi penyakit tanaman cabai. Berdasarkan penelitian tersebut dijelaskan bahwa ketepatan menggunakan metode *computer vision* tergantung dari jenis kamera yang digunakan .
4.	Pengembangan Aplikasi Perangkat Bergerak Identifikasi Penyakit Daun Jeruk Berbasis Android dengan Memanfaatkan Vize AI	(Jupiyandi, Kharisma, & Triwiratno, 2019)	Penelitian ini mengenai penggunaan *web* service untuk menentukan penyakit pada daun tanaman jeruk. Berdasarkan penelitian tersebut dihasilkan 100% tingkat akurasi.

2.2 Penyakit pada Tanaman Cabai

Penyakit pada tanaman cabai beraneka ragam, beberapa diantaranya adalah Layu Fusarium, Layu Bakteri Ralstonia, Busuk Buah Antraknosa, Virus Kuning, dan Bercak Daun. Berikut penjelasan penyakit tersebut:

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

Penyakit Layu Fusarium dapat diamati pada bagian daun tanaman cabai. Hal ini ditandai dengan gejala serangan berupa layu pada daun bagian bawah. Bagian yang terinfeksi akan tertutup seperti kapas yang disebut hifa putih. Apabila penyakit sudah mencapai batang maka buah kecil dari cabai akan gugur. Gambar 2.1 merupakan contoh tanaman yang terserang penyakit Layu Fusarium.

Gambar 2.1 Penyakit Layu Fusarium pada Tanaman Cabai

Sumber: (Meilin, 2014)

2.2.2 Layu Bakteri Ralstonia (*Ralstonia solanacearum*)

Penyakit Layu Bakteri Ralstonia dapat menyerang pada tanaman cabai muda ataupun tua. Pada tanaman cabai muda, gejala serangan dapat diamatai pada daun bagian atas tanaman yang mulai tampak layu. Sedangkan pada tanaman cabai tua, gejala serangan dapat diamati pada daun bagian bawah tanaman yang mulai tampak layu. Setelah beberapa hari tanaman cabai tiba-tiba akan layu keseluruhan pada daunnya dan kadang-kadang juga mengalami kekuningan. Penyakit ini juga bias diketahui dengan cara memotong melintang bagian batang dan dicelupkan pada air jernih, maka akan batang tersebut akan mengelurkan cairan keruh menyerupai kepulan asap. Dan pada bagian buah akan terlihat kekuningan dan busuk. Pada Gambar 2.2 merupakan contoh penyakit Layu Bakteri Ralstonia yang menyerang pada tanaman cabai.

Gambar 2.2 Penyakit Layu Bakteri Ralstonia pada Tanaman Cabai

Sumber: (Meilin, 2014)

2.2.3 Busuk Buah Antraknosa (*Collectrotichum gloeospoiroides*)

Penyakit Bsusuk Buah Antraknosa menyerang bagian buah pada tanaman cabai. Gejala awal tanaman yang diterjangkit penyakit ini ialah ditandai dengan munculnya bercak agak mengkilap, berair, berwarna hitam, orange, dan coklat. Luka yang ditimbulkan akan melebar dan membentuk sebuah lingkaran dengan diameter sekitar 30 mm. Dalam waktu yang tidak lama maka buah akan membusuk ditandai dengan berubahnya warna menjadi coklat kehitaman. Serangan penyakit ini membuat seluruh buah keriput dan mengering. Gambar 2.3 merupakan contoh buah yang terserang penyakit ini.

Gambar 2.3 Penyakit Busuk Buah Antraknosa pada Tanaman Cabai

Sumber: (Meilin, 2014)

2.2.4 Virus Kuning (Gemini Virus)

Penyakit virus kuning ini menyerang bagian daun pada tanaman cabai. Hal ini ditandai dengan adanya gejala berupa helai daun mulai berwarna kuning, tulang daun menebal, dan daun menggulung ke atas. Apabila serangan sudah pada tahap lanjut maka daun akan mengecil, berwarna kuning terang, tanaman kerdil, dan tidak berbuah. Gambar 2.4 merupakan contoh tanaman cabai yang terserang penyakit ini.

Gambar 2.4 Penyakit Virus Kuning pada Tanaman Cabai

Sumber: (Meilin, 2014)

2.2.5 Bercak Daun (*Cercospora sp.*)

Penyakit bercak daun menyebabkan kerusakan pada bagian daun, akar, dan juga batang tanaman cabai. Gejala serangan penyakit ini adalah terlihatnya bercak bulat berwarna coklat pada daun dan kering, ukuran besarnya bercak adalah sekitar 1 inci. Dalam pusat bercaknya berwarna putih dengan tepi memiliki warna agak lebih tua. Gambar 2.5 merupakan contoh gambar tanaman cabai yang terserang penyakit ini.

Gambar 2.5 Penyakit Bercak Daun pada Tanaman Cabai

Sumber: (Meilin, 2014)

2.3 Hama pada Tanaman Cabai

Hama yang menyerang tanaman cabai antara lain adalah Thrips, Lalat Buah, Kutu Kebul, Kutu Daun Persik, Kutu Daun, dan Tungau. Berikut penjelasan beberapa hama tersebut:

2.3.1 Thrips (*Thrips parvispinus* Karny) (Thripidae: Thysanoptera)

Thrips menyerang tanaman cabai dengan cara mengisap cairan permukaan bawah daun. Daun yang terserang akan berubah warna menjadi cokelat, mengeriting, dan pada akhirnya akan mati. Pada serangan lanjut daun akan menggulung ke dalam dan muncul benjolan. Gambar 2.6 merupakan contoh gambar tanaman cabai yang terserang hama Thrips.

Gambar 2.6 Hama Thrips pada Tanaman Cabai

(Meilin, 2014)

2.3.2 Lalat Buah (*Bactrocera sp.*)

Hama Lalat Buah ini menyerang bagian buah pada tanaman cabai. Serangan Lalat Buah ini dapat diketahui gejala awalnya yaitu adanya titik hitam pada pangkal buah. Titik hitam ini muncul karena aktifitas lalat buah dewasa yang memasukkan telurnya pada buah. Telur tersebut akan berkembang dan menetas di dalam buah. Buah yang terserang akan membusuk dan jatuh ke tanah. Gambar 2.7 merupakan contoh gambar buah tanaman cabai yang terserang hama Lalat Buah.

Gambar 2.7 Hama Lalat Buah pada Tanaman Cabai

(Meilin, 2014)

2.3.3 Kutu Kebul (*Bemisia tabaci*)

Hama Kutu Kebul menyerang bagian daun tanaman cabai dengan gejala serangan yaitu munculnya bercak nekrotik (bercak hitam kecokelatan dan di

tengahnya terdapat bercak putih). Hal tersbut dikarenakan rusaknya sel akibat serangan nimfa dan serangga dewasa. Gambar 2.8 merupakan contoh gambar dari hama Kutu Kebuk yang menyerang daun tanaman cabai.

Gambar 2.8 Hama Kutu Kebul pada Tanaman Cabai

(Meilin, 2014)

2.3.4 Kutu Daun Persik (*Myzus persicae*)

Kutu Daun Persik menyerang pada bagian bawah daun dengan cara mengisap cairan daun yang muda dan atau bagian tanaman yang masih muda. Pada bagian yang terserang akan terlihat kutu yang bergerombol. Akan terlihat pula bercak-bercak dan daun akan mengeriting. Bila gejala sudah lanjut maka daun akan layu dan mati. Populasi hama ini meningkat pada musim kemarau (Meilin, 2014).

2.3.5 Kutu Daun (*Aphididae*)

Hama Kutu Daun biasanya menyerang pada bagian pucuk tanaman dan daun muda. Gejala serangan ini adalah daun mengkerut, mengeriting, dan pertumbuhan tanaman terhambat. Hama ini dapat mengeluarkan cairan manis (embun madu) yang dapat menarik semut dan cendawan untuk datang. Cendawan ini dapat menurunkan kualitas buah dari tanaman cabai. Gambar 2.9 merupakan contoh gambar hama Kutu Daun yang menyerang pada tanaman cabai.

Gambar 2.9 Hama Kutu Daun pada Tanaman Cabai

(Meilin, 2014)

2.3.6 Tungau (*Polyphagotarsonemus latus* dan *Tetranychus sp.*)

Hama Tungau ini menyerang pada daun muda dengan mengisap cairan tanaman dan mengakibatkan perubahan bentuk daun. Daun menjadi menebal, warna berubah menjadi tembaga atau kecokelatan, kaku, melengkung ke bawah, dan keriting. Gambar 2.10 merupakan contoh gambar daun tanaman cabai yang terserang Tungau.

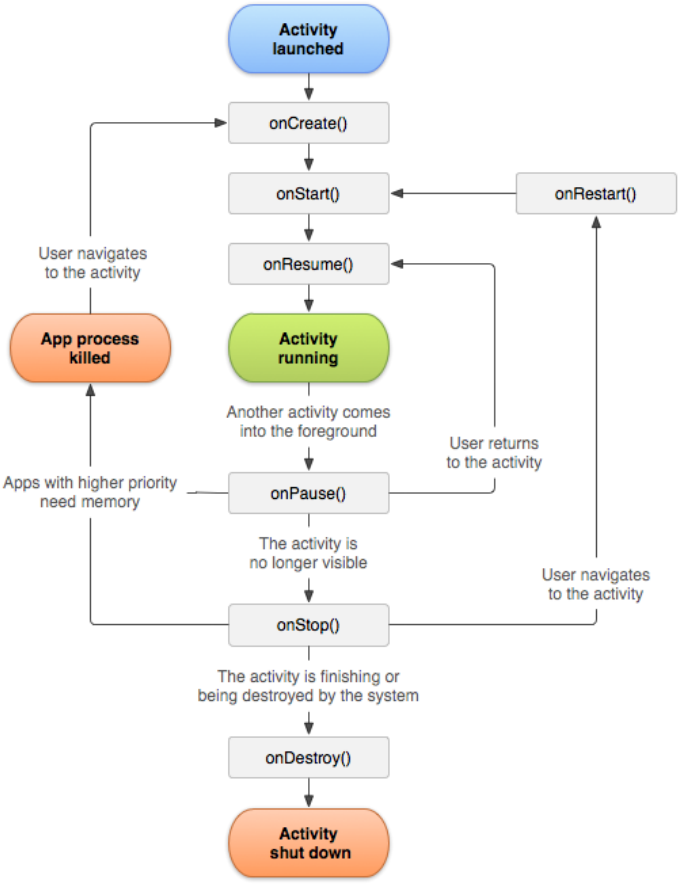
Gambar 2.10 Hama Tungau pada Tanaman Cabai

(Meilin, 2014)

2.4 Android

Android merupakan sistem operasi yang bersifat *open source* dan berbasis Linux yang dibuat untuk berbagai bentuk perangkat. Dengan menggunakan *kernel* Linux maka Android dapat menjalankan fungsi-fungsi yang mendasar pada perangkat misalnya untuk manajemen memori. Selain itu menggunakan *kernel* Linux juga memungkinkan Android untuk mendapatkan keuntungan utama yaitu

fitur keamanan dan juga memungkinkan produsen perangkat untuk mengembangkan *driver* perangkat keras dengan *kernel* yang cukup terkenal ini (Developers, 2019b). Sebuah aplikasi Android terdiri dari dari satu atau lebih *activity,* dimana *activity* tersebut memiliki *lifecycle* seperti yang ada pada Gambar 2.11.



Gambar 2.11 Ilustrasi dari *activity lifecycle*

Sumber: (Developers, 2019d)

2.5 Architecture Components

Android *architecture components* merupakan kumpulan alat yang dapat membantu untuk merancang aplikasi Android yang lebih baik, dapat diuji, dan mudah untuk dipelihara (Developers, 2019a). *Architecture components* sendiri merupakan bagian dari Android Jetpack yang diperkenalkan dalam acara tahunan Google pada tahun 2018. Dimana Android Jetpack adalah serangkaian alat dan panduan untuk membantu pengembang Android membuat aplikasi berkualitas tinggi dengan lebih mudah.

Dalam *architecture components* terdapat dua bagian yang dibahas, yaitu siklus hidup komponen anatarmuka dan pengelolaan data. Untuk siklis hidup komponen dalam *architecture* components menggunakan bantuan kelas ViewModel dan untuk pengelolaan data menggunakan Room. Dengan *architecture components* ini

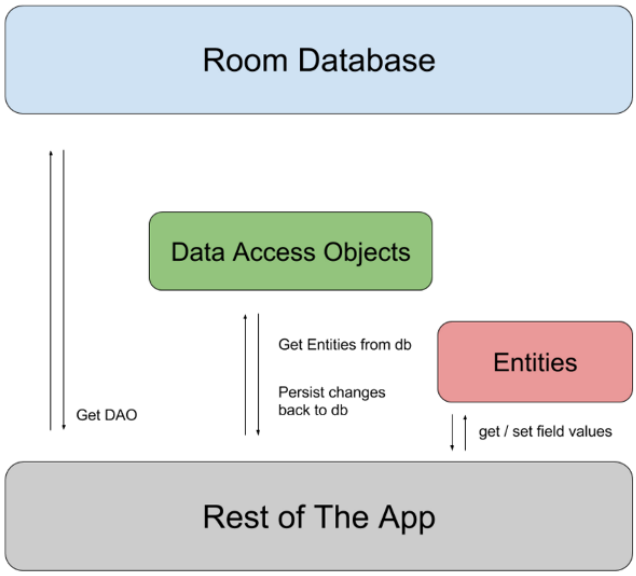
memungkin pembuatan *design pattern* Model-View-ViewModel pada pengembangan aplikasi Android.

2.5.1 ViewModel

Komponen antarmuka yang baik adalah yang dapat mempertahankan datanya saat terjadi perubahan konfigurasi. Solusi untuk mempertahankan data ini adalah dengan menggunakan ViewModel. ViewModel sendiri akan mempertahankan data yang ada di anatarmuka meskipun ada perubahan konfigurasi dalam anatarmuka. ViewModel ini mengelola siklus hidup komponen antarmuka, jadi kelas ini mampu mendeteksi komponen antarmukanya. Dengan begitu ViewModel mampu mempertahankan data yang ada dalam anatarmuka (Developers, 2019e).

2.5.2 Room

Room menyediakan lapisan abstraksi di atas SQLite yang memungkinkan untuk mengakses database yang lancar. SQLite merupakan SQL database yang digunakan sebagai penyimpanan lokal dalam Android. Dengan menggunakan Room maka penerapan penyimpanan lokal dalam Android akan semakin mudah. Dalam Room sendiri terdapat tiga komponen utama seperti yang ada pada Gambar 2.12. *Database* yang berfungsi sebagai pemegang database dan berfungsi sebagai akses utama ke data relasional yang ada pada aplikasi. Selanjutnya *Entities* yang merupakan representasi tabel yang ada dalam database. Dan yang terakhir adalah *Data Access Obects* yang meruapakan kelas yang berisi metode yang digunakan untuk mengakses database.



Gambar 2.12 Komponen Room

Sumber: (Developers, 2019c)

2.6 Ximilar Custom Image Recognition

Ximilar Custom Image Recognition merupakan platform menyediakan layanan *machine learning*, dimana dengan *machine learning* mampu memberikan komputer kemampuan untuk belajar tanpa deprogram secara eksplisit. Ximilar Custom Image Recognition menggabungkan teknik *machine learning* dan model *neural network* untuk membuat sistem pengenalan dan kalsifikasi objek. Teknologi yang digunakan dalam Ximilar Custom Image Recognition meliputi penggunakan layanan TensorFlow dan juga bahasa pemrograman Python. Untuk menggunakan Ximilar Custom Image Recognition secara umum meliputi mendefinisikan label dan mengunggah gambar, melatih sistem dengan gambar yang telah diunggah, dan mencoba mengenali sebuah gambar uji coba (Ximilar, 2019b).

2.7 Firebase Cloud Storage

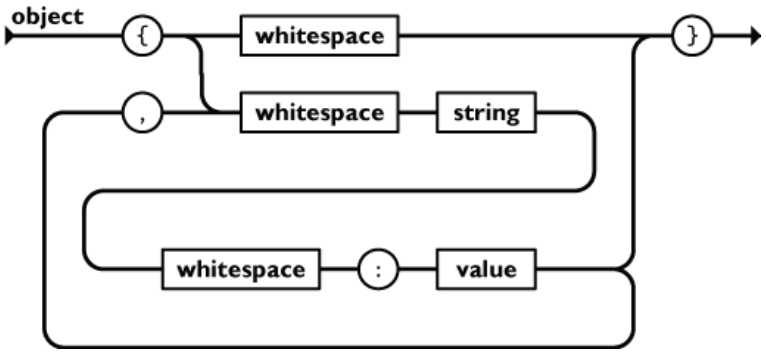
Firebase Cloud Storage merupakan sistem penyimpanan yang memudahkan pengembang aplikasi untuk menyimpan dan menyajikan data penggunanya. Cloude Storage dapat digunakan untuk menyimpan berupa gambar, video, audio, bahkan konten lainnya. Dalam penerpannya pada aplikasi perangkat bergerak Android dapat melalui SDK (*Software Development Kit*) yang telah disediakan dari Firebase Cloud Storage (Firebase, 2019a). Dalam penelitian ini, Cloud Storage digunakan untuk menyimpan gambar yang dideteksi. Dengan mengunggah gambar ke Cloud Storage maka akan didapatkan alamat gambar yang digunakan untuk proses pendeteksian.

2.8 JSON (*JavaScript Object Notation*)

JavaScript Object Notation atau JSON merupakan sebuah format pertukaran data yang ringan. JSON bersifat independen tidak bergantung oleh bahasa pemrograman manapun. Format JSON adalah teks yang menggunakan bahasa yang umum digunakan. Hal tersebut membuat JSON sebagai format pertukaran data yang ideal untuk digunakan (JSON, 2019).

Struktur dalam JSON adalah *object* dan *array*. *Object* sendiri adalah kumpulan nama atau nilai yang tersusun secara tidak teratur yang dimulai dengan kurung buka kurawal '{' dan ditutup dengan kurung tutup kurawal '}', setiap nama diikuti oleh titik dua ':' dan setiap pasangan nama atau nilai dipisahkan oleh koma ','. Sedangkan untuk *array* adalah kumpulan nilai yang tersusun secara urut yang dimulai dengan braket kiri '[' kemudian ditutup dengan braket kanan ']' dan setiap nilai dipisahkan oleh koma ','. Pada Gambar 2.13 merupakan bentuk *object* dalam JSON dan Gambar 2.14 merupakan bentuk *array* dalam JSON.

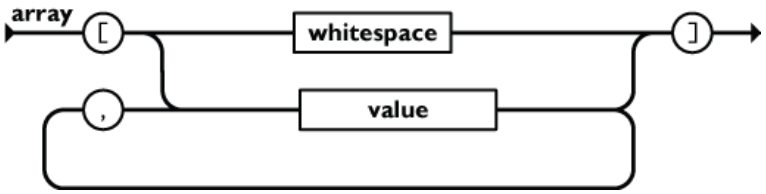
1. *Object*



Gambar 2.13 *Object* dalam JSON

Sumber: (JSON, 2019**)**

2. *Array*



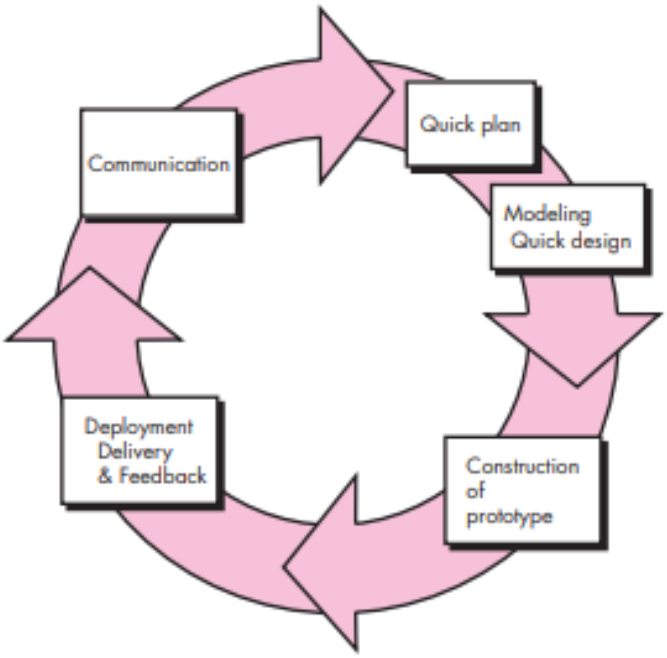
Gambar 2.14 *Array* dalam JSON

Sumber: (JSON, 2019)

2.9 Metode *Prototyping*

Metode *prototyping* merupakan sebuah metode pengembangan perangakat lunak dimana pengembang memulainya dengan membuat sebuah *prototype* terlebih dahulu. Dalam *prototyping* terdapat beberapa jenis yaitu *rapid prototyping* atau *throwaway prototyping* dan *evolutionary prototyping*. Dalam penelitian ini digunakan *evolutionary prototyping* karena pembuatan *prototype* dimulai berdasarkan kebutuhan awal yang belum pasti (Sherrell, 2013).

Diagram *prototyping* yang dijelaskan pada Gambar 2.15 dimulai dengan tahap komunikasi anatara pengembang dan juga pengguna. Selanjutnya dilakukan perencanaan pembuatan *prototype* dengan cepat dan melakukan pemodelan dalam bentuk rancangan antarmuka. Kemudian perancangan tersebut digunakan sebagai dasar kontruksi pembuatan *prototype*. Dan pada akhirnya *prototype* yang dibuat diberikan ke pengguna untuk dievaluasi dan mencari umpan-balik yang berguna untuk memperbaiki analisis kebutuhan. Metode *prototyping* dalam penelitian ini digunakan sebagai metode menganalisis kebutuhan.



Gambar 2.15 Diagram *Prototyping*

Sumber: (Pressman, 2010)

2.10 Pengujian

2.10.1 Pengujian Validasi

Pengujian validasi dalam artian sederhana adalah bahwa validasi berhasil ketika sistem melakukan fungsinya sesuai dengan yang diharapkan pengguna (Sommerville, 2016). Pengujian validasi bertujuan untuk memvalidasi pada keluaran hasil aplikasi sesuai dengan kasus uji yang dilakukan pada setiap kebutuhan fungsionalitas. Hasil dari pengujian validasi ini dipresentasikan dalam bentuk tabel yang menyatakan apakah keluaran hasil aplikasi tersebut valid atau tidak valid.

2.10.2 Pengujian Akurasi

Pengujian akurasi merupakan pengujian yang digunakan untuk mengukur seberapa akurat hasil deteksi penyakit pada tanaman cabai dari aplikasi dengan cara mencoba aplikasi pendeteksi penyakit tanaman cabai secara langsung dengan munggunakan data latih dan juga data uji yang ada.

2.10.3 Pengujian *Usability*

Pengujian *usability* bertujuan untuk mengetahui bagaimana respon pengguna pada saat menjalankan aplikasi. Pengujian *usability* ini juga untuk menilai seberapa berguna dan juga mudah sebuah aplikasi digunakan oleh pengguna. Dalam pengujian *usability* diperlukan responden sebanyak 5 orang. Hasil dari pengujian *usability* dapat menampilkan kepuasan pengguna terhadap penggunaan aplikasi (Nielsen, 2004).

System Usability Scale (SUS) merupakan skala yang digunakan untuk penilaian sebuah sistem dengan menggunakan sepuluh pertanyaan sederhana yang mampu menggambarkan secara umum sistem tersebut. *System Usability Scale* (SUS) sendiri merupakan skala Likert dimana terdapat pertanyaan dan kemudian pengguna memilih tingkat persetujuan dan ketidaksetujuan dengan pernyataan skala lima poin (Brooke, 1986). *System Usability Scale* (SUS) digunakan sebagai instrumen pengujian *usability*.

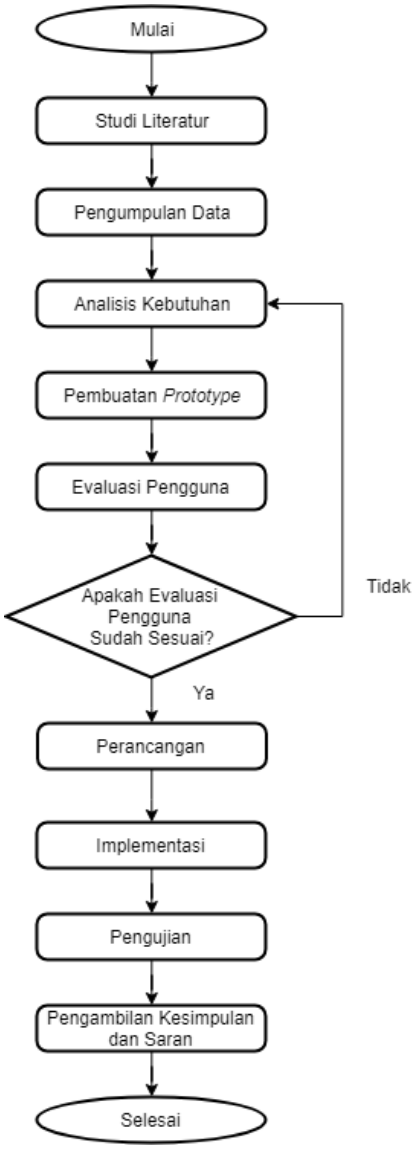
2.10.4 Pengujian *Compatibility*

Pengujian *compatibility* merupakan pengujian yang sering dilakukan pada aplikasi seluler. Pengujian *usability* bertujuan untuk memvalidasi ketergantungan pada aplikasi yang diuji dan lingkungan yang digunakan untuk menjalankannya (Zhang, Gao, Cheng, & Uehara, 2015). Pengujian ini dilakukan dengan menjalankan aplikasi pada beberapa perangkat yang berbeda-beda konfigurasinya.

Instrumen yang digunakan dalam pengujian *compatibility* ini adalah Firebase Test Lab. Dengan Firebase Test Lab memungkinkan untuk menguji sebuah aplikasi seluler dengan beberapa perangkat yang berbeda konfigurasinya. Firebase Test Lab juga mampu mensimulasikan penggunaan aplikasi menyerupai pengguna sebenarnya dan juga sebagai alat yang membantu menentukan penyebab kegagalan aplikasi saat dijalankan (Firebase, 2019b).

BAB 3 METODOLOGI PENELITIAN

Penelitian ini bertipe implementatif dengan pendekatan pengembangan, yaitu untuk membuat sebuah artefak perangkat lunak dengan menerapkan prinsip-prinsip rekayasa perangkat lunak. Lokasi dari penelitian ini dilakukan pada Balai Pengkajian Teknologi Pertanian Jawa Timur. Dalam bab ini akan menjelaskan semua tahapan yang akan dilakukan sepanjang penelitian pengembangan aplikasi, sebuah metode sudah pasti dibutuhkan untuk dijadikan landasan pengembangan agar proses pengembangan tersebut bisa terstruktur dengan baik. Dalam melakukan penelitian ini, peneliti menggunakan metode *prototyping* untuk menggali kebutuhan. Adapun alur dari tiap tahapan metode yang akan digunakan tersebut dijelaskan pada Gambar 3.1.



Gambar 3.1 Alur Metodologi Penelitian

3.1 Studi Literatur

Studi literatur digunakan untuk mengumpulkan dan mempelajari literatur yang berhubungan dengan penelitian ini dengan tujuan untuk mendapatkan informasi tambahan dan sebagai acuan dalam penelitian ini. Dalam penelitian ini literatur yang digunakan adalah yang berhubungan dengan anatara lain:

1. Penyakit pada tanaman cabai
2. Hama pada tanaman cabai
3. Pengembangan aplikasi Android
4. Android *architecture components* yang digunakan sebagai arsitektur pengembangan aplikasi dalam penelitian ini
5. Gambaran umum proses yang dilakukan untuk menggunakan Ximilar Custom Image Recognition
6. JSON (*JavaScript Obejct Notation*) sebagai format pertukaran data
7. Metode *prototyping*
8. Pengujian validasi dan akurasi

3.2 Pengumpulan Data

Tahap pengumpulan data merupakan tahap untuk mendapatkan data yang akan digunakan dalam penelitian. Pada penelitian ini data berasal dari Balai Pengkajian Teknologi Pertanian Jawa Timur. Proses pengambilan data gambar penyakit pada tanaman cabai dengan menggunakan kamera Sony XZ Premium dengan bantuan latarbelakang berupa kertas berwarna abu-abu. Kemudian proses pengambilan data untuk kebutuhan sistem dilakukan dengan cara wawancara langsung secara individual terhadap pakar penyakit dan hama pada tanaman.

3.3 Analisis Kebutuhan

Tahap analisis kebutuhan dilakukan bertujuan untuk mendapatkan kebutuhan yang diperlukan untuk mengembangkan sistem. Kebutuhan tersebut berasal dari tahap pengumpulan data maupun dari umpan balik yang dihasilkan dari evaluasi pengguna. Instrumen yang digunakan dalam analisis kebutuhan ini adalah *object-oriented analysis*. Dimana dilakukan pembuatan *use case diagram* dan *use case scenario.* Agar mempermudah dalam memenuhi kebutuhan yang ada maka penulis mengelompokan menjadi kebutuhan fungsional dan non-fungsional.

3.4 Pembuatan *Prototype*

Pada tahap pembuatan *prototype* dilakukan pembuatan *prototype* dalam bentuk *prototype* sistem setengah jadi. Pembuatan *prototype* ini bertujuan untuk memudahkan dalam melakukan evaluasi kebutuhan yang sudah ada oleh pengguna.

3.5 Evaluasi Pengguna

Pada tahap evaluasi pengguna, *prototype* yang sudah ada diujikan ke pengguna untuk mendapatkan umpan balik dari pengguna. Umpan balik yang didapat bisa

berupa perbaikan kebutuhan yang sudah ada atau penambahan kebutuhan yang baru. Setelah tahap evaluasi pengguna selesai maka dilakukan pengecekan apakah hasil evaluasi sudah sesuai dengan yang dibutuhkan pengguna. Jika belum sesuai maka dilakukan analisis kebutuhan kembali untuk selanjutnya dilakukan perancangan dan pembuatan *prototype* kembali dan jika hasil evaluasi sudah sesuai maka dilanjutkan pada tahap implementasi.

3.6 Perancangan

Pada tahap ini dilakukan pembuatan rancangan arsitekture sistem *activity diagram*, *sequence diagram*, dan *class diagram.* Setelahnya dilakukan pembuatan rancangan basis data, rancangan algoritme, dan yang terakhir adalah pembuatan rancangan antarmuka untuk memudahkan implementasi antarmuka dari kebutuhan yang sudah ada. Pembuatan rancangan ini berdasarkan kebutuhan yang dihasilkan pada tahap analisis kebutuhan.

3.7 Implementasi

Pada tahap implementasi dilakukan pembuatan aplikasi yang berdasarkan pada rancangan yang sudah ada dan dalam tahap ini juga terdapat pengintegrasian Ximilar Custom Image Recognition sebagai pendeteksi penyakit pada tanaman cabai. Pengimplementasian aplikasi ini menggunakan bantuan *integrated development environment* (IDE) yaitu Android Studio dengan menggunakan bahasa pemrograman Java.

3.8 Pengujian

Pada tahap ini dilakukan pengujian terhadap apliaksi hasil dari implementasi. Pengujian pada tahap ini mencakup pengujian fungsional dan juga pengujian akurasi. Untuk pengujian fungsional dilakukan dengan menggunakan metode *blackbox testing*. Dan untuk pengujian akurasi dilakukan bertujuan untuk mengukur seberapa akurat hasil deteksi penyakit tanaman cabai yang dilakukan dengan aplikasi ini. Pengujian akurasi dilakukan dengan menggunakan data latih dan data uji yang ada. Serta terdapat pengujian *usability* untuk mengetahui kemudahan penggunaan aplikasi oleh pengguna dan pengujian *compatibility* untuk menguji aplikasi dapat berjalan pada perangkat Android dengan minimal level API 21.

3.9 Pengambilan Kesimpulan dan Saran

Pada tahap pengambilan kesimpulan dilakukan berdasarkan pada rumusan masalah yang telah ditentukan dan disimpulkan setelah semua tahapan penelitian selesai. Penulisan saran dilakukan untuk memberikan saran perbaikan untuk penelitian selanjutnya.

BAB 4 ANALISIS KEBUTUHAN

Bab ini membahas mengenai analisis kebutuhan aplikasi pendeteksi penyakit pada tanaman cabai. Dalam bab analisis kebutuhan ini terdapat beberapa hal seperti analisis kebutuhan yang terdapat indentifikasi aktor, kebutuhan fungsional dan non fungsional, dan *use case diagram* serta *use case scenario*.

4.1 Gambaran Umum

Aplikasi yang dikembangkan dalam penelitian ini adalah aplikasi pendeteksi penyakit pada tanaman cabai atau dalam penelitian ini disebut Agrai. Tujuan utama aplikasi Agrai adalah untuk membantu pengguna dalam mendeteksi penyakit yang menyerang tanaman cabai. Dalam penelitian ini menggunakan *web service* untuk mendeteksi penyakit tanaman cabai, *web service* yang digunakan adalah Ximilar Custom Image Recognition. Ximilar Custom Image Recognition mampu mengklasifikasikan gambar sesuai dengan data latih yang telah dibuat. Selain itu, aplikasi ini juga akan menampilkan cara pengendalian sesuai dengan penyakit yang terdeteksi. Hasil dari deteksi juga akan disimpan dalam penyimpanan lokal dari *smartphone* pengguna.

4.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mendapatkan kebutuhan yang sesuai dengan kebutuhan pengguna aplikasi pendeteksi penyakit pada tanaman cabai. Analisis kebutuhan ini dilakukan dengan cara studi literatur dan wawancara langsung ke pakar penyakit dan hama tanaman. Hasil studi literatur dan wawancara dari pakar penyakit dan hama tanaman disimpulkan menjadi beberapa poin penting seperti yang ditunjukan Tabel 4.1.

Tabel 4.1 Hasil Analisis Kebutuhan Awal

No.	Hasil Analisis Kebutuhan Awal
1.	Pengguna melakukan identifikasi penyakit pada tanaman cabai secara konvensional yaitu dengan mengamati secara langsung ciri-ciri yang terlihat pada tanaman.
2.	Sebagian pengguna mengalami kesulitan untuk menentukan penyakit apa yang menyerang tanaman cabainya.
3.	Sebagian pengguna merasa kesulitan untuk mengetahui gejala untuk setiap penyakit yang menyerang tanaman cabai.
4.	Informasi yang diperlukan pengguna adalah informasi gajala serangan, dan cara pengendalian.

4.3 Iterasi *Prototype* Pertama

Dalam iterasi *prototype* yang pertama dilakukan pembuatan *prototype* berdasarkan hasil analisis kebutuhan awal. Pada *prototype* yang pertama terdapat tiga kebutuhan fungsional yaitu mengambil gambar, mendeteksi penaykit, dan melihat informasi penyakit. Setelah melakukan pembuatan *prototype*, maka dilakukan evaluasi pengguna untuk mengetahui apakah kebutuhan fungsional yang ada sudah sesuai dengan yang diharapkan pengguna dan juga untuk menggali lebih banyak kebutuhan yang lainnya.

Hasil evaluasi pengguna pada uji *prototype* yang pertama terdapat kebutuhan baru yaitu pengguna mengharapkan bisa melihat riwayat hasil deteksi penyakit yang telah dilakukan sebelumnya. Selain itu juga terdapat informasi yang kurang pada kebutuhan fungsional yang sudah ada. Dalam kebutuhan fungsional melihat informasi penyakit pengguna juga menginginkan dapat melihat pestisida yang bisa digunakan untuk mengendalikan penyakit tersebut.

4.4 Iterasi *Prototype* Kedua

Dalam iterasi *prototype* yang kedua dilakukan perbaikan *prototype* dari hasil evaluasi pada tahap iterasi pertama. Dilakukan penambahan kebutuhan fungsional yang didapat dari hasil evaluasi *prototype* yang pertama. Setelah melakukan perbaikan *protorype*, maka dilakukan evaluasi pengguna kembali guna mengetahui apakah kebutuhan fungsional yang ada pada *prototype* kedua sudah sesuai dengan yang diharapkan dan juga untuk mengetahui apakah ada kebutuhan yang lainnya.

Evaluasi pengguna pada iterasi kedua menghasilkan beberapa perbaikan *prototype* namun tidak menambahkan kebutuhan baru. Hal ini menandakan bahwa *prototype* kedua sudah memenuhi kebutuhan yang sesuai dengan pengguna. Perbaikan *prototype* yang didapat adalah penambahan halaman tutorial awal dan juga halaman informasi aplikasi.

4.5 Identifikasi Aktor

Tahap identifikasi aktor bertujuan untuk mengidentifikasi aktor yang berinteraksi langsung dengan aplikasi pendeteksi penyakit pada tanaman cabai. Hasil dari identifikasi aktor dijelaskan pada Tabel 4.2.

Tabel 4.2 Identifikasi Aktor

Aktor	**Deskripsi**
Pengguna	Merupakan orang yang berinteraksi langsung dengan aplikasi Agrai dan ingin mendeteksi penyakit pada tanaman cabai. Pengguna dapat seorang petani ataupun peneliti tanaman cabai.

4.6 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan utama yang harus dipenuhi oleh sistem. Terpenuhinya kebutuhan fungsional ini supaya aplikasi dapat digunakan oleh pengguna dan aplikasi dapat berjalan sesuai dengan yang diharapkan. Kebutuhan fungsional merupakan hasil dari analisis kebutuhan. Berikut adalah tabel kebutuhan fungsional yang ditunjukan pada Tabel 4.4.

Untuk mempermudah dalam proses analisis, perancangan, dan implementasi maka setiap kebutuhan diberikan kode sebagai penunjuk kebutuhan. Dalam pemberian kode setiap kebutuhan sesuai dengan penjelasan kode kebutuhan yang ada pada Tabel 4.3.

Tabel 4.3 Penjelasan Kode Kebutuhan

SRS-F/NF-ARG-XX	
SRS	SRS merupakan singkatan dari *system requirement specification* atau spefikasi kebutuhan sistem.
F/NF	F merupakan kode penunjuk untuk kebutuhan fungsional, sedangkan NF untuk kebutuhan non fungsional.
AGR	AGR merupakan inisial untuk aplikasi pendeteksi penyakit tanaman cabai.
XX	XX merupakan kode penunjuk untuk menyatakan nomor kebutuhan.

Tabel 4.4 Deskripsi Kebutuhan Fungsional

Kode kebutuhan	**Deskripsi**	***Use case***
SRS-F-AGR-01	Sistem harus mampu menyediakan fungsi untuk mengambil gambar melalui kamera atau dari galeri.	Mengambil gambar
SRS-F-AGR-02	Sistem harus mampu menyediakan fungsi untuk mendeteksi penyakit tanaman cabai.	Mendeteksi penyakit
SRS-F-AGR-03	Sistem harus mampu menyediakan fungsi untuk melihat informasi penyakit pada tanaman cabai.	Melihat informasi penyakit
SRS-F-AGR-04	Sistem harus mampu menyediakan fungsi untuk melihta riwayat hasil deteksi penyakit tanaman cabai.	Melihat riwayat deteksi

4.7 Kebutuhan Non Fungsional

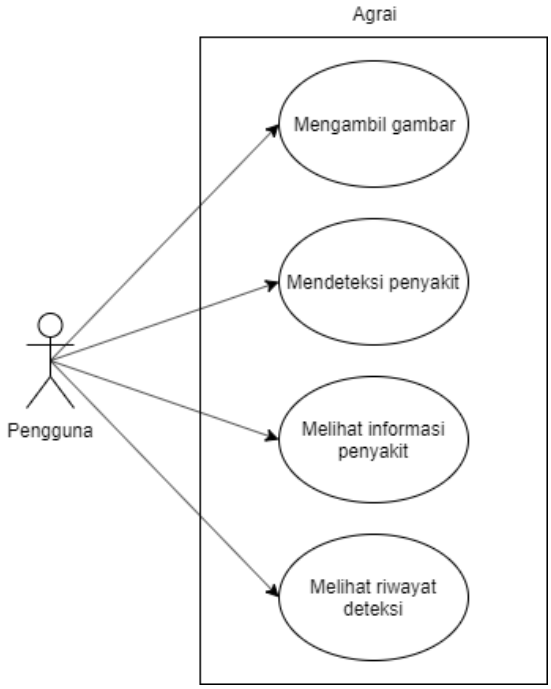
Kebutuhan non fungsional adalah kebutuhan yang tidak harus dipenuhi oleh sistem. Namun terpenuhinya kebutuhan ini mampu memberikan pengalaman pengguna yang lebih baik pada aplikasi. Pada Tabel 4.5 merupakan kebutuhan non fungsional yang ada pada aplikasi pendeteksi penyakit pada tanaman cabai ini.

Tabel 4.5 Deskripsi Kebutuhan Non Fungsional

Kode kebutuhan	Nama Kebutuhan	Deskripsi
SRS-NF-AGR-01	*Usability*	Pengguna dapat menjalankan aplikasi dengan mudah.
SRS-NF-AGR-02	*Compatibility*	Aplikasi dapat berjalan pada perangkat Android dengan minimal level API 21.

4.8 *Use Case Diagram*

Use case diagram menggambarkan aksi yang dapat dilakukan oleh pengguna terhadap sistem. Pemodelan *use case diagram* pada aplikasi ini digambarkan pada Gambar 4.1. Terdapat satu aktor dalam pemodelan *use case diagram* pada aplikasi ini yaitu aktor pengguna. Dimana aktor pengguna dapat melakukan aksi mengambil gambar, mendeteksi penyakit, melihat informasi penyakit, dan melihat riwayat deteksi.



Gambar 4.1 *Use Case Diagram*

4.9 *Use Case Scenario*

Use case scenario dibuat setelah pemodelan *use case diagram* telah selesai. *Use case scenario* dibuat untuk menjelaskan setiap *use case* yang ada. Dimana *use case scenario* menjelaskan diantaranya adalah kondisi sistem sebelum tindakan (Pra-kondisi), tindakan yang dilakukan, dan kondisi sistem setelah tindakan (Post-kondisi). *Use case scenario* untuk aplikasi ini dijelaskan pada Tabel 4.6 sampai Tabel 4.10.

Tabel 4.6 *Use Case Scenario* Mengambil Gambar

Nama *Use* Case	Mengambil gambar
Kode Kebutuhan	SRS-AGR-01
Aktor	Pengguna
Tujuan	Pengguna dapat mengambil gambar tanaman cabai melalui kamera atau galeri
Pra-kondisi	Pengguna telah berada pada halaman utama
Tindakan Utama	1. Pengguna menekan tombol deteksi 2. Sistem menampilkan pilihan untuk mengambil gambar melalui kamera atau galeri 3. Pengguna mengambil gambar melalui kamera atau galeri dan memotong gambar sesuai yang diinginkan 4. Sistem menampilkan halaman periksa
Post-kondisi	Pengguna mendapatkan gambar yang diinginkan
Alternatif	-

Tabel 4.7 *Use Case Scenario* Mendeteksi Penyakit

Nama *Use* Case	Mendeteksi penyakit
Kode Kebutuhan	SRS-AGR-02
Aktor	Pengguna
Tujuan	Pengguna mengetahui penyakit yang menyerang tanaman cabainya
Pra-kondisi	Pengguna telah memilih gambar yang akan dideteksi
Tindakan Utama	1. Pengguna menekan tombol deteksi 2. Sistem memproses gambar untuk dideteksi dan menampilkan penyakit hasil deteksi ke layar

Tabel 4.8 *Use Case Scenario* Mendeteksi Penyakit (lanjutan)

Post-kondisi	Pengguna mengetahui penyakit yang menyerang tanaman cabainya, yang dapat diketahui pengguna adalah berupa nama penyakit, tingkat akurasi, dan gambar yang dideteksi
Alternatif	Jika koneksi internet pada *smartphone* pengguna tidak ada, maka sistem akan menampilkan pesan bahwa koneksi internet tidak tersedia

Tabel 4.9 *Use Case Scenario* Melihat Informasi Penyakit

Nama *Use* Case	Melihat informasi penyakit
Kode Kebutuhan	SRS-AGR-03
Aktor	Pengguna
Tujuan	Pengguna memperoleh informasi penyakit yang menyerang tanaman cabainya
Pra-kondisi	Sistem telah mengetahui penyakit yang menyerang tanaman cabainya
Tindakan Utama	1. Pengguna menekan tombol cara pengendalian 2. Sistem menampilkan informasi penyakit
Post-kondisi	Pengguna memperoleh informasi penyakit berupa gejala, cara pengendalian, dan pestisida yang dapat digunakan
Alternatif	-

Tabel 4.10 *Use Case Scenario* Melihat Riwayat Deteksi

Nama *Use* Case	Melihat riwayat deteksi
Kode Kebutuhan	SRS-AGR-04
Aktor	Pengguna
Tujuan	Pengguna memperoleh daftar riwayat deteksi yang telah dilakukan sebelumnya
Pra-kondisi	Pengguna telah berada pada halaman utama
Tindakan Utama	1. Pengguna menekan tombol riwayat 2. Sistem menampilkan daftar riwayat deteksi
Post-kondisi	Pengguna memperoleh riwayat deteksi berdasarkan tanggal deteksi

Tabel 4.11 *Use Case Scenario* Melihat Riwayat Deteksi (lanjutan)

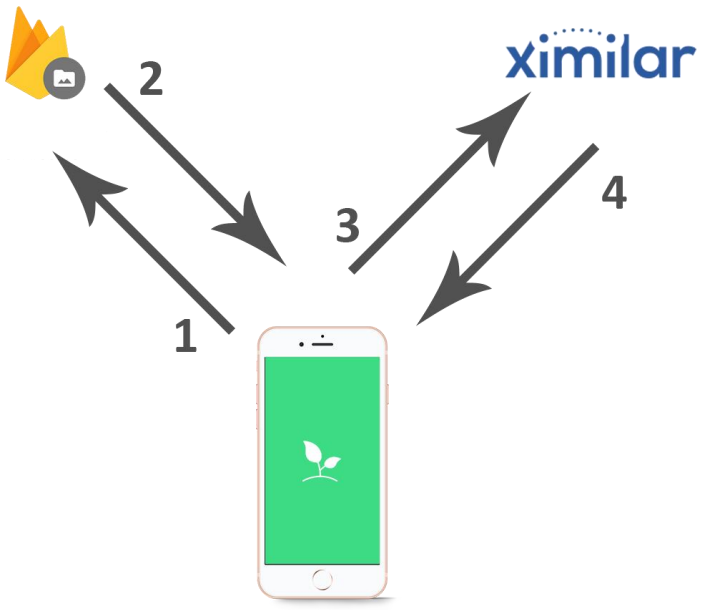
Alternatif	Jika riwayat deteksi belum ada maka sistem menampilkan pesan riwayat deteksi belum ada

BAB 5 PERANCANGAN

Bab ini membahas tentang perancangan aplikasi pendeteksi penyakit pada tanaman cabai. Dalam perancangan ini meliputi pembuatan rancangan arsitektur sistem, *sequence diagram*, *class diagram*, perancangan basis data, perancangan algoritme, dan perancangan antarmuka.

5.1 Perancangan Arsitektur Sistem

Perancangan arsitektur sistem pada aplikasi pendeteksi penyakit pada tanaman cabai digambarkan seperti pada Gambar 5.1. Pada rancangan arsitektur sistem tersebut menggambarkan bahwa teradpat proses pengiriman data antara aplikasi pendeteksi penyakit pada tanaman cabai dengan *web service* yaitu Ximilar Custom Image Recognition. Proses pengiriman data pertama kali dilakukan oleh aplikasi pendeteksi penyakit pada tanaman cabai kepada Firebase Cloud Storage. Kemudian akan mengembalikan pesan balasan berupa alamat dari gambar yang akan dideteksi. Selanjutnya aplikasi mengirimkan alamat tersebut kepada Ximilar Custom Image Recognition, kemudian Ximilar Custom Image Recognition akan memberikan balasan berupa pesan dengan format JSON. Balasan tersebut akan digunakan sebagai informasi pada aplikasi pendeteksi aplikasi pendeteksi penyakit pada tanaman cabai.



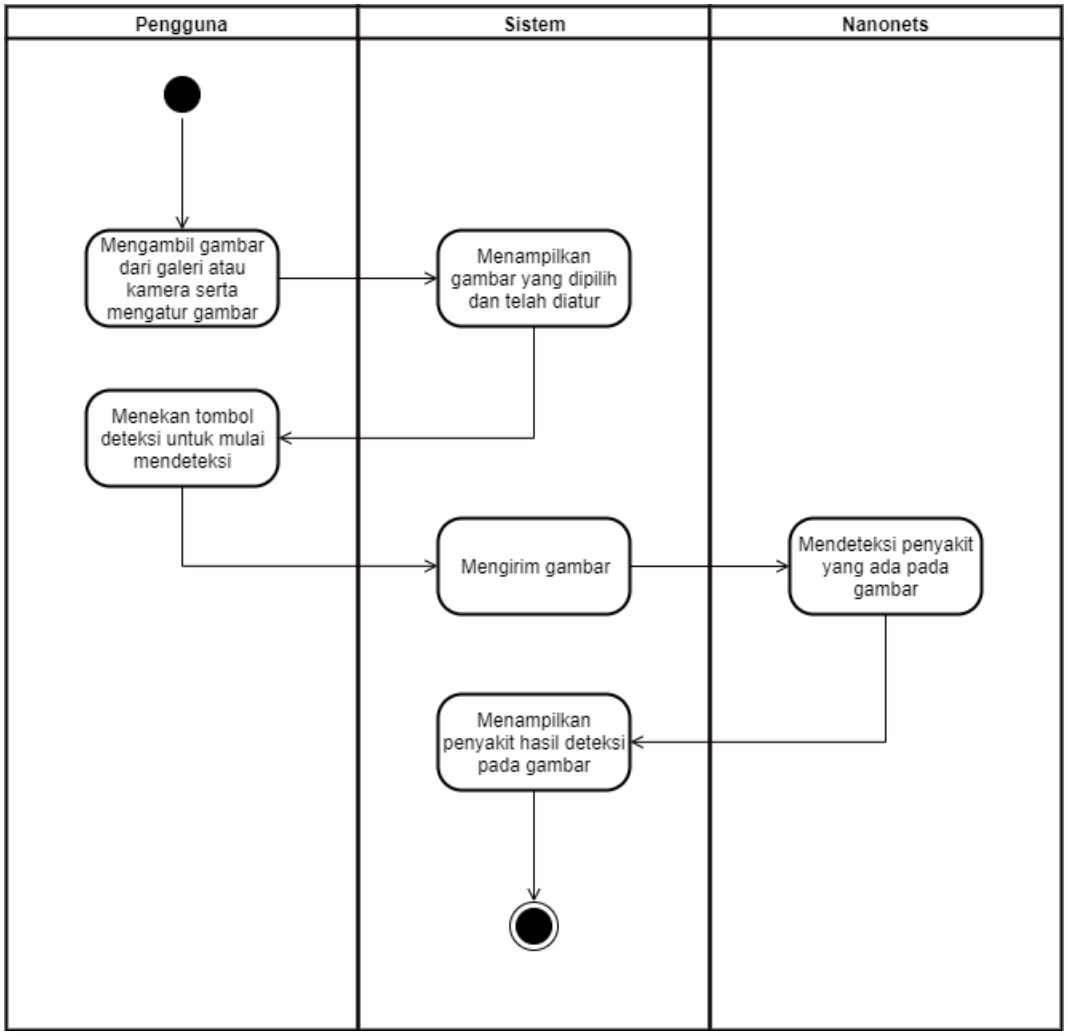
Gambar 5.1 Rancangan Arsitektur Sistem

Perancangan arsitektur sistem tidak hanya menjaskan komunikasi antara aplikasi pendeteksi penyakit pada tanaman cabai dengan layanan Ximilar Custom

Image Recognition. Namun dalam aplikasi pendeteksi penyakit pada tenaman cabai sendiri dilakukan pembuatan percancangan arsitektur meliputi pembuatan *design pattern*. *Design pattern* yang dirancang pada aplikasi pendeteksi penyakit pada tanaman cabai ini adalah berbasis Model-View-ViewModel (MVVM). Model sendiri merupakan bagian yang bertugas menyediakan data. Kemudian ViewModel merupakan bagian yang bertugas menyediakan data untuk View. Dan View merupakan bagian yang bertugas menampilkan data ke pengguna. Dalam penelitian ini penerapan *design pattern* dilakukan dengan menggunakan sekumpulan *library* yang terdapat pada *Architecture Components*. Dalam penerapannya terdapat sedikit modifikasi pada bagian model, dimana dalam model diterapkan *repository pattern* yang bertugas tidak hanya menyediakan data namun juga bagian yang mengatur pengelolaan data.

5.2 *Activity Diagram*

Activity diagram menjelaskan mengenai aktifitas yang terjadi antara pengguna, sistem, dan Ximilar Custom Image Recognition dalam menjalakan aplikasi pendeteksi penyakit pada tanaman cabai ini. Pada Gambar 5.2 merupakan *activity diagram* untuk sistem ini. Dalam *activity diagram* tersebut dijelaskan terdapat 6 aktifitas. Aktifitas ini dimulai dari pengguna mengambil gambar dari galeri atau kamera kemudian menyesuaikan gambar yang dipilih. Selanjutnya sistem menampilkan gambar yang telah dipilih dan disesuaikan. Pengguna menekan tombol deteksi untuk memulai mendeteksi gambar. Selanjutnya sistem akan mengirimkan gambar kepada Ximilar Custom Image Recognition. Ximilar Custom Image Recognition akan mendeteksi penyakit dari gambar yang dikirimkan sistem. Kemudian pada aktifitas terakhir sistem akan menampilkan hasil deteksi yang dilakukan oleh Ximilar Custom Image Recognition.

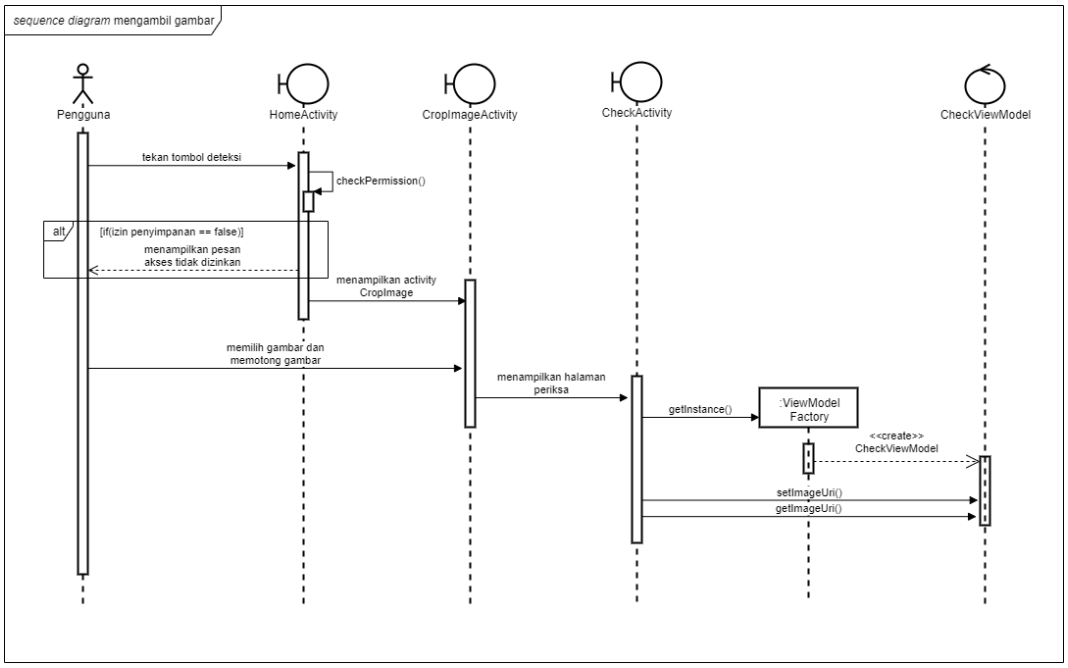


Gambar 5.2 Perancangan *Activity Diagram*

5.3 *Sequence Diagram*

Pada *sequence diagram* menjelaskan urutan proses yang ada dalam aplikasi untuk mencapai tujuan yang ada dalam kebutuhan fungsional. Pada bagian ini akan digambarkan beberapa *sequence diagram* yang sesuai dengan tujuan kebutuhan fungsional, antara lain adalah *sequence diagram* mengambil gambar, mendeteksi penyakit, melihat informasi penyakit, dan melihat riwayat deteksi.

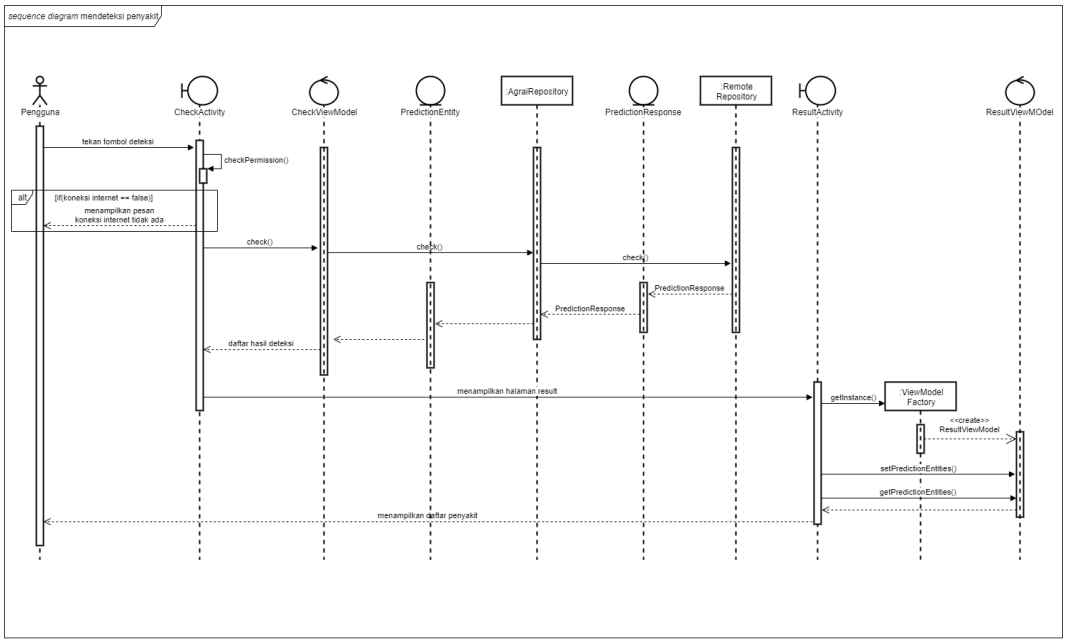
5.3.1 *Sequence Diagram* Mengambil Gambar



Gambar 5.3 *Sequence Diagram* Mengambil Gambar

Sequence diagram mengambil gambar yang terdapat pada Gambar 5.3 menunjukkan bahwa aktor pengguna menekan tombol deteksi pada halaman HomeActivity. Kemudian aplikasi mengecek izin penyimpanan dengan memanggil fungsi `checkPermission()`. Kemudian aplikasi akan menampilkan halaman CropImageActivity, pada halaman ini pengguna memilih dan memotong gambar sesuai keinginannya. Setelah menyesuaikan gambar yang dipilih maka akan tampil halaman CheckActivity. Saat halaman CheckActivity dibuat maka dibuat pula objek dari kelas CheckViewModel dengan bantuan objek ViewModelFactory. Kemudian CheckActivity akan memanggil fungsi `setImageUri()` untuk memasukkan alamat gambar ke objek dari CheckViewModel. Terakhir CheckActivity akan mengobservasi nilai dari fungsi `getImageUri()` pada objek dari CheckViewModel.

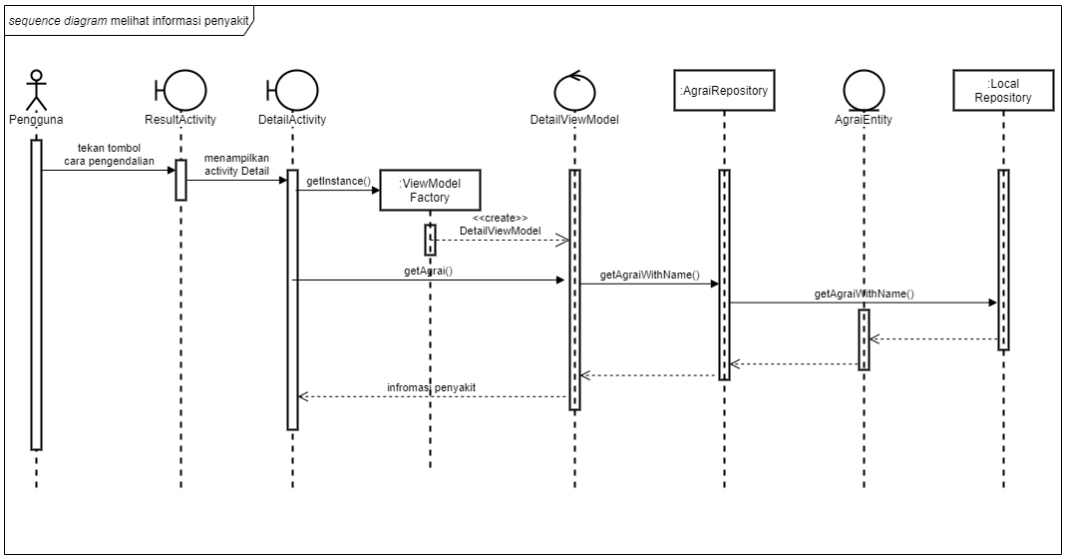
5.3.2 *Sequence Diagram* Mendeteksi Penyakit



Gambar 5.4 *Sequence Diagram* Mendeteksi Penyakit

Sequence diagram mendeteksi penyakit yang terdapat pada Gambar 5.4 menunjukkan bahwa aktor pengguna menekan tombol deteksi pada halaman CheckActivity. Selanjutnya akan dilakukan pemeriksaan apakah koneksi internet tersedia atau tidak. Selanjutnya dilakukan pemanggilan fungsi `check()` dengan cara mengobservasi nilai kembalian dari fungsi tersebut. Kemudian objek dari CheckViewModel akan memanggil fungsi `check()` yang terdapat pada objek AgraiRepository. Dan terjadi pemanggilan fungsi `check()` yang ada pada objek RemoteRepository. Fungsi tersebut memberikan sebuah nilai kembalian berupa PredictionResponse, yang akan diterima oleh objek AgraiRepository. Kemudian fungsi `check()` pada AgraiRepository akan mengembalikan nilai ke CheckViewModel berupa PredictionEntity. Setelah itu akan muncul halaman ResultActivity yang akan menapilkan nilai PredictionEntity. Ketika ResultActivity dibuat maka dibuat pula objek ResultViewModel dengan bantuan objek ViewModelFactory. Selajutnya dilakukan pemanggilan fungsi `setPredictionEntities()` pada objek dari ResultViewModel. Terakhir dilakukan pemanggilan fungsi `getPredictionEntities()` pada objek dari ResultViewModel dengan cara mengobservasi nilai kembalian dari fungsi tersebut.

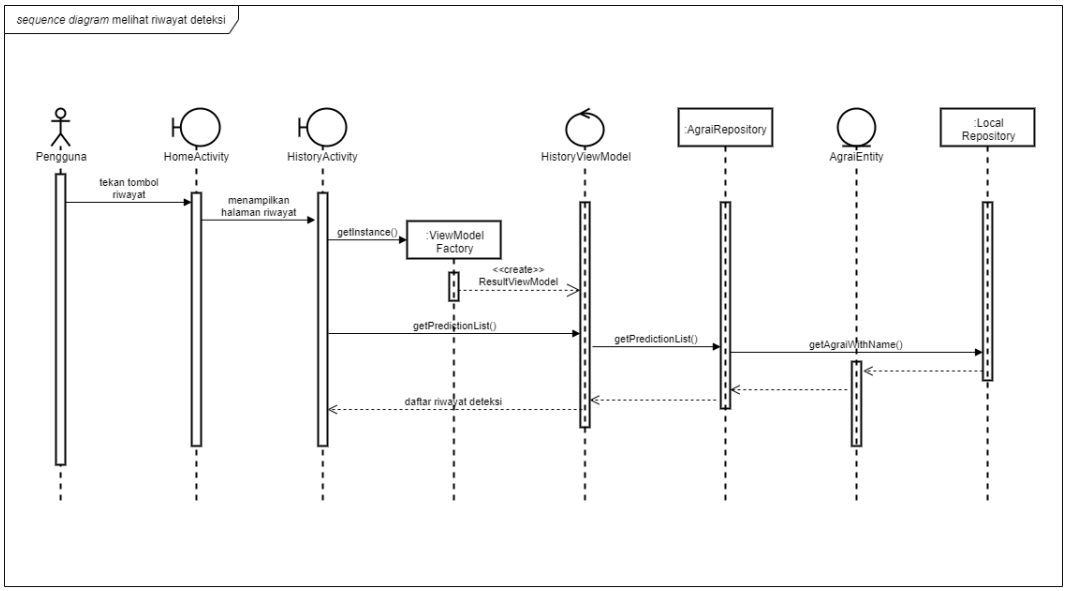
5.3.3 *Sequence Diagram* Melihat Informasi Penyakit



Gambar 5.5 *Sequence Diagram* Melihat Informasi Penyakit

Sequence diagram melihat informasi penyakit yang terdapat pada Gambar 5.5 menunjukkan bahwa akor pengguna menekan tombol cara pengendalian yang ada pada tampilan di halaman ResultActivity. Selanjutnya akan tampil DetailActivity dan memanggil fungsi `getAgrai()` pada objek dari DetailViewModel dengan cara mengobservasi nilai kembalian dari fungsi tersebut. Ketika DetailActivity dibuat maka dibuat pula objek dari DetailViewModel dengan bantuan ViewModelFactory. Kemudian akan memanggil fungsi `getAgraiWithName()` yang ada pada objek dari AgraiRepository dan hal tersbut juga akan memanggil fungsi `getAgraiWithName()` yang ada pada objek dari LocalRepository. Fungsi yang ada pada LocalRepository tersebut akan mengembalikan nilai berupa AgraiEntity yang akan diterima oleh objek dari AgraiRepository. Hal tersebut memicu perubahan nilai kembalian yang ada pada fungsi `getAgrai()` pada objek dari DetailViewModel. Terakhir akan ditampilkan informasi mengenai penyakit yang sesuai dengan hasil deteksi yang ada pada halaman DetailActivity.

5.3.4 *Sequence Diagram* Melihat Riwayat Deteksi

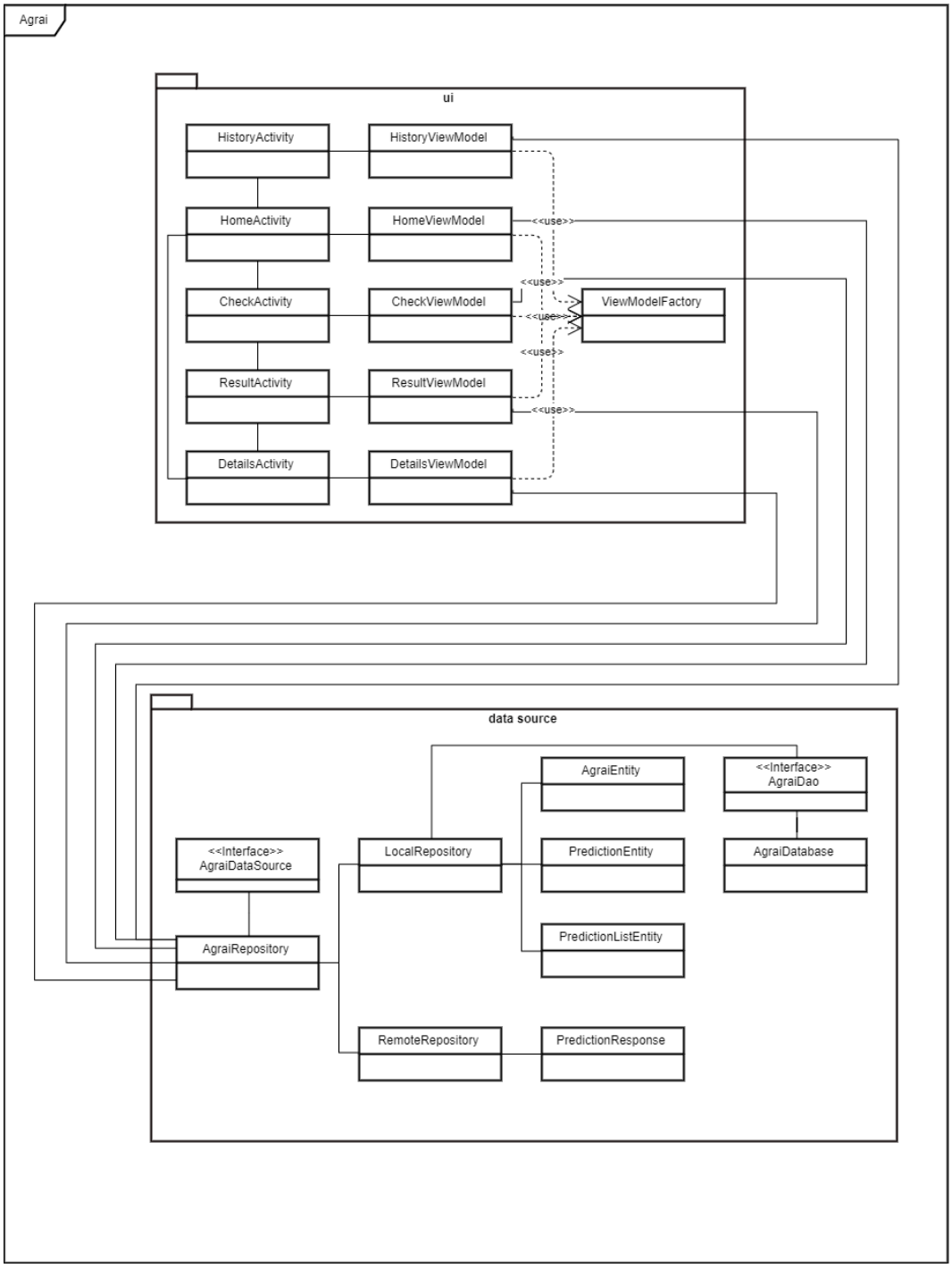


Gambar 5.6 *Sequence Diagram* Melihat Riwayat Deteksi

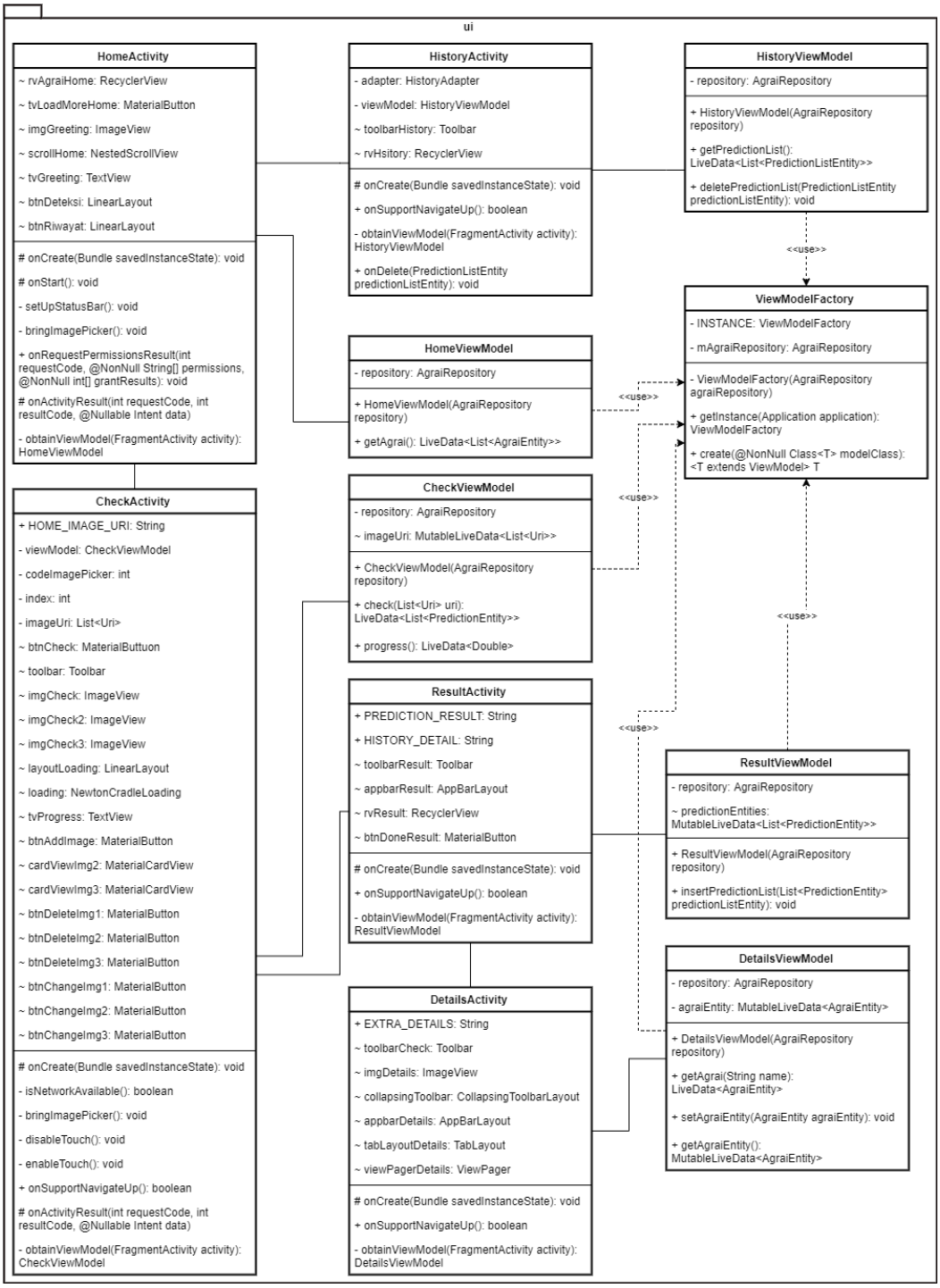
Sequence diagram melihat riwayat deteksi yang terdapat pada Gambar 5.6 menunjukkan bahwa aktor pengguna menekan tombol riwayat yang ada pada halam HomeActivity dan aplikasi akan menampilkan halaman HistoryActivity. Dengan terbentuknya halaman HistoryActivity maka dibuat pula objek dari HistoryViewModel dengan bantuan objek ViewModelFactory. Selanjutnya dilakukan pemanggilan fungsi `getPredictionList()` pada objek dari HistoryViewModel dengan cara mengobservasi nilai kembalian dari fungsi tersebut. Kemudian akan dipanggil fungsi `getPredictionList()` yang ada pada objek dari AgraiRepository dan selanjutnya akan memanggil fungsi `getPredictionList()` yang ada pada objek dari LocalRepository. Fungsi dari objek LocalRepository tersebut akan mengembalikan nilai berupa AgraiEntity yang adan dikirim ke objek dari AgraiRepository. Hal tersebut akan memicu perubahan nilai kembalian dari fungsi `getPredictionList()` yang ada pada objek dari HistoryViewModel. Dan terakhir ditampilkan daftar riwayat deteksi.

5.4 *Class Diagram*

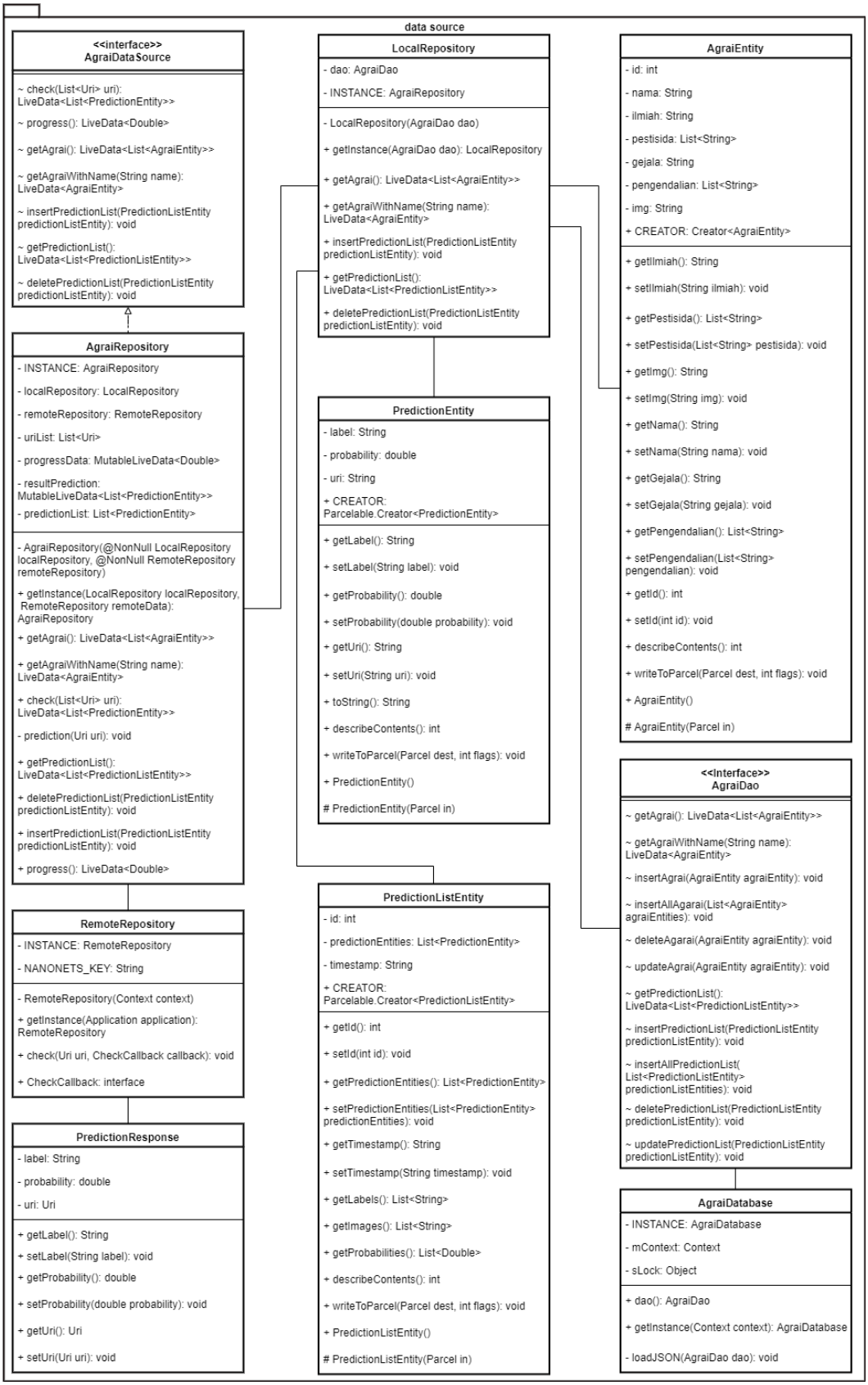
Pembutan *class diagram* bertujuan sebagai pedoman dalam pembetukan kelas yang akan dilakukan pada tahap implementasi aplikasi pendeteksi penyakit pada tanaman cabai. Dalam pembuatan *class diagram* ini dibagi dalam beberapa kelompok paket atau *package*, yaitu ui dan data source. Kelas-kelas yang ada di dalam suatu *package* saling berhubungan dengan kelas lainnya yang ada dalam *package* yang sama ataupun yang berbeda. Pada Gambar 5.7 menunjukkan hubungan kelas-kelas di dalam satu package dan juga terhadapat kelas-kelas yang ada di *package* lainnya.



Gambar 5.7 Rancangan *Class Diagram*



Gambar 5.8 Rancangan *Class Diagram* (ui)



Gambar 5.9 Rancangan *Class Diagram* (*data source*)

Gambar 5.8 menunjukan *class diagram* pada bagian *package* ui. *Package* ui ini berfungsi untuk mempresentasikan data dan juga menampilkannya kepada pengguna. Kelas `HomeActivity` dan `HomeViewModel` digunakan untuk menampilkan halaman utama. Kelas `CheckActivity` dan `CheckViewModel` digunakan untuk menampilkan gambar yang akan dideteksi. Kelas `ResultActivity` dan `ResultViewModel` digunakan untuk menampilkan hasil deteksi dari gambar. Kelas `DetailsActivity` dan `DetailsViewModel` digunakan untuk menampilkan detail dari penyakit diantaranya adalah informasi gejala, cara pengendalian, dan pestisida yang dapat digunakan. Kelas `HistoryActivity` dan `HistoryViewModel` digunakan untuk menampilkan riwayat hasil deteksi yang telah dilakukan. Dan terdapat kelas `ViewModelFactory` yang merupakan kelas yang digunakan untuk membuat objek dari setiap ViewModel yang ada di setiap *activity*.

Gambar 5.9 menunjukan *class diagram* pada bagian *package* data source. *Package* ini berfungsi sebagai pengelola data, baik data dari dan/atau ke layanan Ximilar Custom Image Recognition ataupun penyimpanan lokal. Kelas `AgraiRepository` merupakan kelas yang bertugas memberikan data ke ViewModel ataupun menerima data dari ViewModel. Dalam hal ini `AgraiRepository` mengimplementasikan kelas `AgraiDataSource` yang berguna sebagai kontrak. Kemudian kelas `AgraiRepository` ini juga mengatur apakah data diambil dari penyimpanan lokal atau perlu melakukan permintaan ke layanan Ximilar Custom Image Recognition. Kelas `RemoteRepository` merupakan kelas yang berguna untuk melakukan komunikasi dengan layanan Ximilar Custom Image Recognition. Kelas `PredictionResponse` digunakan sebagai kelas model yang berguna untuk sebagai model dari respon yang diberikan ketika melakukan permintaan ke layanan Ximilar Custom Image Recognition. Selanjutnya kelas `LocalRepository` yang berguna untuk mengelola data yang berada pada penyimpanan lokal. Dalam penyimpanan lokal ini nantinya akan menggunakan sebuah bantuan *library* yaitu Room. Kelas `AgraiEntity`, `PredictionEntity`, dan `PredictionListEntity` berguna sebagai kelas model yang digunakan dalam aplikasi pendeteksi penyakit pada tanaman cabai. Kelas `AgraiDao` merupakan kelas yang berisi metode yang digunakan untuk mengakses database. Dan kelas `AgraiDatabase` merupakan kelas yang digunakan sebagai akses utama ke data relasional yang ada dalam penyimpanan local.

5.5 Perancangan Basis Data

Perancangan basis data dibuat dengan tujuan untuk mengontrol aliran data yang ada pada sistem. Dalam pembuatan aplikasi pendeteksi penyakit pada tanaman cabai ini hanya diperlukan penyimpanan lokal yang ada di *smartphone*. Entitas yang diperlukan ada dua, yaitu AgraiEntity dan PredictionListEntity. Perancangan basis data ditunjukan pada Tabel 5.1 dan Tabel 5.2

Tabel 5.1 Rancangan Basis Data AgraiEntity

No.	Nama Kolom	Tipe Data
1.	id	INTEGER
2.	nama	TEXT
3.	ilmiah	TEXT
4.	pestisida	TEXT
5.	gejala	TEXT
6.	pengendalian	TEXT
7.	img	TEXT

Tabel 5.2 Rancangan Basis Data PredictionListEntity

No.	Nama Kolom	Tipe Data
1.	id	INTEGER
2.	predictionEntities	TEXT
3.	timestamp	TEXT

5.6 Perancangan Algoritme

Perancangan algoritme merupakan cara untuk mempermudah dalam membuat sebuah penyelesaian untuk sebuah masalah. Dengan membuat algoritme akan memberikan gambaran untuk tahap implementasi suatu solusi terhadap masalah. Pada pengembangan aplikasi pendeteksi penyakit pada tanaman cabai ini ada satu algoritme yaitu algoritme mendeteksi penyakit seperti yang ditunjukan pada Tabel 5.3.

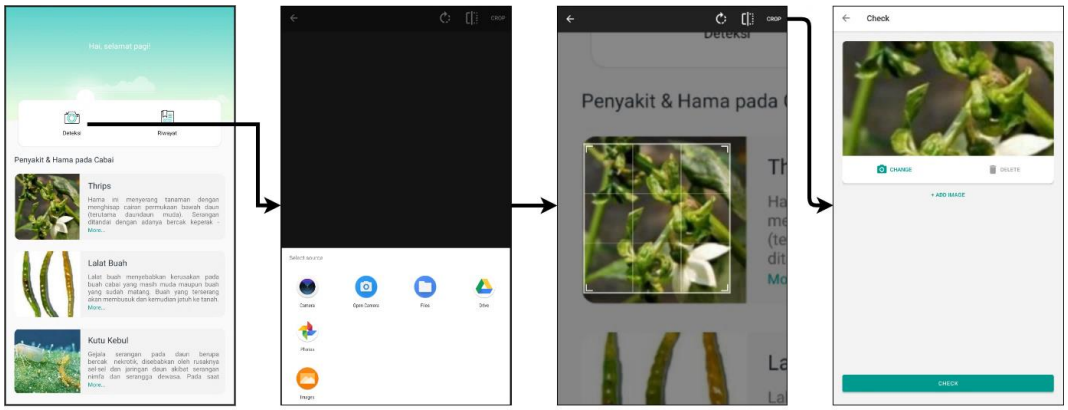
Tabel 5.3 Perancangan Algoritme Mendeteksi Penyakit

```
1   checkImg()
2         image -> File
3         result -> String
4         captureAndCropImage()
5         if(image != null)
6             if(network == true)
7                 result = detection(image)
8             else
9                 show("No network")
10            endif
11        endif
12  end
13  String detection(File image)
14        onResponse(String response)
15            return response
16        onError(Error error)
17            return null
18  end
```

5.7 Perancangan Antarmuka

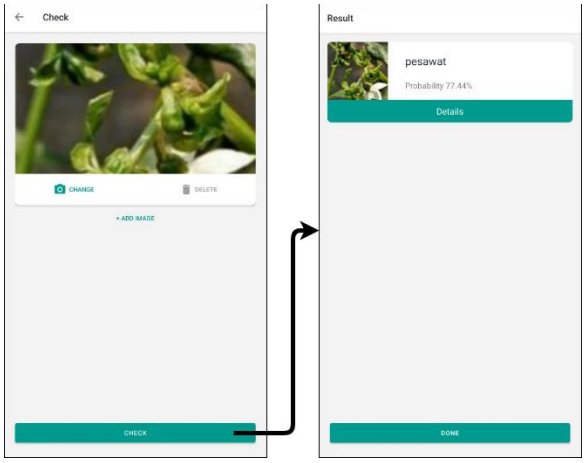
Perancangan antarmuka bertujuan untuk mempermudah dalam pengimplementasian antarmuka. Dengan menggunakan perancangan antarmuka dapat diketahui alur dari setiap perpindahan halaman di aplikasi pendeteksi penyakit pada tanaman cabai.

Pada Gambar 5.10 menunjukkan alur antarmuka dari kebutuhan untuk mengambil gambar. Dimana alur pertama berada pada halaman utama dan berpindah ke halaman mengambil gambar ketika menekan tombol deteksi. Setelah gambar terambil maka berpindah ke halaman untuk memotong gambar. Alur terakhir adalah akan berpindah ke halaman periksa apabila menekan tombol "Crop".



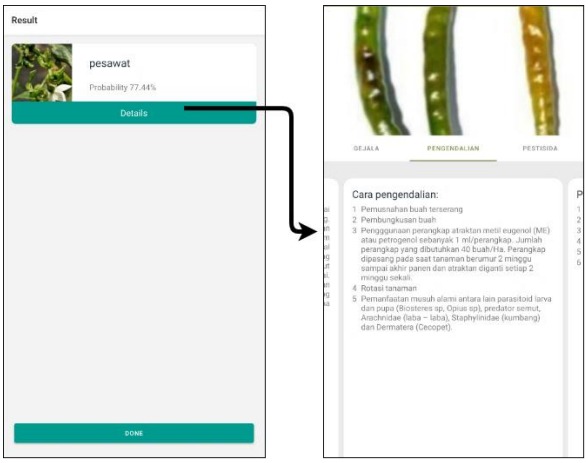
Gambar 5.10 Alur Antarmuka Mengambil Gambar

Pada Gambar 5.11 menunjukkan alur antarmuka untuk kebutuhan mendeteksi penyakit. Pada gambar tersebut terlihat bahwa alur bermula pada halaman periksa dan akan berpindah ke halaman hasil yang menampilkan hasil deteksi penyakit ketika menekan tombol "Check" yang berada pada bawah halaman.



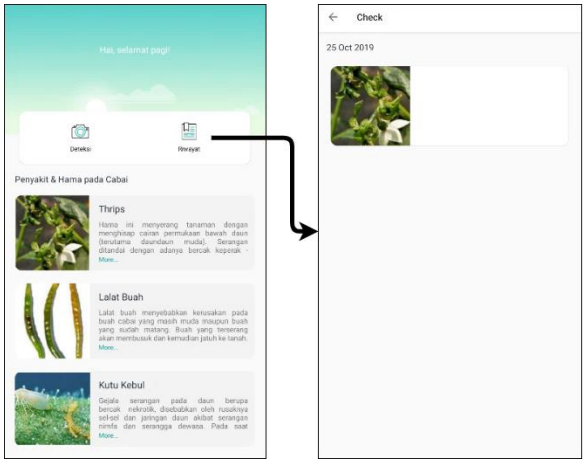
Gambar 5.11 Alur Antarmuka Mendeteksi Penyakit

Pada Gambar 5.12 menunjukkan alur antarmuka untuk kebutuhan melihat informasi penyakit. Terlihat bahwa alur dimulai pada halam hasil dan akan berpindah ke halaman detail yang menampilkan informasi berupa cara pengendalian ketika menekan tombol cara pengendalian yang ada pada tampilan kartu yang ada pada halaman hasil.



Gambar 5.12 Alur Antarmuka Melihat Informasi Penyakit

Pada Gambar 5.13 menunjukkan alur antarmuka untuk kebutuhan melihat riwayat deteksi. Terlihat bahwa alur dimulai pada halaman utama dan akan berpindah ke halaman riwayat yang menunjukkan daftar riwayat deteksi yang pernah dilakukan sebelumnya ketika menekan tombol riwayat yang ada pada halaman utama.



Gambar 5.13 Alur Antarmuka Melihat Riwayat Deteksi

BAB 6 IMPLEMENTASI

Dalam bab ini menjelaskan terkait implementasi sistem atau aplikasi yang meliputi batasan implementasi, spesifikasi perangkat keras dan lunak, implementasi basis data, integrasi Ximilar Custom Image Recognition, implementasi kode program, dan implementasi antarmuka pengguna.

6.1 Batasan Implementasi

Aplikasi pendeteksi penyakit pada tanaman cabai memiliki beberapa batasan implementasi dalam proses pengembangannya. Batasan implementasi tersebut adalah:

1. Aplikasi pendeteksi penyakit pada tanaman cabai hanya mampu berjalan pada sistem operasi Android dengan level API minimal 21.
2. Aplikasi pendeteksi penyakit pada tanaman cabai memerlukan koneksi internet untuk menjalankannya.
3. Aplikasi pendeteksi penyakit pada tanaman cabai ini menggunakan layanan Ximilar Custom Image Recognition sebagai proses deteksi penyakit pada tanaman cabai.
4. Aplikasi dikembangkan dengan menggunakan Android Studio sebagai *Integrated Development Environtment*.

6.2 Spesifikasi Sistem

Spesifikasi sistem menggambarkan spesifikasi lingkungan yang digunakan dalam proses pengembangan aplikasi pendeteksi penyakit pada tanaman cabai. Spesifikasi sistem meliputi dua lingkungan yaitu spesifikasi perangkat keras dan spesifikasi perangkat lunak.

6.2.1 Spesifikasi Perangkat Keras

Pengembangan aplikasi pendeteksi penyakit pada tanaman cabai dilakukan dengan menggunakan laptop ASUS X450J dengan spesifikasi dijelaskan pada Tabel 6.1 dan untuk pengujian aplikasi dilakukan di perangkat bergerak yaitu Sony Xperia XZ Premium dengan spesifikasi dijelaskan pada Tabel 6.2.

Tabel 6.1 Spesifikasi Perangkat Keras Komputer

Komponen	**Spesifikasi**
Model	ASUS X450J
Prosesor	Intel Core i7 4720HQ
Memori	8 GB DDR3L
Penyimpanan	240 GB (SSD) + 1 TB (HDD)
Grafis	NVIDIA GeForce GT 940M 2GB DDR3

Tabel 6.2 Spesifikasi Perangkat Keras Perangkat Bergerak

Komponen	**Spesifikasi**
Model	Sony Xperia XZ Premium (G8142)
Chipset	Qualcomm MSM8998 Snapdragon 835 (10 nm)
Prosesor	Octa-core (4x2.45 GHz Kryo & 4x1.9 GHz Kryo)
Grafis	Adreno 540
Memori	4 GB
Penyimpanan	64 GB
Kamera	19 MP

6.2.2 Spesifikasi Perangkat Lunak

Dalam mengembangkan aplikasi pendeteksi penyakit pada tanaman cabai membutuhkan spesifikasi perangkat lunak yang mendukung proses pengembangan tersebut. Spesifikasi perangkat lunak yang digunakan untuk mengembangan dijelaskan padaTabel 6.3 dan spesifikasi perangkat lunak pada perangkat bergerak yang digunakan dijelaskan pada Tabel 6.4.

Tabel 6.3 Spesifikasi Perangkat Lunak Komputer

Komponen	**Spesifikasi**
Sistem Operasi	Windows 10 Pro
Bahasa Pemrograman	Java
IDE (*Integrated Development Environtment*)	Android Studio 3.5.2
Editor Dokumentasi	Microsoft Office Word 2016
Editor Perancangan	Draw.io

Tabel 6.4 Spesifikasi Perangkat Lunak Perangkat Bergerak

Komponen	**Spesifikasi**
Sistem Operasi	Android 9.0 (Pie)

6.3 Implementasi Basis Data

Basis data yang dikembangkan pada aplikasi pendeteksi penyakit pada tanaman cabai berdasarkan pada perancangan basis data yang telah dibuat sebelumnya. Terdapat 2 tabel yang dibuat dalam implementasi ini, Tabel 6.5 menjelaskan implementasi untuk tabel AgraiEntity dan TABEL menjelaskan

implementasi untuk tabel PredictionListEntity. Kedua tabel tersebut merupakan basis data yang tersimpan pada perangkat bergerak yang menjalankan aplikasi, untuk pengimplementasiannya menggunakan Room sebagai penyimpanan pada aplikasi Android.

Tabel 6.5 Implementasi Basis Data AgraiEntity

No	Tabel AgraiEntity
1	`@Entity`
2	`public class AgraiEntity {`
3	`    @PrimaryKey(autoGenerate = true)`
4	`    private int id;`
5	`    private String nama;`
6	`    private String ilmiah;`
7	`    @TypeConverters(Converters.class)`
8	`    private List<String> pestisida;`
9	`    private String gejala;`
10	`    @TypeConverters(Converters.class)`
11	`    private List<String> pengendalian;`
12	`    private String img;`
13	`    public String getIlmiah() {`
14	`        return ilmiah;`
15	`    }`
16	`    public void setIlmiah(String ilmiah) {`
17	`        this.ilmiah = ilmiah;`
18	`    }`
19	`    public List<String> getPestisida() {`
20	`        if (pestisida == null) {`
21	`            return new ArrayList<>();`
22	`        }`
23	`        return pestisida;`
24	`    }`
25	`    public void setPestisida(List<String> pestisida) {`
26	`        this.pestisida = pestisida;`
27	`    }`
28	`    public String getImg() {`
29	`        return img;`
30	`    }`
31	`    public void setImg(String img) {`
32	`        this.img = img;`
33	`    }`
34	`    public String getNama() {`
35	`        return nama;`
36	`    }`
37	`    public void setNama(String nama) {`
38	`        this.nama = nama;`
39	`    }`
40	`    public String getGejala() {`
41	`        return gejala;`
42	`    }`
43	`    public void setGejala(String gejala) {`
44	`        this.gejala = gejala;`
45	`    }`
46	`    public List<String> getPengendalian() {`
47	`        if (pengendalian == null) {`
48	`            return new ArrayList<>();`
49	`        }`
50	`        return pengendalian;`
51	`    }`
52	`    public void setPengendalian(List<String> pengendalian) {`
53	`        this.pengendalian = pengendalian;`
54	`    }`
55	`    public int getId() {`

Tabel 6.6 Implementasi Basis Data AgraiEntity (lanjutan)

```
56          return id;
57      }
58      public void setId(int id) {
59          this.id = id;
60      }
61  }
```

Tabel 6.7 Implementasi Basis Data PredictionListEntity

No	Tabel PredictionListEntity

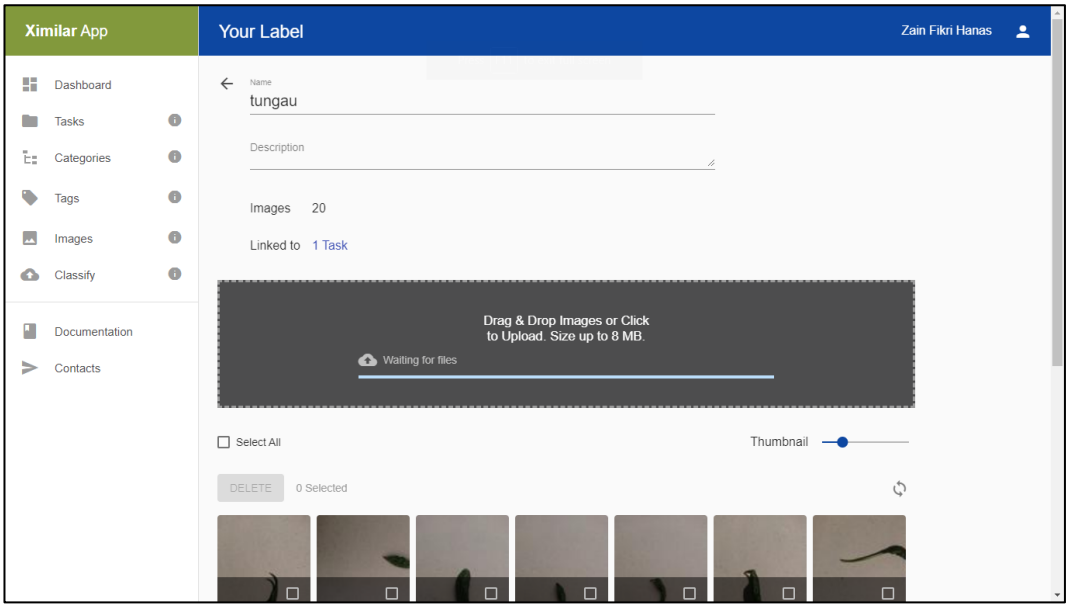
```
1   @Entity
2   public class PredictionListEntity {
3       @PrimaryKey(autoGenerate = true)
4       private int id;
5       @TypeConverters(ConvertersPrediction.class)
6       private List<PredictionEntity> predictionEntities;
7       private String timestamp;
8       public int getId() {
9           return id;
10      }
11      public void setId(int id) {
12          this.id = id;
13      }
14      public List<PredictionEntity> getPredictionEntities() {
15          return predictionEntities;
16      }
17      public   void   setPredictionEntities(List<PredictionEntity>
18  predictionEntities) {
19          this.predictionEntities = predictionEntities;
20      }
21      public String getTimestamp() {
22          return timestamp;
23      }
24      public void setTimestamp(String timestamp) {
25          this.timestamp = timestamp;
26      }
27      public List<String> getLabels() {
28          List<String> labelList = new ArrayList<>();
29          for      (PredictionEntity      predictionEntity      :
30  predictionEntities) {
31              labelList.add(predictionEntity.getLabel());
32          }
33          return labelList;
34      }
35      public List<String> getImages() {
36          List<String> uriList = new ArrayList<>();
37          for      (PredictionEntity      predictionEntity      :
38  predictionEntities) {
39              uriList.add(predictionEntity.getUri());
40          }
41          return uriList;
42      }
43      public List<Double> getProbabilities() {
44          List<Double> probabilityList = new ArrayList<>();
45          for      (PredictionEntity      predictionEntity      :
46  predictionEntities) {
47
48  probabilityList.add(predictionEntity.getProbability());
49          }
50          return probabilityList;
51      }
52  }
```

6.4 Integrasi Ximilar Custom Image Recognition

Aplikasi pendeteksi penyakit pada tanaman cabai ini menggunakan *web service* Ximilar Custom Image Recognition untuk proses pengidentifikasian penyakit. Pada proses integrasinya memerlukan beberapa proses, secara umum ada tiga proses. Yang pertama mendefinisikan kategori atau tanda dan mengunggah gambar, kemudian melatih sistem untuk mengenali kategori atau tanda, dan mengirimkan gambar untuk dikenali oleh sistem yang telah dibuat. Pengintegrasian Ximilar Custom Image Recognition dengan aplikasi pendeteksi penyakit pada tanaman cabai menggunakan API sebagai antarmuka komunikasi.

6.4.1 Mendefinisikan Kategori atau Tanda dan Mengunggah Gambar

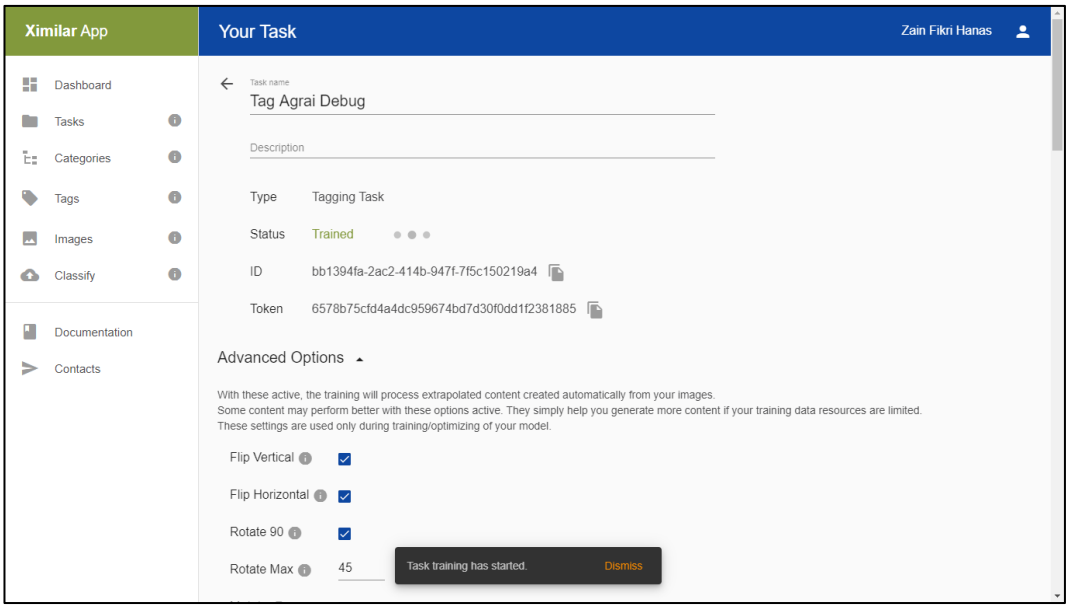
Langkah awal yaitu mendefinisikan kategori atau tanda. Kategori disini digunakan untuk mengelompokan gambar dengan objek yang memiliki satu label. Dan tanda digunakan untuk mengelompokan gambar dengan objek yang memiliki banyak label. Contoh untuk kategori adalah untuk membedakan antara gambar mobil dan pesawat terbang. Dan contoh untuk tanda adalah membedakan warna sepatu. Dalam penelitian ini menggunakan tanda karena objek yang akan dikenali pada penelitian ini adalah daun yang memiliki penyakit berbeda-beda. Kemudian setelah mendefinisikan kategori dilakukan pengunggahan gambar sesuai dengan kategorinya. Pada Gambar 6.1 memperlihatkan proses pendefinisian tanda dan proses pengunggahan gambar.



Gambar 6.1 Mendefinisikan Kategori atau Tanda

6.4.2 Melatih Ximilar Custom Image Recognition

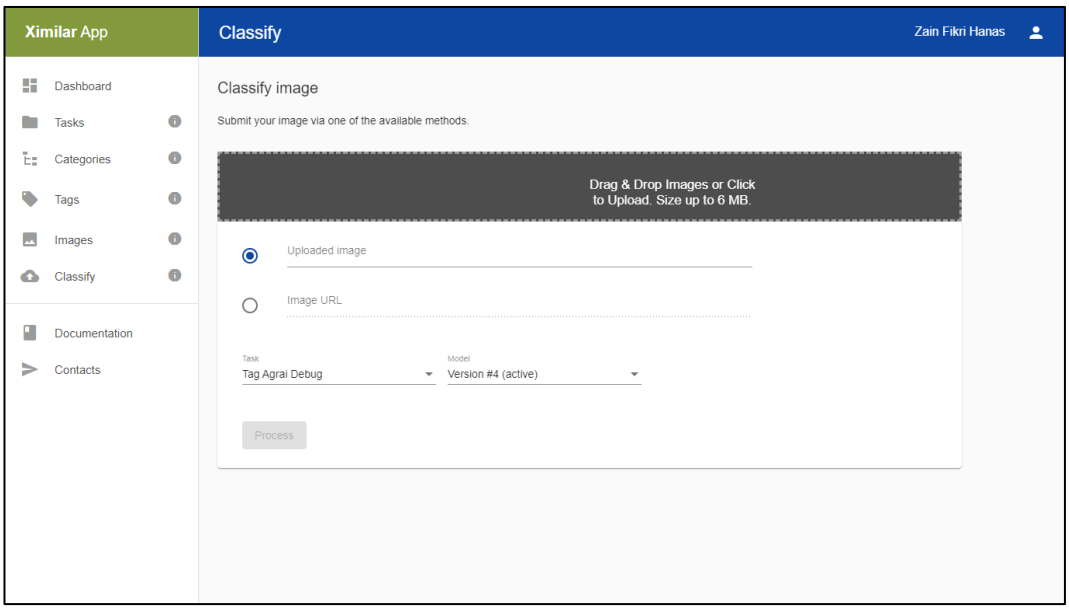
Langkah selanjutnya adalah melatih sistem yang dibuat agar mengenali pola dari setiap label gambar yang telah didefinisikan. Proses ini memerlukan beberapa waktu tergantung banyaknya gambar yang didefinisikan. Pada Gambar 6.2 memperlihatkan proses melatihnya.



Gambar 6.2 Melatih Sistem

6.4.3 Mengirimkan Gambar

Langkah selanjutnya ada melakukan tes ke sistem yang telah dibuat dengan cara mengirimkan gambar yang selanjutnya akan diprediksi sesuai hasil latih sistem. Gambar yang dikirimkan dapat berupa gambar yang diunggah sendiri atau dari alamat gambar lainnya. Gambar 6.3 memperlihatkan proses mengirimkan gambar.



Gambar 6.3 Mengirimkan Gambar

6.5 Implementasi Algoritme

Algoritme yang diimplementasikan berasal dari hasil perancangan algoritme yang telah dilakukan sebelumnya. Algoritme yang diimplementasikan merupakan

fungsi utama pada aplikasi pendeteksi penyakit pada tanaman cabai. Fungsi tersebut adalah mendeteksi penyakit. Pada Tabel 6.8 menjelaskan fungsi mendeteksi penyakit. Pada kode program yang ada pada tabel tersebut fungsi `checkImg()` merupakan fungsi utama yang dipanggil ketika akan mendeteksi gambar. Kemudian akan dilakukan pengecekan koneksi internet dengan menggunakan fungi `isNetworkAvailable()`, jika internet tidak tersedia maka akan ditampilkan pesan bahwa tidak terdapat koneksi internet dan jika internet tersedia maka akan melakukan pemanggilan fungsi `check()` yang selanjutnya akan mengirimkan data ke layanan Ximilar Custom Image Recognition.

Tabel 6.8 Implementasi Algortime Mendeteksi Penyakit

No	Kode Program
1	`private void checkImg() {`
2	`        if (!isNetworkAvailable()) {`
3	`            Toast.makeText(CheckActivity.this, "Tidak koneksi`
4	`internet", Toast.LENGTH_SHORT).show();`
5	`            return;`
6	`        } else {`
7	`            index = 1;`
8	`            layoutLoading.setVisibility(View.VISIBLE);`
9	
10	`loading.setLoadingColor(getResources().getColor(R.color.colorAc`
11	`cent));`
12	`            loading.start();`
13	`            disableTouch();`
14	
15	`            viewModel.check(imageUri).observe(this,`
16	`resultEntity -> {`
17	`                if (resultEntity != null) {`
18	`                    if(resultEntity.size() > 0){`
19	
20	`layoutLoading.setVisibility(View.INVISIBLE);`
21	`                        loading.stop();`
22	
23	`                        Toast.makeText(CheckActivity.this,`
24	`resultEntity.toString(), Toast.LENGTH_LONG).show();`
25	`                        Log.d("Result Prediction",`
26	`resultEntity.toString());`
27	`                        enableTouch();`
28	
29	`                        ArrayList<PredictionEntity>`
30	`predictionEntityList = new ArrayList<>(resultEntity);`
31	`                        Intent intent = new`
32	`Intent(CheckActivity.this, ResultActivity.class);`
33	
34	`intent.putParcelableArrayListExtra(PREDICTION_RESULT,`
35	`predictionEntityList);`
36	`                        startActivity(intent);`
37	`                        finish();`
38	`                    }`
39	`                }`
40	`            });`
41	
42	`            viewModel.progress().observe(this, progress -> {`
43	`                if (progress != null) {`
44	
45	`tvProgress.setText(String.format("%s%s%%s", "Uploading ",`
46	`String.valueOf(progress.intValue()), " (" +`
47	`String.valueOf(index) + ")"));`
48	`                    if (progress.intValue() == 100) {`

Tabel 6.9 Implementasi Algortime Mendeteksi Penyakit (lanjutan)

```
49                          index++;
50                      }
51                  }
52              });
53          }
54      }
55
56  private boolean isNetworkAvailable() {
57          ConnectivityManager        connectivityManager        =
58  (ConnectivityManager)
59  getSystemService(Context.CONNECTIVITY_SERVICE);
60          assert connectivityManager != null;
61          NetworkInfo             activeNetworkInfo             =
62  connectivityManager.getActiveNetworkInfo();
63          return      activeNetworkInfo      !=      null      &&
64  activeNetworkInfo.isConnectedOrConnecting();
65      }
66
67  public void check(Uri uri, CheckCallback callback) {
68
69  AndroidNetworking.post("https://api.ximilar.com/recognition/v2/
70  classify")
71                  .addJSONObjectBody(jsonObject)
72                  .addHeaders("Content-Type", "application/json")
73                  .addHeaders("Authorization",              "Token
74  6578b75cfd4a4dc959674bd7d30f0dd1f2381885")
75                  .setPriority(Priority.HIGH)
76                  .setTag("Ximilar")
77                  .build()
78                  .getAsJSONObject(new
79  JSONObjectRequestListener() {
80                      @Override
81                      public void onResponse(JSONObject response)
82  {
83                          Log.d("XIMILAR", response.toString());
84                          List<PredictionResponse>
85  predictionResponseList = new ArrayList<>();
86
87                          for (int i = 0; i < uriList.size(); i++)
88  {
89                              try {
90                                  PredictionResponse
91  predictionResponse = new PredictionResponse();
92                                  JSONObject     jsonObject     =
93  response.getJSONArray("records").getJSONObject(i).getJSONObject
94  ("best_label");
95
96  predictionResponse.setLabel(jsonObject.getString("name"));
97
98  predictionResponse.setProbability(jsonObject.getDouble("prob"))
99  ;
100 //
101 predictionResponse.setUri(uri.get(i));
102
103 predictionResponse.setUri(uriList.get(i));
104
105 predictionResponseList.add(predictionResponse);
106                             } catch (JSONException e) {
107                                 e.printStackTrace();
108                             }
109                         }
110
111 callback.onSuccess(predictionResponseList);
```

Tabel 6.10 Implementasi Algortime Mendeteksi Penyakit (lanjutan)

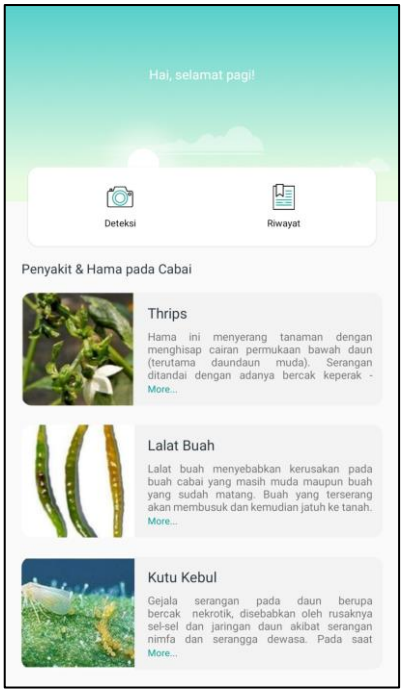
```
112                         }
113
114                         @Override
115                         public void onError(ANError anError) {
116                             callback.onFailure();
117                             Log.e("ERROR", anError.getErrorBody());
118                         }
119                     });
120     }
```

6.6 Implementasi Antarmuka Pengguna

Implementasi antarmuka pengguna berdasarkan rancangan antarmuka yang telah dibuat sebelumnya. Pada penelitian ini menggunakan metode *prototyping* dimana *prototype* yang digunakan saat analisis adalah sistem setengah jadi maka antara rancangan anatarmuka dan implementasi antarmuka tidak berbeda jauh.

6.6.1 Implementasi Antarmuka Halaman Utama

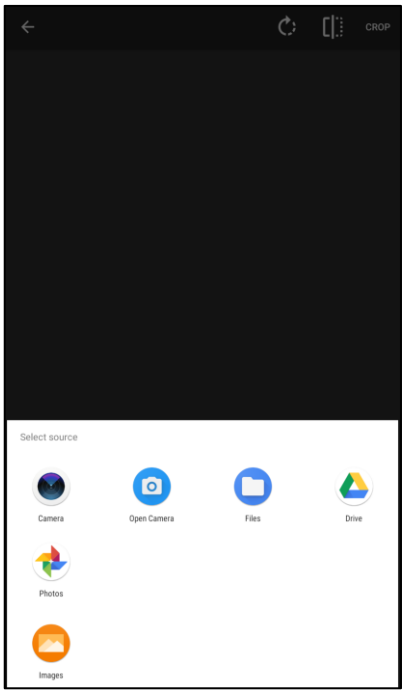
Pada Gambar 6.4 menunjukan halaman utama aplikasi. Dimana terdapat dua tombol yaitu yang pertama tombol untuk mengambil gambar dan tombol yang kedua untuk melihat riwayat deteksi. Dan juga ditampilkan informasi mengenai penyakit pada tanaman cabai.



Gambar 6.4 Implementasi Antarmuka Halaman Utama

6.6.2 Implementasi Antarmuka Halaman Mengambil Gambar

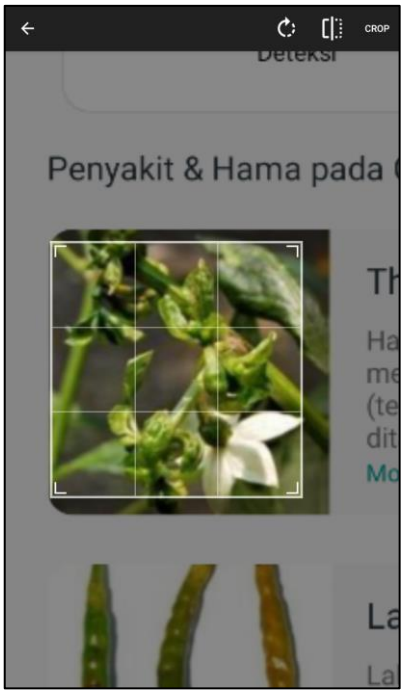
Pada Gambar 6.5 menggambarkan hasil implementasi halaman mengambil gambar. Terlihat bahwa gambar bias diambil melaluli kamera atau penyimpanan local dari perangkat bergerak.



Gambar 6.5 Implementasi Antarmuka Halaman Mengambil Gambar

6.6.3 Implementasi Antarmuka Halaman Memotong Gambar

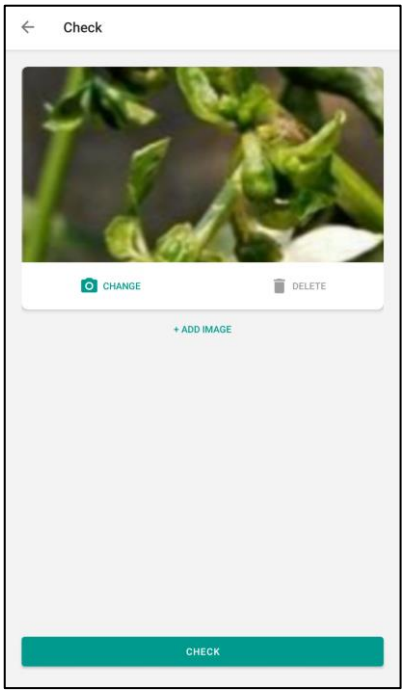
Pada Gambar 6.6 menggambarkan hasil implementasi halaman memotong gambar. Pada halaman tersebut pengguna dapat mengatur gambar sesuai yang diinginkan dan jika sudah sesuai dapat menekan tombol “Crop”.



Gambar 6.6 Implementasi Antarmuka Halaman Memotong Gambar

6.6.4 Implementasi Antarmuka Halaman Mendeteksi Gambar

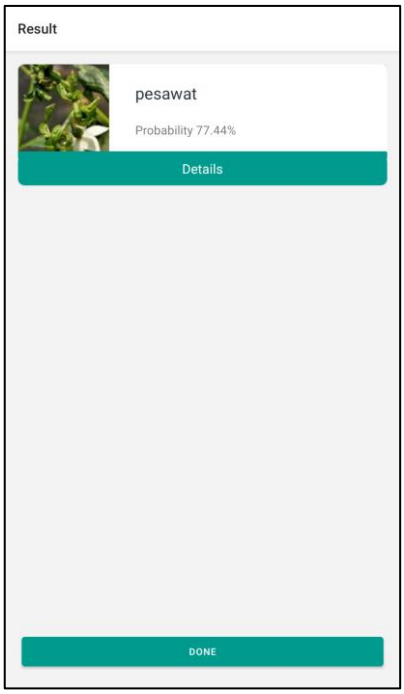
Pada Gambar 6.7 menggambarkan hasil implementasi halaman mendeteksi gambar. Terdapat tombol ubah gambar untuk mengubah gambar yang telah dipilih dan ada tombol menghapus gambar untuk menghapus gambar yang telah dipilih. Terdapat juga tombol untuk menambahkan gambar lain, maksimal 3 gambar. Dan tombol “Check” untuk memulai mengirimkan gambar untuk dideteksi.



Gambar 6.7 Implementasi Antarmuka Halaman Mendeteksi Gambar

6.6.5 Implementasi Antarmuka Halaman Hasil Deteksi Gambar

Pada Gambar 6.8 menggambarkan hasil deteksi gambar. Terdapat nama penyakit hasil prediksi dan akurasinya. Terdapat tombol “Details” untuk melihat informasi penyakit dan tombol “Done” untuk menutup halaman hasil deteksi.



Gambar 6.8 Implementasi Antarmuka Halaman Hasil Deteksi Gambar

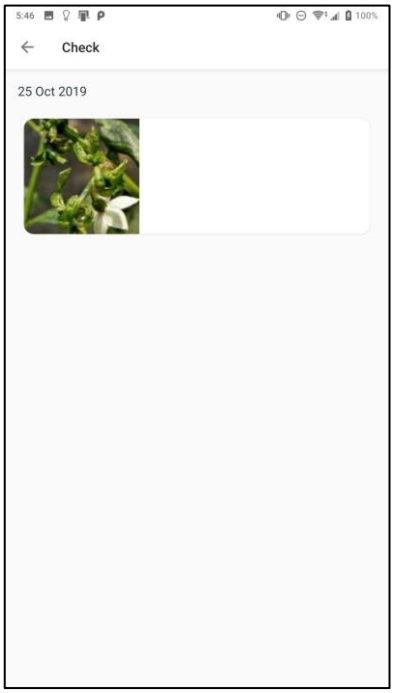
6.6.6 Implementasi Antarmuka Halaman Informasi Penyakit

Pada Gambar 6.9 menggambarkan hasil implementasi halaman informasi penyakit. Terdapat informasi gambar, gejala, cara pengendalian, dan juga pestisida yang sesuai dengan penyakit tersebut.

Gambar 6.9 Implementasi Antarmuka Halaman Informasi Penyakit

6.6.7 Implementasi Antarmuka Halaman Riwayat Deteksi

Pada Gambar 6.10 menggambarkan hasil implementasi halaman riwayat deteksi. Terdapat daftar hasil deteksi penyakit yang dikelompokan sesuai tanggal dilakukannya deteksi. Dan jika ditekan untuk setiap daftar hasil deteksi maka akan membuka halaman hasil deteksi untuk riwayat tersebut.



Gambar 6.10 Implementasi Antarmuka Halaman Riwayat Deteksi

BAB 7 PENGUJIAN

Pengujian aplikasi merupakan langkah yang bertujuan untuk mengetahui apakah aplikasi sudah memenuhi kebutuhan yang telah didapat pada analisis kebutuhan. Pengujian aplikasi pendeteksi penyakit pada tanaman cabai ini meliputi pengujian akurasi, pengujian validasi, pengujian *usability*, dan pengujian *compatibility*. Pengujian aplikasi dilakukan berdasarkan hasil implementasi sistem.

7.1 Pengujian Akurasi

Pengujian akurasi merupakan pengujian yang dilakukan untuk mengetahui keakuratan deteksi dari aplikasi yang telah dibuat. Untuk mengetahui nilai akurasi dari aplikasi pendeteksi penyakit pada tanaman cabai dengan cara mencoba secara langsung aplikasi untuk mendeteksi gambar penyakit.

Pada aplikasi ini terdapat 6 label yang diunggah ke Ximilar Custom Image Recognition berdasarkan hasil pengumpulan data pada Balai Pengkajian Tekonologi Pertanian Jawa Timur. Dan juga terdapat gambar untuk data uji yang telah diambil yaitu sebanyak 77 gambar. Rincian dari jumlah gambar setiap label dan jumlah gambar uji ditunjukan pada Tabel 7.1.

Dilakukan percobaan deteksi terhadap data uji dengan menggunakan aplikasi ini. Dihasilkan 58 gambar dideteksi dengan benar sesuai penyakitnya dan 19 gambar yang hasil deteksinya tidak sesuai. Pada

Tabel **7.2** menunjukan rincian hasil pengujian terhadap data uji. Dari hasil tersebut maka persentasi dari akurasinya adalah 75,32% yang didapatkan dari total gambar yang dideteksi dengan benar dibagi seluruh gambar uji kemudian dikalikan 100.

Tabel 7.1 Jumlah Gambar pada Setiap Label dan Data Uji

Nama Label	Jumlah Gambar	Jumlah Gambar Uji
Bercak Daun	73	18
Busuk Buah Antraknosa	23	5
Pythophthora	103	25
Thrips	46	11
Tungau	20	4
Virus Kuning	57	14

Tabel 7.2 Hasil Pengujian Data Uji

Nama Label	Jumlah Benar	Jumlah Salah
Bercak Daun	11	7

Busuk Buah Antraknosa	5	0

Tabel 7.3 Hasil Pengujian Data Uji (lanjutan)

Pythophthora	21	4
Thrips	8	3
Tungau	3	1
Virus Kuning	10	4

7.2 Pengujian Validasi

Pengujian validasi merupakan pengujian yang dilakukan untuk mengetahui kesesuaian hasil implementasi terhadap kebutuhan yang telah didefinisikan diawal. Pengujian validasi merupakan sebuah pengujian *blackbox*, dimana focus pengujian adalah hasil keluaran aplikasi. Pengujian dianggap valid apabila hasil keluaran aplikasi sesuai dengan yang diharapkan diawal yaitu sesuai dengan kebutuhan yang telah didefinisikan.

7.2.1 Pengujian Validasi Mengambil Gambar

Pada Tabel 7.4 menjelaskan pengujian validasi untuk kasus uji mengambil gambar melalui kamera. Dengan hasil pengujian yaitu valid. Penentuan hasil pengujian berdasarkan hasil yang diekspektasikan terwujud, yaitu gambar berhasil diambil dan menampilkan halaman *crop* selanjutnya sistem menampilkan halaman *check*.

Tabel 7.4 Kasus Uji Mengambil Gambar Melalui Kamera

Nama Kasus Uji	Mengambil gambar melalui kamera
Prosedur	1. Pengguna menekan tombol deteksi 2. Pengguna menekan tombol kamera 3. Pengguna mengambil gambar 4. Pengguna menekan tombol *crop*
Expected Result	Gambar berhasil diambil melalui kamera dan sistem menampilkan halaman *check.*
Actual Result	Gambar berhasil diambil dan menampilkan halaman *crop* selanjutnya sistem menampilkan halaman *check*.
Validasi	Valid

Pada Tabel 7.5 menjelaskan pengujian validasi untuk kasus uji mengambil gambar melalui galeri. Dengan hasil pengujian yaitu valid. Ekspektasi hasil pengujian terwujud dengan hasil gambar berhasil diambil dan menampilkan halaman *crop* selanjutnya sistem menampilkan halaman *check*.

Tabel 7.5 Kasus Uji Mengambil Gambar Melalui Galeri

Nama Kasus Uji	Mengambil gambar melalui galeri
Prosedur	5. Pengguna menekan tombol deteksi 6. Pengguna menekan tombol galeri 7. Pengguna memilih gambar pada galeri 8. Pengguna menekan tombol *crop*
Expected Result	Gambar berhasil diambil melalui kamera dan sistem menampilkan halaman *check.*
Actual Result	Gambar berhasil diambil dan menampilkan halaman *crop* selanjutnya sistem menampilkan halaman *check*.
Validasi	Valid

7.2.2 Pengujian Validasi Mendeteksi Penyakit

Pada Tabel 7.6 menjelaskan pengujian untuk kasus uji mendeteksi penyakit tanpa koneksi internet. Dan hasil aktual dari pengujian tersebut adalah menampilkan pesan “Tidak ada koneksi internet”, maka status validasi pengujian ini adalah valid.

Tabel 7.6 Kasus Uji Mendeteksi Penyakit Tanpa Koneksi Internet

Nama Kasus Uji	Mendeteksi penyakit tanpa koneksi internet
Prosedur	1. Pengguna menekan tombol *check*
Expected Result	Menampilkan pesan tidak terdapat koneksi internet
Actual Result	Menampilkan pesan “Tidak ada koneksi internet”
Validasi	Valid

Pada Tabel 7.7 menjelaskan pengujian dengan kasus uji mendeteksi penyakit dengan internet. Hasil aktual dari pengujian tersebut adalah sistem menampilkan dialog *loading* dan sistem berpindah ke halaman hasil deteksi. Status validasi dari pengujian ini adalah valid.

Tabel 7.7 Kasus Uji Mendeteksi Penyakit dengan Internet

Nama Kasus Uji	Mendeteksi penyakit dengan internet
Prosedur	1. Pengguna menekan tombol *check*
Expected Result	Menampilkan *loading* dan selanjutnya sistem menampilkan halaman hasil deteksi
Actual Result	Menampilkan dialog *loading* dan sistem berpindah ke halaman hasil deteksi.
Validasi	Valid

7.2.3 Pengujian Validasi Melihat Informasi Penyakit

Pada Tabel 7.8 menunjukan pengujian untuk kasus uji melihat informasi penyakit. Hasil aktual dari pengujian adalah menampilkan halaman informasi penyakit yang berisi informasi gejala, pengendalian, dan pestisida mengenai penyakit yang bersangkutan. Nilai validasi untuk pengujian ini adalah valid.

Tabel 7.8 Kasus Uji Melihat Informasi Penyakit

Nama Kasus Uji	Melihat informasi penyakit
Prosedur	1. Pengguna menekan tombol *detail* pada item hasil deteksi
Expected Result	Menampilkan halaman informasi penyakit
Actual Result	Menampilkan halaman informasi penyakit yang berisi informasi gejala, pengendalian, dan pestisida mengenai penyakit yang bersangkutan
Validasi	Valid

7.2.4 Pengujian Validasi Melihat Riwayat Deteksi

Pada Tabel 7.9 menjelaskan pengujian untuk kasus uji melihat riwayat deteksi. Hasil pengujian dari kasus uji ini adalah valid. Dengan hasil aktual adalah menampilkan halaman riwayat deteksi yang berisi daftar hasil deteksi yang pernah dilakukan.

Tabel 7.9 Kasus Uji Melihat Riwayat Deteksi

Nama Kasus Uji	Melihat riwayat deteksi
Prosedur	1. Pengguna menekan tombol riwayat
Expected Result	Menampilkan halaman riwayat deteksi
Actual Result	Menampilkan halaman riwayat deteksi yang berisi daftar hasil deteksi yang pernah dilakukan
Validasi	Valid

7.3 Pengujian *Usability*

Pengujian *usability* merupakan pengujian yang bertujuan untuk menilai subjektif seseorang terhadap kegunaan aplikasi yang telah dibuat dan dilakukan dengan waktu yang singkat (Brooke, 2013). Dalam penelitian ini pengujian *usability* dilakukan dengan intrumen pengujian yaitu SUS (*System Usability Scale*). Karena SUS merupakan sebuah kuesioner yang mana sebuah kuesioner biasanya dijawab dengan cepat maka untuk menghindari bias jawaban maka dilakukan pembuatan pertanyataan positif dan negatif. Hal ini bertujuan agar responden membaca secara baik dari setiap pernyataan. Kuesioner SUS memilki 10 pernyataan yang menggambarkan pengalaman responden dalam menggunakan

aplikasi. Pada Tabel 7.11 merupakan pernyataan yang ada dalam kuesioner SUS. Pengujian ini dimulai dengan memberikan sebuah skenario yang harus dilakukan kepada responden. Kemudian responden mengisi kuesioner yang telah dibuat. Pada Tabel 7.10 menunjukan skenario yang harus dilakukan oleh responden.

Tabel 7.10 Skenario Pengujian *Usability*

No	Skenario
1	Mengambil gambar dari kamera atau galeri 1. Menekan tombol “Deteksi” 2. Memilih untuk mengambil gambar dari kamera atau galeri 3. Mengambil gambar 4. Memotong gambar
2	Mendeteksi penyakit 1. Menekan tombol “Check”
3	Melihat informasi penyakit 1. Menekan tombol “Details” pada hasil deteksi
4	Melihat riwayat deteksi 1. Menekan tombol “Riwayat” 2. Menekan salah satu riwayat penyakit

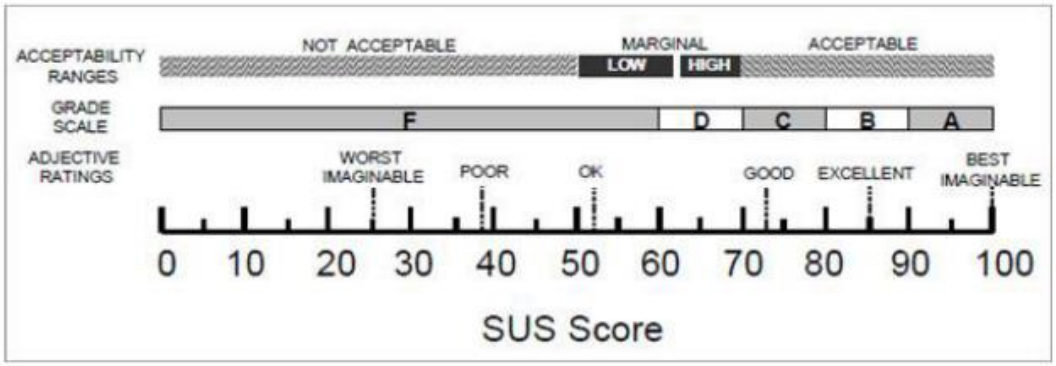
Tabel 7.11 Kuesioner SUS (*System Usability Scale*)

No	Pernyataan	Sangat tidak setuju				Sangat setuju
		1	2	3	4	5
1	Saya merasa bahwa saya akan sering menggunakan aplikasi ini					
2	Saya menemukan kerumitan yang tidak diperlukan dalam aplikasi ini					
3	Saya merasa bahwa aplikasi ini mudah digunakan					
4	Saya memerlukan bantuan orang teknis untuk menjelaskan penggunaan aplikasi ini					

Tabel 7.12 Kuesioner SUS (*System Usability Scale*) (lanjutan)

5	Saya merasa bahwa kebanyakan fungsi dalam aplikasi ini terintegrasi dengan baik					
6	Saya menemukan ketidakkonsistenan yang terlalu banyak dalam aplikasi ini					
7	Saya membayangkan kebanyakan orang akan mudah dalam penggunaan aplikasi ini					
8	Saya merasa bahwa aplikasi ini rumit untuk digunakan					
9	Saya percaya diri dalam menggunakan aplikasi ini					
10	Saya butuh belajar banyak sebelum menggunakan aplikasi ini					

Sistem penilaian yang digunakan untuk menilai hasil kuesioner SUS tersebut adalah dengan cara mengkonversi skor. Mengurangi skor pada pernyataan nomor ganjil dengan 1 dan untuk pernyataan nomor genap dengan cara 5 dikurangi skor pada pernyataan tersebut. Kemudian untuk mendapatkan nilai SUS secara keseluruhan adalah dengan menjumlahkan semua skor hasil konversi dan dikalikan dengan 2,5.



Gambar 7.1 Kategori Nilai SUS (*System Usability Scale*)

Hasil yang didapat dari penilaian SUS secara keseluruhan adalah 0 sampai 100. Kemudian dilakukan pengkategorian hasil nilai SUS. Pada Gambar 7.1 menunjukan pembagian kategori untuk hasil nilai SUS. Berdasarkan pembagian kategori tersebut maka pengkategorian pada pengujian *usability* penelitian ini seperti pada Tabel 7.13.

Tabel 7.13 Kategori Skor SUS (*System Usability Scale*)

Skor	Kategori
>85	Sangat Baik
73-85	Baik
52-72	Cukup
39-51	Buruk
<39	Sangat Buruk

Pada Tabel 7.14 merupakan tabel hasil konversi skor SUS yang didapat pada pengujian *usability* aplikasi pendeteksi penyakit pada tanaman cabai ini. Terdapat 5 responden dan juga skor dari setiap responden untuk setiap pernyataan yang diberikan. Diakhir tabel terdapat nilai keseluruhan SUS yang didapat dari jumlah seluruh skor setiap pernyataan dari seluruh responden dikalikan dengan 2,5.

Tabel 7.14 Konversi Skor Pengujian *Usability* dengan SUS

Nomor Pernyataan	Nama Responden				
	Sugianto	Parmodianto	Panut	Purwanto	Marsono
1	4	4	3	4	3
2	3	4	4	4	3
3	4	4	4	4	4
4	4	3	1	2	3
5	3	3	3	4	4
6	3	3	3	1	2
7	4	3	3	4	3
8	3	3	3	4	4
9	3	2	2	2	3
10	2	3	2	1	2
Total	33	32	28	30	31
Dikali dengan 2,5	82,5	80	70	75	77,5
Rata-rata	77				

7.4 Pengujian *Compatibility*

Pengujian *compatibility* merupakan pengujian yang bertujuan untuk mengetahui apakah aplikasi dapat berjalan pada perangkat bergerak dengan sistem operasi Android yang berbeda level API. Pada penelitian ini pengujian *compatibility* dilakukan pada perangkat bergerak Android level API 21 sampai 29 dengan menggunakan bantuan layanan Test Lab dari Firebase. Pada Tabel 7.15 merupakan hasil pengujian *compatibility* yang dilakukan dengan layanan Test Lab dari Firebase.

Tabel 7.15 Pengujian *Compatibility*

	Lulus	**Tidak Lulus**
Level API 21	√	
Level API 22	√	
Level API 23	√	
Level API 24	√	
Level API 25	√	
Level API 26	√	
Level API 27	√	
Level API 28	√	
Level API 29	√	

7.5 Analisis Hasil Pengujian

Analisis hasil pengujian yang telah dialukan pada aplikasi pendeteksi penyakit pada tanaman cabai meliputi pengujian validasi, pengujian *usability*, dan pengujian *compatibility*.

7.5.1 Analisis Hasil Pengujian Akurasi

Hasil pengujian akurasi pada aplikasi pendeteksi penyakit pada tanaman cabai dengan menggunakan Ximilar Custom Image Recognition ini adalah 75,32%. Berdasarkan 6 label yang telah diunggah pada Ximilar Custom Image Recognition dan 77 gambar uji. Keakurasian dari aplikasi ini tergolong belum akurat. Hal ini disebabkan karena gambar latih yang diunggah kurang banyak dan belum cukup mewakili semua jenis penyakit.

7.5.2 Analisis Hasil Pengujian Validasi

Hasil pengujian validasi pada aplikasi ditunjukan pada Tabel 7.4 sampai Tabel 7.9. Kasus uji yang dilakukan berdasarkan kebutuhan yang telah didefinisikan. Banyaknya kasus uji berdasarkan kemungkinan yang ada pada setiap hasil

implementasi dari kebutuhan yang ada. Dari 6 kasus uji yang dilakukan memiliki nilai validasi yang valid semua.

7.5.3 Analisis Hasil Pengujian *Usability*

Hasil pengujian *usability* yang didapat terhadap 5 responden dengan menggunakan instrument kuesioner SUS (*System Usability Scale*) pada aplikasi pendeteksi penyakit pada tanaman cabai ini menunjukan nilai rata-rata yaitu 77. Nilai tersebut dikategorikan baik.

7.5.4 Analisis Hasil Pengujian *Compatibility*

Hasil pengujian *compatibility* yang dilakukan pada aplikasi dengan menggunakan Firebase Test Lab ditunjukan pada Tabel 7.15. Dengan pengujian pada perangkat Android level API 21 sampai 29. Dari pengujian terhadap 9 perangkat Android tersebut didapatkan hasil lulus pada semua perangkat.

BAB 8 PENUTUP

Bagian ini berisi kesimpulan mengenai penelitian pengembangan aplikasi pendeteksi penyakit pada tanaman cabai dengan menggunakan Ximilar Custom Image Recognition dan saran untuk penelitian selanjutnya.

8.1 Kesimpulan

Berdasarkan tahap pengembangan aplikasi pendeteksi penyakit pada tanaman cabai yang meliputi tahap analisi kebutuhan, perancangan, implementasi, dan pengujian, didapatkan kesimpulan sebagai berikut:

1. Proses analisis kebutuhan awal pada pengembangan aplikasi pendeteksi penyakit pada tanaman cabai didapatkan berdasarkan studi pustaka mengenai sistem yang mirip yang sudah dikembangkan dan proses iterasi penggalian kebutuhan dengan metode *prototyping* dengan cara melakukan wawancara kepada pakar penyakit dan hama tanaman di Balai Pengkajian Teknologi Pertanian Jawa Timur didapatkan 4 kebutuhan fungsional dan 2 kebutuhan non-fungsional. Kebutuhan fungsional tersebut berdasarkan iterasi pertama *prototyping* yang menghasilkan 3 kebutuhan fungsiobal dan iterasi kedua yang menghasilkan 1 kebutuhan fungsional.
2. Integrasi Ximilar Custom Image Recognition dengan aplikasi pendeteksi penyakit pada tanaman cabai menggunakan API yang telah disediakan oleh *web service* Ximilar Custom Image Recognition. Untuk tahapan integrasi meliputi mendefinisikan tanda dan mengunggah gambar latih, melatih Ximilar Custom Image Recognition berdasarkan gambar latih yang sudah diunggah, mengirimkan gambar sebagai uji coba dari sistem yang sudah dibuat dengan gambar latih yang ada.
3. Aplikasi pendeteksi penyakit pada tanaman cabai ini memiliki akurasi sebesar 75,32% berdasarkan pengujian gambar uji sebanyak 77 gambar. Akurasi pada aplikasi ini tergolong belum akurat disebabkan jumlah gambar yang ada belum mampu mewakili untuk setiap jenis penyakit tanaman cabai. Hasil dari pengujian *usability* pada aplikasi pendeteksi penyakit pada tanaman cabai tehadap 5 responden dikategorikan baik dengan nilai rata-rata 77. Untuk pengujian *compatibility* yang dilakukan dengan menggunakan Firebase Test Lab menghasilkan lulus pada semua perangkat yang diujikan.

8.2 Saran

Berdasarkan tahapan penelitian yang telah dilakukan, terdapat beberapa saran yang diberikan untuk pengembangan sistem yang serupa selanjutnya, antara lain:

1. Jenis penyakit yang lain pada tanaman cabai perlu ditambahkan, sehingga mampu mendeteksi semua penyakit pada tanaman cabai.

2. Proses pengambilan gambar latih dapat dilakukan dengan cara lain. Misalnya dengan alat bantu pencahayaan yang cukup agar lebih baik kualitasnya.
3. Perlu diteliti lebih lanjut dengan menambahkan jumlah gambar latih yang lebih banyak untuk mengetahui hasil akurasinya.

DAFTAR REFERENSI

Brooke, J. (1986). SUS - A quick and dirty usability scale. *Usability Evaluation in Industry*. Retrieved from https://cui.unige.ch/isi/icle-wiki/_media/ipm:test-suschapt.pdf

Brooke, J. (2013). SUS: A Retrospective. *Journal of Usability Studies*, *8*(2), 29–40.

Developers, A. (2019a). Android Architecture Components. Retrieved September 27, 2019, from https://developer.android.com/topic/libraries/architecture/

Developers, A. (2019b). Platform Architecture. Retrieved July 16, 2019, from Google Developers website: https://developer.android.com/guide/platform

Developers, A. (2019c). Save data in a local database using Room. Retrieved July 24, 2019, from https://developer.android.com/training/data-storage/room/index.html

Developers, A. (2019d). Understand the Activity Lifecycle. Retrieved July 24, 2019, from https://developer.android.com/guide/components/activities/activity-lifecycle

Developers, A. (2019e). ViewModel Overview. Retrieved September 27, 2019, from https://developer.android.com/topic/libraries/architecture/viewmodel

DitjenHorti (Direktorat Jendral Hortikultura). (2015). *Statistika Produksi Hortikultura Tahun 2014*. Jakarta: Kementerian Pertanian.

Firebase. (2019a). Cloud Storage. Retrieved December 8, 2019, from https://firebase.google.com/docs/storage?authuser=0

Firebase. (2019b). Firebase Test Lab Robo tests. Retrieved October 1, 2019, from https://firebase.google.com/docs/test-lab/android/robo-ux-test

JSON. (2019). Introducing JSON. Retrieved September 15, 2019, from https://www.json.org/

Jupiyandi, S., Kharisma, A. P., & Triwiratno, A. (2019). *Pengembangan Aplikasi Perangkat Bergerak Identifikasi Penyakit Daun Jeruk Berbasis Android dengan Memanfaatkan Vize AI*. *3*(4), 3993–3998.

Kementerian Pertanian. (2016). Outlook Komoditas Pertanian Sub Sektor Hortikultura Cabai Merah. *Outlook Komoditas Pertanian Sub Sektor Hortikultura Pusat Data Dan Informasi Pertanian*, 1–92.

Meilin, A. (2014). Hama dan Penyakit pada Tanaman Cabai serta Pengendaliannya. In *Balai Pengkajian Teknologi Pertanian Jambi* (Vol. 1).

Muslim, A. A. dkk. (2015). Sistem Pakar Diagnosa Hama Dan Penyakit Cabai Berbasis Teorema Bayes. *Jutisi*, *4*(3), 867–876.

Nielsen, J. (2004). Card Sorting: How Many Users to Test. Retrieved from https://www.nngroup.com/articles/card-sorting-how-many-users-to-test/

Pressman, R. S. (2010). Software Engineering A Practitioner's Approach 7th Edition. In *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman* (7th ed.). https://doi.org/10.1017/CBO9781107415324.004

Ralahalu, M. A. (2013). *Jurnal budidaya tanaman* (Vol. 2). Vol. 2.

Sherrell, L. (2013). Evolutionary Prototyping. In *Encyclopedia of Sciences and Religions*. https://doi.org/10.1007/978-1-4020-8265-8_201039

Sommerville, I. (2016). *Software Engineering: Global Edition, Tenth Edition*. https://doi.org/10.1136/bmj.1.5802.756-b

Techinasia. (2018). Jumlah Pengguna Smartphone di Indonesia 2018. Retrieved July 15, 2019, from https://d26bwjyd9l0e3m.cloudfront.net/wp-content/uploads/2014/12/Indonesia-to-be-worlds-fourth-largest-smartphone-by-2018-surpass-100-million-users-chart1.jpg

Wibowo, A. P. W. (2017). Penerapan Teknik Computer Vision Pada Bidang Fitopatologi Untuk Diteksi Penyakit dan Hama Tanaman Cabai. *Jurnal Informatika: Jurnal Pengembangan IT Poltek Tegal*, *2*(2), 102–108. Retrieved from http://ejournal.poltektegal.ac.id/index.php/informatika/article/view/528

Wijaya, E. H., & Hidayat, N. (2018). *Diagnosis Penyakit Cabai Dengan Menggunakan Metode Forward Chaining – Dempster-Shafer*. *2*(12), 7202–7208.

Ximilar. (2019a). Custom Image Recognition (previously Vize.ai). Retrieved December 8, 2019, from https://docs.ximilar.com/services/recognition/

Ximilar. (2019b). Image Recognition. Retrieved December 8, 2019, from https://www.ximilar.com/services/image-recognition/

Zhang, T., Gao, J., Cheng, J., & Uehara, T. (2015). Compatibility testing service for mobile applications. *Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015*, *30*(April), 179–186. https://doi.org/10.1109/SOSE.2015.35

LAMPIRAN A HASIL ITERASI PROTOTYPE

LAMPIRAN B SKENARIO *USABILITY TESTING*

LAMPIRAN C HASIL KUISIONER *USABILITY TESTING*