PENGEMBANGAN SISTEM INFORMASI MANAJEMEN TERNAK BURUNG LOVEBIRD BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Akhsana Zufar Masyhuda
NIM: 155150207111027

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENGEMBANGAN SISTEM INFORMASI MANAJEMEN TERNAK BURUNG LOVEBIRD BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Akhsana Zufar Masyhuda

NIM: 155150207111027

Skripsi ini telah diuji dan dinyatakan lulus pada

18 Juli 2019

Telah diperiksa dan disetujui oleh:

Pembimbing I	Pembimbing II
Nurudin Santoso, S.T., M.T.	Edy Santoso, S.Si, M.Kom.
NIP: 19740916 200012 1 001	NIP: 19740414 200312 1 004

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 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 pustaka.

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, 18 Juli 2019

Akhsana Zufar Masyhuda

NIM: 155150207111027

KATA PENGANTAR

Puji syukur saya panjatkan kehadirat Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga laporan skripsi yang berjudul "Pengembangan Sistem Informasi Ternak Burung *Lovebird* Berbasis Android" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terimakasih kepada:

1. Bapak Nurudin Santoso, S.T., M.T., dan Bapak Edy Santoso , S.Si., M.Kom., selaku Pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini,
2. Bapak Agus Wahyu Widodo, S.T., M.Cs., selaku Ketua Program Studi Teknik Informatika,
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D., selaku Ketua Jurusan Teknik Informatika,
4. Adam Hendra Brata, S.Kom., M.T., M.Sc., selaku dosen Penasihat Akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi,
5. Ayahanda dan Ibunda dan seluruh keluarga besar atas segala nasihat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti-hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini,
6. Seluruh civitas academica Informatika Universitas Brawijaya yang telah dan banyak memberi bantuan dan dukungan selama penulis menempuh studi di Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 18 Juli 2019

Penulis

Email: zufarr@ub.ac.id

ABSTRAK

Lovebird merupakan salah satu jenis burung beo yang memiliki warna bulu yang menarik, selain itu suaranya yang unik juga menjadi salah faktor yang membuat burung yang berasal dari afrika ini memiliki nilai jual yang lumayan tinggi dan beberapa orang tertarik untuk melakukan budidaya burung *lovebird*. Saat ini para peternak burung belum menyadari pentingnya melakukan proses *recording* pada burung ternaknya. Hal tersebut dapat mengakibatkan informasi data burung yang diberikan kepada calon pembeli menjadi tidak akurat. Permasalah lain yang sering muncul dalam bidang *aviculture* adalah para peternak masih mencatat segala sesuatu terkait dengan keuangan masih dengan cara yang manual. Solusi yang diberikan adalah dengan disediakan layanan aplikasi portal yang memudahkan para peternak burung *lovebird*. Pada aplikasi portal terdapat *subsystem* bernama *Bird farm management system* yang dapat membantu para peternak untuk melakukan pendataan dan segala sesuatu yang berkaitan dengan kegiatan aktivitas ternak burung baik dari sisi pendataan burung-burung ternaknya maupun dari sisi keuangan yang digunakan selama proses budidaya burung berlangsung. Dalam penelitian ini pengembangan sistem dilakukan menggunakan metode *Rapid Application Development* pengggunaan metode RAD sangat tepat untuk mengembangkan aplikasi dengan waktu yang relative singkat. Sistem yang dikembangkan menggunakan pendekatan berorientasi objek. Pada penelitian ini sistem yang dikembangkan berjalan pada *platform android* untuk mempermudah para peternak melakukan proses *recording* dimanapun dan kapanpun. Dalam penelitian ini terdapat 3 faktor yang diuji yaitu: validasi, unit, dan *usability*. Pengujian validasi menggunakan metode *blackbox* menghasilkan 100% valid. Pengujian unit menggunakan *metode whitebox* menghasilkan 100% valid. Sedangkan untuk menguji *usability* menggunakan metode *post-study system questionnaire* mendapatkan poin secara keseluruhan 0.94 yang menandakan bahwa sistem dapat diandalkan.

Kata kunci: Sistem Informasi, Android, *Rapid Application Development*, *Post-study System Questionnaire.*

ABSTRACT

Lovebird is one kind of parrot that has an attractive feather color and distintinctive sound. That's make birds form Africa have a fairly high selling value and some people are interested doing lovebird bird cultiviation. Nowadays, the bird breeders have not yet realized the importance of recording on cultivated bird. This can result in information on bird data given to prospective buyers to be inaccurate. Another problem that often arises in the field of aviculture is that farmers still record everything related to finance still in a manual way. The solution for those problems is by providing a portal application service that makes it easy for lovebird breeders. In the portal application, there is a subsystem called the Bird farm management system that can help farmers to collect data and everything related to activities of cultivation activities both in terms of data collection of cultivated birds and from the financial side used during the bird cultivation process. In this research, system development was carried out using the Rapid Application Development method. The use of the RAD method is very appropriate for developing applications with a relatively short time. The system developed using an object oriented approach. In this research, there were 3 factors tested validation, unit, and usability. Validation testing using the blackbox method produces 100% valid. Testing units using the whitebox method produces 100% valid. Whereas to test usability using the post-study system questionnaire method to get overall points 0.94 which indicates that the system is reliable.

Keywords: *information system, Android, Rapid Application Development, Post-study System Questionnaire.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Lovebird merupakan salah satu jenis burung beo yang berasal dari wilayah afrika dan madagaskar yang berukuran relatif kecil. Burung ini memiliki daya tarik yang berasal dari warna bulunya yang mencolok, burung ini juga memiliki sifat yang tergolong aktif. Dengan adanya keunikan yang terdapat dalam burung *lovebird* banyak orang tertarik untuk memelihara burung tersebut, selain itu *lovebird* juga dibudidayakan dan dilakukan mutasi pada warna bulu untuk menghasilkan warna burung yang lebih menarik (David, 2003). Budidaya burung biasanya dikenal dengan istilah *Aviculture* memiliki potensi untuk dapat dimanfaatkan sebagai wirausaha. *Lovebird* sendiri dapat dibudidayakan dan dijadikan sebagai potensi usaha dengan menggunakan keindahan yang terdapat pada warna bulunya yang dapat dimanfaatkan sebagai daya tarik pembeli.

Pada era modern, perkembangan teknologi berkembang dengan sangat pesat. *Smartphone* merupakan salah satu inovasi yang memberikan banyak manfaat bagi penggunanya, salah satu manfaat *smartphone* digunakan untuk mendapatkan dan menyebarkan informasi (Choudrie et al., 2014). Kemudahan dalam mengangkses informasi darimanapun dan kapanpun serta ukurannya yang mudah untuk dibawa menjadikan *smartphone* menjadi salah satu *gadget* yang digunakan untuk membantu mempermudah kegiatan sehari-hari (Wang, Xiang and Fesenmaier, 2016). Sosial media merupakan salah satu produk dari pesatnya pertumbuhan teknologi informasi yang biasanya dimanfaatkan untuk mendapatkan, bertukar dan menyebarluaskan informasi (Mulawarman and Nurfitri, 2017). Hal tersebut dapat dimanfaatkan oleh peternak burung *lovebird* untuk dapat mengakses informasi-informasi baru yang berkaitan tentang budidaya *lovebird*. Selain itu para peternak *lovebird* dapat memasarkan burung *lovebird* yang ingin mereka jual melalui sosial media. Akan tetapi, penggunaan sosial media sebagai sarana bertukar informasi bukan merupakan hal yang kurang tepat. Karena selain cakupannya terlalu luas, keterbatasan ruang yang terdapat pada sosial media juga mengakibatkan informasi tentang burung yang akan dijual menjadi tidak lengkap dan tidak akurat.

Informasi data burung yang tidak akurat disebabkan oleh para peternak yang bergerak dalam bidang *aviculture* tidak melakukan identifikasi atau *recording* secara jelas pada burung-burung yang dibudidayakan. *Recording* merupakan salah satu aspek yang dapat mempermudah para peternak dalam melakukan hewan ternaknya, dengan adanya identifikasi terhadap masing-masing individu hewan ternak akan membuat para peternak dapat mudah dalam menentukan sikap atau aksi yang akan dilakukan terhadap hewan ternaknya (G.L.M. Chappell, 2006). Saat ini belum terdapat *platform* khusus yang menjadi pedukung untuk para pecinta maupun peternak burung *lovebird* untuk dapat mempermudah melakukan proses *recording*.

Permasalah lain yang sering muncul dalam bidang *aviculture* adalah para peternak masih mencatat segala sesuatu terkait dengan keuangan masih dengan cara yang manual. Pencatatan secara manual memiliki peluang lebih besar terjadinya kesalahan selama proses pencatatan transaksi. Dengan cara pencatatan yang manual, para peternak juga akan mengalami kesulitan apabila ingin melihat transaksi yang terjadi pada bulan atau tahun tertentu. Hal tersebut dinilai tidak efektif dan efisien sehingga dapat memperlambat jalannya proses bisnis.

Berdasarkan dengan adanya permasalahan diatas menjadikan peluang yang sangat besar untuk disediakan layanan aplikasi portal yang memudahkan para peternak burung *lovebird*. Pemanfaatan aplikasi portal dalam komunitas dapat mendorong para penggunanya berkontribusi, dan merupakan wadah khusus bagi para penggunanya untuk saling berinteraksi (Kondratova et al., 2015). Dalam aplikasi portal tersebut terdiri dalam dua bagian yaitu *bird farm management system* dan *marketplace. Bird farm management system* merupakan suatu sistem yang dapat membantu para peternak untuk melakukan pendataan dan segala sesuatu yang berkaitan dengan kegiatan aktivitas ternak burung baik dari sisi pendataan burung-burung ternaknya maupun dari sisi keuangan yang digunakan selama proses budidaya burung berlangsung. Pendataan burung sendiri dilakukan dengan cara mendaftarkan nomor unik untuk setiap jenis burung yang ada serta melengkapi data burung seperti spesies, tipe burung, jenis kelamin, umur, foto burung, foto dna, serta data induknya. Penomoran ini dilakukan agar setiap hewan yang ada memiliki *record* yang jelas dan tidak tertukar antara satu dengan yang lainnya (G.L.M. Chappell, 2006). Selain pendataan kepada setiap burung yang ada, dalam *bird farm management system* juga melakukan pendataan pada burung yang akan dibudidayakan, data tersebut berisikan data induk burung yang akan dibudidayakan dan aktivitas yang terjadi selama proses budidaya burung berlangsung. Untuk memudahkan para peternak *lovebird* dalam mengelola keungan selama budidaya burung maka dalam *bird farm management* sistem akan ditambahkan fitur yang dapat membuat laporan keuangan yang berisikan segala transaksi pengeluaran/pemasukan yang terjadi dalam kurun waktu tertentu.

Dengan ini maka dilakukan penelitian dengan judul "Pengembangan Aplikasi Manajemen Ternak Burung Berbasis Android", sehingga dapat membantu para peternak *lovebird* dalam mengelola budidaya burung.

1.2 Rumusan masalah

berlandaskan masalah yang dipaparkan pada latar belakang, maka akan dibuat rumusan masalah dalam penilitian ini sebagai berikut:

1. Apa saja kebutuhan yang terdapat dalam pengembangan sistem informasi manajemen ternak burung *lovebird* berbasis *Android*?
2. Bagaimana hasil rancangan dari sistem informasi manajemen ternak burung *lovebird* berbasis *Android* dari hasil yang didapatkan pada saat proses analisis kebutuhan?

3. Bagaimana hasil implementasi dari sistem informasi manajemen ternak burung *lovebird* berbasis *Android*?
4. Bagaimana hasil pengujian dari sistem informasi manajemen ternak burung *lovebird* berbasis *Android*?

1.3 Tujuan

Terdapat beberapa tujuan yang ingin dicapai dari penelitian ini, hal tersebut adalah sebagai berikut:

1. Mengetahui kebutuhan yang terdapat dalam pengembangan sistem informasi.
2. Mengetahui hasil rancangan yang berdasarkan pada hasil analisis kebutuhan yang telah dilakukan.
3. Mengetahui hasil implementasi berdasarkan pada hasil rancangan yang telah dilakukan.
4. Mengetahui kualitas dari sistem informasi yang dinilai dari aspek kemudahan dan kesusaian sistem dengan kebutuhan.

1.4 Manfaat

1. Untuk penulis, memberikan pengetahuan dan pengalaman baru dalam tentang proses pengembangan aplikasi berbasis *android* yang dapat digunakan dalam membantu pekerjaan orang lain.
2. Pemilik budidaya burung *lovebird* dapat dengan mudah mengatur segala sesuatu hal yang berkaitan dengan manajemen budidaya burung *lovebird*.

1.5 Batasan masalah

Dalam penelitian ini untuk dapat membatasi ruang lingkup atau cakupan penelitan maka terdapat beberapa batasan masalah, diantaranya:

1. Data yang digunakan dalam pengujian merupakan data *dummy*, agar kerahasiaan informasi tentang *user* dapat dilindungi
2. Sistem informasi yang dikembangkan berjalan pada *smartphone* dengan OS *android* yang memiliki versi OS 6.0(*Marshmallow*) keatas.

1.6 Sistematika pembahasan

Sistematika pembahasan merupakan urutan langkah yang dilakukan selama proses penelitian berlangsung untuk mendapatkan hasil yang sesuai dengan penelitian, yaitu :

1. Bab I PENDAHULUAN

Pada Bab I berisikan terkait latar belakang, identifikasi masalah, rumusan masalah, tujuan, manfaat penelitian, ruang lingkup, dan sistematika pembahasan pada penelitian.

2. Bab II LANDASAN KEPUSTAKAAN

Pada Bab II berisikan rincian terkait kajian pustaka, dasar-dasar teori yang menjadi dasar selama proses penulisan dan penelitian.

3. Bab III METODOLOGI PENELITIAN

Pada Bab III menjelaskan terkait metode yang digunakan untuk pengumpulan data dan diagram alir yang merupakan tahapan-tahapan yang dilakukan selama proses penelitian.

4. Bab IV ANALISIS KEBUTUHAN

Pada Bab IV menjelaskan terkait proses mendapatkan kebutuhan (*fungsional dan non-fungsional*) dan *stackholder* yang terdapat dalam sistem.

5. Bab V PERANCANGAN DAN IMPLEMENTASI

Pada Bab V memberikan rincian proses perancangan dalam pengembangan sistem yang mengacu pada hasil analisis kebutuhan. Hasil yang didapatkan dari proses perancangan akan dijadikan dasar untuk tahapan implementasi.

6. Bab VI PENGUJIAN DAN ANALISIS

Pada Bab VI membahas tentang pengujian kualitas sistem yang dinilai dari aspek *usability*, *validasi*, dan *unit*.

7. Bab VII PENUTUP

Pada Bab VII ini berisi tentang kesimpulan yang merupakan jawaban dari pertanyaan yang terdapat pada rumusan masalah dan saran yang diberikan oleh peneliti untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada tahun 2010 Suryati Putri melakukan penelitian dengan judul "Pembangunan Sistem Informasi Pendataan Rakyat Miskin untuk Program Beras Miskin (Raskin) Pada Desa Mantren Kecamatan Kebonagung Kabupaten Pacitan". Latar belakang dari penelitian ini adalah pengelolaan data yang masih dilakukan secara manual, memiliki peluang timbul masalah yang lebih besar yang berasal dari faktor manusia. Hal tersebut mengakibatkan tidak efektifnya program Raskin yang menyebabkan program tersebut menjadi tidak tepat sasaran. Sehingga untuk mengatasi permasalahan tersebut dibuatlah sistem informasi yang dapat melakukan pengelolaan data dengan lebih efektif dan efisien. Hasilnya adalah sistem ini dapat mempermudah dalam melakukan pengolahan dan pengelolaan data, serta menghasilkan data yang lebih akurat. Selain itu sistem juga dapat mempermudah petugas dalam proses pembuatan laporan (Suryati, 2010).

Pada tahun 2017 Neetu Mitta melakukan penelitian dengan judul "*Blood Ban information System Using Android Application*". Pada penelitian tersebut penulis menjelaskan penggunaan aplikasi android dalam mengembangkan sistem informasi dinilai memiliki nilai *user-friendly* yang lebih tinggi dan aplikasi tersebut dinilai lebih mudah untuk digunakan (Mittal and Snotra, 2017).

Pada tahun 2013 Elvi Fetrina melakukan penelitian dengan judul "*Inventory Management Information System Development at* BPRTIK KEMKOMINFO Jakarta". Pada penelitian tersebut penulis menggunakan SDLC *Rapid Application Development* (RAD) dan dengan pendekatan *Object-Oriented* menggunakan *Unified Modeling Language.* Penggunaan RAD dinilai memiliki kelebihan dalam proses komunikasi dan dalam proses pengumpulan informasi (Fetrina et al., 2013).

Berdasarkan penelitian-penelitian tersebut, penulis menjadikan referensi mengenai latar belakang pada penelitian pertama (Putri, Suryati, 2010). Hal yang menjadi latar belakang dalam penelitian tersebut memiliki kemiripan dengan aplikasi yang akan dikembangkan oleh penulis. Dimana data yang dimasukan secara manual memiliki peluang untuk tidak sesuai dengan fakta yang ada. Hal itu dapat mengakibatkan penyampaian informasi yang tidak sesuai dengan yang seharusnya. Penulis juga menjadikan referensi dari penelitian kedua (Mitta, Neetu, 2017) yang mengembangkan sistem menggunakan aplikasi android. Penulis memilih untuk menggunakan aplikasi yang berbasis android karena penggunaan android dinilai memiliki nilai *user-friendly* yang lebih tinggi dan lebih mudah untuk digunakan. Dalam penelitian ketiga (Fetrina et al., 2013) penulis menjadikan referensi tentang metode yang digunakan dalam pengembangan sistem. Penulis memilih RAD sebagai metode yang digunakan karena RAD dinilai dapat menghasilkan sistem dengan lebih cepat dengan waktu yang relatif lebih singkat. RAD juga mempermudah penulis dalam proses komunikasi dan dalam proses pengumpulan informasi yang berkaitan dengan pengembangan aplikasi yang akan dibuat.

2.2 Data

Data merupakan representasi fakta-fakta atau kejadian yang berasal dari kenyataan. Kumpulan dari angka, huruf, ataupun simbol-simbol khusus dapat juga disebut sebagai data. Data yang masih belum memiliki manfaat biasanya disebut dengan data mentah. Pengolahan data atau biasa disebut dengan *Data Processing* merupakan proses untuk mengolah data mentah menjadi data yang lebih terstruktur sehingga data tersebut memiliki manfaat tertentu. Data yang sudah diolah dan memiliki manfaat biasa disebut sebagai informasi (Valacich and Schneider, 2012).

2.3 Sistem Informasi

Sistem informasi merupakan sistem yang memiliki peran untuk membantu mempermudah jalannya suatu proses bisnis seperti seluruh aktivitas yang berhubungan dengan transaksi, kegiatan, strategi, segala bentuk laporan yang dibutuhkan oleh organisasi tertentu. Definisi sistem informasi dapat juga diartikan sebagai pengelola atau rancangan kerja yang dapat mengorganisasi sumber daya manusia maupun sumber daya yang berupa komputer untuk mengubah suatu data mentah yang kemudian diolah menjadi keluaran berupa informasi yang dibutuhkan suatu perusahaan untuk mencapai target-target yang telah ditentukan. Kumpulan komponen yang terdiri dari prosedur kerja, data mentah, informasi, sumber daya dan teknologi informasi yang terorganisir akan membentuk sistem informasi yang dapat digunakan untuk pengambilan keputusan untuk menyelesaikan suatu kebutuhan yang terdapat dalam suatu organisasi atau perusahaan (Valacich and Schneider, 2012). Beberapa contoh tipe sistem informasi yang biasa digunakan pada suatu organisasi dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tipe-Tipe Sistem Informasi pada *Level* Organisasi

Tipe Sistem Informasi	Deskripsi
Transaction Proccessing System	Memproses seluruh transaksi operasional yang terjadi pada organisasi
Management Information System	Menghasilkan informasi secara detail untuk membantu proses pengelolaan data pada suatu organisasi
Decission Support System	Menyediakan alat analisis untuk mendukung pengambilan keputusan kuantitatif
Electronic Commerce System	Memungkinkan pelanggan melakukan pembelian baik barang ataupun jasa
Geographical Information System	Mengelola yang berhubungan dengan data spasial

Pengembangan sistem informasi pada dasarnya bertujuan untuk mengolah data yang belum diolah kedalam bentuk yang memiliki manfaat untuk menyelesaikan suatu proses bisnis. Hasil olahan data yang sudah memiliki manfaat dapat dikatakan sebagai informasi. Terdapat 3 aspek penilaian yang membuat suatu informasi dapat dikatakan bermanfaat antaralain: aspek keterkaitan (relevan), aspek waktu, dan aspek keakuratan.

2.4 Komponen Sistem Informasi

Menurut (Bourgeois and Bourgeois, 2014), sistem informasi memiliki komponen penyusun yang masing-masing dari komponen tersebut memiliki peranan yang vital dalam suatu sistem informasi. Sistem informasi dapat bekerja secara maksimal apabila setiap komponen telah terpenuhi. Berikut penjelasan dari tiap-tiap komponen sistem informasi:

a. Perangkat keras (*Hardware*)
Hardware merupakan peralatan-peralatan yang terlihat secara fisik yang dapat menunjang jalannya sistem informasi, biasanya *hardware* berupa perangkat-perangkat komputer yang terlihat secara fisik. Contohnya: *keyboard, printer, mouse, monitor.*

b. Perangkat lunak (*Software*)
Perangkat lunak merupakan tempat nantinya akan dilakukan data yang masih mentah akan diolah dan dimanipulasi untuk mendapatkan informasi.

c. Prosedur
Prosedur berisikan kumpulan aturan atau cara yang harus dipenuhi dalam proses manipulasi data sehingga dapat menghasilkan informasi yang sesuai dengan yang diharapkan

d. Pengguna
Pengguna merupakan seluruh pihak yang memegang peran dan terlibat secara langsung dalam ruang lingkup sistem informasi, baik dari tahap pengembangan, proses manipulasi data, maupun pada tahap penggunaan dari informasi pada sistem informasi.

e. Data Base

Database merupakan suatu kumpulan data yang saling terhubung dengan data yang lain, data tersebut tersimpan dalam perangkat keras komputer dan perangkat lunak melakukan manipulasi terhadap data tersebut, diantaranya: data sistem dan data *user*.

2.5 Aviculture

Aviculture merupakan suatu kegiatan budidaya burung untuk memelihara dan membesarkan burung dilakukan dalam penangkaran (Stephen Fronefield, 2019).

Terdapat beberapa faktor yang mendorong dilakukannya *aviculture* antaralain: hobi, bisnis, penelitian, dan konservasi burung yang terancam punah. Selain melakukan budidaya pada burung tujuan lain dari aviculture juga melakukan perawatan terhadap habitat alami burung dan memberikan edukasi yang berhubungan terkait dengan nutrisi, penyakit, dan bagaimana cara melakukan budidaya burung supaya menghasilkan hasil yang lebih baik.

2.6 Lovebird

Lovebird merupakan burung berukuran kecil yang memiliki warna burung yang menarik. *Lovebird* termasuk dalam jenis burung beo dengan *taxonomy* (pengelompokan ilmiah) seperti yang terdapat pada Tabel 2.1.

Tabel 2.2 *Taxonomy* Burung *Lovebird*
(Sumber: (Myers et al., 2019))

Lovebird	
Ordo	*Psittaciformes*
Famili	*Psittacidae*
Sub-Famili	Psittacinae
Genus	*Agapornis*
Spesies	*Agapornis canus, Agapornis fischeri, Agapornis lilianae, Agapornis nigrigenis, Agapornis personatus*

Psittaculture merupakan sebutan yang ditujukan untuk orang-orang yang berfokus pada budidaya burung yang berjenis burung beo. Seseorang yang berada dalam bidang psittaculture biasanya disebut dengan istilah *psittaculturist*. Psittaculturist terbagi menjadi 4 tingkatan yaitu:

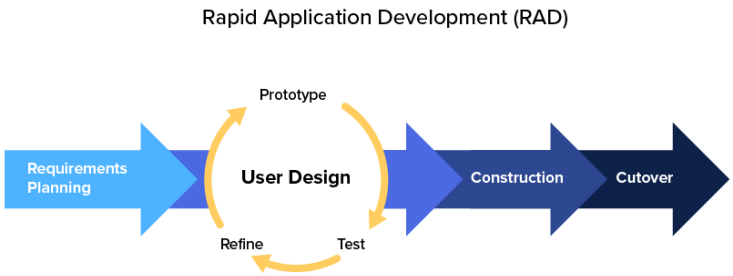
1. *Psittaculturist* yang mempunyai burung jenis beo hanya sebagai hewan peliharaan.
2. *Psittaculturist* yang memelihara burung beo sebagai hobi, biasanya melakukan budidaya dalam sekala kecil.
3. *Psittaculturist* yang memelihara burung beo dan melakukan budidaya dengan skala yang lebih besar.
4. *Psittaculturist* yang menjadikan budidaya burung beo sebagai sumber penghasilan.

Dalam proses budidaya burung *lovebird* yang menjadi indikasi keberhasilan proses budidaya dinilai dari tingkat *mortality* (kematian) pada saat proses budidaya dalam kurun waktu tertentu. Adapun faktor-faktor yang mempengaruhi keberhasilan dari proses budidaya burung antaralain: Umur induk, ukuran dari

kandang burung, parasit, ketersediaan makanan, pengaruh dari manusia, dan iklim dimana proses budidaya berlangsung (Egwumah, 2018).

2.7 Rapid Application Development

Rapid Application Development model adalah *model* pengembangan perangkat lunak yang digunakan dengan tujuan menghasilkan sistem yang memiliki kualitas yang baik dengan waktu yang digunakan dalam proses pengembangan yang relatif singkat dan anggaran biaya yang digunakan rendah (Beynon-Davies et al., 1999). Pada Gambar 2.1 merupakan gambar dari *Rapid Application Development model* (RAD), RAD memiliki 4 fase utama yaitu fase *requirements planning, user design, contruction, cutover*.



Gambar 2.1 *Rapid Application Development*

(Sumber : Kissflow.com, 2018)

1. *Requirement Planning*

Pada fase Requirement Planning bertujuan untuk membangun pemahaman terkait masalah masalah yang ada dalam ruang lingkup bisnis, selain itu hal ini pertujuan untuk mengidentifikasi proses bisnis yang dapat didukung oleh aplikasi yang akan dikerjakan.

2. *User Design*

Pada fase user design merupakan fase untuk membangun design prototipe dengan berbagai iterasi. Pada proses user design menghasilkan tapilan sistem secara garis besar, rencana implementasi, dan detail dari model sistem.

3. *Construction*

Pada fase construction merupakan fase implementasi dilakukan, selain itu pada fase construction akan dilakukan pengujian unit dan pengujian sistem.

4. *Cutover*

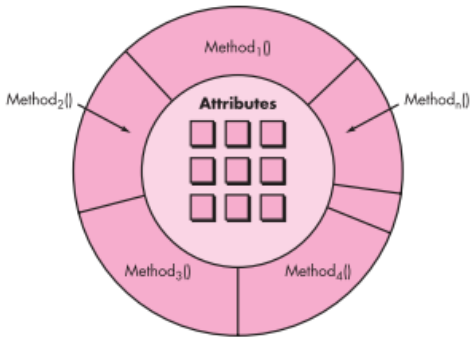
Pada fase cutover merupakan fase peluncuran produk dimana hasil yang diluncurkan adalah hasil dari finalisasi sistem yang telah diuji secara penggunaannya dan telah memenuhi persetujuan dengan pengguna yang akan menggantikan sistem yang lama menjadi sistem yang baru.

2.8 Pendekatan *Object Oriented* (OO)

Object oriented merupakan suatu metode pendekatan pemrograman yang menjadikan objek sebagai basisnya. Dalam pendekatan menggunakan *object oriented*. Terdapat beberapa istilah yang sering ditemui pada pemrograman berorientasi objek, diantaranya:

1. Encapsulation (pembungkusan)

Encapsulation atau pembungkusan bertujuan membatasi akses langsung kebeberapa komponen objek. Pada Gambar 2.2 dijelaskan untuk dapat mengakses atribut adalah dengan mengakses fungsi-fungsi (*method)* yang ada. Hal tersebut mebuat resiko-resiko yang terjadi karena adanya perubahan akan berkurang (Pressman, 2009).



Gambar 2.2 Representasi *Object* dan *Class*
(Sumber : Pressman, 2009)

2. Inheritance (pewarisan)

Inheritance atau pewarisan merupakan pembeda dalam suatu sistem yang berorientasi objek (Pressman, 2009). Semua fungsi dan atribut yang terdapat pada *parent class (super class)* akan diwariskan ke *child class (subclass)*.

3. Polymorphism (perubahan bentuk)

Dalam pemorgraman berbasis objek terdapat istilah *polymorphism* yang artinya perubahan bentuk. *Polymorphism* sendiri adalah karateristik dari *subclass* yang dapat memiliki bentuk yang berbeda-beda (Oracle, 2019).

2.9 Unified Modeling Language (UML)

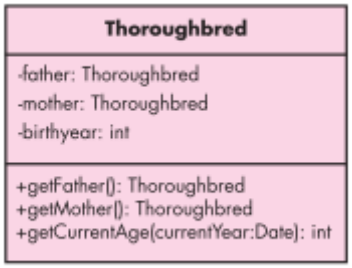
UML merupakan bahasa yang menjadi standar bahasa dalam proses perancangan perangkat lunak (*software blueprints)*. Selama proses perancangan perangkat lunak UML digunakan membantu *developer* perangkat lunak untuk membuat rancangan untuk yang dimodelkan dalam bentuk diagram (Pressman, 2009). Beberapa bentuk diagram yang dimodelkan untuk membantu proses perancangan perangkat lunak, diantaranya:

1. Class Diagram

Dalam UML terdapat *class diagram* yang digunakan untuk menggambarkan isi yang terdapat pada suatu *class* (atribut dan fungsi). *Class* diagram juga

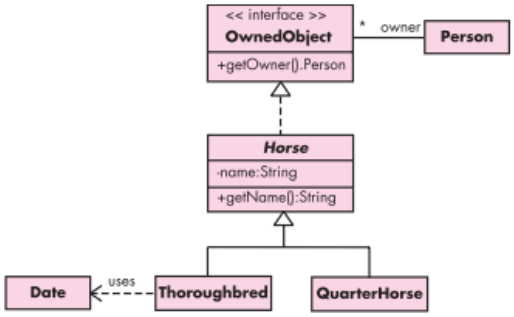
menggambarkan hubungan suatu *class* terhadap kumpulan *class* yang terdapat dalam sistem(Pressman, 2009).

Suatu *class* atau *interface* pada *class diagram* digambarkan sebagai sebuah kotak yang terbagi menjadi 3 bagian. Nama *class* yang terdapat pada *class diagram* diletakkan pada kotak bagian yang paling atas. Pada bagian tengah kotak, berisikan atribut-atribut yang ada pada sebuah *class*. Pada bagian paling bawah dari sebuah kotak, berisikan kumpulan fungsi (*method*) yang dimiliki oleh suatu *class.* Gambar 2.3 menunjukkan representasi *class* pada *class diagram* dengan nama *class* yang terletak pada bagian paling atas pada kotak (*thoroughbred)*, *class* tersebut memiliki atribut berjumlah 3 buah yang terletak pada bagian tengah kotak, dan fungsi yang terdapat dalam *class* tersebut berjumlah 3 yang dituliskan pada bagian paling bawah pada kotak. Komponen-komponen yang terdapat pada *class* tersebut terdiri dari nama komponen, tipe data komponen, dan akses *modifier*. Tipe data dari atribut pada *class* diagram dituliskan terpisah dari nama atribut menggunakan titik dua (:). Akses *modifier* pada *class* diagram diindikasikan menggunakan simbol-simbol diantaranya, - (*private*), # (*protected*), + (*public*), dan ~ (*package*) (Pressman, 2009).



Gambar 2.3 *Class Diagram*
(Sumber: Pressman, 2009)

Relasi dari suatu *class* pada *class diagram* ditunjukan dengan adanya garis yang memiliki panah pada salah satu ujungnya. *Class* yang merupakan *subclass* pada *class diagram* akan terhubung dengan *class* lain melalui garis dengan panah yang menunjuk suatu *class* yang merupakan *superclass* dari sebuah *subclass*. Dalam UML hubungan antara *subclass* dengan *superclass* dinamakan *generalization* (Pressman, 2009). Pada Gambar 2.4 merupakan contoh hubungan antar *class*. Pada gambar tersebut *class throughbred* terhubung dengan *class horse* dengan tanda panah yang mengarah ke *class horse* yang merupakan *abstract class* sekaligus merupakan *superclass* dari *class throughbred*. Untuk menggambarkan hubungan suatu *class* dengan *class interface* ditandai dengan garis putus-putus dengan tanda panah mengarah ke *class interface*. Hubungan *class* yang mengimplementasi sebuah *interface* pada UML disebut sebagai *realization* (Pressman, 2009). Contoh penerapan *realization* dapat dilihat pada Gambar 2.4, *class Horse* terhubung dengan *class OwnedObject* melalui garis putus-putus dengan tanda panah mengarah pada *class OwnedObject*, menandakan bahwa *class Horse mengimplementasikan class interface OwnedObject*.

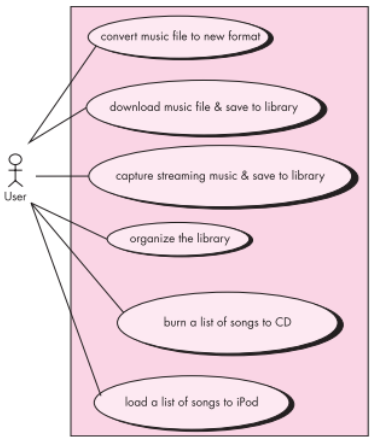


Gambar 2.4 Class diagram regarding horses
(Sumber : Pressman, 2009)

2. Use Case Diagram

Representasi fungsionalitas atau kebutuhan yang terdapat dalam sistem yang dilihat dari sisi pengguna dalam UML dikenal dengan sebutan *use case diagram* (Pressman, 2009). Pada Gambar 2.5 merupakan contoh dari *use case* diagram. Dimana terdapat seorang yang bertindak sebagai *actor* yang digambarkan dengan gambar *stickman*, dalam gambar tersebut *stickman* yang ada merupakan representasi dari satu jenis *actor* yaitu *user*. Suatu sistem yang kompleks bisa memiliki beberapa jenis *actor* didalamnya (Pressman, 2009).

Setiap fungsi yang terdapat pada sistem akan dituliskan pada bangun datar berbentuk *oval*. Untuk menghubungkan antara *actor* dengan *usecase* akan dihubungkan menggunakan sebuah garis, dimana garis tersebut menandakan bahwa *actor* terlibat atau memiliki wewenang untuk dapat menjalankan fungsi tersebut (Pressman, 2009). Langkah ataupun tahapan untuk dapat menjalankan fungsi tersebut tidak akan dijelaskan pada *use case* diagram, akan tetapi penjelasan dari tahapan untuk menjalankan fungsi tersebut akan dituliskan terpisah dengan *use case diagram*. fungsi yang terdapat di dalam suatu *use case* diagram akan dibungkus oleh suatu kotak yang merupakan representasi dari sistem sedangkan *actor* yang ada tidak berada didalamnya, hal tersebut berarti bahwa *use case* berda di dalam sistem sedangkan *actor* terpisah dari sistem.

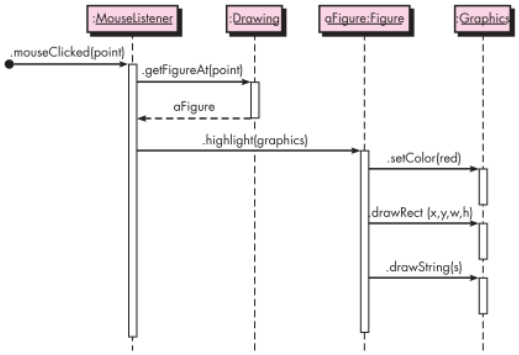


Gambar 2.5 Use Case Diagram
(Sumber : Pressman, 2009)

3. Sequence Diagram

Dalam UML yang digunakan untuk memberikan gambaran tentang bagaimana komunikasi antar objek ketika suatu proses sedang berlangsung disebut dengan *Sequence diagram*. *Sequence* diagram menampilkan urutan kejadian tentang perpindahan pesan dari objek satu ke objek yang lain yang bertujuan untuk menyelesaikan suatu proses yang bertujuan untuk memenuhi suatu kebutuhan. *Sequence diagram* merupakan representasi dari interaksi yang terjadi dalam satu skenario untuk mencapai memenuhi suatu kebutuhan dalam sistem (Pressman, 2009).

Pada Gambar 2.6 merupakan contoh *sequence* diagram untuk program menggambar. Diagram tersebut menampilkan runtutan kejadian yang terjadi pada saat terdapat interaksi dengan gambar (dalam kasus pada Gambar 2.6 merupakan interaksi apa bila ingin meningkatkan tingkat kecerahan pada gambar). Tiap kotak yang berada diatas diagram biasanya berhubungan dengan suatu objek, tetapi tidak menutup kemungkinan bahwa yang terdapat dalam kotak tersebut adalah sebuah *model* ataupun yang lainnya, seperti *class* (Pressman, 2009).



Gambar 2.6 Sequence Diagram
(Sumber : Pressman, 2009)

2.10 Graphql

Graphql merupakan *Application Programming Interface* (API) yang berisikan bahasa *query* untuk memenuhi suatu permintaan yang berdasarkan pada data yang ada dalam kurun waktu tertentu. *Graphql* memberikan deskripsi data yang terdapat dalam API secara lengkap dan mudah untuk dipahami, *graphql* membantu *client* untuk mendapatkan data hanya yang sesuai dengan apa yang mereka butuhkan dan tidak membuang *resource* dengan mengambil data-data yang *client* sebenarnya tidak membutuhkannya, sehingga mempermudah dalam mengembangkan API kedepannya (graphql, 2019).

2.11 Prisma

Prisma merupakan jembatan yang menghubungkan antara database dengan *resolver graphql*. Prisma merupakan pengganti dari *Object-Relational Mapping*

(ORM) tradisional yang membuatnya mudah untuk mengimplementasikan server *graphql* yang fleksibel dan siap untuk digunakan. Dalam server prisma memiliki mesin dengan performa tinggi untuk menghasilkan *database query* yang aktual (prisma, 2019).

2.12 Arsitektur MVP (Model-View-Presenter)

MVP (*model-view-presenter*) adalah *design pattern* yang merupakan pengembangan dari *design pattern* MVC (*model-view-controller*). *View* hanya berisikan komponen yang berhubungan langsung dengan pengguna, komponen-komponen tersebut seperti activity/fragment, dan XML file. Model merepresentasikan kelas-kelas yang mendeskripsikan data dan *business logic*. Model juga berisikan *business rules* yang merupakan cara tentang bagaimana data dapat dimanipulasi. Hal yang membedakan antara MVP dan MVC adalah pada MVP *controller* digantikan dengan presenter yang menghubungkan antara *view* dengan *model*, presenter sendiri dapat melakukan manipulasi pada data dan memodifikasi/memperbarui tampilan (Sinhal, 2017).

2.13 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan proses untuk memastikan sistem berjalan seperti yang semestinya dan untuk menelusuri apakah terdapat kecacatan sebelum sistem benar-benar digunakan. Pengujian perangkat lunak secara garis besar memiliki 2 tujuan utama. Pertama, membuktikan kepada *stackholder* bahwa sistem telah sesuai dengan kebutuhan. Proses untuk membuktikan hal tersebut dapat dilakukan dengan melakukan proses pengujian validasi. Pengujian tersebut mempunyai sekumpulan kasus uji bagaimana seharusnya sistem digunakan. Kedua, bertujuan untuk menemukan adanya kecacatan dalam sistem. Pengujian ini dilakukan dengan cara menyediakan kasus uji yang secara acak atau tidak sesuai dengan prosedur sebagaimana seharusnya sistem digunakan(Sommerville, 2010).

2.13.1 Black Box Testing

Black-box testing merupakan salah satu jenis pengujian perangkat lunak yang menguji tentang fungsionalitas tanpa harus mengetahui bagaimana kode program ataupun faktor internal lain di dalam perangkat lunak tersebut. *Black-box testing* biasanya disebut dengan *specification-based testing* yang dalam pengaplikasiannya *black-box testing* dapat diterapkan dalam pengujian integrasi, sistem, dan *acceptance*.

2.13.2 White Box Testing

White box testing adalah salah satu jenis pengujian perangkat lunak dimana *test-case*-nya diambil dari struktur kode program (Pressman, 2009). Teknik pengujian yang dilakukan dalam *white box testing* dikenal dengan istilah *Basis path testing*. *Basis path testing* digunakan untuk mendapatkan kompleksistas dari suatu algoritma yang terdapat di dalam sistem dengan cara menentukan *flow*

graph. *Flow graph* merupakan sebuah alur kode program untuk menyelesaikan suatu tugas/perintah yang digambarkan dengan kumpulan *independent path*. *Independent path* merupakan satu dari banyak jalur yang ada dalam suatu kode program dimana jalur tersebut merupakan kumpulan *node* yang telah terhubung dengan minimal satu buah *edge. Node* merupakan representasi dari pernyataan atau perintah yang terdapat dalam kode program. Kumpulan *node* tersebut dihubungkan dengan garis yang disebut *edge. Edge* merupakan representasi dari urutan perintah yang harus dikerjakan dalam kode program.

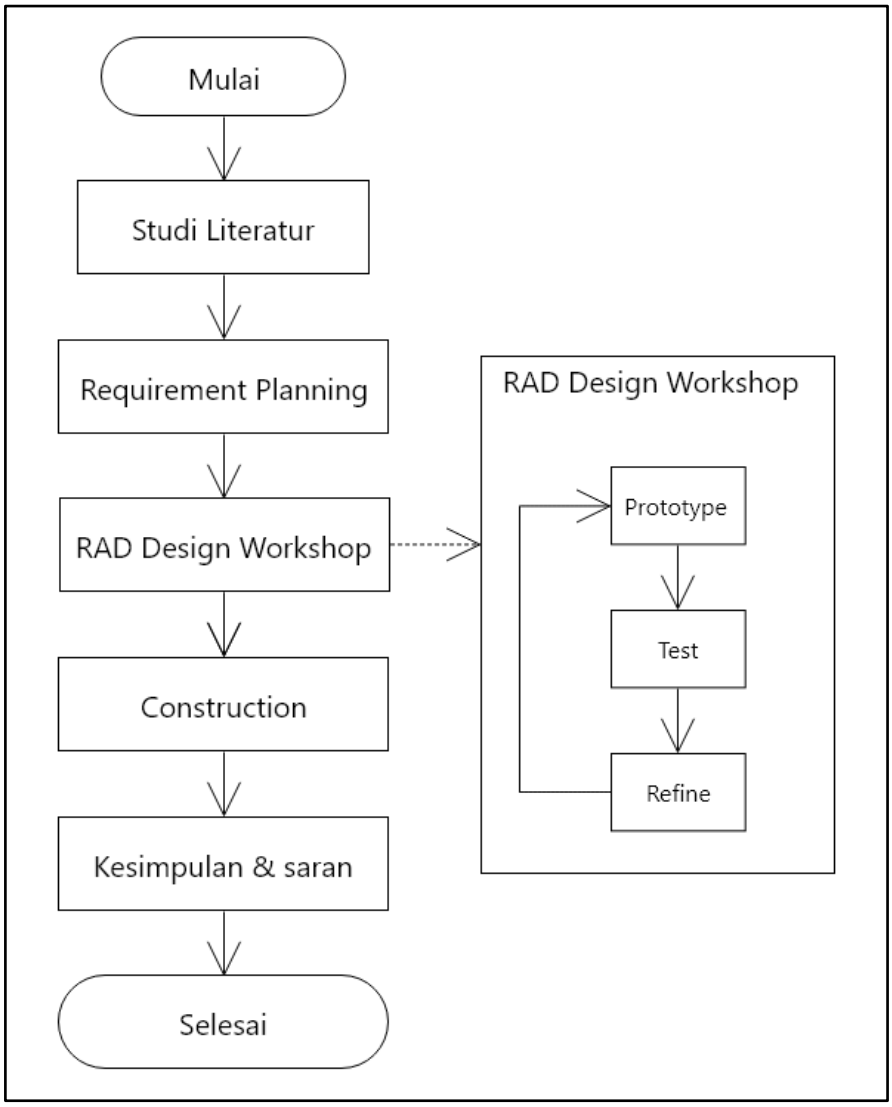
2.13.3 Usability Testing

Usability testing digunakan sebagai acuan kepuasan calon pengguna terhadap sistem dan penilaian kualitas berdasarkan perasaan yang dirasakan oleh pengguna selama menggunakan sistem (Bevan, 1995). Hal yang dapat dilakukan untuk mendapatkan informasi dari calon pengguna salah satunya adalah dengan memberikan kuesioner kepada para calon pengguna. Pertanyaan yang terdapat dalam kuesioner biasanya berbentuk pilihan ganda ataupun skala penilaian dari responden. Terdapat beberapa metode kuesioner yang dijadikan starndar untuk mendapatkan informasi dari pengguna salah satunya adalah dengan metode *Post-study System Usability Questionnaire*.

Post-study system usability questionnaire terdiri dari 15 pernyataan yang memiliki skala jawaban 1-7. Semakin besar nilai yang berikan oleh responden menandakan semakin setuju responden dengan pernyataan pada soal kuesioner. Pada *post-study system usability* terdapat beberapa aspek yang dinilai yaitu kualitas dari sistem, kualitas informasi pada sistem, dan kualitas antarmuka dari sistem(Sauro and Lewis, 2012).

BAB 3 METODOLOGI

Bab metodologi penelitian membahas terkait tahapan-tahapan yang dilakukan ketika melakukan proses penelitian untuk mengembangkan sistem informasi manajemen ternak burung *lovebird*. Tahapan-tahapan tersebut terbagi menjadi beberapa bagian seperti yang dijelaskan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Tahapan Penelitian

3.1 Studi Literatur

Pada tahapan ini, penggunaan studi literatur bertujuan untuk mengetahui dasar-dasar teori yang digunakan selama proses penelitian. Teori-teori tersebut akan digunakan untuk membantu jalannya penelitian. Teori yang dijadikan sebagai studi literatur berasal dari beberapa sumber ilmiah yang berasal dari jurnal

ilmiah, buku, dan literatur yang berasal dari internet. Beberapa studi literatur yang digunakan untuk menunjang penelitian ini meliputi:

1. Kajian Pustaka
2. Sistem Informasi
3. Rapid Application Development Model
4. Pendekatan Object Oriented
5. Pemodelan UML
6. Graphql
7. prisma
8. Arsitektur MVP (Model-View-presenter)
9. Teori Pengujian

3.2 Requirement Planning

Pada tahapan ini penulis dan pengguna sistem bertemu untuk berdiskusi terkait kebutuhan yang harus tersedia pada sistem. Pada fase ini akan dituliskan kebutuhan list kebutuhan baik dari segi *fungsional* maupun *non-fungsional* yang nantinya akan diimplementasikan kedalam sistem, dan menuliskan aktor yang terdapat dalam sistem yang akan dibuat. Hasil analisis yang sudah didapatkan oleh penulis kemudian akan dituliskan dalam bentuk *Work Breakdown Structure*.

3.3 RAD Design Workshop

Pada fase RAD Design Workshop merupakan fase perancangan dari sistem dengan kebutuhan yang telah didapatkan dari fase *requirement planning*. Perancangan merupakan dasar yang nantinya akan menjadi gambaran untuk proses implementasi sistem. Terdapat beberapa tahapan yang dilalui pada fase RAD Design Workshop yaitu: prototype, test, dan refine.

3.3.1 Prototype

Terdapat 3 buah jenis perancangan yang akan dikerjakan sebelum menuju ke fase selanjutnya, diantaranya:

1. Perancangan Antarmuka

Perancangan antarmuka akan menggambarkan bagaimana tampilan sistem yang akan dibuat. Dalam hal ini tata letak dari setiap komponen pada suatu *activity* akan digambarkan. Perancangan antarmuka akan dijadikan gambaran ketika proses implementasi antarmuka sistem.

2. Perancangan Komponen

Untuk dapat memenuhi suatu kebutuhan atau menyelesaikan suatu proses, terdapat langkah-langkah yang harus dilalui. Langkah-langkah yang harus

dilakukan akan dituliskan pada saat perancangan komponen. Dalam penulisannya langkah-langkah tersebut dituliskan dalam bentuk *pseudocode.*

3. Perancangan Data

Perancangan data memuat tentang bagaimana struktur *database* yang terdapat dalam sistem. Perancangan data berisikan tabel-tabel yang terdapat dalam *database* sistem yang akan dikembangan.

3.3.2 Test

Pada proses ini setiap hasil rancangan yang dibuat akan dituliskan kedalam *Roadmap* proses pengerjaan sistem yang kemudian akan ditunjukan langsung kepada *stackholder* untuk mengetahui pendapat dari *stackholder* tersebut.

3.3.3 Refine

Pada tahapan ini saran yang didapatkan dari *stackholder* akan ditinjau kembali yang kemudian akan dilakukan perbaikan terhadap hasil rancangan yang sebelumnya sudah dibuat berdasarkan pada masukkan yang diberikan oleh *stackholder*.

3.4 Construction

Dalam *Rapid Application Development* fase *construction* merupakan fase rancangan akan diimplementasikan, dan hasil yang didapatkan dari proses implementasi akan diuji.

3.4.1 Implementasi

Implementasi sistem adalah hasil rancangan yang sudah dilakukan akan dimplementasikan. Beberapa proses yang dilakukan pada saat implementasi adalah sebagai berikut:

1. Implementasi kebutuhan sistem adalah proses mengimplementasikan seluruh kebutuhan yang telah dijelaskan pada saat proses *requirement planning* kedalam sistem.
2. Implementasi basis data adalah proses implementasi dari rancangan basis data yang sudah dijelaskan pada tahapan RAD *Design Workshop.*
3. Implementasi kode program merupakan proses mengimplementasikan *pseudocode* yang sudah dibuat pada fase RAD *Design Workshop* kedalam bahasa pemrograman *java*. Pada tahap implementasi kode program seluruh kode akan menggunakan *design pattern MVP(model view presenter)* untuk menyelesaikan suatu proses.
4. Implementasi antarmuka merupakan proses implementasi dari hasil perancangan antarmuka pada fase RAD *Design Workshop*. Untuk mengimplementasikan antarmuka pada sistem yang berbasis *android* bahasa pemrograman yang digunakan peneliti adalah XML.

3.4.2 Pengujian

Proses pengujian pada sistem dilakukan untuk mengevaluasi terhadap hasil implementasi sistem dengan kesesuaian kebutuhan yang sudah dituliskan pada fase *requirement planning*. Proses pengujian bertujuan untuk mengetahui apakah terdapat kesalahan dan memastikan untuk mengurangi peluang terjadinya kesalahan dalam sistem. Dalam penelitian ini terdapat beberapa aspek yang diuji untuk memberikan penilaian pada kualitas sistem, diantaranya:

1. Unit Testing

Pengujian unit merupakan pengujian yang dilakukan untuk memastikan setiap komponen yang terdapat dalam penelitian ini telah berfungsi. Komponen tersebut meliputi *class*, atribut (objek), dan fungsi. Pada penelitian ini dilakukan pengujian dengan mengambil 3 buah fungsi yang dijadikan kasus uji. Pengujian unit dalam penelitian ini dilakukan menggunakan metode *white-box*.

2. Validation Testing

Validation testing merupakan pengujian yang dilakukan untuk memastikan bahwa semua fungsi yang telah diimplementasikan kedalam sistem sudah berjalan dan sesuai dengan apa yang digambarkan sebelumnya. Pada penelitian ini *validation testing* dijalankan menggunakan metode *black-box*.

3. Usability Testing

Pada pengujian ini akan dilakukan pengujian terhadap sistem dari sisi kemudahan calon pengguna ketika menggunakan sistem secara langsung. Dalam penelitian ini peneliti ingin mengetahui beberapa faktor untuk menilai *usability* dari sistem, faktor-faktor tersebut antaralain: kualitas dari sistem, kualitas informasi yang terdapat dalam sistem, dan kualitas antarmuka sistem. Untuk mendapatkan respon dari calon pengguna peneliti memanfaatkan metode *post-study system usability questionnaire* yang merupakan kuesioner berisikan 15 pernyataan.

3.5 Kesimpulan dan Saran

Kesimpulan yang didapatkan dari hasil pembahasan diatas digunakan untuk menjawab semua pertanyaan yang dituliskan pada rumusan masalah. Saran dituliskan untuk menyempurnakan kekurangan yang ditemukan selama proses penelitian berlangsung untuk dapat dikembangkan pada penelitian selanjutnya.

BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran Umum Sistem

Sistem informasi manajemen ternak burung *lovebird* merupakan suatu aplikasi yang dibuat untuk mempermudah para peternak maupun pecinta burung *lovebird* dalam urusan mengelola burung maupun hal yang terkait dengan keuangan dalam pengelolaan budidaya ternaknya. Kemudahan dalam pengelolaan hewan dan keuangan yang terjadi selama proses budidaya akan mempermudah para pelaku budidaya untuk mendapatkan data yang akurat yang berkaitan dengan budidaya. Sistem informasi ini dapat memberikan informasi berkaitan dengan detail dari masing-masing burung yang dibudidayakan, selain itu para pelaku budidaya juga dapat menambahkan informasi baru apabila burung yang dibudidayakan bertambah. Para pelaku budidaya juga dapat menambahkan catatan yang terjadi selama proses budidaya burung berlangsung, dan mencatat selama proses pembesaran burung mulai dari masih telur hingga berumur 1 bulan. Selain mencatat hal-hal yang berkaitan dengan burung yang dibudidayakan sistem ini juga menyediakan fitur yang dapat mengelola keuangan. Para pelaku budidaya dapat mencatatkan segala bentuk pengeluaran dan pemasukan yang didapatkan selama proses budidaya burung berlangsung. Catatan yang berkaitan dengan pengeluaran dan pemasukan dalam kurun waktu tertentu dapat dilihat oleh para pelaku budidaya dalam bentuk laporan laba-rugi.

4.2 Identifikasi Aktor

Aktor merupakan suatu individu yang akan melakukan interaksi dengan sistem. Pada sistem ini aktor yang terlibat digambarkan pada Tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
Guest	*Guest* merupakan orang-orang yang belum terdaftar dalam sistem dan belum dapat menggunakan sebagian besar fitur yang ada dalam sistem.
Pengguna	Pengguna merupakan orang-orang yang telah terdaftar dan masuk kedalam sistem.

4.3 Kebutuhan Sistem

Kebutuhan sistem adalah fitur yang harus tersedia dalam sistem untuk dapat suatu masukan, maupun menanggapi suatu kondisi tertentu yang berguna untuk menyelesaikan suatu permasalahan. Kebutuhan sistem dibagi menjadi 2 jenis kebutuhan yang masing masing dituliskan menggunakan kode unik, BFM-F-XX untuk kebutuhan fungsional dan BFM-NF-XX untuk kebutuhan non-fungsional.

BFM diambil dari fitur utama dari sistem yang akan dikembangkan yaitu *Bird Farm Management*. Huruf F adalah kode unik yang merepresentasikan kebutuhan fungsional yang terdapat dalam sistem. Sedangkan untuk kebutuhan non-fungsional yang terdapat dalam sistem direpresentasikan dengan kode NF. Simbol XX pada kode menandakan nomor dari kebutuhan sistem.

4.3.1 Kebutuhan Fungsional Sistem

Tabel 4.2 Kebutuhan Fungsional Sistem

No.	Kode Unik	Nama Kebutuhan	Deskripsi
1.	BFM-F-01	Login	Sistem dapat menyediakan fitur untuk guest untuk mendaftar.
2.	BFM-F-02	Signup	Sistem dapat menyediakan fitur login untuk guest agar dapat masuk ke sistem.
3.	BFM-F-03	Melihat profile pengguna	Sistem menyediakan fitur untuk pengguna dapat melihat *profile* pengguna.
4.	BFM-F-04	Mengubah informasi pengguna	Sistem menyediakan fitur untuk pengguna dapat mengubah informasi pada *profile* pengguna.
5.	BFM-F-05	Melihat list burung	Sistem dapat menyediakan fitur untuk pengguna dapat melihat *list* burung yang dimiliki.
6.	BFM-F-06	Melihat detail burung	Sistem dapat menyediakan fitur untuk pengguna dapat melihat detail dari burung yang dimiliki.
7.	BFM-F-07	Mengubah informasi burung	Sistem dapat menyediakan fitur untuk pengguna dapat mengubah data burung yang dimiliki.
8.	BFM-F-08	Menambah burung	Sistem dapat menyediakan fitur untuk pengguna dapat menambahkan burung baru.
9.	BFM-F-09	Menambah induk burung	Sistem dapat menyediakan fitur untuk pengguna dapat menambahkan induk burung.

Tabel 4.2 Kebutuhan Fungsional Sistem (Lanjutan)

10.	BFM-F-10	Melihat list induk burung	Sistem dapat menyediakan fitur untuk pengguna dapat melihat *list* induk burung yang dimiliki.
11.	BFM-F-11	Melihat list batch	Sistem dapat menyediakan fitur untuk pengguna dapat melihat *list batch* proses budidaya yang ada pada masing masing induk.
12.	BFM-F-12	Membuat batch baru	Sistem dapat menyediakan fitur untuk pengguna dapat membuat *batch* proses budidaya yang baru.
13.	BFM-F-13	Melihat detail log batch	Sistem dapat menyediakan fitur untuk pengguna dapat melihat detail *log* dari suatu *batch* proses budidaya yang ada.
14.	BFM-F-14	Menambahkan log pada batch	Sistem dapat menyediakan fitur untuk pengguna dapat menambahkan detail *log* kedalam suatu *batch* proses budidaya.
15.	BFM-F-15	Mengakhiri batch	Sistem dapat menyediakan fitur untuk pengguna dapat mengakhiri suatu *batch* proses budidaya
16.	BFM-F-16	Melihat galeri burung	Sistem dapat menyediakan fitur untuk pengguna dapat melihat galeri burung.
17.	BFM-F-17	Menambahkan foto pada galeri burung	Sistem dapat menyediakan fitur untuk pengguna dapat menambahkan foto pada galeri burung
18.	BFM-F-18	Filter burung	Sistem dapat menyediakan fitur untuk pengguna dapat melakukan pencarian berdasarkan pada nomor ring burung ataupun jenis mutasi warna burung.

Tabel 4.2 Kebutuhan Fungsional Sistem (Lanjutan)

19.	BFM-F-19	Melihat list jurnal keuangan	Sistem dapat menyediakan fitur untuk pengguna dapat melihat *list* jurnal keuangan yang telah dibuat
20.	BFM-F-20	Membuat jurnal keuangan	Sistem dapat menyediakan fitur untuk pengguna dapat membuat jurnal keuangan
21	BFM-F-21	Melihat list transaksi	Sistem dapat menyediakan fitur untuk pengguna dapat melihat *list* transaksi keuangan
22.	BFM-F-22	Membuat transaksi	Sistem dapat menyediakan fitur untuk pengguna dapat membuat transaksi pada jurnal keuangan
23.	BFM-F-23	Melihat laporan laba-rugi	Sistem dapat menyediakan fitur untuk pengguna dapat melihat laporan laba-rugi yang terjadi dalam kurun waktu satu bulan

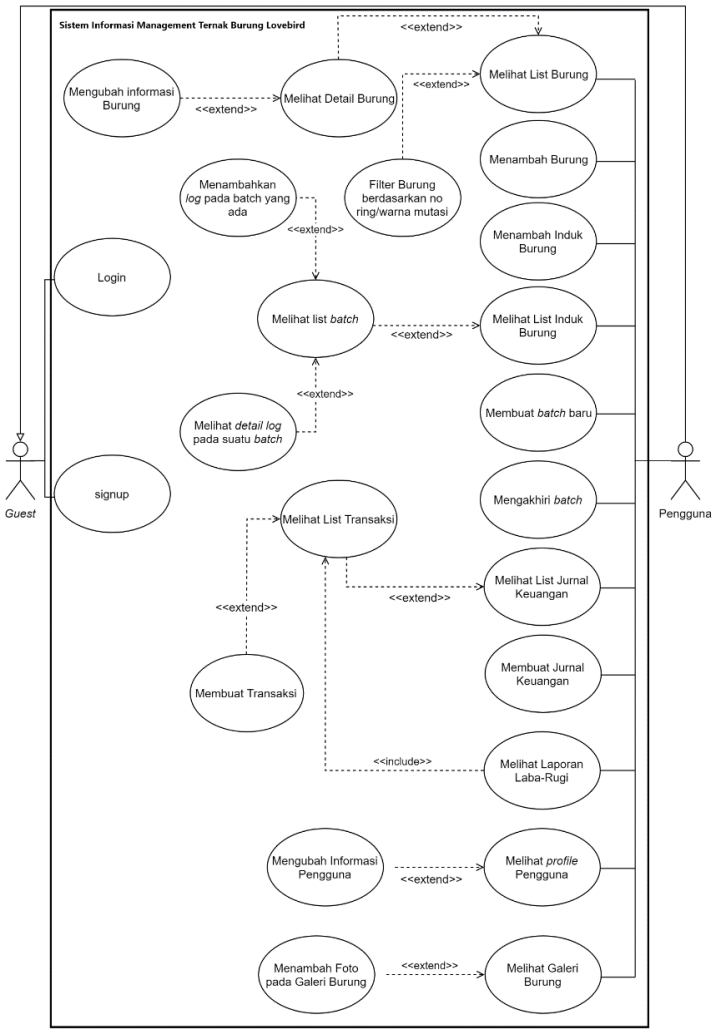
4.3.2 Kebutuhan Non-fungsional Sistem

Tabel 4.3 Kebutuhan Non-fungsional Sistem

No.	Kode Unik	Deskripsi
1.	BFM-NF-01	Sitem harus memiliki antarmuka yang menarik dan mudah untuk dimengerti saat digunakan

4.4 Diagram Use Case

use case diagram menggambarkan bagaimana aktor berinteraksi dengan sistem. Kebutuhan fungsional menjadi dasar untuk dapat memodelkan diagram ini. Gambar 4.1 merupakan *use case diagram* dari sistem yang dikembangkan. Dalam sistem tersebut terdapat 2 buah aktor yaitu *guest* dan pengguna, pengguna sendiri merupakan generalisasi dari aktor *guest*. Selain itu pada Gambar 4.1 terdapat kebutuhan-kebutuhan yang telah dituliskan pada Tabel 4.2 yang berjumlah 23 buah. Garis akan menghubungkan antara aktor dengan masing masing kebutuhan. Pada sistem yang akan dibuat aktor *guest* hanya memiliki 2 buah kebutuhan. Untuk dapat menggunakan fungsi yang terdapat dalam sistem aktor *guest* harus login terlebih dahulu.



Gambar 4.1 *Use Case Diagram* Sistem Informasi Manajemen Ternak Burung *lovebird*

4.5 Use Case Scenario

Skenario *use case* menggambarkan prosedur yang harus dilalui oleh aktor untuk dapat berinteraksi dengan sistem. Pembuatan skenario yang terdapat dalam sistem informasi manajemen ternak burung *lovebird* berdasarkan pada *use case diagram* yang terdapat pada Gambar 4.1. Tabel 4.4 hingga Tabel 4.25 adalah *usecase skenario* untuk masing masing kebutuhan.

Tabel 4.4 Skenario *Use Case Login*

login	
Objective	Aktor dapat login ke sistem
Aktor	Guest
Pre-condition	-

Tabel 4.4 Skenario *Use Case Login* (Lanjutan)

Main flow	1. Sistem menampilkan *activity login* 2. Aktor memasukan *email* dan *password* 3. Aktor menekan tombol login 4. Sistem menampilkan *activity dashboard*
Alternative flow	4.1. Sistem menampilkan pesan error saat login
Post-condition	Sistem menampilkan *activity dashboard* yang berarti aktor sudah masuk kedalam sistem.

Tabel 4.5 Skenario *Use Case Signup*

Signup	
Objective	Aktor dapat mendaftar sebagai pengguna
Actor	Guest
Pre-condition	-
Main flow	1. Sistem menampilkan *activity login* 2. Aktor menekan tombol *signup* 3. Sistem menampilkan *activity signup* 4. Aktor mengisi form untuk *signup* 5. Aktor menekan tombol register 6. Sistem menampilkan dialog sukses
Alternative flow	5.1. Sistem menampilkan pesan error saat *signup*
Post-condition	Sistem menampilkan dialog registrasi sukses dan aktor terdaftar sebagai pengguna.

Tabel 4.6 Skenario *Use Case* Melihat *Profile*

Melihat *Profile*	
Objective	Aktor dapat melihat *profile* yang berisikan informasi diri dari pengguna
Actor	Pengguna
Pre-condition	Aktor berada dalam *activity dashboard*
Main flow	1. Aktor menekan gambar *account* yang berada di sisi kiri atas pada *activity dashboard* 2. Sistem menampilkan *activity profile* yang berisikan informasi terkait data diri pengguna
Alternative flow	-
Post-condition	Sistem menampilkan *activity profile* yang berisikan informasi terkait data diri pengguna

Tabel 4.7 Skenario *Use Case* Mengubah Informasi Pengguna

Mengubah Informasi Pengguna	
Objective	Aktor dapat mengganti informasi tentang data diri pengguna
Actor	Pengguna
Pre-condition	Aktor berada dalam *activity profile*
Main flow	1. Aktor menekan tombol edit *profile* yang berada pada *activity profile* 2. Sistem menampilkan form berisikan informasi data diri pengguna 3. Aktor memasukan informasi baru sesuai dengan label yang tersedia dalam form 4. Aktor menekan tombol *save* 5. Sistem menampilkan *dialog* sukses mengganti perubahan
Alternative flow	-
Post-condition	Sistem menampilkan *dialog* sukses menyimpan perubahan dan informasi terbaru akan ditambahkan ke *database*

Tabel 4.8 Skenario *Use Case* Melihat *List* Burung

Melihat *list* burung	
Objective	Aktor dapat melihat koleksi burung yang dimiliki
Actor	Pengguna
Pre-condition	Aktor berada dalam *activity dashboard*
Main flow	1. Aktor menekan menu *bird management* 2. Sistem menampilkan *activity birdManagement* yang berisikan koleksi burung dari pengguna
Alternative flow	-
Post-condition	Sistem menampilkan *activity birdManagement* yang berisikan koleksi burung dari pengguna

Tabel 4.9 Skenario *Use Case* Melihat Detail Burung

Melihat detail burung	
Objective	Aktor dapat melihat detail informasi dari salah satu koleksi burungnya
Actor	Pengguna
Pre-condition	Aktor berada dalam *activity birdManagement*
Main flow	1. Aktor menekan salah satu gambar burung yang berada dalam *list* burung pada *activity birdManagement*

Tabel 4.9 Skenario *Use Case* Melihat Detail Burung (Lanjutan)

	2. Sistem menampilkan *activity birdDetail* yang berisikan informasi burung sesuai dengan gambar yang aktor pilih
Alternative flow	-
Post-condition	Sistem menampilkan *activity birdDetail* yang berisikan informasi burung sesuai dengan gambar yang dipilih aktor

Tabel 4.10 Skenario *Use Case* Mengubah Informasi *Detail* Burung

Mengubah Informasi *Detail* Burung	
Objective	Aktor dapat mengganti informasi terkait data burung yang dimiliki
Actor	Pengguna
Pre-condition	Aktor berada dalam *activity bird detail*
Main flow	1. Aktor menekan tombol *edit* Data Burung yang berada pada *activity birdDetail* 2. Sistem menampilkan *form* yang berisikan informasi sesuai dengan burung yang berada dalam *activity birdDetail* 3. Aktor mengisikan form dengan data baru yang ingin diganti sesuai pada label yang terdapat dalam *form* 4. Aktor menekan tombol *update* 5. Sistem menampilkan *dialog* sukses
Alternative flow	-
Post-condition	Sistem menampilkan *dialog* sukses dan data terkait burung yang diubah tersimpan didalam *database.*

Tabel 4.11 Skenario *Use Case* Menambah Burung

Menambah burung	
Objective	Aktor dapat menambahkan koleksi burungnya
Actor	Pengguna
Pre-condition	Aktor berada dalam *activity birdManagement*
Main flow	1. Aktor menekan *floating action button* yang ada dalam *activity birdManagement* 2. Aktor memilih menu *register* burung 3. Sistem menampilkan *form* untuk registrasi burung baru 4. Aktor mengisi *form* untuk registrasi burung baru 5. Aktor menekan tombol *register* Burung 6. Sistem menampilkan *dialog* sukses

Tabel 4.11 Skenario *Use Case* Menambah Burung (Lanjutan)

Alternative flow	4.1. Sistem menampilkan pesan *error* dikarenakan pengguna tidak mengisi *form* secara lengkap 4.2. Sistem menampilkan pesan *error* dikarenakan pengguna memasukan nomor ring yang sudah terdaftar 5.2. Sistem menyimpan data burung tanpa dilengkapi informasi induk burung
Post-condition	Sistem menampilkan *dialog* sukses dan data burung berhasil tersimpan di *database*

Tabel 4.12 Skenario *Use Case* Menambah Induk Burung

Menambah induk burung	
Objective	Aktor dapat menambahkan koleksi induk untuk dibudidayakan
Actor	Pengguna
Pre-condition	Aktor berada dalam *activity birdManagement*
Main flow	1. Aktor menekan *floating action button* yang ada dalam *activity birdManagement* 2. Aktor memilih menu *register* induk 3. Sistem menampilkan *form* untuk registrasi induk 4. Aktor mengisi *form* untuk registrasi induk baru 5. Aktor menekan tombol *register* Induk Burung 6. Sistem menampilkan *dialog* sukses
Alternative flow	-
Post-condition	Sistem menampilkan *dialog* sukses dan data induk berhasil tersimpan di *database*

Tabel 4.13 Skenario *Use Case* Melihat *List* Induk Burung

Melihat *list* induk	
Objective	Aktor dapat melihat *list* induk yang dimiliki
Actor	Pengguna
Pre-condition	Aktor berada pada *activity dashboard*
Main flow	1. Aktor memilih menu *breeding record* 2. Sistem menampilkan *activity breedingRecord* yang berisikan *list* induk yang dimiliki pengguna
Alternative flow	-

Tabel 4.13 Skenario *Use Case* Melihat *List* Induk Burung (Lanjutan)

Post-condition	Sistem menampilkan *activity breedingRecord* yang berisikan *list* induk yang dimiliki oleh pengguna

Tabel 4.14 Skenario *Use Case* Melihat *List Batch* Budidaya

Melihat *list* produk budidaya	
Objective	Aktor dapat melihat *list* produk budidaya dari induk yang dimiliki
Actor	Pengguna
Pre-condition	Aktor berada pada *activity breedingRecord*
Main flow	1. Aktor menekan pada salah satu *list* induk burung yang tersedia 2. Sistem menampilkan *activity listProduct* yang berisikan *list batch* budidaya yang pernah terjadi pada induk yang dipilih aktor
Alternative flow	-
Post-condition	Sistem menampilkan *activity listProduct* yang berisikan *list batch* budidaya yang pernah terjadi pada induk yang dipilih aktor

Tabel 4.15 Skenario *Use Case* Menambahkan *Log* Kegiatan Pada *Batch* Budidaya

Menambahkan detail kegiatan budidaya	
Objective	Aktor dapat menambahkan *log* kegiatan pada *batch* budidaya
Actor	Pengguna
Pre-condition	Aktor berada pada *activity list product* atau *activity detail log record*
Main flow	1. Aktor menekan *floating Action Menu* pada *activity listProduct* 2. Aktor memilih tambah record 3. Sistem menampilkan dialog berisikan form tentang data batch terakhir 4. Aktor mengisi form menambah log kegiatan 5. Aktor menekan tombol Tambah Log Sistem menampilkan dialog sukses
Alternative flow	5.1. Sistem menampilkan pesan *error* karena user memasukan *log* baru kedalam *batch* yang sudah selesai
Post-condition	Sistem menampilkan dialog sukses dan *log* kegiatan dari *batch* proses budidaya berhasil tersimpan di *database*

Tabel 4.16 Skenario *Use Case* Melihat *Detail Log Batch* Proses Budidaya

Melihat *detail log batch* proses budidaya	
Objective	Aktor dapat melihat *detail log* kegiatan dari salah satu *batch* proses budidaya
Actor	Pengguna
Pre-condition	Aktor berada pada activity list product
Main flow	1. Aktor memilih salah satu *batch* yang terdapat dalam *activity listProduct* 2. Sistem menampilkan *activity detailLogRecord* yang berisikan *list log* kegiatan yang terjadi dalam suatu *batch* selama proses budidadaya berlangsung
Alternative flow	-
Post-condition	Sistem menampilkan *activity detailLogRecord* yang berisikan *list log* kegiatan yang terjadi dalam suatu *batch* selama proses budidadaya berlangsung

Tabel 4.17 Skenario *Use Case* Menambah *Batch* Proses Budidaya

Menambah *batch* proses budidaya	
Objective	Aktor dapat menambahkan *batch* baru untuk proses budidaya yang dilakukan oleh suatu induk burung
Actor	Pengguna
Pre-condition	Aktor berada pada *activity breedingRecord* atau *activity listProduct*
Main flow	1. Aktor menekan *Floating Action Menu* yang terdapat dalam *activity* 2. Aktor memilih tambah *batch* baru 3. Sistem Menampilkan *form* berbentuk *dialog* yang mempunyai *spinner* yang berisikan *list* induk burung yang dimiliki pengguna 4. Aktor memilih induk burung yang akan ditambahakan *batch* proses budidaya yang baru
	5. Aktor menekan tombol tambah batch 6. Sistem menampilkan dialog sukses
Alternative flow	5.1. Sistem menampilkan pesan *error* karena *batch* sebelumnya belum selesai
Post-condition	Sistem menampilkan *dialog* sukses dan data *bach* berhasil tersimpan di *database*

Tabel 4.18 Skenario *Use Case* Mengakhriri *Batch* Proses Budidaya

Mengakhiri *batch* proses budidaya	
Objective	Aktor dapat mengakhiri suatu *batch* proses budidaya yang dilkaukan suatu induk burung
Actor	Pengguna
Pre-condition	Aktor berada pada *activity detail log record*
Main flow	1. Pengguna menekan tombol *close batch* pada *activity detail log record* 2. Sistem menampilkan *dialog* konfirmasi penutupan *batch* 3. Aktor menekan tombol *close batch* 4. Sistem menampilkan dialog sukses menutup *batch* proses budidaya
Alternative flow	-
Post-condition	Sistem dapat menampilkan dialog sukses menutup *batch* dan status *batch* yang lama berubah dari on *process* menjadi *finish*

Tabel 4.19 Skenario *Use Case* Melihat Galeri Burung

Melihat galeri burung	
Objective	Aktor dapat Melihat koleksi foto burung
Actor	Pengguna
Pre-condition	Aktor berada pada activity dashboard
Main flow	1. Aktor memilih menu *gallery*
	2. Sistem menampilkan activity gallery yang berisikan kumpulan foto burung milik aktor
Alternative flow	-
Post-condition	Sistem menampilkan *activity gallery* yang berisikan kumpulan foto burung milik aktor

Tabel 4.20 Skenario *Use Case* Menambah Foto Pada Galeri Burung

Menambah foto pada galeri burung	
Objective	Aktor dapat menambahkan foto dari burung yang sedang dibudidayakan
Actor	Pengguna
Pre-condition	Aktor berada pada *activity gallery*
Main flow	1. Aktor menekan tombol "*add image"* yang terdapat pada *activity gallery*

Tabel 4.20 Skenario *Use Case* Menambah Foto Pada Galeri Burung (Lanjutan)

	2. Sistem menampilkan *form* untuk aktor dapat menambahkan foto burung baru 3. Aktor mengisi *form* untuk menambah foto baru 4. Aktor menekan tombol *upload* Foto Burung Sistem menampilkan *dialog* sukses
Alternative flow	-
Post-condition	Sistem dapat menampilkan *dialog* sukses dan foto yang baru di-*upload* tersimpan di *database*

Tabel 4.21 Skenario *Use Case* Melihat *List* Jurnal Keuangan

Melihat *list* jurnal keuangan	
Objective	Aktor dapat melihat *list* jurnal keuangan yang telah dibuat oleh user
Actor	Pengguna
Pre-condition	Aktor berada pada *activity dashboard*
Main flow	1. Aktor memilih menu *finance* 2. Sistem menampilkan *activity journalList* yang berisikan *list* jurnal keuangan yang sudah dibuat oleh user
Alternative flow	-
Post-condition	Sistem menampilkan *activity journalList* yang berisikan *list* jurnal keuangan yang sudah dibuat oleh user

Tabel 4.22 Skenario *Use Case* Membuat Jurnal Keuangan

Membuat jurnal keuangan	
Objective	Aktor dapat membuat jurnal keuangan baru
Actor	Pengguna
Pre-condition	Aktor berada pada *activity journalList*
Main flow	1. Aktor menekan *Floating Action Menu* yang berada pada sisi kanan bawah *activity*
	2. Aktor memilih buat jurnal 3. Sistem menampilkan dialog yang berisikan informasi bulan dan tahun waktu sekarang 4. Aktor menekan tombol buat jurnal keuangan 5. Sistem menampilkan dialog sukses

Tabel 4.22 Skenario *Use Case* Membuat Jurnal Keuangan (Lanjutan)

Alternative flow	4.1 Gagal membuat jurnal karena jurnal keuangan sudah terdaftar
Post-condition	Sistem menampilkan *dialog* sukses membuat jurnal dan jurnal yang baru tersimpan di *database*

Tabel 4.23 Skenario *Use Case* Melihat *List* Transaksi

Melihat list transaksi	
Objective	Aktor dapat melihat list transaksi yang terjadi selama kurun waktu satu bulan
Actor	Pengguna
Pre-condition	Aktor berada pada *activity journal list*
Main flow	1. Aktor menekan salah satu jurnal keuangan yang terdapat pada *activity journal list* 2. Sistem menampilkan *list* transaksi yang terjadi pada jurnal keuangan yang dipilih oleh aktor
Alternative flow	-
Post-condition	Sistem menampilkan *list* transaksi yang terjadi pada jurnal keuangan yang dipilih oleh aktor

Tabel 4.24 Skenario *Use Case* Membuat Transaksi

Membuat transaksi	
Objective	Aktor dapat melihat list transaksi yang terjadi selama kurun waktu satu bulan
Actor	Pengguna
Pre-condition	Aktor berada pada *activity journalList*
Main flow	1. Aktor menekan *Floating Action Menu* yang berada pada sisi kanan bawah *activity* 2. Aktor memilih buat transaksi 3. Sistem menampilkan form yang harus diisi aktor untuk menambah transaksi 4. Aktor melengkapi form untuk tambah transaksi 5. Aktor menekan tombol simpan transaksi 6. Sistem menampilkan *dialog* sukses
Alternative flow	-
Post-condition	Sistem menampilkan *dialog* sukses dan transaksi yang baru tersimpan di *database*

Tabel 4.25 Skenario *Use Case* Melihat Laporan Keuangan

Melihat laporan laba-rugi	
Objective	Aktor dapat melihat laporan laba-rugi yang berisikan informasi transaksi yang terjadi dalam kurun satu bulan
Actor	Pengguna
Pre-condition	Pengguna berada pada *activity detailJournal*
Main flow	1. Aktor menekan tombol "Lihat Laporan" yang terdapat pada *activity detailJournal* 2. Sistem menampilkan Laporan keuangan yang berisikan informasi transaksi yang terjadi dalam kurun waktu satu bulan
Alternative flow	-
Post-condition	Sistem menampilkan Laporan keuangan yang berisikan informasi transaksi yang terjadi dalam kurun waktu satu bulan

4.6 Analisis Data

Analisis data merupakan proses untuk mendapatkan struktur data yang nantinya akan dijadikan tabel dalam *database* sistem informasi management ternak burung *lovebird*, dari hasil analisis data yang dibutuhkan dalam sistem ini adalah sebagai berikut:

a. Data *user* merupakan data yang berisikan informasi terkait pengguna yang menggunakan sistem ini, data *user* memiliki atribut yaitu *id, email, password,* nama*,* alamant*,* lokasi*,* domisili, foto *profile,* tanggal lahir, nomor ktp, foto ktp, nomor *handphone,* jenis kelamin, *birdOwned, birdParent,* dan *Journal*.

b. Data *bird* merupakan data yang berisikan informasi terkait burung yang dimiliki oleh pengguna, data *bird* memiliki atribut yaitu *id, ring, species,* tipe burung, jenis kelamin*,* foto dna*,* usia*, parent,* dan foto burung*.*

c. Data *image* merupakan data yang berisikan data gambar dengan atributnya yaitu *id, src,* deskripsi*.*

d. Data *birdParent* merupakan data yang berisikan informasi terkait data burung yang akan dibudidayakan, data *birdParent* memiliki atribut *id,* data burung jantan*,* data burung betina, dan *breedingRecord*.

e. Data *breedingRecord* merupakan data yang berisikan informasi terkait *batch* yang terjadi pada induk burung selama proses budidaya, dalam *breedingRecord* memiliki atribut-atribut yaitu *id,* nama *batch, log,* dan status *batch.*

f. Data *breedingLog* merupakan detail kegiatan yang terjadi dalam suatu *batch* selama proses budidaya burung berlangsung, data *breedingLog* memiliki atribut yaitu *id,* waktu*,* deskripsi, tipe *log,* telur burung yang lahir, dan telur burung yang mati selama proses *breeding*.

g. Data *Journal* merupakan data yang merupakan laporan keuangan dalam jangka waktu tertentu, data *finance* memiliki beberapa atribut antaralain *id,* transaksi*,* waktu*,* dan saldo*.*

h. Data *transaction* merupakan detail pengeluaran ataupun pemasukan yang terjadi selama proses budidaya berlangsung, data *transaction* memiliki atribut yaitu *id,* tipe transaksi*,* deskripsi transaksi*,* jumlah*,* waktu*.*

BAB 5 PERANCANGAN DAN IMPLEMENTASI

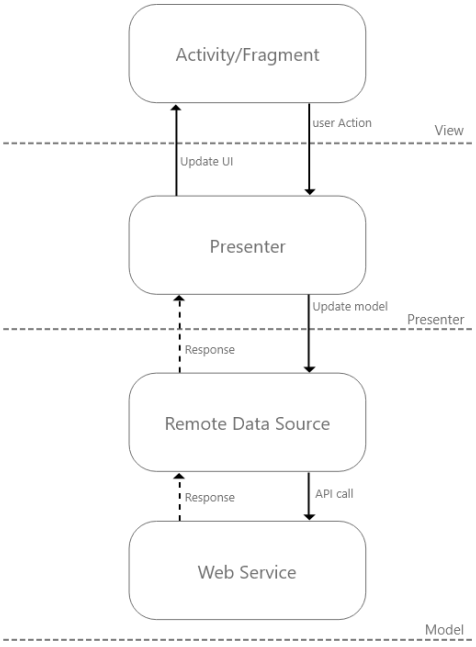
Pada bab ini menjelaskan tentang proses perancangan dan proses implementasi yang dilakukan untuk mengembangkan sistem informasi manajemen ternak burung *lovebird*. Pada tahap perancangan merupakan proses perancangan sistem dan hasil yang didapatkan dari proses perancangan akan menjadi dasar yang akan diimplementasikan untuk membangun sistem.

5.1 Perancangan

Proses perancangan berdasarkan terhadap hasil analisis kebutuhan yang telah didapatkan. Dalam proses perancangan terbagi menjadi empat buah tahapan yang masing-masing adalah perancagan arsitektur, perancangan data, perancangan komponen, perancangan antarmuka.

5.1.1 Perancangan Arsitektur Sistem

Sistem informasi manajemen ternak burung *lovebird* dikembangkan menggunakan *design pattern* MVP (*Model, View, Presenter*). Gambaran arsitektur yang akan dikembangkan akan digambarkan seperti pada Gambar 5.1. Pada gambar tersebut sistem terbagi menjadi 3 buah bagian yaitu *view* yang merupakan *acitivity/fragment* yang berguna menerima perintah dari pengguna sistem. Pada bagian *presenter* berfungsi untuk berinteraksi dengan bagian *model* dan melakukan perubahan tampilan pada *view.* Pada bagian *remote data source* merupakan *Apollo client* yang terhubung dengan *graphql* server berfungsi untuk mengambil data.



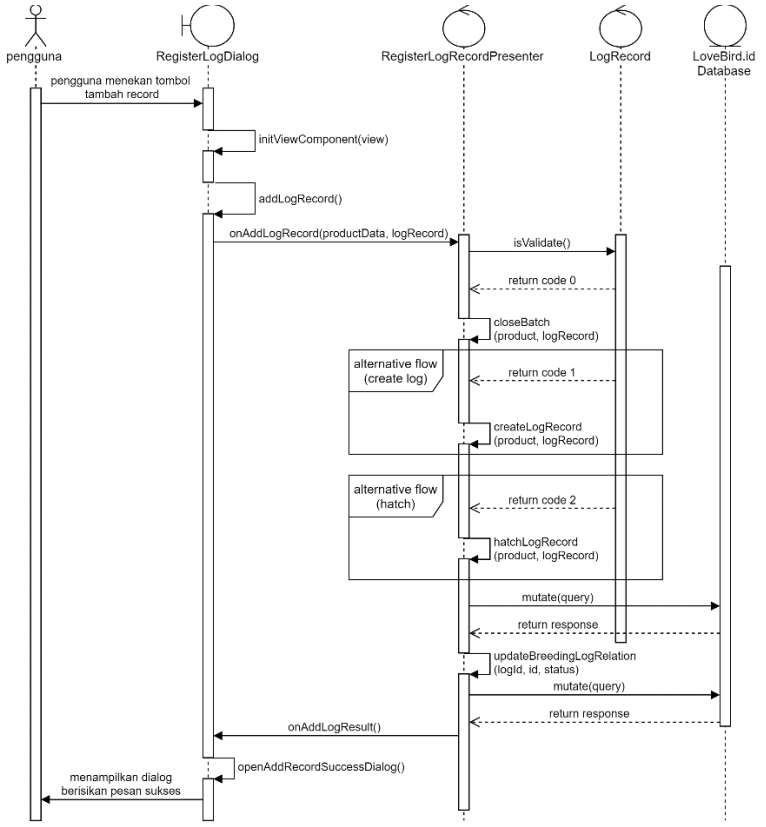
Gambar 5.1 Arsitektur Sistem Informasi Manajemen Ternak Burung Lovebird

5.1.2 Perancangan Diagram Sequence

Perancangan diagram *sequence* menunjukan bagaimana urutan suatu proses yang dijalankan oleh sistem untuk dapat memenuhi suatu kebutuhan yang terdapat dalam sistem. Skenario *use case* menjadi pedoman dalam rancangan pembuatan diagram *sequence*. Pada bagian perancangan ini akan diberikan contoh diagram *sequence* untuk memenuhi kebutuhan yaitu "menambah *Log* kegiatan pada *Batch* proses budidaya", "menambah burung", dan "melihat detail burung".

5.1.2.1 Perancangan Diagram *Sequence* Menambah *Log Batch* Proses Budidaya

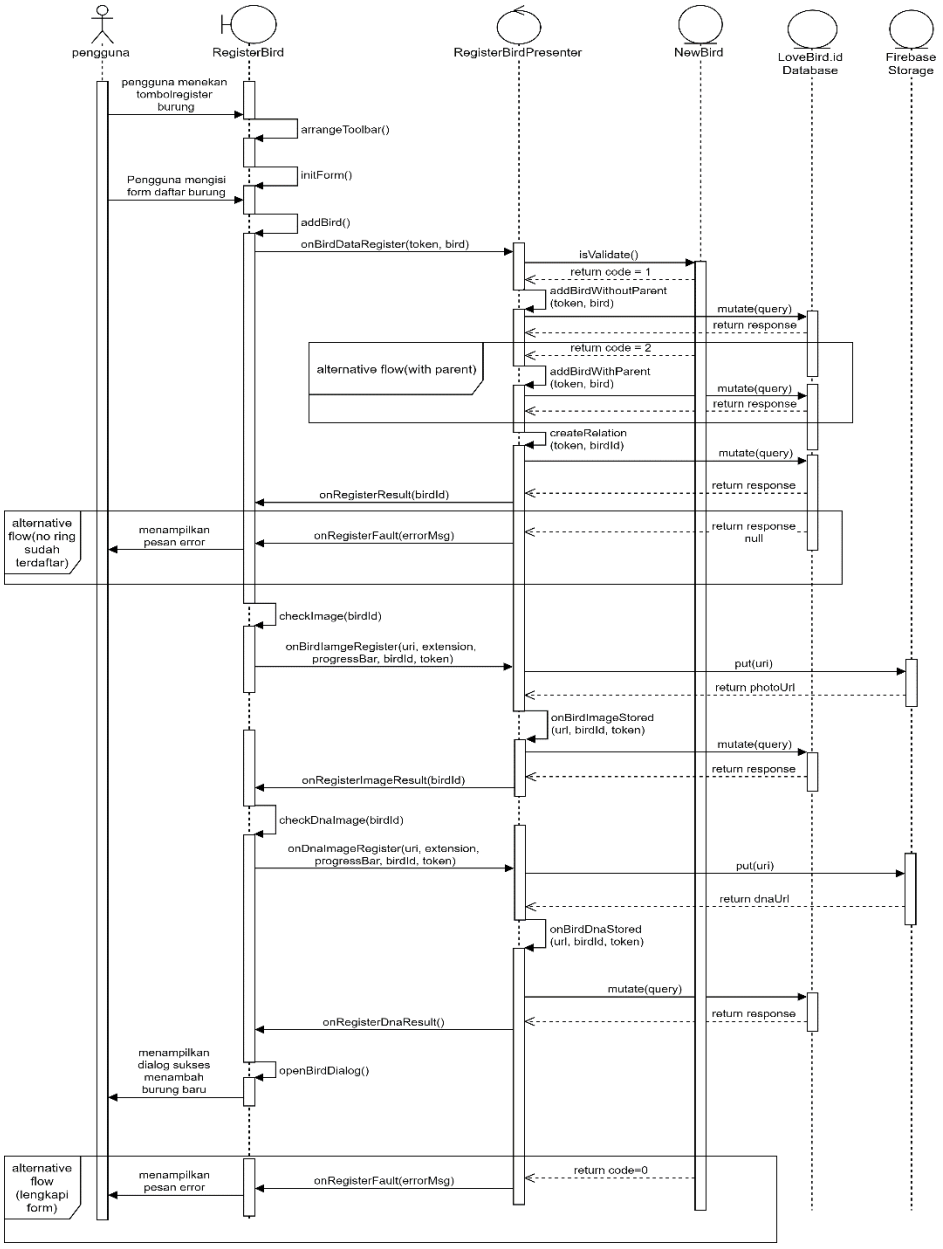
Pada Gambar 5.2 merupakan rancangan diagram *sequence* untuk kebutuhan menambah *log* kegiatan pada *batch* proses budidaya. Terdapat beberapa *class* yang terlibat untuk dapat memenuhi kebutuhan menambah *log* kegiatan pada *batch* proses budidaya yaitu *class RegisterLogRecordDialog* yang merupakan *view, RegisterLogRecordPresenter* merupakan *prensenter* yang berfungsi menghubungkan *view* dengan *database* sistem. Dalam proses menambahkan *log* Kegiatan hal pertama yang harus diisikan oleh pengguna adalah type *log* yang akan ditambahkan oleh pengguna. Tipe-tipe *log* tersebut terbagi menjadi 3 yaitu *breeding log*, bertelur, atau menetas. Setelah menentukan tipe dari *log* kegiatan maka pengguna harus melengkapi form sesuai dengan tipe yang dipilih, yang nantinya data tersebut akan disimpan kedalam database.



Gambar 5.2 Diagram *Sequence* Menambah *Log Batch* Proses Budidaya

5.1.2.2 Perancangan Diagram *Sequence* Menambah Burung

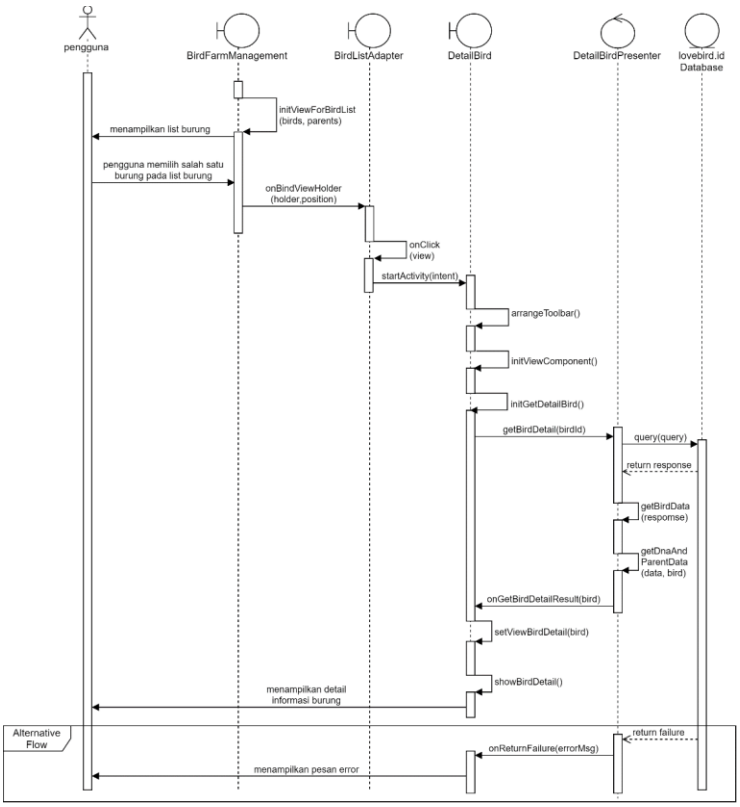
Pada Gambar 5.3 merupakan rancangan diagram *sequence* untuk menambah koleksi burung baru pengguna. Untuk memenuhi kebutuhan ini terdapat beberapa *class* yang terlibat yaitu *class RegisterBird* yang merupakan *view* dan *class RegisterBirdPresenter* yang merupakan *presenter*. Dalam proses menambah burung proses pertama adalah memasukan data terkait informasi burung baru. Setelah data berhasil disimpan di *database* selanjutnya, sistem akan melakukan pemeriksaan terhadap foto. Apabila ditemukan foto maka foto tersebut akan di upload kemudian link dari foto tersebut akan disimpan di *database*.



Gambar 5.3 Diagram *Sequence* Menambah Burung

5.1.2.3 Perancangan Diagram Sequence Melihat Detail Burung

Pada Gambar 5.4 merupakan rancangan diagram *sequence* untuk kebutuhan melihat detail burung yang dimiliki oleh pengguna. Terdapat beberapa *class* yang terlibat untuk dapat memenuhi kebutuhan ini yaitu *class BirdFarmManagement* dan *class DetailBird* yang merupakan *view, DetailBirdPreseneter* yang merupakan *prensenter* yang berfungsi menghubungkan *view* dengan *database* sistem. Dalam *view* terdapat fungsi yang berguna untuk mengatur tampilan serta memanggil fungsi *getBirdDetail* pada *class presenter* yang berguna untuk mengambil objek *bird* yang berisikan informasi burung. Dalam *presenter* terdapat fungsi yang berisikan *query* untuk mengambil data yang berisikan informasi burung dengan nomor *birdId* sesuai dengan yang didapatkan dan memanggil fungsi *onGetBirdDetailResult* untuk memperbarui tampilan sesuai dengan informasi burung yang didapat dari *database*.



Gambar 5.4 Diagram *Sequence* Melihat Detail Burung

5.1.3 Perancangan Class Diagram

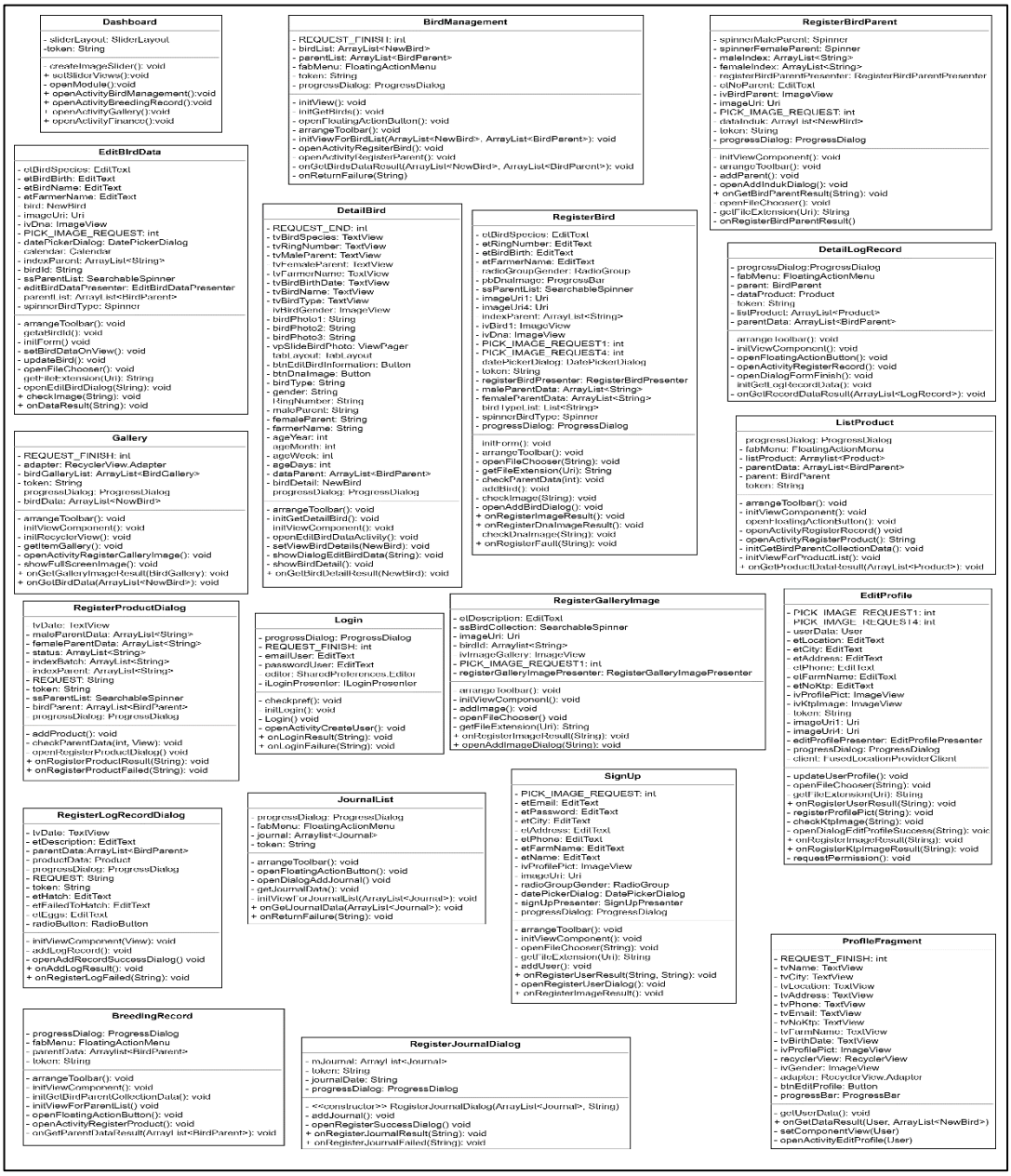
Perancangan *class diagram* bertujuan untuk mengetahui objek yang terdapat pada sistem yang terdapat pada suatu *class* dapat terhubung dengan objek yang terdapat pada *class* yang lain. Pada Gambar 5.5 merupakan gambar class diagram yang terdapat pada sistem informasi managemen ternak burung *lovebird*.

Gambar 5.5 *Class Diagram* Sistem Informasi Manajemen Ternak Burung Lovebird

Pada Gambar 5.5 terdapat beberapa jenis kelas yaitu kelas *activity* (sisi kiri gambar) yang merupakan *views* berfungsi untuk menampilkan sisi antarmuka yang terhubung langsung dengan pengguna. Kelas *presenter* (sisi tengah pada gambar) merupakan kelas yang berisikan perintah untuk mengerjakan suatu perintah. Kelas *model* (sisi kanan gambar) merupakan representasi dari objek-objek yang terdapat dalam sistem.

5.1.3.1 Informasi Kelas View

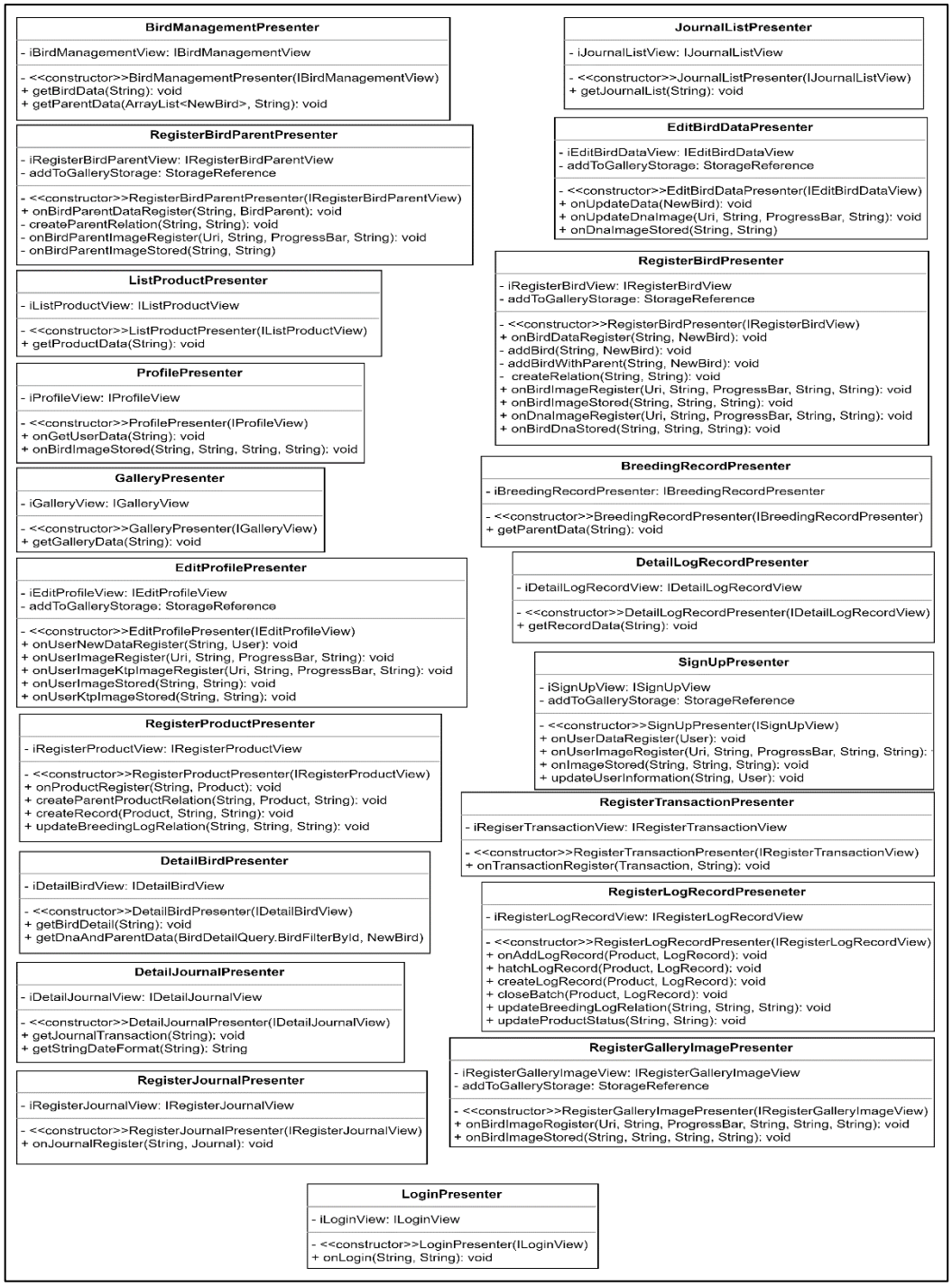
Fungsi dan atribut yang terdapat terdapat di setiap kelas view pada Gambar 5.5 dipresentasikan oleh Gambar 5.6



Gambar 5.6 Informasi *View Class*

5.1.3.2 Informasi Kelas Presenter

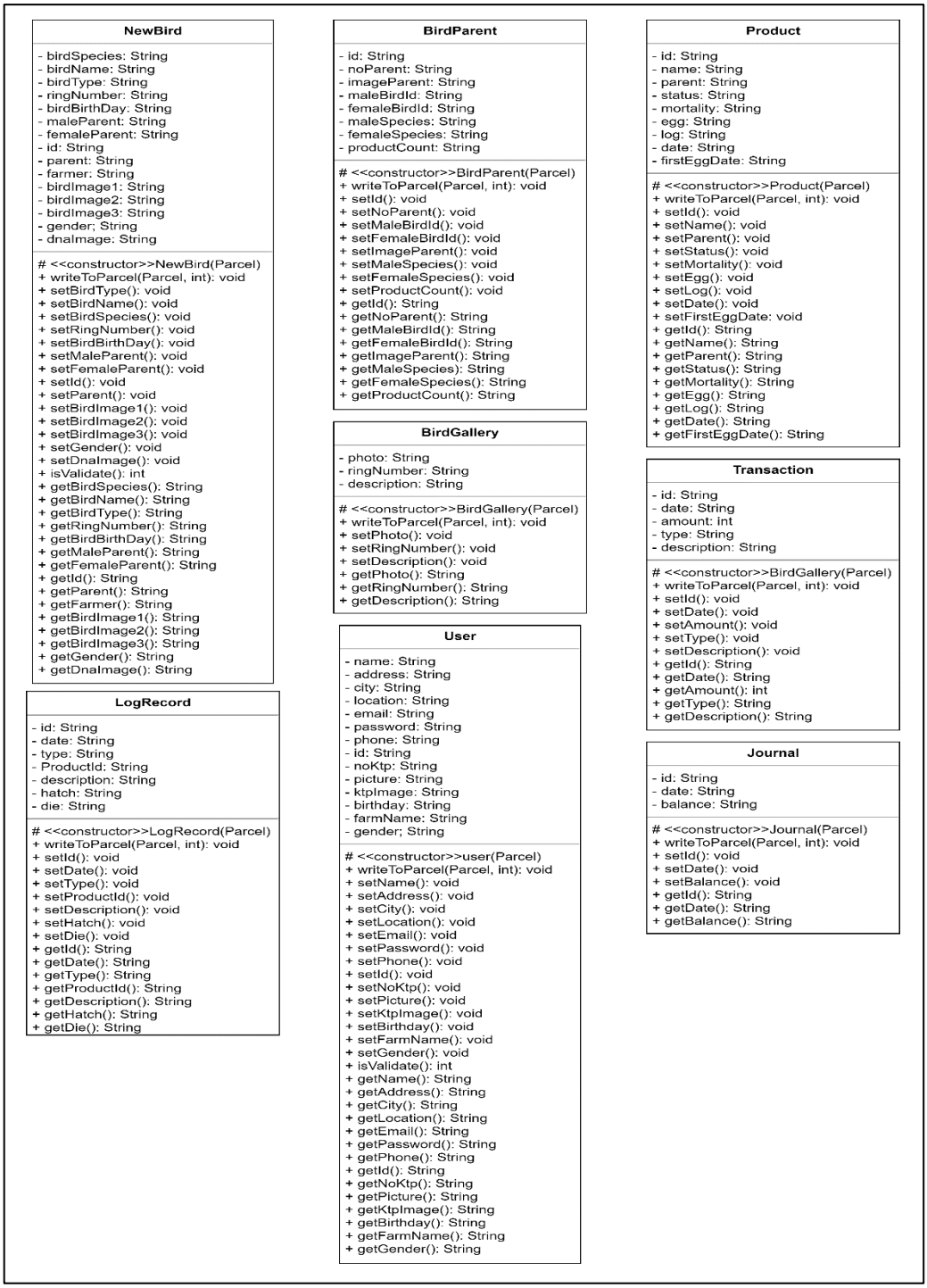
Fungsi dan atribut yang terdapat di kelas presenter pada Gambar 5.5 dipresentasikan oleh Gambar 5.7



Gambar 5.7 Informasi *Presenter Class*

5.1.3.3 Informasi Kelas Model

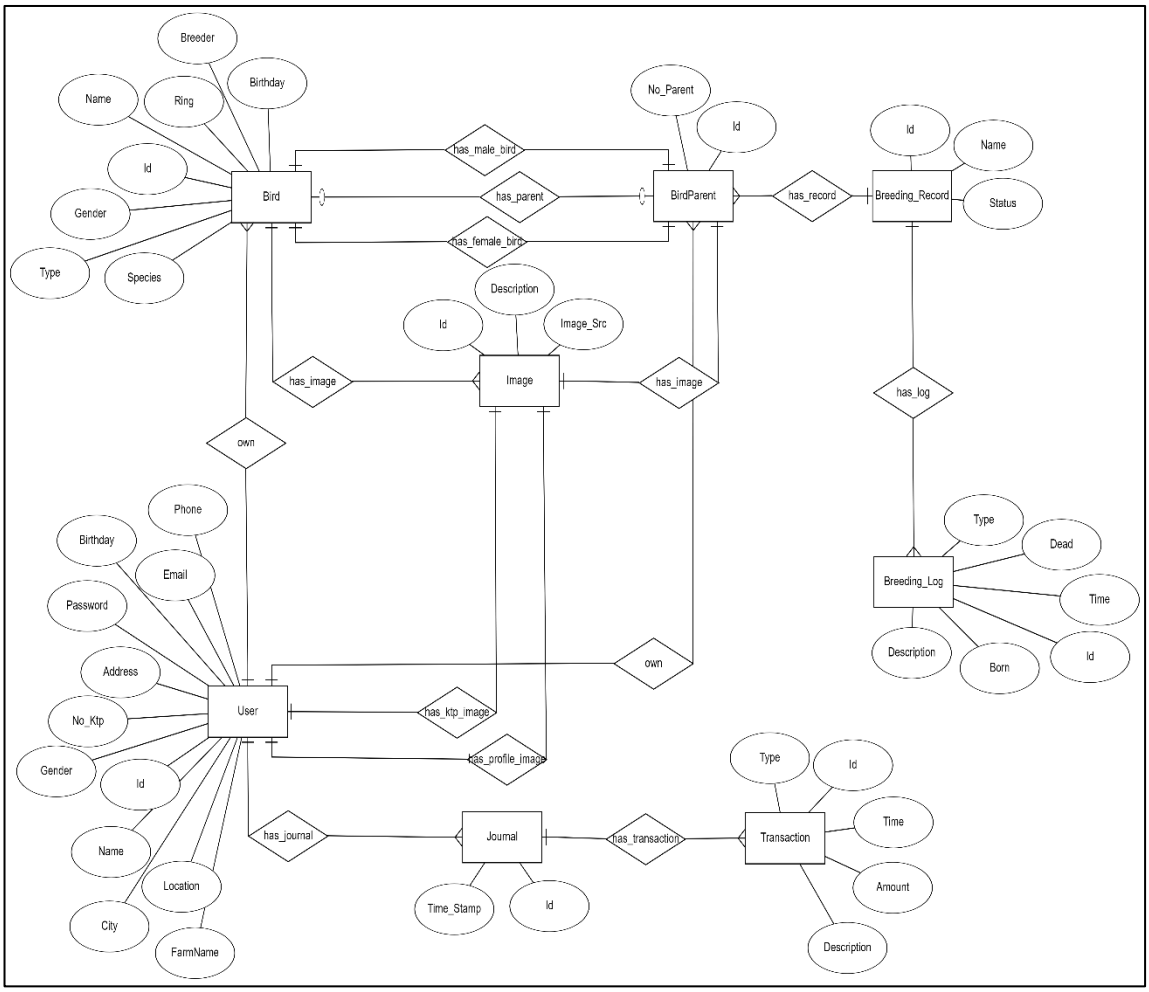
Fungsi dan atribut yang terdapat di kelas model pada Gambar 5.5 dipresentasikan oleh Gambar 5.8



Gambar 5.8 Informasi *Model Class*

5.1.4 Perancangan Data

Perancangan data bertujuan untuk mengetahui masing-masing atribut pada setiap *entity* yang terdapat pada sistem. Atribut dari masing masing *entity* didapatkan dari hasil analisis data yang sudah dilakukan pada tahap sebelumnya. Selain itu perancangan data bertujuan untuk mengetahui hubungan yang terdapat pada masing-masing *entity*. Hasil dari perancangan data akan terbentuk *Entity Relationship Diagram* (ERD) yang digambarkan pada Gambar 5.9.



Gambar 5.9 *Entity Relationship Diagram* Sistem Informasi Manajemen Ternak Burung *Lovebird*

Masing-masing *entity* akan menjadi dasar untuk proses implementasi data. Setiap atribut yang terdapat pada *entity* memiliki tipe data dan fungsinya masing-masing yang akan dijelaskan pada Tabel 5.1 hingga Tabel 5.8.

Pada Tabel 5.1 akan dijelaskan atribut yang terdapat pada *entity User*

1. *Entity User*
   Nama *entity* : *User*
   Total Atribut : 11
   Deskripsi : entity ini berisikan data diri dari pengguna

Tabel 5.1 *Entity User*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan sebagai *primary key* dimana setiap *user* memiliki *id* yang berbeda-beda	*Required*
2.	*Email*	*String*	Atribut yang berisikan *email* dari *user*	*Required*
3.	*Password*	*String*	Atribut yang berisikan *password* dari *user*	*Required*
4.	*Name*	*String*	Atribut yang berisikan nama dari *user*	*Required*
5.	*City*	*String*	Atribut yang berisikan domisili kota tempat *user* menetap	-
6.	*Location*	*String*	Atribut yang berisikan koordinat dari lokasi *user*	-
7.	*Address*	*String*	Atribut yang berisikan alamat tempat tinggal dari *user*	-
8.	*Gender*	*String*	Atribut yang berisikan jenis kelamin dari *user*	-
9.	No_Ktp	*String*	Atribut yang berisikan nomor identitas dari *user*	-
10.	*Phone*	*String*	Atribut yang berisikan nomor telpon dari *user*	-
11.	*FarmName*	*String*	Atribut yang berisikan nama peternakkan dari *user*	-
11.	*Birthday*	*String*	Atribut yang berisikan tanggal lahir dari *user*.	-

2. *Entity Bird*
   Nama entity : *Bird*
   Total Atribut : 8
   deskripsi : etity ini berisakan data tentang burung pengguna

Tabel 5.2 *Entity Bird*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan *primary key* untuk setiap burung yang didaftarkan oleh *user*	*Required*
2.	*Ring*	*String*	Atribut yang berisikan nomor ring dari burung.	*Required*
3.	*Species*	*String*	Atribut yang berisikan spesies dari burung.	*Required*
4.	*Type*	*String*	Atribut yang berisikan jenis dari burung.	*Required*
5.	*Gender*	*String*	Atribut yang berisikan jenis kelamin dari burung.	*Required*
6.	*Name*	*String*	Atribut yang berisikan nama dari burung.	*Required*
7.	*Breeder*	*String*	Atribut yang berisikan nama peternak dari burung.	*Required*
8.	*Birthday*	*String*	Atribut yang berisikan perkiraan tanggal lahir burung.	*Required*

3. *Entity BirdParent*
   Nama entity : *BirdParent*
   Total Atribut : 2
   deskrispi : etity ini berisakan data tentang induk burung yang dimiliki oleh pengguna

Tabel 5.3 *Entity BirdParent*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan *primary key* untuk setiap induk burung yang didaftarkan oleh *user*	*Required*
2.	*No_Parent*	*String*	Atribut yang berisikan perkiraan tanggal lahir burung.	*Required*

4. *Entity Image*
   Nama : *Image*
   Jumlah Atribut : 3
   Fungsi : Menyimpan informasi yang berkaitan dengan data yang berupa gambar yang terdapat

Tabel 5.4 *Entity Image*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan *primary key* untuk setiap foto burung yang diupload oleh *user*	*Required*
2.	*Image_Src*	*String*	Atribut yang berisikan link foto burung.	*Required*
3.	*Description*	*String*	Atribut yang berisikan deskripsi yang diberikan untuk burung.	*Required*

5. *Entity Breeding_Record*
   Nama : *Breeding_Record*
   Jumlah Atribut : 3
   Fungsi : Menyimpan informasi tentang proses *breeding record(batch)* yang dibuat oleh pengguna

Tabel 5.5 *Entity Breeding_Record*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan *primary key* untuk setiap *breeding record* yang dibuat oleh *user*	*Required*
2.	*Name*	*String*	Atribut yang berisikan nama *batch* proses *breeding* burung.	*Required*
3.	*Status*	*String*	Atribut yang berisikan status dari *batch* proses *breeding* burung.	*Required*

6. *Entity Breeding_Log*
   Nama : *Breeding_Log*
   Jumlah Atribut : 6
   Fungsi : Menyimpan informasi tentang *log* selama proses budidaya

Tabel 5.6 *Entity Breeding_Log*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan *primary key* untuk setiap *breeding log* yang dimasukan oleh *user*	*Required*
2.	*Type*	*String*	Atribut yang berisikan type *breeding log* burung.	*Required*
3.	*Description*	*String*	Atribut yang berisikan detail kegiatan saat proses *breeding* burung.	*Required*
4.	*Time*	*String*	Atribut yang berisikan waktu pencatatan *breeding log* burung.	*Required*
5.	*Born*	*String*	Atribut yang berisikan jumlah bayi burung yang lahir saat proses *breeding* burung.	-
6.	*Dead*	*String*	Atribut yang berisikan jumlah bayi burung yang gagal menetas saat proses *breeding* burung.	-

7. *Entity Journal*
   Nama entity : *Journal*
   Total Atribut : 2
   deskripsi : berisikan data yang berkaitan dengan jurnal keuangan

Tabel 5.7 *Entity Journal*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan *primary key* untuk setiap jurnal keuangan yang dibuat oleh *user*	*Required*
2.	*Time_Stamp*	*String*	Atribut yang berisikan bulan dan tahun jurnal keuangan dibuat	*Required*

8. *Entity Transaction*
   Nama entity : *Transaction*
   Total Atribut : 5
   Deskripsi : Menyimpan informasi tentang transaksi keuangan

Tabel 5.8 *Entity Transaction*

No.	*attribute*	*Data type*	*description*	*Required*
1.	*Id*	*ID*	Dijadikan *primary key* untuk setiap transaksi keuangan yang dibuat oleh *user*	*Required*
2.	*Time*	*String*	Atribut yang berisikan waktu transaksi keuangan dibuat	*Required*
3.	*Type*	*String*	Atribut yang berisikan jenis transaksi keuangan yang dibuat	*Required*
4.	*Amount*	*String*	Atribut yang berisikan jumlah uang yang terjadi saat transaksi keuangan yang dibuat	*Required*
5.	*Description*	*String*	Atribut yang berisikan deskripsi tentang kegiatan transaksi keuangan yang dibuat	*Required*

5.1.5 Perancangan Komponen

Perancangan komponen bertujuan untuk memberikan gambaran fungsi yang akan diimplementasikan kedalam sistem. Perancagan sistem berisikan kumpulan proses yang akan berjalan dalam sistem untuk mendapatkan hasil tertentu. Rancangan komponen dituliskan dalam bentuk *pseudocode*. Pada bagian perancangan komponen ini akan diberikan contoh perancangan komponen untuk '*onAddLogRecord'*, '*onBirdDataRegister'*, dan '*getBirdData'*.

5.1.5.1 Perancangan komponen Fungsi "*onAddLogRecord"*

Fungsi *onAddLogRecord* dijalankan untuk menambahkan kegiatan selama proses budidaya burung berlangsung. Fungsi ini terdapat pada kelas *RegisterLogRecordPresenter*. Kode 5.1 merupakan *pseudocode* dari fungsi *onAddLogRecord*. Fungsi ini berfungsi untuk menentukan jenis dari *breeding log* yang terjadi dalam suatu *batch* budidaya.

```
Algoritma fungsi onAddLogRecord(Product product, LogRecord logRecord)
1   if product status = finish
2      menampilkan pesan Batch product sudah selesai, Silahkan Membuat
    Batch Baru
3   else
4      inisialisasi variable validate
5      if validate = 0 maka
6          memanggil method closeBatch
7      else if validate = 1
8          memanggil method createLogRecord
9      else
10         memanggil method hatchLogRecord
11     end if-else
12  end if-else
```

Kode 5.1 Pseudocode fungsi *onAddLogRecord*

5.1.5.2 Perancangan Komponen Fungsi "*onBirdDataRegister*"

Fungsi *onBirdDataRegister* dijalankan untuk mendaftarkan data burung baru yang dimiliki oleh pengguna. Fungsi ini terdapat pada kelas *RegisterBirdPresenter*. Kode 5.2 merupakan *pseudocode* dari fungsi *onBirdDataRegister*.

```
Algoritma fungsi onBirdDataRegister(String token, NewBird newBird)
1   Inisialisasi variabel validation
2   if validation = 0 maka,
3      menjalankan method onRegisterFault(msg)
4   else if validation = 1 maka,
5      memanggil method addBirdWithoutParent
6   else
7      memanggil method addBirdWithParent
8   end if-else
```

Kode 5.2 Pseudocode fungsi *onBirdDataRegister*

5.1.5.3 Perancangan Komponen Fungsi "*getBirdData*"

Fungsi *getBirdData* dijalankan untuk mendapatkan informasi yang berisikan data detail dari salah satu burung yang dimiliki oleh pengguna. Fungsi ini terdapat pada kelas *DetailBirdPresenter*. Kode 5.3 merupakan *pseudocode* dari fungsi *getBirdData*.

```
Algoritma fungsi getBirdData(Response<BirdDetailQuery.Data> response)
1   inisialisasi variable data
2   inisialisasi objek burung
3   setBirdId pada objek burung dengan data.id
4   setBirdName pada objek burung dengan data.name
5   setBirdType pada objek burung dengan data.type
6   setBirdSpecies pada objek burung dengan data.species
7   setRingNumber pada objek burung dengan data.ring
8   setGender pada objek burung dengan data.gender
9   setBirdBirthDays pada objek burung dengan data.age
10  setFarmer pada objek burung dengan data.breeder
11  setBirdImage1 pada objek burung dengan data.image.get(0)
12  if data.image size == 2
13     setBirdImage2 pada objek burung dengan data.image.get(1)
14     getDnaAndParentData(data, burung)
15  else
16     setBirdImage2 pada objek burung dengan data.image.get(1)
17     setBirdImage3 pada objek burung dengan data.image.get(2)
18     getDnaAndParentData(data, burung)
19  end if
```

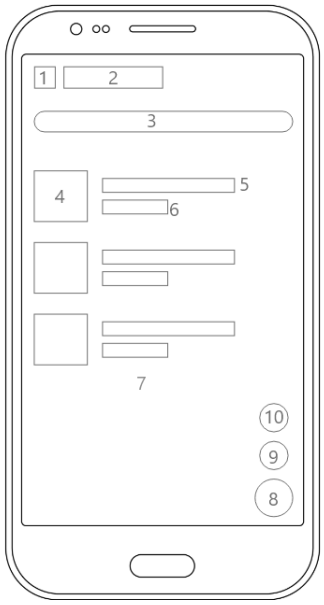
Kode 5.3 Pseudocode fungsi *getBirdData*

5.1.6 Perancangan Antarmuka

Perancangan antarmuka berisikan gambaran rancangan terhadap tampilan dari *activity* yang terdapat dalam sistem informasi manajemen ternak burung *lovebird*. Pada bagian perancagan antarmuka akan diberikan hasil akhir dari contoh perancagan antarmuka untuk *activity* 'tambah *BreedingLog Record*', 'registrasi burung', 'detail burung' setelah dilakukan perbaikan.

5.1.6.1 Perancangan Antarmuka Activity *BreedingLog Record*

Perancangan antarmuka untuk menambahkan *breeding log record* akan digambarkan seperti pada Gambar 5.10.



Gambar 5.10 Rancagan Antarmuka *Activity List* Burung

Setiap komponen yang terdapat pada Gambar 5.10 akan dijelaskan pada Tabel 5.9.

Tabel 5.9 Penjelasan Rancangan Antarmuka *Activity List* Burung

No.	Nama Objek	Tipe	Keterangan
1.	*back button*	*Button*	Tombol yang digunakan untuk kembali ke *activity* sebelumnya.
2.	*logo*	*ImageView*	*Image view* untuk tampilan dari logo sistem
3.	*bird collection tablayout*	*TabLayout*	berguna untk berpindah dari *fragment bird collection* ke *fragment parent collection*
4.	*bird image*	*ImageView*	*Image view* untuk tampilan foto burung

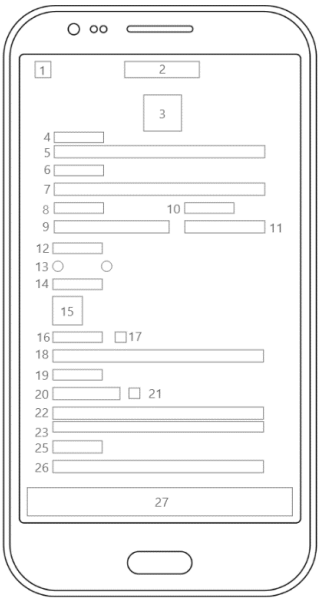
5.	*bird species*	*TextView*	*Text view* yang berisikan species burung

Tabel 5.9 Penjelasan Rancangan Antarmuka *Activity List* Burung (Lanjutan)

6.	*bird ring*	*TextView*	*Text view* yang berisikan nomor *ring* burung
7.	*bird list layout*	*RecyclerView*	*Layout* yang digunakan untuk menampilkan *list* burung
8.	*menu register*	*Floating Action Menu*	*Floating action menu* berisikan pilihan untuk melakukan registrasi
9.	*bird register*	*Floating Action Button*	*Button* yang digunakan untuk masuk kedalam *activity* registrasi burung
10.	*parent register*	*Floating Action Button*	*Button* yang digunakan untuk masuk kedalam *activity* registrasi induk

5.1.6.2 Perancagan Antarmuka *Activity* Registrasi Burung

Perancangan antarmuka untuk melakukan registrasi burung baru akan digambarkan seperti pada Gambar 5.11.



Gambar 5.11 Rancangan Antarmuka *Activity* Registrasi Burung

Setiap komponen yang terdapat pada Gambar 5.11 akan dijelaskan pada Tabel 5.10.

Tabel 5.10 Penjelasan Rancangan Antarmuka *Activity* Registrasi Burung

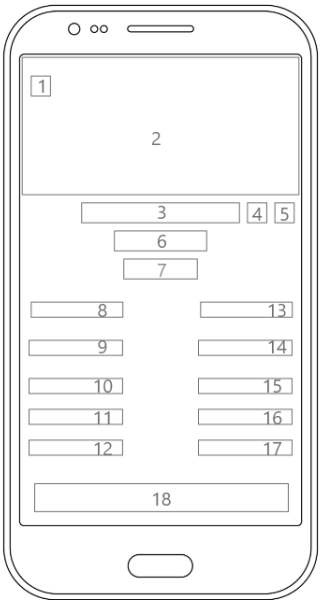
No.	Nama Objek	Tipe	Keterangan
1.	*back button*	*Button*	Tombol yang digunakan untuk kembali ke *activity* sebelumnya.
2.	*title*	*TextView*	*Text view* yang berisikan judul *activity*
3.	*bird photo image*	*ImageView*	*Image view* yang menjadi *preview* gambar yang akan di-*upload*
4.	*no ring text*	*TextView*	*Text view* yang berisikan tulisan 'no. ring'
5.	*no ring input*	*EditText*	*Edit text* tempat pengguna memasukan nomor *ring burung*
6.	*bird species text*	*TextView*	*Text view* yang berisikan tulisan 'warna mutasi'
7.	*bird species input*	*EditText*	*Edit text* tempat pengguna memasukan species burung
8.	*Bird type text*	*TextView*	*Text view* yang berisikan tulisan 'jenis burung'
9.	*bird type spinner*	*Spinner*	*Spinner* yang berisikan jenis-jenis burung *lovebird*
10.	*bird name text*	*TextView*	*Text view* yang berisikan tulisan 'Nama'
11.	*bird name input*	*EditText*	*Edit text* tempat pengguna memasukan nama burung
12.	*gender text*	*TextView*	*Text view* yang berisikan tulisan 'jenis kelamin'
13.	*gender option*	*RadioButton*	*Radio button* yang berisikan pilihan jenis kelamin
14.	*dna text*	*TextView*	*Text view* yang berisikan tulisan 'dna'
15.	*dna image*	*ImageView*	*Image view* yang menjadi preview gambar yang akan di-*upload*

Tabel 5.10 Penjelasan Rancangan Antarmuka *Activity* Registrasi Burung (Lanjutan)

16.	*birth text*	*TextView*	*Text view* yang berisikan tulisan 'birth'
17.	*calendar*	*ImageView*	*Image view* yang menjadi tombol untuk membuka *datepicker*
18.	*birthdate input*	*EditText*	*Edit text* yang berisikan nilai yang sesuai dengan tanggal pada *datepicker* yang telah dipilih pengguna
19.	*parent text*	*TextView*	*Text view* yang berisikan tulisan 'induk'
20.	*parent spinner*	*Spinner*	*Spinner* yang berisikan daftar id dari induk burung yang dimiliki oleh pengguna
21	*search button*	*ImageView*	*Image view* yang merupakan button yang apabila ditekan oleh pengguna akan menampilkan informasi dari induk burung
22	*male parent text*	*TextView*	*Text view* yang berisikan informasi dari induk jantan. *Text view* ini hanya muncul ketika *search button* ditekan oleh pengguna
23	*female parent text*	*TextView*	*Text view* yang berisikan informasi dari induk betina. *Text view* ini hanya muncul ketika *search button* ditekan oleh pengguna
24	*farmer text*	*TextView*	*Text view* yang berisikan tulisan 'peternak'
25	*farmer name input*	*EditText*	*Edit text* tempat pengguna memasukan nama peternak burung
26	*photo text*	*TextView*	*Text view* yang berisikan tulisan 'foto burung'
27	*register button*	*Button*	Tombol untuk melakukan registrasi burung baru

5.1.6.3 Perancangan Antarmuka *Activity* Detail Burung

Perancangan antarmuka untuk melihat detail dari koleksi burung yang dimiliki pengguna akan digambarkan seperti pada Gambar 5.12.



Gambar 5.12 Rancangan Antarmuka *Activity* Detail Burung

Setiap komponen yang terdapat pada Gambar 5.12 akan dijelaskan pada Tabel 5.11.

Tabel 5.11 Penjelasan Rancangan Antarmuka *Activity* Detail Burung

No.	Nama Objek	Tipe	Keterangan
1.	*back button*	*Button*	Tombol yang digunakan untuk kembali ke *activity* sebelumnya.
2.	*bird image*	*ImageSlider*	*Image Slider* untuk tampilan koleksi foto burung
3.	*bird species text*	*TextView*	*Text view* yang berisikan species burung
4.	*bird gender*	*ImageView*	*Image view* untuk tampilan logo jenis kelamin burung
5.	*dna button*	*Button*	Tombol yang digunakan untuk membuka *dialog fragment* yang berisikan foto dna burung

Tabel 5.11 Penjelasan Rancangan Antarmuka *Activity* Detail Burung (Lanjutan)

6.	*bird ring text*	*TextView*	*Text view* yang berisikan nomor *ring* burung
7.	*bird age text*	*TextView*	*Text view* yang berisikan umur burung
8.	*text "type* burung"	*TextView*	*Text view* dengan tulisan "*type burung"*
9.	*text "*nama burung"	*TextView*	*Text view* dengan tulisan "nama burung*"*
10.	*text "*induk jantan"	*TextView*	*Text view* dengan tulisan "induk jantan*"*
11.	*text "*induk betina"	*TextView*	*Text view* dengan tulisan "induk betina*"*
12.	*text "*peternak"	*TextView*	*Text view* dengan tulisan "peternak*"*
13.	*bird type text*	*TextView*	*Text view* yang berisikan tipe burung
14.	*bird name text*	*TextView*	*Text view* yang berisikan nama dari burung
15.	*male bird parent text*	*TextView*	*Text view* yang berisikan nomor ring induk jantan
16.	*female bird parent text*	*TextView*	*Text view* yang berisikan nomor ring induk betina
17.	*Farmer name text*	*TextView*	*Text view* yang berisikan nama dari peternak burung
18.	*Edit Information button*	*Button*	Tombol yang digunakan untuk berpindah ke *Activity EditBirdData*

5.2 Implementasi Sistem

Proses perancangan yang sudah dilakukan akan menjadi dasar dalam proses implementasi sistem. Tahap yang terjadi saat proses implementasi sistem terbagi menjadi tiga buah tahapan yang masing-masing adalah implemetasi data, implementasi kode program, dan implementasi antarmuka. Pada tahap implementasi sistem akan dijelaskan juga spesifikasi terkait perangkat yang digunakan dalam proses pengerjaan sistem. Spesifikasi yang akan dibahas antaralain spesifikasi perangkat lunak, spesifikasi perangkat keras, dan spesifikasi sistem operasi.

5.2.1 Spesifikasi Sistem

5.2.1.1 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak berisikan informasi terkait versi ataupun bahasa pemrograman yang digunakan untuk mendukung pengembangan sistem. Perangkat lunak tersebut dijelaskan pada tabel 5.12.

Tabel 5.12 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Editor Dokumentasi	Microsoft Office Word 2016
Editor Perancangan	Draw.io, Adobe XD 17.0.12.11 (Starter)
Editor Pemrograman	Android Studio 3.4
Bahasa Pemrograman	Java, XML

5.2.1.2 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras berisikan informasi terkait jenis perangkat keras yang digunakan dalam proses pengembangan sistem. Perangkat keras tersebut dijelaskan pada tabel 5.13.

Tabel 5.13 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Laptop	1. Asus X450JF 2. Processor Intel Core i7 4700HQ CPU @ 2.40GHz 3. Memory 8 GB DDR3L 4. Grafis nVidia GT 745M 2GB

5.2.1.3 Spesifikasi Sistem Operasi

Spesifikasi sistem operasi berisikan informasi terkait jenis dan versi dari sistem operasi yang digunakan dalam proses pengembangan sistem. Spesifikasi sistem operasi tersebut dijelaskan pada tabel 5.15.

Tabel 5.14 Spesifikasi Sistem Operasi

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 10 Home 64-bit

5.2.2 Implementasi Data

Implementasi data pada sistem akan menggunakan *graphql* API, yang berarti hasil dari perancangan data yang sudah dibuat akan diubah kedalam bentuk *schema* dari masing-masing *entity* yang ada. *Schema* tersebut akan secara otomatis dibuat menjadi *database* oleh prisma. Dalam Kode 5.4 merupakan implementasi *schema graphql* yang dibuat.

Schema.graphql

```
type User {
    id: ID!
    email: String!
    password: String!
    name: String
    farmName: String
    city: String
    location: String
    address: String
    phone: String
    journal: [Journal]
    birthday: String
    ktp: String
    ktpImage: Image
    gender: String
    image: Image
    birdOwned: [Bird]
    birdParent: [BirdParent]
}
type Journal {
    id: ID!
    transaction: [Transaction]
    timeStamp: String
}
type Transaction {
    id: ID!
    type: String
    description: String
    amount: String
    timeStamp: String
    isFinish: Boolean!
}
type Bird {
    id: ID!
    name: String
    ring: String!
    owner: User!
    breeder: String!
    species: String!
    type: String!
    gender: String!
    age: String!
    image: [Image]
    parent: BirdParent
    dna: Image    dna: Image
}
type BirdParent {
    id: ID!
    noParent: String!
    image: Image
    breedingRecord: [BreedingRecord]
    male: Bird!
    female: Bird!
}
type Image {
    id: ID!
    src: String!
    description: String!
}
```

Kode 5.4 GraphQL Schema

```
Schema.graphql
60  type BreedingRecord {
61      id: ID!
62      name: String
63      status: String
64      parent: BirdParent
65      log: [BreedingLog]
66  }
67  type BreedingLog {
68      id: ID!
69      type: String
70      born: String
71      dead: String
72      timeStamp: String!
73      description: String!
74  }
```

Kode 5.4 GraphQl Schema (Lanjutan)

5.2.3 Implementasi Kode Program

Implementasi ini kode program bertujuan untuk mengetahui kode program yang terdapat dalam sistem yang bertujuan untuk mencapai suatu tujuan tertentu. Beberapa kode program yang diambil untuk menjadi contoh adalah kode program untuk melihat list burung, menambah data burung baru, dan melihat informasi detail burung.

5.2.3.1 Implementasi Kode Program Fungsi "*onAddLogRecord*"

Fungsi ini memiliki tujuan untuk menambahkan *record* aktivitas yang terjadi selama proses budidaya burung berlangsung. Fungsi ini berada pada *class RegisterLogRecordPresenter.* Isi kode program yang terdapat dalam fungsi ini dijelaskan pada Kode 5.5.

```
fungsi onAddLogRecord(Product product, LogRecord logRecord)
1       public void onAddLogRecord(Product product, LogRecord logRecord)
    {
2           if(product.getStatus().equals("finish")){
3               iRegisterLogRecordView.onRegisterLogFailed("Batch Sudah
    Selesai, Silahkan Membuat Batch Baru");
4           }else{
5               int validate = logRecord.isValidate();
6               if (validate == 0) {
7                   closeBatch(product, logRecord);
8               } else if(validate == 1) {
9                   createLogRecord(product, logRecord);
10              } else if( validate == 2){
11                  hatchLogRecord(product, logRecord);
12              }
13          }
14      }
```

Kode 5.5 Implementasi Kode Program Fungsi *onAddLogRecord*

5.2.3.2 Implementasi Kode Program Fungsi "*onBirdDataRegister*"

Fungsi ini memiliki tujuan untuk mendaftarkan data burung baru kedalam database. Fungsi ini berada pada *class RegisterBirdPresenter*. Isi kode program yang terdapat di dalam fungsi *onBirdDataRegister* akan dijelaskan pada Kode 5.6.

```
fungsi onBirdDataRegister(String token, NewBird newBird)
1     public void onBirdDataRegister(String token, NewBird newBird) {
2         int validation = newBird.isValidate();
3         if (validation == 0) {
4             iRegisterBirdView.onRegisterFault("Please fill all
      form");
5         } else if (validation == 1) {
6            addBird(token, newBird);
7         }
8         else {
9             addBirdWithParent(token, newBird);
10        }
11    }
```

Kode 5.6 Implementasi Kode Program *onBirdDataRegister*

5.2.3.3 Implementasi Kode Program "*getBirdData*"

Fungsi ini memiliki tujuan untuk mendapatkan data burung sesuai dengan yang ingin dilihat oleh pengguna. Fungsi ini terdapat pada *class BirdDetailPresenter*. Isi kode program yang terdapat dalam fungsi ini dapat dilihat pada Kode 5.7.

```
fungsi getBirdData(Response<BirdDetailQuery.Data> response)
1         private void getBirdData(Response<BirdDetailQuery.Data>
      response) {
2         BirdDetailQuery.BirdFilterById data =
      response.data().birdFilterById();
3         NewBird newBird = new NewBird();
4         newBird.setId(data.id());
5         newBird.setBirdName(data.name());
6         newBird.setBirdType(data.type());
7         newBird.setBirdSpecies(data.species());
8         newBird.setRingNumber(data.ring());
9         newBird.setGender(data.gender());
10        newBird.setBirdBirthdays(data.age());
11        newBird.setFarmer(data.breeder());
12        newBird.setBirdImage1(data.image().get(0).src());
13        if (data.image().size() == 2) {
14            newBird.setBirdImage2(data.image().get(1).src());
15            getDnaAndParentData(data, newBird);
16        } else {
17            newBird.setBirdImage2(data.image().get(1).src());
18            newBird.setBirdImage3(data.image().get(2).src());
19            getDnaAndParentData(data, newBird);
20        }
21        iDetailBirdView.onGetBirdDetailResult(newBird);
22    }
```

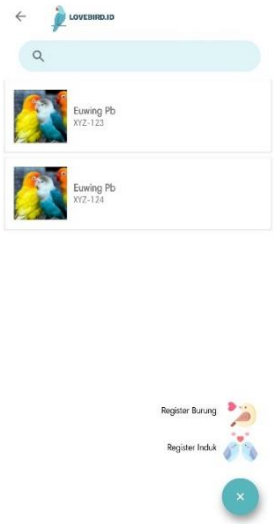
Kode 5.7 Implementasi Kode Program *getBirdData*

5.2.4 Implementasi Antarmuka

Implementasi antarmuka berisikan hasil implementasi dari tampilan sistem informasi manajemen ternak burung *lovebird* berbasis android yang telah dibuat. Beberapa contoh tampilan yang diambil adalah tampilan untuk menampilkan *list* burung yang dimiliki oleh pengguna, tampilan untuk detail informasi burung yang dimiliki pengguna, dan form yang digunakan oleh pengguna untuk menambah daftarkan burung baru.

5.2.4.1 Implementasi Antarmuka *Activity BirdManagement*

Pada tampilan antarmuka *activity BirdManagement* terdapat *list* yang berisikan informasi burung-burung yang sudah didaftarkan oleh pengguna. Selain itu diatas *list* terdapat *search-bar* yang digunakan untuk melakukan *filter* terhadap tampilan *list* burung sesuai dengan ketentuan yang dimasukan oleh user. Selain itu pada bagian kanan bawah dari *activity* terdapat *floating action menu* yang berisikan button untuk melakukan registrasi burung ataupun registrasi induk burung. Pada Gambar 5.13 merupakan hasil implementasi antarmuka dari *Activity BirdManagement.*



Gambar 5.13 Implementasi Antarmuka *Activity BirdManagement*

5.2.4.2 Implementasi Antarmuka *Activity DetailBird*

Pada tampilan antarmuka *activity Detailbird* berisikan *list* yang merupakan informasi tentang data diri burung. Pada bagian atas dari *activity* tersebut terdapat gambar yang merupakan foto dari burung. Apabila informasi terkait DNA burung sudah didaftarkan oleh pengguna maka akan muncul tombol bertuliskan DNA yang dapat digunakan untuk melihat foto bukti DNA yang merupakan bukti terkait jenis kelamin burung. Pada bagian bawah dari *activity* terdapat tombol yang digunakan untuk memperbarui data burung. Pada Gambar 5.14 merupakan hasil implementasi antarmuka dari *activity DetailBird.*

Gambar 5.14 Implementasi Antarmuka *Activity DetailBird*

5.2.4.3 Implementasi Antarmuka *Activity RegisterBird*

Pada tampilan antarmuka *activity RegisterBird* merupakan *form* yang digunakan pengguna untuk mendaftarkan burung baru. Pada bagian bawah dari *activity* terdapat tombol yang digunakan untuk melakukan proses registrasi data burung baru. Pada Gambar 5.15 merupakan hasil implementasi antarmuka dari *activity RegisterBird.*

Gambar 5.15 Implementasi Antarmuka *Activity RegisterBird*

BAB 6 PENGUJIAN

Pada tahap pengujian hasil yang didapatkan dari proses implementasi akan diuji. Tujuan dilakukannya pengujian seteleah proses implementasi adalah untuk mengetahui hasil yang diimplementasikan telah sesuai dengan proses perancangan dan memenuhi kebutuhan sistem. Proses pengujian pada penelitian ini menggunakan metode pengujian *white-box, black-box,* dan pengujian *usability.* Penggunaan metode pengujian *white-box* akan dilakukan pada saat pengujian *unit* dan pengujian integrasi dan penggunaan metode pengujian *black-box* akan dilakukan pada proses pengujian validasi.

6.1 Pengujian Unit

Pengujian unit bertujuan untuk melakukan verifikasi unit-unit yang terdapat pada perangkat lunak yang dikembangkan. Unit-unit tersebut merupakan fungsi ataupun prosedur yang dijalankan untuk mendapatkan hasil yang diinginkan. Pengujian unit juga bertujuan agar hasil yang didapatkan dari fungsi telah sama dengan hasil perancangan sistem. Penggunaan metode pengujain *white-box* dilakukan dengan menggunakan basis path testing. Basis path testing merupakan seluruh kemungkinan alur kode program yang dapat dijalankan untuk mendapatkan suatu hasil.

6.1.1 Pengujian Unit Fungsi "onAddLogRecord"

1. *Pseudocode*

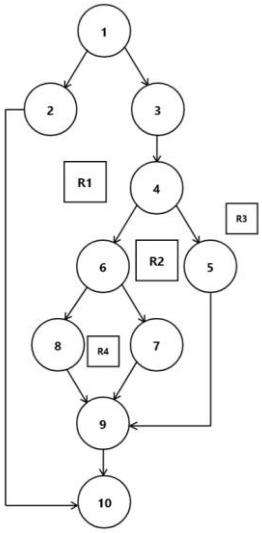
Fungsi *onAddLogRecord* terdapat pada kelas *RegisterLogRecrodPresenter*. Kode 6.1 merupakan *pseudocode* dari fungsi *onAddLogRecord*.

```
Algoritma fungsi onAddLogRecord(Product product, LogRecord logRecord)
1   if product status = finish
2      menampilkan pesan Batch product sudah selesai, Silahkan Membuat
    Batch Baru
3   else
       inisialisasi variable validate
4      if validate = 0 maka
5          memanggil method closeBatch
6      else if validate = 1
7          memanggil method createLogRecord
8      else
           memanggil method hatchLogRecord
9      end if-else
10  end if-else
```

Kode 6.1 Pseudocode Fungsi *onAddLogRecord*

2. *Basis Path Testing*

Hasil *flow graph* yang didapatkan dari *pseudocode* pada kode 6.1 digambarkan pada Gambar 6.1.



Gambar 6.1 *Flow Graph* fungsi *onAddLogRecord*

2.1. Cyclomatic Complexity
- V(G) = 4, terdapat 4 buah region (R1, R2, R3, R4)
- V(G) = 12 *edge(s)* – 10 *node(s)* + 2 = 4
- V(G) = 3 *predicate node(s)* + 1 = 4

2.2. Independent Path
- Jalur 1: 1-2-10
- Jalur 2: 1-3-4-5-9-10
- Jalur 3: 1-3-4-6-8-9-10
- Jalur 4: 1-3-4-6-7-9-10

Dari hasil *Basis path testing* tedapat 4 *independent path*. Masing-masing *path* merupakan kemungkinan alur kode program yang mungkin terjadi. Dilihat dari jumlah alur kode program yang mungkin terjadi, dapat dikatakan bahwa fungsi *onAddLogRecord* merupakan fungsi yang baik karena memiliki jumlah kurang dari 10, hal tersebut berarti algoritma yang terdapat pada fungsi *onAddLogRecord* memiliki kemungkinan *error* yang relatif lebih kecil. Pada Tabel 6.1. merupakan hasil dan kasus uji yang dijalankan pada fungi *onAddLogRecord.*

Tabel 6.1 Hasil Pengujian Unit Fungsi *onAddLogRecord* pada Kelas *RegisterLogRecordPresenter*

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	Memberikan nilai pada variable status dengan nilai = "*finish*"	menampilkan tulisan "*test passed*" dan "Batch produk sudah selesai, silahkan membuat batch baru"	menampilkan tulisan "*test passed*" dan "Batch produk sudah selesai, silahkan membuat batch baru"	Valid

Tabel 6.1 Hasil Pengujian Unit Fungsi *onAddLogRecord* pada Kelas *RegisterLogRecordPresenter (Lanjutan)*

2.	2	Memberikan nilai pada variable status dengan nilai = "*on process*" dan variable validate = 0	menampilkan tulisan "*test passed*" dan "*close batch*"	menampilkan tulisan "*test passed*" dan "*close batch*"	Valid
3.	3	Memberikan nilai pada variable status dengan nilai = "*on process*" dan variable validate = 1	menampilkan tulisan "*test passed*" dan "*Create Log Record*"	menampilkan tulisan "*test passed*" dan "*Create Log Record*"	Valid
4.	4	Memberikan nilai pada variable status dengan nilai = "*on process*" dan variable validate = 2	menampilkan tulisan "*test passed*" dan "*Hatch Log Record*"	menampilkan tulisan "*test passed*" dan "*Hatch Log Record*"	Valid

6.1.2 Pengujian Unit Fungsi "onBirdDataRegister"

1. *Pseudocode*

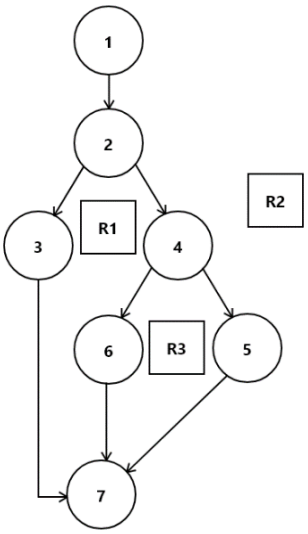
Fungsi *onBirdDataRegister* terdapat pada kelas *RegisterBirdPresenter*. Kode 6.2 merupakan *pseudocode* dari fungsi *onBirdDataRegister*.

```
Algoritma fungsi onBirdDataRegister(String token, NewBird newBird)
1   Inisialisasi variabel validation
2   if validation = 0 maka,
3      menjalankan method onRegisterFault(msg)
4   else if validation = 1 maka,
5      memanggil method addBirdWithoutParent
6   else
       memanggil method addBirdWithParent
7   end if-else
```

Kode 6.2 Pseudocode *onBirdDataRegister*

2. *Basis Path Testing*

Hasil *flow graph* yang didapatkan dari *pseudocode* pada kode 6.2 digambarkan pada Gambar 6.2.



Gambar 6.2 *Flow Graph* Fungsi *onBirdDataRegister*

2.1. Cyclomatic Complexity
- V(G) = 3, terdapat 3 buah region (R1, R2, R3)
- V(G) = 8 *edge(s)* – 7 *node(s)* + 2 = 3
- V(G) = 2 *predicate node(s)* + 1 = 3

2.2. Independent Path
- Jalur 1: 1-2-3-7
- Jalur 2: 1-2-4-6-7
- Jalur 3: 1-2-4-5-7

Dari hasil *Basis path testing* tedapat 3 *independent path*. Masing-masing *path* merupakan kemungkinan alur kode program yang mungkin terjadi. Dilihat dari jumlah alur kode program yang mungkin terjadi, dapat dikatakan bahwa fungsi *onBirdDataRegister* merupakan fungsi yang baik karena memiliki jumlah kurang dari 10, hal tersebut berarti algoritma yang terdapat pada fungsi *onBirdDataRegister* memiliki kemungkinan *error* yang relatif lebih kecil. Pada Tabel 6.2. merupakan hasil dan kasus uji yang dijalankan pada fungi *onBirdDataRegister.*

Tabel 6.2 Hasil Pengujian Unit Fungsi *onBirdDataRegister* pada Kelas *RegisterBirdPresenter*

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	Memberikan nilai pada variable validation dengan nilai = 0	menampilkan tulisan “test passed” dan “*please fill all form*”	menampilkan tulisan “test passed” dan “*please fill all form*”	Valid

Tabel 6.2 Hasil Pengujian Unit Fungsi *onBirdDataRegister* pada Kelas *RegisterBirdPresenter* (Lanjutan)

2.	2	Memberikan nilai pada variable validation dengan nilai = 1	menampilkan tulisan "test passed" dan "*add bird without parent*"	menampilkan tulisan "test passed" dan "*add bird without parent*"	Valid
3.	3	Memberikan nilai pada variable validation dengan nilai = 2	menampilkan tulisan "test passed" dan "*add bird with parent*"	menampilkan tulisan "test passed" dan "*add bird with parent*"	Valid

6.1.3 Pengujian Unit Fungsi"getBirdData"

1. *Pseudocode*

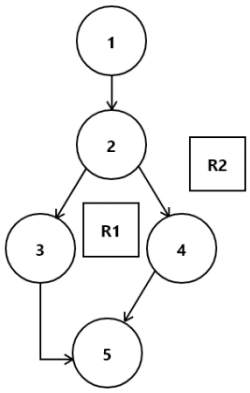
Fungsi *getBirdData* terdapat pada kelas *DetailBirdPresenter*. Kode 6.3 merupakan *pseudocode* dari fungsi *getBirdData*.

```
Algoritma fungsi getBirdData(Response<BirdDetailQuery.Data> response)
1   inisialisasi variable data
    inisialisasi objek burung
    setBirdId pada objek burung dengan data.id
    setBirdName pada objek burung dengan data.name
    setBirdType pada objek burung dengan data.type
    setBirdSpecies pada objek burung dengan data.species
    setRingNumber pada objek burung dengan data.ring
    setGender pada objek burung dengan data.gender
    setBirdBirthDays pada objek burung dengan data.age
    setFarmer pada objek burung dengan data.breeder
    setBirdImage1 pada objek burung dengan data.image.get(0)
2   if data.image size == 2
3      setBirdImage2 pada objek burung dengan data.image.get(1)
       getDnaAndParentData(data, burung)
4   else
       setBirdImage2 pada objek burung dengan data.image.get(1)
       setBirdImage3 pada objek burung dengan data.image.get(2)
       getDnaAndParentData(data, burung)
5   end if
```

Kode 6.3 Pseudocode Fungsi *getBirdData*

2. *Basis Path Testing*

Hasil *flow graph* yang didapatkan dari *pseudocode* pada kode 6.3 digambarkan pada Gambar 6.3.



Gambar 6.3 *Flow Graph* Fungsi *getBirdDetail*

2.1. Cyclomatic Complexity
- V(G) = 2, terdapat 2 buah region (R1, R2)
- V(G) = 5 *edge(s)* – 5*node(s)* + 2 = 2
- V(G) = 1 *predicate node(s)* + 1 = 2

2.2. Independent Path
- Jalur 1: 1-2-3-5
- Jalur 2: 1-2-4-5

Dari hasil *Basis path testing* tedapat 2 *independent path*. Masing-masing *path* merupakan kemungkinan alur kode program yang mungkin terjadi. Dilihat dari jumlah alur kode program yang mungkin terjadi, dapat dikatakan bahwa fungsi *getBirdDetail* merupakan fungsi yang baik karena memiliki jumlah kurang dari 10, hal tersebut berarti algoritma yang terdapat pada fungsi *getBirdDetail* memiliki kemungkinan *error* yang relatif lebih kecil. Pada Tabel 6.3. merupakan hasil dan kasus uji yang dijalankan pada fungi *getBirdDetail.*

Tabel 6.3 Hasil Pengujian Unit Fungsi *getBirdDetail* Pada Kelas *DetailBirdPresenter*

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	memberikan nilai pada data().image().size =2	menampilkan tulisan “test passed” dan “*object bird has* 2 *images*”	menampilkan tulisan “*test passed*” dan “*object bird has* 2 *images*”	Valid
2.	2	memberikan nilai pada data().image().size =3	menampilkan tulisan “test passed” dan “*object bird has* 3 *images*”	menampilkan tulisan “test passed” dan “*object bird has* 3 *images*”	Valid

6.2 Pengujian Validasi

Pengujian validasi merupakan pengujian terhadap fungsi-fungsi kebutuhan yang terdapat dalam sistem. Pengujian ini bertujuan untuk memastikan bahwa setiap fungsi yang terdapat pada perangkat lunak telah memenuhi kebutuhan dan sesuai dengan yang telah diharapkan. Untuk melakukan pengujian validasi pada penelitian ini menggunakan metode *black-box*. Pengujian dengan menggunakan metode *black-box* akan mendapatkan seluruh kemungkinan hasil yang didapatkan dari suatu *input* pada suatu kebutuhan.

6.2.1 Pengujian Validasi Sign Up

Pengujian validasi *sign up* bertujuan untuk mengetahui apakah calon pengguna perangkat lunak dapat mendaftarkan diri ke sistem. Kasus Uji validasi *sign up* dijelaskan pada Tabel 6.4 dan Tabel 6.5

Tabel 6.4 Hasil Pengujian Validasi *Sign Up*

Kode Kebutuhan	BFM-F-01
Nama Kasus Uji	*Sign Up*
Prosedur	1. Menekan tombol *sign up* pada *activity login* 2. Mengisi seluruh *form* yang tersedia 3. Menekan tombol *register*
Hasil yang diharapkan	Sistem menyimpan data pengguna
Hasil	Sistem menyimpan data pengguna
Status	Valid

Tabel 6.5 Hasil Pengujian Validasi *Sign Up* Alternatif 1

Kode Kebutuhan	BFM-F-01
Nama Kasus Uji	*Sign Up*
Prosedur	1. Menekan tombol *sign up* 2. Mengisi seluruh *form* dengan *email* yang sudah terdaftar 3. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan error "*email* telah terdaftar"
Hasil	Sistem menampilkan pesan error "*email* telah terdaftar"
Status	Valid

6.2.2 Pengujian Validasi Login

Pengujian validasi *login* bertujuan untuk apakah pengguna yang sudah terdaftar bisa masuk kedalam sistem. Kasus Uji validasi *login* dijelaskan pada Tabel 6.6 dan Tabel 6.7

Tabel 6.6 Hasil Pengujian Validasi *Login*

Kode Kebutuhan	BFM-F-02
Nama Kasus Uji	*Login*
Prosedur	1. Mengisi *email* dan *password* dengan benar 2. Menekan tombol *login*
Hasil yang diharapkan	Sistem menampilkan *activity dashboard*
Hasil	Sistem menampilkan *activity dashboard*
Status	Valid

Tabel 6.7 Hasil Pengujian Validaasi *Login* Alternatif 1

Kode Kebutuhan	BFM-F-02
Nama Kasus Uji	*Login*
Prosedur	1. Mengisi *email* dan *password* secara acak 2. Menekan tombol *login*
Hasil yang diharapkan	Sistem menampilkan pesan error "*email* atau *password salah*"
Hasil	Sistem menampilkan pesan error "*email* atau *password salah*"
Status	Valid

6.2.3 Pengujian Validasi Melihat Profile

Pengujian validasi melihat *profile* bertujuan untuk mengetahui apakah calon pengguna dapat melihat *profile* pengguna. Kasus Uji validasi melihat *profile* dijelaskan pada Tabel 6.8

Tabel 6.8 Hasil Pengujian Validasi Melihat *Profile*

Kode Kebutuhan	BFM-F-03
Nama Kasus Uji	Melihat *profile*
Prosedur	1. Menekan *icon profile* pada *activity dashboard*
Hasil yang diharapkan	Sistem menampilkan *activity profile*

Tabel 6.8 Hasil Pengujian Validasi Melihat *Profile* (Lanjutan)

Hasil	Sistem menampilkan *activity profile*
Status	Valid

6.2.4 Pengujian Validasi Edit Profile

Pengujian validasi *edit profile* bertujuan untuk mengetahui apakah calon pengguna dapat melihat *profile* pengguna. Kasus Uji validasi *edit profile* dijelaskan pada Tabel 6.9

Tabel 6.9 Hasil Pengujian Validasi *Edit Profile*

Kode Kebutuhan	BFM-F-04
Nama Kasus Uji	Melihat *profile*
Prosedur	1. Menekan tombol *edit profile* yang ada di *activity profile* 2. Mengubah data yang terdapat pada *form edit profile* 3. Menekan tombol *update profile*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan Sistem menyimpan data pengguna yang baru
Hasil	Sistem menampilkan pesan sukses dan Sistem menyimpan data pengguna yang baru
Status	Valid

6.2.5 Pengujian Validasi Melihat List Burung

Pengujian validasi melihat *list* burung bertujuan untuk mengetahui apakah pengguna dapat melihat burung yang dimiliki pengguna dan terdaftar di dalam sistem. Kasus Uji validasi melihat *list* burung dijelaskan pada Tabel 6.10

Tabel 6.10 Hasil Pengujian Validasi Melihat *List* Burung

Kode Kebutuhan	BFM-F-05
Nama Kasus Uji	Melihat *list* burung
Prosedur	1. Menekan *menu birdfarm management*
Hasil yang diharapkan	Sistem menampilkan *activity birdfarm management* yang berisikan *list* burung yang dimiliki pengguna
Hasil	Sistem menampilkan *activity birdfarm management* yang berisikan *list* burung yang dimiliki pengguna
Status	Valid

6.2.6 Pengujian Validasi Melihat Detail Burung

Pengujian validasi melihat *detail* burung bertujuan untuk mengetahui apakah pengguna dapat melihat detail burung yang dimiliki oleh pengguna. Kasus Uji validasi melihat *detail* burung dijelaskan pada Tabel 6.11

Tabel 6.11 Hasil Pengujian Validasi Melihat *Detail* Burung

Kode Kebutuhan	BFM-F-06
Nama Kasus Uji	Melihat *detail* burung
Prosedur	1. Menekan salah satu foto burung yang terdapat pada *list* burung yang tersedia
Hasil yang diharapkan	Sistem menampilkan *activity detail bird* yang berisikan informasi burung yang dipilih pengguna
Hasil	Sistem menampilkan *activity detail bird* yang berisikan informasi burung yang dipilih pengguna
Status	Valid

6.2.7 Pengujian Validasi Edit Informasi Burung

Pengujian validasi *edit* informasi burung bertujuan untuk mengetahui apakah pengguna dapat mengubah informasi burung yang dimiliki. Kasus Uji validasi *edit* informasi burung dijelaskan pada Tabel 6.12

Tabel 6.12 Hasil Pengujian Validasi *Edit* Informasi Burung

Kode Kebutuhan	BFM-F-07
Nama Kasus Uji	*Edit* informasi burung
Prosedur	1. Menekan tombol *edit information* yang terdapat pada *activity detail bird* 2. Mengubah data yang terdapat pada *form edit bird* 3. Menekan tombol *update information*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan Sistem menyimpan informasi burung yang sudah diperbarui
Hasil	Sistem menampilkan pesan sukses dan Sistem menyimpan informasi burung yang sudah diperbarui
Status	Valid

6.2.8 Pengujian Validasi Menambah Burung Baru

Pengujian validasi menambah burung baru bertujuan untuk mengetahui apakah pengguna dapat menyimpan data burung baru kedalam sistem. Kasus Uji validasi menambah burung baru dijelaskan pada Tabel 6.13, Tabel 6.14, dan Tabel 6.15

Tabel 6.13 Hasil Pengujian Validasi Menambah Burung Baru

Kode Kebutuhan	BFM-F-08
Nama Kasus Uji	Menambah burung baru
Prosedur	1. Menekan tombol *Floating Action Menu* pada *activity birdfarm management* 2. Memilih menu registrasi burung 3. Mengisi *form* secara lengkap 4. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan Sistem menyimpan informasi burung yang baru
Hasil	Sistem menampilkan pesan sukses dan Sistem menyimpan informasi burung yang baru
Status	Valid

Tabel 6.14 Hasil Pengujian Validasi Menambah Burung Baru Alternatif 1

Kode Kebutuhan	BFM-F-08
Nama Kasus Uji	Menambah burung baru
Prosedur	1. Menekan tombol *Floating Action Menu* pada *activity birdfarm management* 2. Memilih menu registrasi burung 3. Tidak mengisi *form* secara lengkap 4. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan error "*Please fill all form"*
Hasil	Sistem menampilkan pesan error "*Please fill all form"*
Status	Valid

Tabel 6.15 Hasil Pengujian Validasi Menambah Burung Baru Alternatif 2

Kode Kebutuhan	BFM-F-08
Nama Kasus Uji	Menambah burung baru
Prosedur	1. Menekan tombol *Floating Action Menu* pada *activity birdfarm management* 2. Memilih menu registrasi burung 3. Mengisi *form* dengan data burung yang sudah terdaftar 4. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan error
Hasil	Sistem menampilkan pesan error*"*
Status	Valid

6.2.9 Pengujian Validasi Menambah Induk Burung Baru

Pengujian validasi menambah induk burung baru bertujuan untuk mengetahui apakah pengguna dapat menyimpan data burung baru kedalam sistem. Kasus Uji validasi menambah induk burung baru dijelaskan pada Tabel 6.16

Tabel 6.16 Hasil Pengujian Validasi Menambah Induk Burung Baru

Kode Kebutuhan	BFM-F-09
Nama Kasus Uji	Menambah induk burung baru
Prosedur	1. Menekan tombol *Floating Action Menu* pada *activity birdfarm management* 2. Memilih menu registrasi Induk 3. Mengisi *form* secara lengkap 4. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan Sistem menyimpan informasi induk burung yang baru
Hasil	Sistem menampilkan pesan sukses dan Sistem menyimpan informasi induk burung yang baru
Status	Valid

6.2.10 Pengujian Validasi Melihat List Induk Burung

Pengujian validasi melihat *list* induk burung bertujuan untuk mengetahui apakah pengguna dapat melihat induk burung yang dimiliki pengguna dan terdaftar di dalam sistem. Kasus Uji validasi melihat *list* induk burung dijelaskan pada Tabel 6.17.

Tabel 6.17 Hasil Pengujian Validasi Melihat *List* Induk Burung

Kode Kebutuhan	BFM-F-10
Nama Kasus Uji	Melihat *list* induk burung
Prosedur	1. Menekan *menu breeding record*
Hasil yang diharapkan	Sistem menampilkan *activity breeding record* yang berisikan *list* induk burung yang dimiliki pengguna
Hasil	Sistem menampilkan *activity birdfarm management* yang berisikan *list* burung yang dimiliki pengguna
Status	Valid

6.2.11 Pengujian Validasi Melihat List Batch

Pengujian validasi melihat *list* bertujuan untuk mengetahui apakah pengguna dapat melihat *batch* yang dimiliki oleh masing-masing induk burung. Kasus Uji validasi melihat *list batch* dijelaskan pada Tabel 6.18.

Tabel 6.18 Hasil Pengujian Validasi Melihat *List Batch*

Kode Kebutuhan	BFM-F-11
Nama Kasus Uji	Melihat *list batch* induk
Prosedur	1. Menekan salah satu induk burung yang terdapat pada *list* induk burung
Hasil yang diharapkan	Sistem menampilkan *activity list product* yang berisikan *list batch* yang dimiliki induk burung
Hasil	Sistem menampilkan *activity list product* yang berisikan *list batch* yang dimiliki induk burung
Status	Valid

6.2.12 Pengujian Validasi Membuat Batch Baru

Pengujian validasi membuat *batch* baru bertujuan untuk mengetahui apakah pengguna dapat membuat *batch* baru untuk proses budidaya. Kasus Uji validasi membuat *batch* baru dijelaskan pada Tabel 6.19 dan Tabel 6.20.

Tabel 6.19 Hasil Pengujian Validasi Membuat *Batch* Baru

Kode Kebutuhan	BFM-F-12
Nama Kasus Uji	Membuat *batch* baru
Prosedur	1. Menekan tombol *Floating Action Menu* 2. Memilih menu register *batch* 3. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan data *batch* baru tersimpan
Hasil	Sistem menampilkan pesan sukses dan data *batch* baru tersimpan
Status	Valid

Tabel 6.20 Hasil Pengujian Validasi Membuat *Batch* Baru Alternatif 1

Kode Kebutuhan	BFM-F-12
Nama Kasus Uji	Membuat *batch* baru
Prosedur	1. Menekan tombol *Floating Action Menu* 2. Memilih menu register *batch* 3. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan error "*batch sebelumnya belum selesai"*
Hasil	Sistem menampilkan pesan error "*batch sebelumnya belum selesai"*
Status	Valid

6.2.13 Pengujian Validasi Melihat Detail Log

Pengujian validasi melihat *detail log* bertujuan untuk mengetahui apakah pengguna dapat melihat *list log* yang terjadi selama suatu *batch* proses budidaya berlangsung. Kasus Uji validasi melihat *detail log* dijelaskan pada Tabel 6.21.

Tabel 6.21 Hasil Pengujian Validasi Melihat *Detail Log*

Kode Kebutuhan	BFM-F-13
Nama Kasus Uji	Melihat *detail log*
Prosedur	1. Menekan salah satu proses *batch* induk burung
Hasil yang diharapkan	Sistem menampilkan *activity detail log* yang berisikan *list log* yang terjadi selama proses *batch* budidaya
Hasil	Sistem menampilkan *activity detail log* yang berisikan *list log* yang terjadi selama proses *batch* budidaya
Status	Valid

6.2.14 Pengujian Validasi Menambah Log

Pengujian validasi menambah *log* bertujuan untuk mengetahui apakah pengguna dapat melihat menambahkan *log* kegiatan yang terjadi selama suatu *batch* proses budidaya berlangsung. Kasus Uji validasi menambah *log* dijelaskan pada Tabel 6.22 dan Tabel 6.23.

Tabel 6.22 Hasil Pengujian Validasi Menambah *Log*

Kode Kebutuhan	BFM-F-14
Nama Kasus Uji	Menambah *log*
Prosedur	1. Menekan tombol *Floating Action Menu* 2. Memilih menu *register log* 3. Isi *form register log* 4. Menekan tombol *add log*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan *log* kegiatan baru tersimpan
Hasil	Sistem menampilkan pesan sukses dan *log* kegiatan baru tersimpan
Status	Valid

Tabel 6.23 Hasil Pengujian Validasi Menambah *Log* Alternatif 1

Kode Kebutuhan	BFM-F-14
Nama Kasus Uji	Menambah *log*
Prosedur	1. Menekan tombol *Floating Action Menu* 2. Memilih menu *register log* 3. Isi *form register log*

Tabel 6.23 Hasil Pengujian Validasi Menambah *Log* Alternatif 1 (Lanjutan)

	4. Menekan tombol add log
Hasil yang diharapkan	Sistem menampilkan pesan error "*batch* sudah selesai, silahkan membuat *batch* baru"
Hasil	Sistem menampilkan pesan error "*batch* sudah selesai, silahkan membuat *batch* baru"
Status	Valid

6.2.15 Pengujian Validasi Mengakhiri Batch

Pengujian validasi melihat mengakhiri *batch* bertujuan untuk mengetahui apakah pengguna dapat melihat mengakhiri suatu proses budidaya ternak burung. Kasus Uji validasi mengakhiri *batch* dijelaskan pada Tabel 6.24.

Tabel 6.24 Hasil Pengujian Validasi Mengakhiri *Batch*

Kode Kebutuhan	BFM-F-15
Nama Kasus Uji	Mengakhiri *batch*
Prosedur	1. Menekan tombol "*close batch*" pada activity *list log* Menekan tombol "*Ok"*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan status *batch* berubah menjadi *finish*
Hasil	Sistem menampilkan pesan sukses dan status *batch* berubah menjadi *finish*
Status	Valid

6.2.16 Pengujian Validasi Melihat Galeri Burung

Pengujian validasi melihat galeri burung bertujuan untuk mengetahui apakah pengguna dapat melihat galeri burung yang berisikan foto-foto burung yang di-*upload* oleh pengguna. Kasus Uji validasi melihat galeri burung dijelaskan pada Tabel 6.25.

Tabel 6.25 Hasil Pengujian Validasi Melihat Galeri Burung

Kode Kebutuhan	BFM-F-16
Nama Kasus Uji	Melihat galeri burung
Prosedur	1. Menekan *menu galeri*
Hasil yang diharapkan	Sistem menampilkan *gallery* yang berisikan foto burung
Hasil	Sistem menampilkan *gallery* yang berisikan foto burung
Status	Valid

6.2.17 Pengujian Validasi Menambah Foto Burung

Pengujian validasi menambah foto burung bertujuan untuk mengetahui apakah pengguna dapat melihat galeri burung yang berisikan foto-foto burung yang di-*upload* oleh pengguna. Kasus Uji validasi menambah foto burung dijelaskan pada Tabel 6.26.

Tabel 6.26 Hasil Pengujian Validasi Menambah Foto Burung

Kode Kebutuhan	BFM-F-17
Nama Kasus Uji	Menambah foto burung
Prosedur	1. Menekan tombol *Add photo* yang terdapat pada *activity gallery* 2. Pilih foto dan isi *form* untuk upload foto
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan foto burung berhasil tersimpan
Hasil	Sistem menampilkan pesan sukses dan foto burung berhasil tersimpan
Status	Valid

6.2.18 Pengujian Validasi Filter List Burung

Pengujian validasi *filter list* burung bertujuan untuk mengetahui apakah pengguna melakukan pencarian dari *list* burung sesuai dengan keinginan pengguna. Kasus Uji validasi *filter list* burung dijelaskan pada Tabel 6.27 dan Tabel 6.28.

Tabel 6.27 Hasil Pengujian Validasi *Filter List* Burung

Kode Kebutuhan	BFM-F-18
Nama Kasus Uji	*Filter list* burung
Prosedur	1. Mengisikan nomor ring burung yang ingin dicari pada *search bar* pada *activity birdfarm management*
Hasil yang diharapkan	Sistem menampilkan *list* burung yang memiliki nomor ring seperti yang dimasukan oleh pengguna
Hasil	Sistem menampilkan *list* burung yang memiliki nomor ring seperti yang dimasukan oleh pengguna
Status	Valid

Tabel 6.28 Hasil Pengujian Validasi *Filter List* Burung Alternatif 1

Kode Kebutuhan	BFM-F-18
Nama Kasus Uji	*Filter list* burung
Prosedur	1. Mengisikan warna mutase burung yang ingin dicari pada *search bar* pada *activity birdfarm management*

Tabel 6.28 Hasil Pengujian Validasi *Filter List* Burung Alternatif 1 (Lanjutan)

Hasil yang diharapkan	Sistem menampilkan *list* burung yang mengandung mutase warna seperti yang dimasukan oleh pengguna
Hasil	Sistem menampilkan *list* burung yang mengandung mutase warna seperti yang dimasukan oleh pengguna
Status	Valid

6.2.19 Pengujian Validasi Melihat List Journal Keuangan

Pengujian validasi melihat *list* jurnal keuangan bertujuan untuk mengetahui apakah pengguna dapat melihat *list* jurnal keuangan yang pernah dibuat. Kasus Uji validasi melihat *list* jurnal keuangan dijelaskan pada Tabel 6.29.

Tabel 6.29 Hasil Pengujian Validasi Melihat *List Journal* Keuangan

Kode Kebutuhan	BFM-F-19
Nama Kasus Uji	Melihat *list* jurnal keuangan
Prosedur	1. Menekan *menu finance*
Hasil yang diharapkan	Sistem menampilkan *activity finance* yang berisikan *list* jurnal keuangan yang pernah dibuat oleh pengguna
Hasil	Sistem menampilkan *activity finance* yang berisikan *list* jurnal keuangan yang pernah dibuat oleh pengguna
Status	Valid

6.2.20 Pengujian Validasi Membuat Journal Keuangan

Pengujian validasi menambah jurnal keuangan bertujuan untuk mengetahui apakah pengguna dapat menyimpan data jurnal keuangan baru kedalam sistem. Kasus Uji validasi menambah jurnal keuangan baru dijelaskan pada Tabel 6.30 dan Tabel 6.31

Tabel 6.30 Hasil Pengujian Validasi Membuat *Journal* Keuangan

Kode Kebutuhan	BFM-F-20
Nama Kasus Uji	Menambah jurnal keuangan baru
Prosedur	1. Menekan tombol *Floating Action Menu* 2. Memilih menu buat jurnal keuangan 3. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan Sistem menyimpan data jurnal baru
Hasil	Sistem menampilkan pesan sukses dan Sistem menyimpan data jurnal baru
Status	Valid

Tabel 6.31 Hasil Pengujian Validasi Membuat *Journal* Keuangan Alternatif 1

Kode Kebutuhan	BFM-F-20
Nama Kasus Uji	Menambah jurnal keuangan baru
Prosedur	1. Menekan tombol *Floating Action Menu* 2. Memilih menu buat jurnal keuangan 3. Menekan tombol *register*
Hasil yang diharapkan	Sistem menampilkan pesan error "jurnal sudah ada"
Hasil	Sistem menampilkan pesan error "jurnal sudah ada"
Status	Valid

6.2.21 Pengujian Validasi Melihat List Transaksi Keuangan

Pengujian validasi melihat *list* transaksi keuangan bertujuan untuk mengetahui apakah pengguna dapat melihat *list* transaksi yang terjadi didalam jurnal keuangan tertentu (kurun waktu 1 bulan). Kasus Uji validasi melihat *detail log* dijelaskan pada Tabel 6.32.

Tabel 6.32 Hasil Pengujian Validasi Melihat *List* Transaksi Keuangan

Kode Kebutuhan	BFM-F-21
Nama Kasus Uji	Melihat *list* transaksi keuangan
Prosedur	1. Menekan salah satu jurnal keuangan yang terdapat pada *list* jurnal keuangan
Hasil yang diharapkan	Sistem menampilkan *list* transaksi yang terjadi pada jurnal keuangan yang dipilih oleh pengguna
Hasil	Sistem menampilkan *list* transaksi yang terjadi pada jurnal keuangan yang dipilih oleh pengguna
Status	Valid

6.2.22 Pengujian Validasi Menambah Transaksi Keuangan

Pengujian validasi menambah transaksi keuangan bertujuan untuk mengetahui apakah pengguna dapat menambahkan proses transaksi kedalam jurnal keuangan yang dimiliki. Kasus Uji validasi menambah transaksi keuangan dijelaskan pada Tabel 6.33.

Tabel 6.33 Hasil Pengujian Validasi Menambah Transaksi Keuangan

Kode Kebutuhan	BFM-F-22
Nama Kasus Uji	Menambah transaksi keuangan
Prosedur	1. Menekan tombol *Floating Action Menu* 2. Memilih menu tambah transaksi keuangan

Tabel 6.33 Hasil Pengujian Validasi Menambah Transaksi Keuangan (Lanjutan)

	3. Isi form tambah transaksi 4. Menekan tombol add
Hasil yang diharapkan	Sistem menampilkan pesan sukses dan data transaksi baru berhasil tersimpan
Hasil	Sistem menampilkan pesan sukses dan data transaksi baru berhasil tersimpan
Status	Valid

6.2.23 Pengujian Validasi Melihat Laporan Laba-Rugi

Pengujian validasi melihat laporan laba-rugi bertujuan untuk mengetahui apakah pengguna dapat melihat laporan laba-rugi yang terjadi dalam kurun waktu satu bulan. Kasus Uji validasi melihat laporan laba-rugi dijelaskan pada Tabel 6.34.

Tabel 6.34 Hasil Pengujian Validasi Melihat Laporan Laba-Rugi

Kode Kebutuhan	BFM-F-19
Nama Kasus Uji	Melihat laporan laba-rugi
Prosedur	1. Menekan tombol "*lihat laporan"*
Hasil yang diharapkan	Sistem menampilkan laporan laba-rugi yang terjadi dalam kurun waktu satu bulan
Hasil	Sistem menampilkan laporan laba-rugi yang terjadi dalam kurun waktu satu bulan
Status	Valid

6.3 Pengujian Usability

Pengujian *usability* dilakukan dengan cara memberikan sistem yang akan diuji kepada beberapa responden yang dipilih secara acak untuk mencoba menggunakan sistem tersebut. Kualitas dari *usability* sistem akan dinilai menggunakan metode *post-study system usability questionnaire* (PSSUQ). Dalam PSSUQ terdapat beberapa aspek yang diuji antara lain: kualitas dari sistem, kualitas informasi pada sistem, dan kualitas antarmuka.

6.3.1 Prosedur Pengujian

Pengujian *usability* berlangsung dengan memberikan kuesioner kepada responden yang nantinya akan menjadi calon pengguna sistem yang dipilih secara acak. Responden akan mencoba sistem terlebih dahulu sebelum memberikan pendapat pada kuesioner. Dalam penilaian *usability* sistem menggunakan metode PSSUQ terdapat 15 buah pertanyaan yang mewakili 3 aspek penilaian. Jawaban dari pertanyaan berupa skala 1-7 dimana semakin besar nilai yang diberikan oleh

responden menandakan bahwa responden semakin setuju dengan pernyataan yang ada. Pertanyaan yang terdapat pada kuesioner dapat dilihat pada Tabel 6.35

Tabel 6.35 Daftar Pertanyaan PSSUQ

No	Kategori	Pertanyaan
1	*System Quality*	Secara keseluruhan, Saya puas karena dapat mengoperasikan sistem ini dengan mudah.
2		Sangat mudah untuk menggunakan Sistem ini.
3		Saya dapat dengan mudah menyelesaikan tugas dengan cepat menggunakan sistem ini
4		Saya merasa nyaman ketika menggunakan sistem ini
5		Sangat mudah untuk mempelajari bagaimana cara menggunakan sistem ini
6		Saya percaya bahwa Saya bisa menjadi lebih produktif dengan adanya sistem ini
7	*Information Quality*	Apabila terjadi kesalahan, sistem ini memberikan pesan error yang memberitahukan bagaimana cara memperbaiki kesalahan tersebut
8		Informasi yang diberikan oleh sistem (pesan pada layar) sangat jelas
9		Sangat mudah untuk mencari informasi yang Saya butuhkan dalam sistem ini
10		Informasi yang diberikan membatu Saya dalam menyelesaikan tugas
11		Informasi pada sistem Sangat dibutuhkan
12	*Interface Quality*	Saya suka dengan Antarmuka pada sistem ini
13		Antarmuka pada sistem ini enak untuk dilihat
14		Sistem ini memiliki kemampuan dan fungsi yang Saya harapkan harus ada pada sistem
15		Secara keseluruhan, Saya puas dengan sistem ini

6.3.2 Hasil Pengujian Usability

Setelah proses uji *usability* sistem selesai maka selanjutnya kuesioner yang sudah diisi oleh responden akan dianalisis berdasarkan aspek yang sudah dijadikan dasar penilaian. Pada Tabel 6.36 merupakan hasil penilaian *usability* dari aspek kualitas sistem. Pada Tabel 6.37 merupakan hasil penilaian *usability* dari aspek

kualitas informasi. Pada Tabel 6.38 merupakan hasil penilaiain *usability* dari aspek antarmuka sistem.

Tabel 6.36 Hasil Penilaian Usability dari Aspek Kualitas Sistem

Nama / Pertanyaan	Responden 1	Responden 2	Responden 3	Responden 4	Responden 5
1	7	6	7	7	7
2	6	6	7	7	7
3	7	5	5	6	5
4	7	7	7	7	7
5	7	7	7	7	7
6	6	6	5	6	7
Jumlah	40	37	38	40	40
Rata-Rata	0.95	0.88	0.90	0.95	0.95
Total Rata-Rata	0.93				

Tabel 6.37 Hasil Penilaian Usability dari Aspek Kualitas Informasi

Nama / Pertanyaan	Responden 1	Responden 2	Responden 3	Responden 4	Responden 5
7	7	5	7	7	7
8	6	7	7	7	7
9	7	7	7	6	6
10	7	7	7	7	7
11	6	6	6	7	7
Jumlah	33	32	34	34	34
Rata-Rata	0.94	0.91	0.97	0.97	0.97
Total Rata-Rata	0.95				

Tabel 6.38 Hasil Penilaian Usability dari Aspek Kualitas Antarmuka

Nama / Pertanyaan	Responden 1	Responden 2	Responden 3	Responden 4	Responden 5
12	7	7	7	7	7
13	7	7	7	7	7
14	7	7	6	6	5

Tabel 6.38 Hasil Penilaian Usability dari Aspek Kualitas Antarmuka (Lanjutan)

Jumlah	21	21	20	20	19
Rata-Rata	1.00	1.00	0.95	0.95	0.90
Total Rata-Rata	0.96				

Setelah para responden mengisi kuesioner selanjutnya akan dilakukan perhintungan untuk memberikan penilaian untuk *usability* sistem. Dari hasil yang diperoleh secara keseluruhan sistem yang diuji mendapatkan poin 0.94, serta memiliki nilai diatas 0.8 poin dari setiap aspek yang menandakan bahwa sistem mudah digunakan dan dapat diandalkan oleh pengguna. Hasil perhitungan nilai pengujian *usability* sistem menggunakan metode *post-study system usability questionnaire* untuk masing-masing aspek yang diuji dapat dilihat pada Tabel 6.39.

Tabel 6.39 Hasil Akhir Penilaian *Usability* Sistem

Aspek Penilaian	Total Nilai
System Quality	0.93
Information Quality	0.95
Interface Quality	0.96
Overall	0.94

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Hasil yang didapatkan dari penelitian yang sudah dijalankan akan menjadi jawaban dari rumusan masalah yang sudah difenisikan sebelumnya. Kesimpulan yang didapatkan dari penelitian ini adalah:

1. Pada proses analisis kebutuhan pada sistem informasi manajemen ternak burung *lovebird* berbasis *android*, didapatkan 5 buah *module* kebutuhan utama yang masing-masing adalah *module birdfarm management* yang berhubungan dengan burung ternak, *module breeding* yang berhubungan dengan proses budidaya ternak, *module farmer* yang berhubungan dengan informasi peternak, *module gallery* yang berisikan foto-foto burung, dan *module finance* yang berhubungan dengan informasi keuangan. Dari 5 buah *module* tersebut menghasilkan 23 kebutuhan *fungsional* yang direpresentasikan dalam bentuk *use case diagram* dan *use case* scenario, selain itu juga terdapat 1 kebutuhan *non-fungsional*. Dari proses tersebut juga menghasilkan struktur data yang nantinya akan dijadikan sebagai tabel pada *database*.

2. Pada proses perancangan dihasilkan bentuk arsitektur dari sistem, diagram kelas, diagram sequence, perancagan data yang berupa *Entity Relational Diagram* (ERD), perancangan komponen yang berupa *pseudocode*, dan perancangan antarmuka yang merupakan rancangan tampilan sistem.

3. Pada proses implementasi dihasilkan skema yang merupakan representasi dari *Entity Relational Diagram*, yang nantinya secara otomatis akan dijadikan sebagai struktur tabel database oleh prisma. Selain itu akan dihasilkan kode program sesuai dengan *pseudocode* dan antarmuka sesuai dengan rancangan antarmuka yang sudah dituliskan sebelumnya.

4. Pada proses pengujian dalam penelitian ini terdapat 3 jenis pengujian yaitu validasi menggunakan metode *blackbox* dengan 23 kasus uji, pengujian unit menggunakan metode *whitebox* dengan 3 kasus uji dan pengujian *usability* menggunakan metode *PSSUQ* dengan menyebarkan kuesioner untuk 3 orang responden. Hasil yang didapatkan dari pengujian validasi dan pengujian unit adalah 100% valid. Pada pengujian *usability* dari semua aspek memiliki nilai diatas 0.8 yang menandakan bahwa sistem mudah digunakan dan dapat diandalkan oleh calon penggunanya.

7.2 Saran

Saran yang diberikan peneliti sebagai acuan untuk pengembangan lanjut sistem informasi manajemen ternak burung *lovebird* agar dapat mendukung proses bisnis selama budidaya burung berlangsung adalah:

1. Menambahkan sensor pada kandang burung serta fitur notifikasi dan *reminder*. Sensor untuk mengukur suhu dan kelembaban yang terdapat pada

sarang burung yang terhubung dengan perangkat *smartphone* peternak sehingga memudahkan para peternak melakukan *monitoring* kandang selama proses budidaya berlangsung.

2. Memanfaatkan sistem pakar untuk dapat memprediksi hasil anakan perkawian antara dua jenis burung *lovebird*. Hal tersebut dapat berguna untuk para peternak dapat mengetahui jenis anakan yang dihasilkan dari proses perkawinan sejak dini.

DAFTAR PUSTAKA

Alderton, D., 2003. *The Ultimate Encyclopedia of Caged and Aviary Birds*. London, England: Hermes House. pp. 216–219.

Bevan, N., 1995. Usability As Quality of Use. *Software Quality Journal*, 4, pp.115–130.

Beynon-Davies, P., Came, C., Mackay, H. and Tudhope, D., 1999. Rapid application development (Rad): An empirical review. *European Journal of Information Systems*, 8(3), pp.211–232.

Bourgeois, D. and Bourgeois, D.T., 2014. *Chapter 1: What Is an Information System?* Available at: <https://bus206.pressbooks.com/chapter/chapter-1/> [Accessed 28 Jun. 2019].

Choudrie, J., Pheeraphuttharangkoon, S., Zamani, E. and Giaglis, G., 2014. Investigating the Adoption and Use of Smartphones in the Uk: a Silver-Surfers Perspective. *Twenty Second European Conference on Information Systems*, [online] 2015, pp.1–19. Available at: <http://aisel.aisnet.org/ecis2014/proceedings/track16/8/>.

Egwumah, F.A., 2018. Paramount factors influencing the breeding performance of Lovebird Agapornis pullaria. (June).

Fetrina, E., Rustamaji, E., Nuraeni, T. and Durrachman, Y., 2013. Inventory Management Information System Development. pp.2–5.

G.L.M. Chappell, 2006. *SHEEP IDENTIFICATION SYSTEMS*. [online] Available at: <http://www2.ca.uky.edu/agcomm/pubs/asc/asc130/asc130.htm> [Accessed 12 Jan. 2019].

graphql, 2019. *GraphQL | A query language for your API*. [online] Available at: <https://graphql.org/> [Accessed 4 Mar. 2019].

Kondratova, I., Canada, C., Goldfarb, I. and Canada, C., 2015. Design Concepts for Virtual Research and Collaborative Environments NRC Publications Archive (NPArC) Archives des publications du CNRC (NPArC) Design Concepts for Virtual Research and Collaborative Environments Kondratova , Irina ; Goldfarb , Ilia. (November).

Mittal, N. and Snotra, K., 2017. Blood Bank Information System Using Android Application. 3, pp.3–8.

Mulawarman, M. and Nurfitri, A.D., 2017. Perilaku Pengguna Media Sosial beserta Implikasinya Ditinjau dari Perspektif Psikologi Sosial Terapan. *Buletin Psikologi*, 25(1), pp.36–44.

Myers, P., Espinosa, R., Parr, C.S., T. Jones, G.S.H. and Dewey, T.A., 2019. *ADW: Agapornis: CLASSIFICATION*. [online] Available at: <https://animaldiversity.org/accounts/Agapornis/classification/#Agapornis> [Accessed 26 Jun. 2019].

Oracle, 2019. *Polymorphism (The Java™ Tutorials > Learning the Java Language > Interfaces and Inheritance)*. [online] Available at: <https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html> [Accessed 12 Jan. 2019].

Pressman, R.S., 2009. *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman*. *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman*.

prisma, 2019. *Prisma Introduction: What, Why & How - Prisma Docs*. [online] Available at: <https://www.prisma.io/docs/understand-prisma/prisma-introduction-what-why-how-j9ff/> [Accessed 4 Mar. 2019].

Sauro, J. and Lewis, J.R., 2012. *QUANTIFYING THE USER EXPERIENCE PRACTICAL STATTISTICS FOR USER RESEARCH*. 1st ed.

Sinhal, A., 2017. *MVC, MVP and MVVM Design Pattern – Ankit Sinhal – Medium*. [online] Available at: <https://medium.com/@ankit.sinhal/mvc-mvp-and-mvvm-design-pattern-6e169567bbad> [Accessed 14 Jan. 2019].

Sommerville, I., 2010. *Software Engineering, Ninth Edition*. *Software Engineering*.

Stephen Fronefield, 2019. *AAV Aviculture Committee - Association of Avian Veterinarians*. [online] Available at: <https://www.aav.org/page/aviccommittee?> [Accessed 26 Jun. 2019].

Suryati, P., 2010. Pembangunan Sistem Informasi Pendataan Rakyat Miskin Untuk Program Beras Miskin (Raskin) Pada Desa Mantren Kecamatan Kebonagung Kabupaten Pacitan. *Communication*.

Valacich, J. and Schneider, C., 2012. Information Systems Today : Managing in the Digital World. [online] p.667. Available at: <www.myMISlab.com>.

Wang, D., Xiang, Z. and Fesenmaier, D.R., 2016. Smartphone Use in Everyday Life and Travel. *Journal of Travel Research*, 55(1), pp.52–63.

LAMPIRAN A ROADMAP PENGEMBANGAN SISTEM INFORMASI MANAJEMEN TERNAK BURUNG LOVEBIRD BERBASIS ANDROID

System: Birdfarm Management
Status: P0 / P1 / P2/ **Final**

No.	Deskripsi Task	Milestone	Progress	Status
1.	**Bird Collection**		**100%**	
	Menambah Burung		100%	Finish
	Melihat Detail Burung		100%	Finish
	Melihat list bird collection		100%	Finish
	Menambah induk burung		100%	Finish
2.	**Farmer Management**		100%	
	Masuk kedalam sistem		100%	Finish
	Menambah peternak baru		100%	Finish
	Mengubah profile peternak		100%	Finish
	Melihat profile peternak		100%	Finish
3.	**Gallery Burung**		100%	
	Menambahkan foto		100%	Finish
	Melihat foto pada gallery		100%	Finish
4.	**Breeding Record**		100%	
	Melihat list induk burung		100%	Finish
	Menambahkan Data Produk baru		100%	Finish
	Menambah kegiatan Produk		100%	Finish
	Melihat detail kegiatan dari Produk		100%	Finish
5.	**Jurnal Keuangan**		100%	
	Melihat Jurnal Keuangan		100%	Finish
	Laporan Laba Rugi		100%	Finish
	Membuat Jurnal Keuangan		100%	Finish
		TOTAL	**100%**	

Log:

25 Februari 2019

Prototype untuk modul bird collection: menambah burung, menambah induk burung, melihat detail burung, dan melihat list burung.

Prototype untuk modul Gallery burung: melihat koleksi foto-foto burung

Feedback dari *stackholder*:

- Memberikan masukan untuk item yang harus terdapat pada saat registrasi burung, tampilan list burung menggunakan viewpager yang berisikan list burung dan list induk burung.
- Memberikan deskripsi pada setiap foto yang ada pada gallery burung dan menambahkan fitur upload pada gallery burung

04 Maret 2019

Hasil perbaikan prototype sesuai masukan yang diberikan oleh stackholder untuk modul bird collection dan gallery burung:

- Membuat slot image untuk pengguna mengupload image berisikan bukti dna burung
- Menambahkan kolom untuk pengisian nama burung, warna mutasi burung
- Mengubah kolom untuk mengisikan jenis burung menjadi list yang berisikan 12 jenis burung lovebird
- Mengganti tampilan menjadi hanya list burung. List induk burung dipindahkan kedalam modul breeding record.
- Memberikan fitur upload image pada gallery burung

Prototype untuk modul breeding record: membuat batch, menambahkan breeding log, melihat list batch, melihat list log.

Feedback dari *stackholder*:

- Membuat tampilan awal dari bentuk navigation drawer kedalam bentuk dashboard
- Memberikan masukan untuk tampilan yang harus terlihat pada list batch dan isi dari form untuk breeding log
- Menambahkan lokasi dan ktp pengguna untuk keperluan yang terdapat pada sub-sistem lain (marketplace dan GIS).
- Mengurangi slot image pada saat registrasi burung dan mengganti form untuk registrasi induk jantan dan induk betina kedalam bentuk list

02 Mei 2019

Hasil perbaikan prototype sesuai masukan yang diberikan *stackholder* untuk modul bird collection, modul farmer management, dan breeding record:

- Mengganti tampilan kedalam bentuk dashboard
- Memberikan tampilan log terakhir dan status pada masing-masing batch
- Memberikan informasi mortalitas pada proses batch
- Menambahkan location finder yang terhubung dengan gmaps dan menambahkan slot image ktp
- Mengganti form registrasi induk jantan dan induk betina kedalam bentuk list

06 Mei 2019

Pengerjaan prototype untuk fitur finance: melihat laporan, membuat jurnal keuangan, membuat transaksi keuangan

Feedback dari *stackholder*:

- Perbaikan tampilan untuk form tambah transaksi (memberikan kode untuk kredit dan debit)
- Perbaikan tampilan untuk laporan laba-rugi

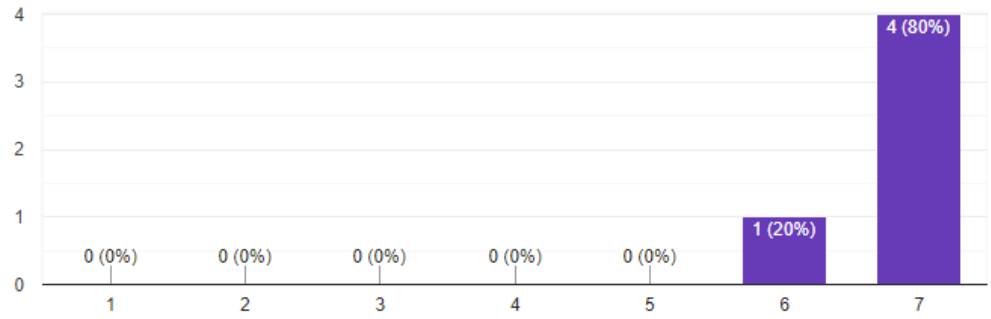
17 Mei 2019

Finalisasi prototype yang akan diimplementasikan

LAMPIRAN B HASIL POST-STUDY SYSTEM USABILITY QUESTIONNAIRE

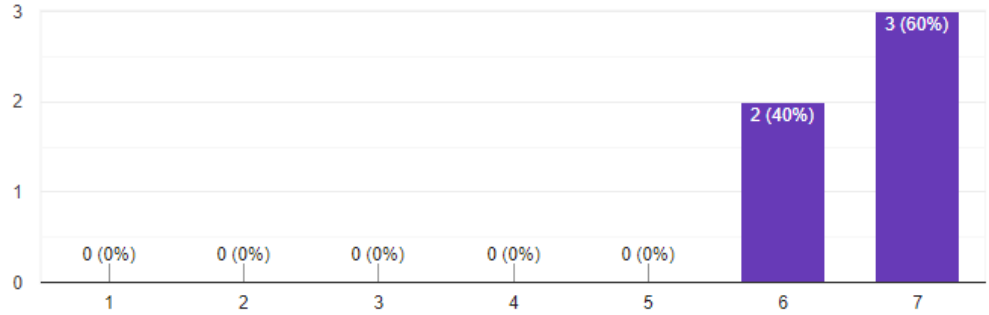
1. Secara keseluruhan, Saya puas karena dapat mengoperasikan sistem ini dengan mudah.

5 responses



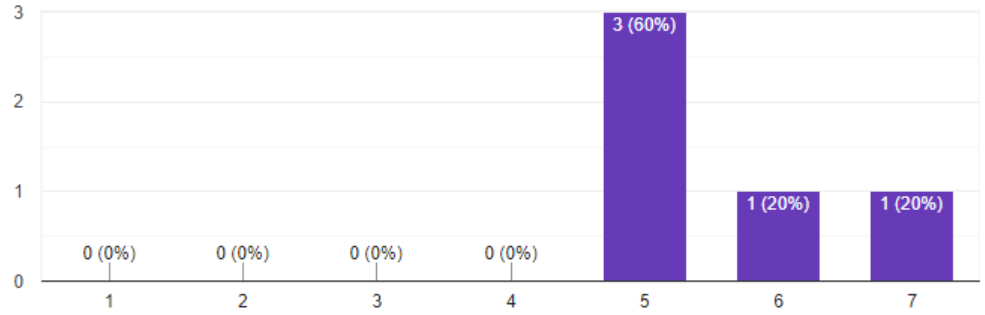
2. Sangat mudah untuk menggunakan Sistem ini.

5 responses



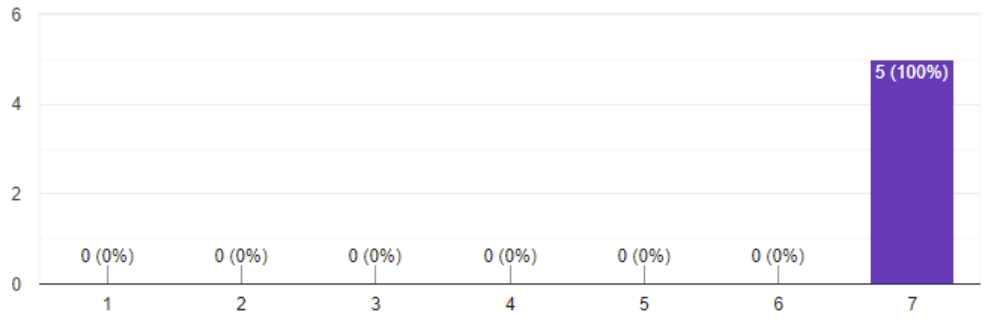
3. Saya dapat dengan mudah menyelesaikan tugas dengan cepat menggunakan sistem ini.

5 responses



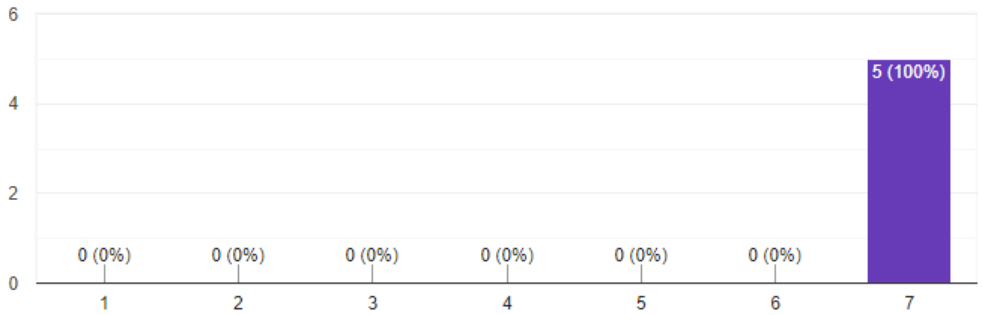
4. Saya merasa nyaman ketika menggunakan sistem ini.

5 responses



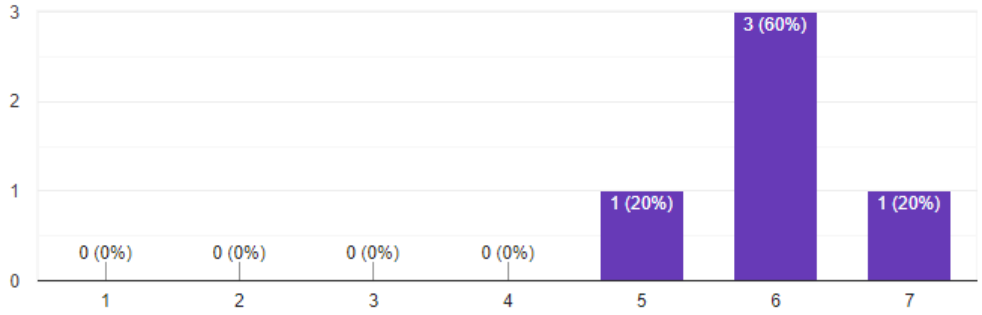
5. Sangat mudah untuk mempelajari bagaimana cara menggunakan sistem ini.

5 responses



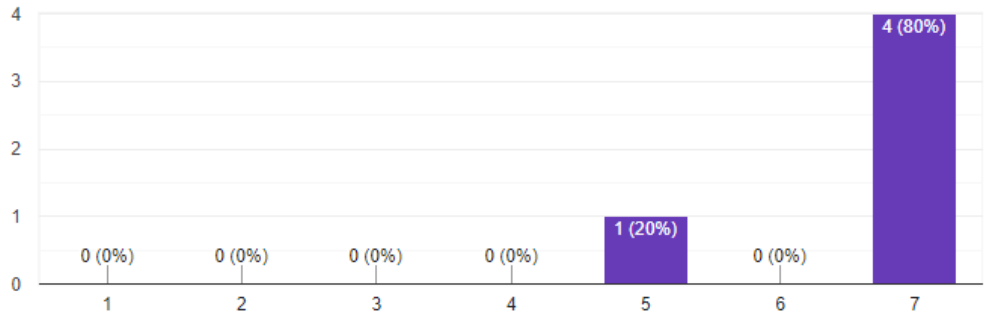
6. Saya percaya bahwa Saya bisa menjadi lebih produktif dengan adanya sistem ini.

5 responses



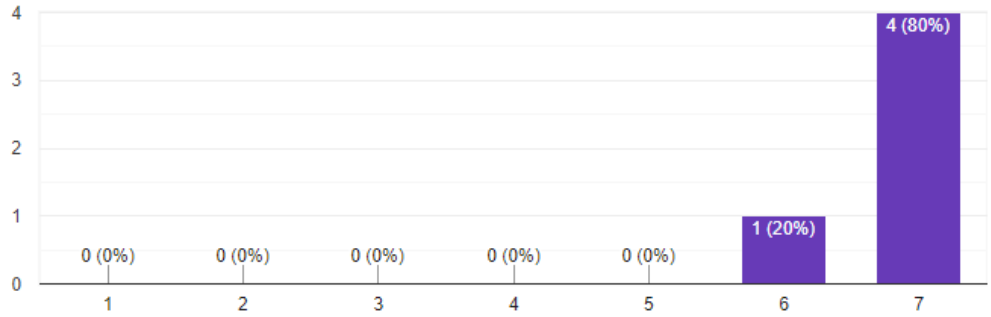
7. Apabila terjadi kesalahan, sistem ini memberikan pesan error yang memberitahukan bagaimana cara memperbaiki kesalahan tersebut.

5 responses



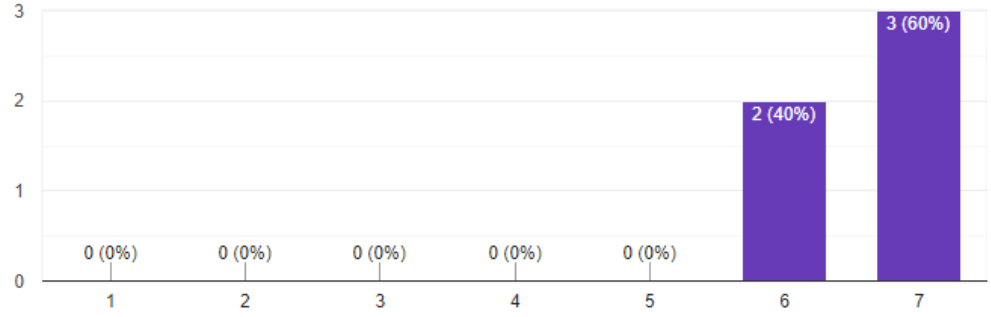
8. Informasi yang diberikan oleh sistem (pesan pada layar) sangat jelas.

5 responses



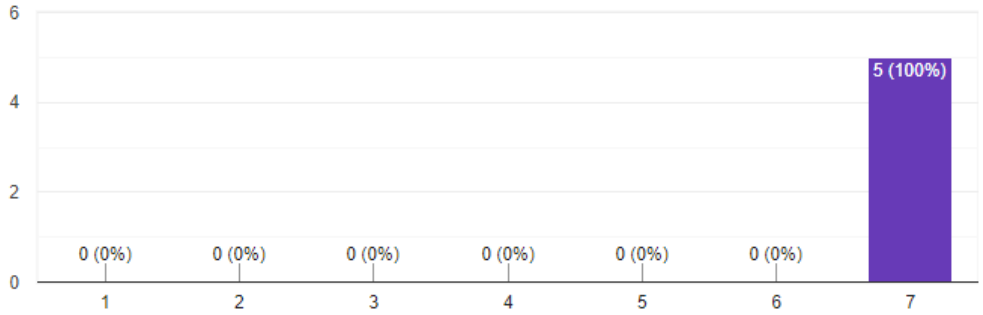
9. Sangat mudah untuk mencari informasi yang Saya butuhkan dalam sistem ini.

5 responses



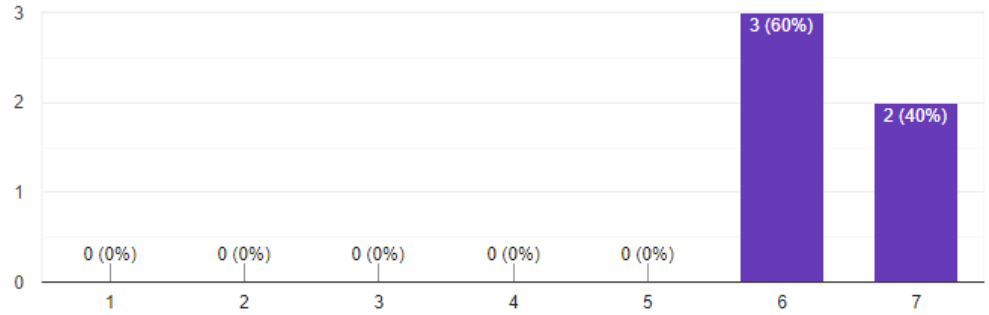
10. Informasi yang diberikan membatu Saya dalam menyelesaikan tugas.

5 responses



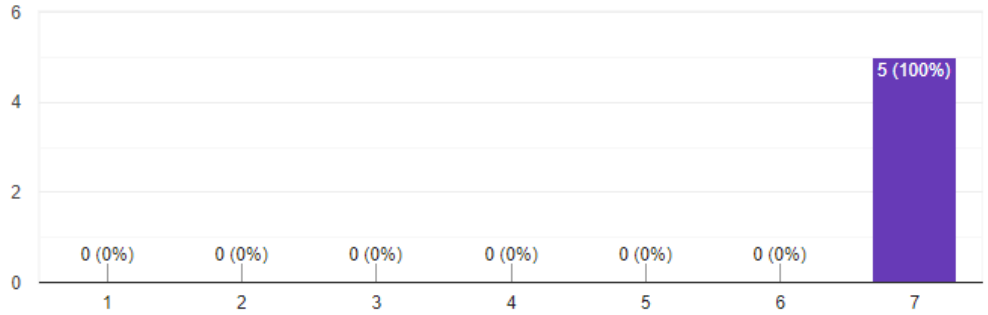
11. Informasi pada sistem Sangat dibutuhkan.

5 responses



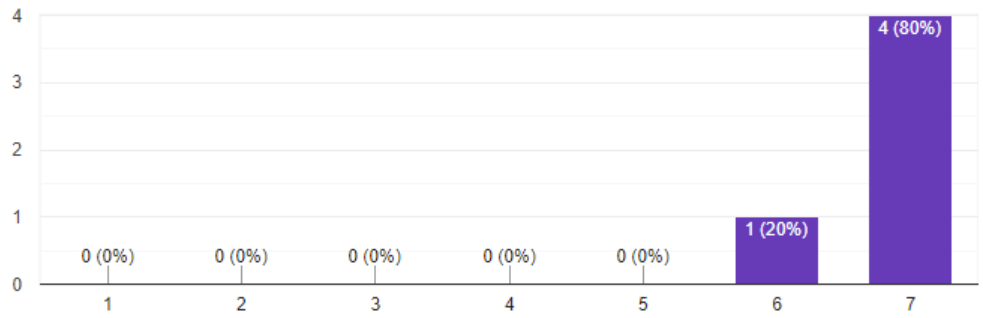
12. Saya suka dengan Antarmuka pada sistem ini.

5 responses



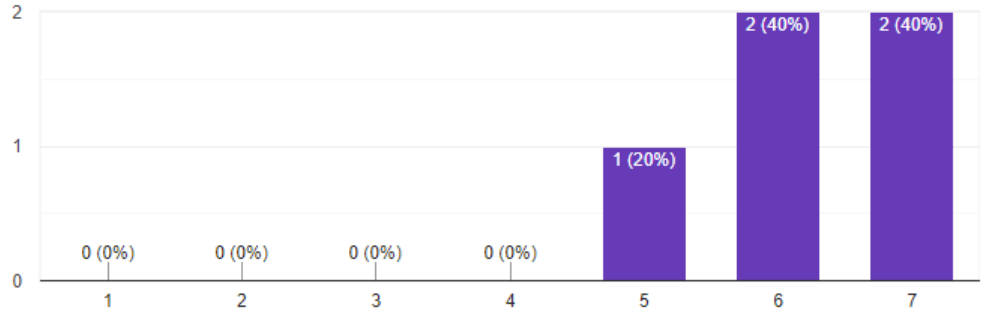
13. Antarmuka pada sistem ini enak untuk dilihat.

5 responses



14. Sistem ini memiliki kemampuan dan fungsi yang Saya harapkan harus ada pada sistem.

5 responses



15. Secara keseluruhan, Saya puas dengan sistem ini.

5 responses