REKAYASA PERANGKAT LUNAK PENGENAL SUATU BINATANG DENGAN METODE CLASSIFICATION AND REGRESSION TREE (CART)

TUGAS AKHIR
Diajukan untuk Memenuhi Salah Satu Syarat Kelulusan
Program Pendidikan Sarjana

Oleh:
Sandi
2015130025

JURUSAN INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER – LIKMI
BANDUNG
2020

REKAYASA PERANGKAT LUNAK PENGENAL SUATU BINATANG DENGAN METODE CLASSIFICATION AND REGRESSION TREE (CART)

Oleh:
Sandi
2015130025

Bandung, 16 Juli 2020
Menyetujui,

Yusup Jauhari Shandi, M.Kom.	Dhanny Setiawan, S.T., M.T.
Dosen Pembimbing	Ketua Jurusan

JURUSAN INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER – LIKMI
BANDUNG
2020

ABSTRAK

Semakin berkembangnya teknologi informasi, menyebabkan kebutuhan akan informasi sudah tidak lagi membutuhkan alat peraga. Bagi kebanyakan orang kecepatan dan keefisienan dalam memperoleh suatu informasi adalah hal yang sangat penting. Berbagai bentuk informasi terhadap objek atau benda yang ditemukan ingin langsung didapatkan, akan tetapi masalah seringkali muncul karena pengenalan suatu objek yang dikenali terbatas. Berdasarkan dari permasalahan tersebut, maka diusulkan dalam pembuatan tugas akhir yang berjudul "**REKAYASA PERANGKAT LUNAK PENGENAL SUATU BINATANG DENGAN METODE *CLASSIFICATION AND REGRESSION TREE* (CART)"** dengan tujuan mempermudah seseorang untuk mengenal suatu binatang yang ditemukan secara langsung atau tiba-tiba. Pemilihan binatang sebagai objek yang dikenal, karena memiliki bentuk objek yang tergolong *consistent* sehingga dapat meningkatkan nilai keakuratan dalam melakukan penelitian. Pemilihan metode *classification and regression tree* (CART) yang digunakan, karena memiliki performa yang akurat dalam pengambilan keputusan. Adapun pemilihan *Clarifai API* sebagai sistem pendukung perangkat lunak ini, karena memiliki kemampuan baik dalam *computer vision* untuk mengidentifikasi dan menganalisis suatu objek melalui gambar atau video.

Pada proses pengembangan perangkat lunak, digunakan model proses *prototyping* yang dilakukan dengan merancang *prototype* perangkat lunak secara berulang sampai perangkat lunak sesuai dengan ketentuan yang telah dispesifikasikan. Paradigma pengembangan berorientasi objek juga digunakan baik pada fase pemodelan maupun pengimplementasian, meliputi: perancangan *use case, activity, class,* dan *sequence diagram.* Pada tahap akhir perancangan perangkat lunak, dilakukan pengujian fungsi dan tampilan untuk memastikan kembali perangkat lunak dapat berjalan dengan baik dan sesuai dengan ketentuan yang telah dispesifikasikan.

Metode *classification and regression tree* (CART) dapat diterapkan pada perangkat lunak ini dengan dilakukan beberapa tahapan, meliputi: pengumpulan ciri suatu binatang terhadap *response* dari *clarifai API,* penempatan ciri *response* sebagai *node* dan *children tree*, dan penyimpulan setiap *children tree* sebagai hasil akhir keputusan.

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat, berkat, karunia, dan pertolongan-Nya hingga terselesaikannya tugas akhir ini yang mengangkat topik "**REKAYASA PERANGKAT LUNAK PENGENAL SUATU BINATANG DENGAN METODE CLASSIFICATION AND REGRESSION TREE (CART)**", sebagai salah satu syarat kelulusan di Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI untuk jenjang strata satu (S1) dengan jurusan Informatika bidang keahlian rekayasa perangkat lunak.

Selain itu juga penulis mengucapkan banyak-banyak terimakasih kepada pihak-pihak yang terlibat dalam penyusunan tugas akhir ini baik yang memberikan dukungan langsung maupun tidak langsung, diantaranya:

1. Bapak Yusup Jauhari Shandi, M. Kom sebagai dosen pembimbing tugas akhir ini yang bersedia memberikan bimbingan dan arahannya untuk penulis mulai dari menyusun bab 1 sampai dengan bab 5.
2. Bapak Budi Tosin selaku CEO PT Budijaya Group Bandung yang telah mengajarkan banyak hal kepada penulis dalam membuat suatu program beserta analisis sistem sehingga memudahkan dalam penyusunan tugas akhir ini.
3. Josephine Adiel Suhendra, yang selalu mendukung dan memberikan semangat selama penyusunan tugas akhir ini.
4. Para staf pengajar STMIK LIKMI yang turut membantu secara tidak langsung dalam memberikan ilmu pengetahuan dan informasi kepada penulis selama mengikuti perkuliahan hingga dapat menyelesaikan penyusunan tugas akhir ini.
5. Kedua orang tua tercinta mama, papa yang selalu memberikan dorongan dan juga semangat serta doa yang tidak putus-putusnya agar penulis dapat menyelesaikan tugas akhir ini.

6. Kakak tercinta, Agnes yang selalu mendukung sekaligus menyemangati penulis agar tugas akhir ini selesai
7. Teman-teman seperjuangan STMIK LIKMI angkatan 2015 yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa penulisan tugas akhir ini masih jauh dari sempurna, oleh karena itu penulis sangat membuka diri untuk kritik dan saran yang membangun dalam perbaikan tugas akhir ini.

Akhir kata penulis berharap, tugas akhir ini dapat bermanfaat bagi penulis dan bagi semua pembaca. Terima kasih semoga Tuhan Yang Maha Esa memberkati kita semua sekarang dan selamanya. Amin.

Bandung, 16 Juli 2020

Penulis

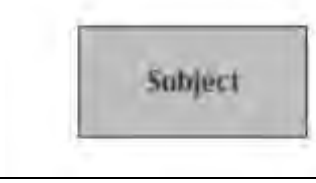
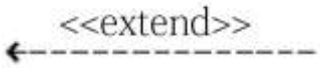
DAFTAR ISI

DAFTAR GAMBAR

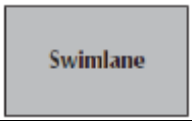
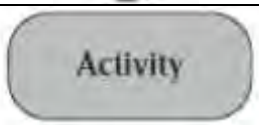
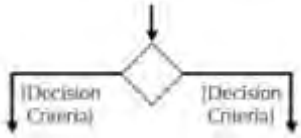
DAFTAR TABEL

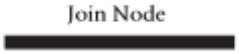
DAFTAR SIMBOL

Use case diagram

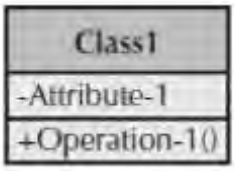
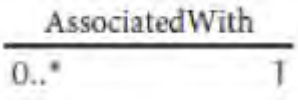
Nama Simbol	Simbol	Nama
Subjek *Boundray*		Merepresentasikan *scope* dari suatu subjek, biasanya subjek dapat berupa sistem atau proses bisnis. Disertai dengan nama subjek di bagian atas.
Aktor (*Actor)*		Orang atau sistem yang mendapatkan manfaat dari sistem perangkat lunak atau yang berinteraksi dengan sistem tersebut. Dapat diasosiasikan dengan aktor lainnya menggunakan spesialisasi. Berada di luar sistem *boundary.*
Use Case		Merepresentasikan bagian-bagian fungsi utama dalam sistem. Berada di dalam sistem *boundary*. Dideskripsikan sebagai kata kerja.
Extend		Merepresentasikan perilaku tambahan atau *optional* yang dapat dilakukan pada suatu *use case* tertentu.

Activity diagram

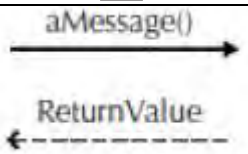
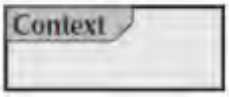
Nama Simbol	Simbol	Nama
Swimlane		Memisahkan *activity diagram* ke dalam baris atau kolom.
Kondisi Awal		Menggambarkan kondisi awal dari sekumpulan *activity* atau *action*.
Aktivitas (*Activity)*		Merepresentasikan sekumpulan dari aksi atau *action*.
Alur Kontrol (*Control Flow*)		Menampilkan urutan pengerjaan dari sekumpulan *activity* atau *action*.
Kondisi atau Percabangan (*Decision*)		Merepresentasikan sebuah tes percabangan untuk menentukan jalur yang akan dipilih.

Nama Simbol	Simbol	Nama
Fork Node (Synchronization)		Memecah suatu tingkah laku menjadi sebuah *activity* atau *action* yang paralel.
Join Node (Synchronization)		Menggabungkan kembali *activity* atau *action* yang paralel.
Kondisi Akhir		Berfungsi untuk menghentikan semua *control flow* dalam suatu *activity diagram*.

Class diagram

Nama Simbol	Simbol	Nama
Kelas (*Class*)		Merepresentasikan benda, orang, atau tempat, yang berhubungan dengan sistem yang nantinya dibutuhkan untuk menangkap dan menyimpan informasi. Memiliki sekumpulan *attribute* dan *operation* di dalamnya.
Atribut Kelas (*Attribute)*	attribute name /derived attribute name	Merepresentasikan properti yang mendeskripsikan status dari suatu objek pada kelas.
Operasi Kelas (*Operation*)	operation name ()	Merepresentasikan aksi atau fungsi yang dilakukan oleh kelas. Disertai dengan tanda dalam kurung yang didalamnya dapat berisi suatu *parameter* atau informasi yang dibutuhkan untuk melakukan operasi tersebut.
Asosiasi (*Association*)		Merepresentasikan relasi antara banyak kelas atau satu kelas ataupun dengan kelas itu sendiri.
Directed Association		Merepresentasikan relasi suatu kelas yang diasosiasikan dengan kelas lain. Relasi ini merupakan *one-way relationship* (satu arah), artinya satu kelas memiliki satu *instance* dari kelas lainnya tapi tidak sebaliknya.

Sequence diagram

Nama Simbol	Simbol	Nama
Aktor (*Actor*)		Orang atau external sistem yang mendapatkan manfaat dari sistem. Aktor berpartisipasi dalam *sequence diagram* dengan mengirimkan pesan atau menerima pesan.
Objek (*Object*)		Objek berpartisipasi dalam *sequence diagram* dengan mengirimkan pesan atau menerima pesan.
Lifeline		Merepresentasikan umur objek dalam *sequence diagram.*
Execution Occurence		Merepresentasikan durasi ketika objek mengirimkan atau menerima pesan.
Message		Menyampaikan informasi dari satu objek ke objek yang lain.
Frame		Merepresentasikan konteks pada *sequence diagram*.

DAFTAR LAMPIRAN

BAB I

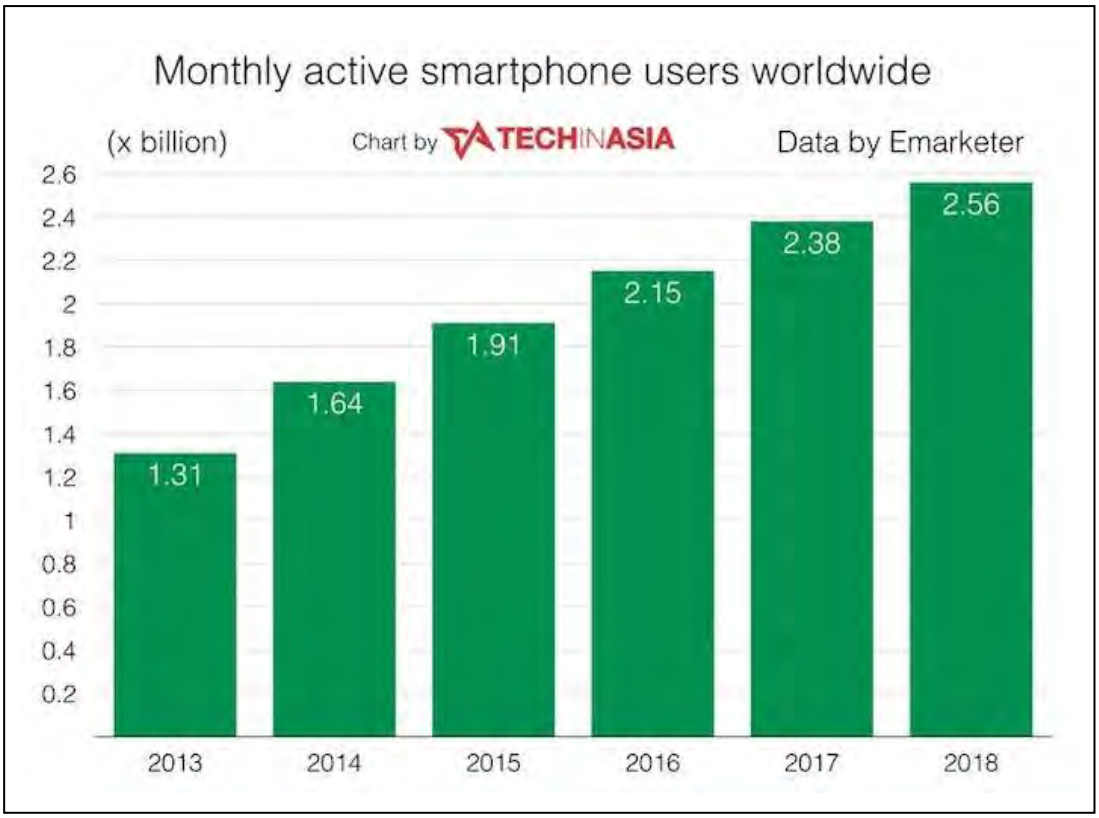
PENDAHULUAN

1.1 Latar Belakang

Dewasa ini, perkembangan akan teknologi sangatlah pesat, sehingga perkembangan teknologi juga mempengaruhi dalam bidang pendidikan dan informasi. Namun, perkembangan teknologi ini belum terlalu dimanfaatkan dalam bidang pendidikan dan informasi karena sering kali membutuhkan alat peraga, contohnya untuk pengenalan binatang pada usia dini. Saat ini pengenalan binatang pada usia dini masih berbentuk gambar dan penjelasan yang dijelaskan oleh seorang pengajar, tanpa langsung melihat pada binatang yang sesungguhnya. Hal ini menyebabkan kurangnya pemahaman dalam mengenal berbagai jenis binatang yang mungkin akan di temui pada masa yang akan datang.

Bagi kebanyakan orang hal ini sering kali menghambat, terutama dalam mengenal suatu binatang yang baru di temui, seperti menentukan nama, jenis makanan, dan habitat asli dari binatang tersebut. Salah satu contohnya, orang-orang sering kali salah dalam menentukan jenis binatang karnivora, herbivora dan omnivora, di mana sering kali dimanfaatkan sebagai bentuk perilaku yang akan dilakukan terhadap binatang tersebut. Orang-orang akan browsing dan mencari tahu mengenai jenis binatang tersebut melalui smartphone ataupun media lainnya. Namun, sering kali orang-orang tidak tahu kata atau *keyword* yang akan mereka tuliskan pada mesin pencarian tersebut. Oleh karena itu, dibutuhkan suatu perangkat lunak pengenal suatu binatang yang dapat secara otomatis mengenal jenis binatang tersebut.

Pada pembuatan tugas akhir ini akan digunakan *clarifai system* di mana dapat digunakan untuk mendeteksi atau mengenal suatu objek melalui gambar dan pada penerapannya akan diterapkan pada *mobile application*. Berdasarkan hasil statistik pada gambar 1.1, tercatat bahwa secara global pengguna aktif *smartphone* di dunia akan meningkat dua kali lipat jumlahnya dari tahun 2013 hingga 2018.



Gambar 1.1
Statistika Pengguna Smartphone Aktif di Dunia
(Sumber: https://id.techinasia.com/jumlah-pengguna-smartphone-di-indonesia-2018)

Hal ini menyebabkan penggunaan *mobile application* menjadi faktor lebih dalam efisiensi waktu dan pemrosesan data, dikarenakan banyaknya pengguna smartphone aktif yang memakai-nya.

Penerapan *clarifai system* digunakan untuk mengidentifikasi dan menganalisis suautu objek melalui gambar maupun video. *Clarifai system* dibuat oleh perusahaan Clarifai Inc. yang merupakan perusahaan berspesialisasi dalam *computer vision* dan menggunakan *machine learning & neural networks* untuk mengidentifikasi dan menganalisis suatu objek. Hasil data *clarifai system* kemudian akan diolah kembali dengan menggunakan metode *classification and regression tree* (CART) agar dapat mengeliminasi perhitungan atau data-data yang kiranya tidak diperlukan sehingga menghasilkan suatu kesimpulan yang akurat.

Dalam pembuatan perangkat lunak ini akan menghubungkan perangkat lunak pengenal suatu binatang dengan kamera pada *smartphone* pengguna. Dengan kamera ini lah, perangkat lunak dapat mendeteksi suatu objek secara langsung sehingga

permasalahan dalam bertemu dengan berbagai jenis binatang yang akan ditemui dapat diselesaikan. Dalam pengenalan suatu binatang ini dapat dilakukan dengan menggunakan dua cara, yaitu dengan memotret langsung objek binatang yang ingin dikenali ataupun juga dengan cara memasukkan foto atau gambar ke dalam perangkat lunak pengenal suatu binatang ini. Dengan adanya fitur memotret langsung, pengguna jadi dapat lebih dimudahkan, terutama dalam menentukan suatu binatang yang akan dijumpai secara langsung.

Berdasarkan uraian di atas, maka diusulkan dalam pembuatan tugas akhir ini yang berjudul "**REKAYASA PERANGKAT LUNAK PENGENAL SUATU BINATANG DENGAN MENGGUNAKAN METODE CLASSIFICATION AND REGRESSION TREE (CART)**".

1.2 Rumusan Masalah

Beberapa rumusan masalah yang akan dikaji dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana mempermudah pengguna khususnya pada usia dini untuk lebih dapat mengenal berbagai jenis binatang yang akan dijumpai?
2. Bagaimana menerapkan *clarifai system* untuk melakukan proses pengenalan suatu jenis binatang melalui sebuah gambar?
3. Bagaimana menerapkan metode *classification and regression tree* (CART) dalam menentukan kesimpulan terhadap hasil *clarifai system*?

1.3 Tujuan

Tujuan dari penulisan tugas akhir ini adalah sebagai berikut:

1. Mempermudah pengguna khususnya usia dini dalam mengenal berbagai jenis binatang yang akan dijumpai.
2. Menerapkan *clarifai system* dalam melakukan proses pengenalan suatu jenis binatang melalui media gambar.

3. Menerapkan metode *classification and regression tree* (CART) pada hasil *clarifai system* agar dapat menghasilkan suatu kesimpulan yang akurat.

1.4 Batasan Masalah

Beberapa batasan masalah dari perangkat lunak ini adalah sebagai berikut:

1. Pengenalan suatu binatang yang dapat dideteksi berupa nama, kelas, habitat, dan golongan binatang yang berbahaya atau tidak.
2. Hasil pengenalan bersifat *single result*, yaitu hanya dapat mendeteksi dan mengenal satu binatang dalam satu gambar atau foto.

1.5 Kegunaan Hasil

Dengan adanya perangkat lunak ini diharapkan dapat membantu kebutuhan pengguna dalam mendeteksi dan mengenal suatu jenis binatang yang akan ditemui melalui foto atau gambar yang akan dimasukkan.

1.6 Sistematika Penulisan

Sistematika penulisan ini disusun dan dibagi kedalam 5 bab, meliputi hal-hal sebagai berikut ini:

BAB I PENDAHULUAN

Pada bab ini akan menjelaskan mengenai latar belakang, rumusan masalah, tujuan pembuatan perangkat lunak, batasan masalah, kegunaan hasil, serta sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini akan menjelaskan mengenai teori rekayasa perangkat lunak, model pengembangan prototyping, dan juga teori-teori lainnya seperti *object recognition, clarifai system,* metode *classification and regression tree* (CART) yang akan digunakan penulis sebagai landasan dalam pembuatan tugas akhir ini.

BAB III ANALISIS DAN PERANCANGAN

Pada bab ini berisi tentang analisis dan rancangan perangkat lunak yang akan dibuat. Pada proses pembuatan tugas akhir ini menggunakan konsep pemrograman berorientasi objek dalam mengembangkan perangkat lunak, maka analisis dan perancangan yang dilakukan meliputi: pembuatan *use case diagram, class diagram, activity diagram, scenario diagram, sequence diagram* serta rancangan antar muka.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan menguraikan mengenai implementasi dari perancangan perangkat lunak pengenal suatu binatang dan membahas tentang hasil pengujian dari perangkat lunak tersebut.

BAB V KESIMPULAN DAN SARAN

Pada bab ini akan memberikan kesimpulan dan saran mengenai pembuatan perangkat lunak pengenal suatu binatang tersebut.

BAB II

LANDASAN TEORI

2.1 Rekayasa Perangkat Lunak

Berisi beberapa penjelasan mengenai definisi rekayasa perangkat lunak, metodologi penelitian, model pengembangan sistem, pemodelan sistem dan *computer-aided software engineering* (CASE) yang akan digunakan.

2.1.1 Definisi Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak adalah suatu disiplin ilmu yang membahas mengenai semua aspek produksi di dalam perangkat lunak (Pressman and Maxim, 2015:4). Menurut Roger S. Pressman dan Bruce R. Maxim dalam bukunya yang berjudul "*Software Engineering A Practitioner's Approach Eighth Edition*", perangkat lunak didefinisikan sebagai berikut (Pressman and Maxim, 2015:4) :

Software is : (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structure that enable the program to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

Berdasarkan kutipan diatas, dijelaskan bahwa perangkat lunak merupakan sekumpulan instruksi berupa program komputer, di mana ketika program komputer itu dijalankan maka akan menyediakan fitur, fungsi, dan kinerja yang diharapkan, disertai dengan struktur data yang dapat memanipulasi informasi dan dokumentasi mengenai cara pengoperasian dan penggunaan program. Roger S. Pressman dan Bruce R. Maxim juga menjelaskan bahwa perangkat lunak memiliki karakteristik utama di mana produk perangkat lunak tidak akan pernah usang, berbeda halnya dengan produk perangkat keras yang pada umumnya memiliki masa hidup. (Pressman dan Maxim, 2015:5)

Roger S. Pressman dan Bruce R. Maxim membagi bidang pada perangkat lunak menjadi tujuh kategori, yaitu (Pressman dan Maxim, 2015:6-7) :

1. *System Software*

   Perangkat lunak ini merupakan sekumpulan dari beberapa program yang dibuat untuk melayani program lain. Biasanya berupa *compiler, editor,* dan *unity* atau bisa

juga termasuk ke dalam komponen sistem operasi, *driver*, dan perangkat lunak untuk suatu jaringan.

2. *Application Software*

   Perangkat lunak yang merupakan *stand-alone program* atau suatu perangkat lunak yang tidak perlu di-*install*. Perangkat lunak ini dibuat untuk menyelesaikan persoalan bisnis secara spesifik. Biasanya perangkat lunak jenis ini lebih ke arah operasional dan manajemen sistem, misalnya: perangkat lunak untuk melakukan pembayaran atau pemesanan.

3. *Engineering / Scientific Software*

   Perangkat lunak yang memiliki tingkat perhitungan yang tinggi. Biasanya diperlukan untuk kepentingan di bidang ilmu analisis, seperti: astronomi, vulkanologi, meteorology, dan lain sebagainya.

4. *Embedded Software*

   Perangkat lunak yang tertanam di dalam suatu produk perangkat keras atau sistem. Biasanya digunakan untuk mengimplementasi dan mengkontrol fitur dan fungsi yang berhubungan dengan perangkat keras atau sistem itu sendiri agar lebih mudah digunakan oleh pengguna, misalnya: kontrol *keypad* pada *microwave* atau fungsi digital pada mobil seperti pemantau bahan bakar, penampilan *dashboard*, dan sistem rem pada kendaraan.

5. *Product-line Software*

   Perangkat lunak yang dirancang untuk menyediakan kemampuan secara spesifik untuk digunakan oleh pengguna yang berbeda. Biasanya berupa perangkat lunak yang ditujukan pada suatu *marketplace* atau untuk ke banyak pengguna.

6. *Web / Mobile Applications*

   Perangkat lunak di mana jaringan sebagai komponen utamanya, mencakup berbagai jenis perangkat lunak, seperi perangkat lunak berbasis *browser* atau bisa juga sebagai perangkat lunak yang tertanam pada perangkat *mobile*.

7. *Artificial Intelligence Software*

   Perangkat lunak yang dalam pengembangannya menggunakan algoritma secara *non-numeric* untuk menyelesaikan masalah yang kompleks, di mana masalah tersebut tidak dapat diselesaikan dalam satu kali tahapan. Biasanya berupa perangkat lunak pada robot, *expert system*, *pattern recognition,* dan lain sebagainya.

Adapun deskripsi lain menurut Ian Sommervile dalam bukunya yang berjudul "*Software Engineering Tenth Edition"*, perangkat lunak didefinisikan sebagai berikut:

"*Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market."* (Sommervile, 2016:20)

Berdasarkan kutipan diatas, dijelaskan bahwa perangkat lunak adalah sekumpulan program komputer dan dokumentasi yang terkait di mana perangkat lunak tersebut dapat dikembangkan untuk pengguna tertentu ataupun secara umum.

Pressman dan Maxim juga menggambarkan rekayasa perangkat lunak sebagai suatu lapisan teknologi di mana setiap pendekatan yang dilakukan oleh pengembang harus didasarkan pada suatu organisasi yang memiliki komitmen untuk menghasilkan perangkat lunak yang berkualitas. (Pressman and Maxim, 2015:15-16)

Gambar 2.1
Lapisan Teknologi Rekayasa Perangkat Lunak
(Sumber : Pressman and Maxim, 2015:16)

Berbeda halnya dengan Eric J. Braude dan Michael E. Bernstein pada bukunya yang berjudul "*Software Engineering Modern Approaches Second Edition"* di mana menjelaskan rekayasa perangkat lunak sebagai berikut:

"*An engineering discipline that involves all aspects of developing and maintaining a software product."* (Braude and Bernstein, 2016:2)

Berdasarkan kutipan diatas, rekayasa perangkat lunak dapat diartikan sebagai suatu disiplin teknik yang mencakup semua aspek di dalam kegiatan pengembangan dan pemeliharaan sebuah produk perangkat lunak.

Braude dan Bernstein menjelaskan juga bahwa rekayasa perangkat lunak tidak dapat dibandingkan dengan disiplin teknik lainnya yang hanya berfokus pada apa yang akan diproduksi, tetapi rekayasa perangkat lunak juga melainkan berfokus pada bagaimana cara memproduksi produk tersebut. Braude dan Bernstein berpendapat bahwa jika dengan menggunakan pendekatan-pendekatan yang telah ditetapkan oleh IEEE, yaitu "*systematic, disciplined, and quantifiable*" pada proses pengembangannya maka akan menghasilkan suatu produk perangkat lunak yang dapat diandalkan, dapat dipelihara, dan sesuai dengan spesifikasi kebutuhan yang diminta. (Braude and Bernstein, 2016:2)

Hal serupa juga dijelaskan oleh Sommervile di dalam bukunya, di mana Sommervile lebih menekankan pengertian rekayasa perangkat lunak ke dalam dua kata kunci, yaitu (Sommervile, 2016:21-22) :

1. Disiplin Teknik

   Perekayasa perangkat lunak dapat membuat perangkat lunaknya bekerja dan juga mengaplikasikan teori-teori, metode-metode dan alat-alat yang sesuai dengan perangkat lunaknya secara selektif atau melalui pertimbangan dan apabila tidak ditemukan teori ataupun metode yang dapat digunakan, maka perekayasa perangkat lunak dapat menemukan solusi dari masalah-masalah tersebut.

2. Semua aspek dalam produksi perangkat lunak

   Rekayasa perangkat lunak tidak hanya memperhitungkan proses secara teknis, tetapi juga dapat memperhitungkan aktivitas, seperti manajemen, pengembangan peralatan, teori, dan metode yang mendukung pengembangan suatu perangkat lunak.

Berdasarkan uraian di atas, maka dapat disimpulkan bahwa pengertian dari rekayasa perangkat lunak adalah suatu disiplin teknik di mana didalamnya menyangkut semua aspek dalam pengembangan perangkat lunak baik dari tahapan awal perencanaana ataupun sampai tahap akhir proses pengembangan atau bahkan sampai ketika perangkat lunak tersebut telah dipasarkan ke semua aspek tersebut dengan memiliki tujuan yang sama, yaitu untuk menghasilkan perangkat lunak yang berkualitas.

2.1.2 Metodologi Penelitian Perangkat Lunak Berorientasi Objek

Metodologi yang akan digunakan dalam penelitian perangkat lunak ini adalah metodologi penelitian berorientasi objek. Menurut Rosa dan Shalahuddin dalam bukunya yang berjudul "Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek", di mana metodologi penelitian berorientasi objek merupakan suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya (Rosa dan Shalahuddin, 2013:100). Rosa dan Shalahuddin juga menambahkan bahwa metodologi penelitian berorientasi objek adalah suatu cara bagaimana sistem perangkat lunak dibandung melalui pendekatan objek secara sistematis, didasarkan pada penerapan prinsip-prinsip pengolahan kompleksitas, dan didalamnya meliputi semua rangkaian aktivitas berorientasi objek dari analisis sampai pengujian. (Rosa dan Shalahuddin, 2013:100)

Hal ini juga dijelaskan oleh Dennis, Wixom dan Tegarden dalam bukunya yang berjudul "*System Analysis and Design An Object-Oriented Approach with UML Fifth Edition*", di mana menjelaskan bahwa metodologi penelitian berorientasi objek berada pada metodologi penelitian lainnya, seperti proses sentrik ataupun data sentrik yang berfokus pada model proses saja atau data saja sebagai konsep utama yang diterapkan pada sistemnya, tetapi metodologi penelitian berorientasi objek berfokus pada kedua-duanya, baik proses maupun data, dengan menggabungkan keduanya kedalam satu model atau yang disebut juga sebagai objek (Dennis et al., 2015:5). Pada proses penelitiannya secara garis besar akan mendefinisikan elemen-elemen utama yang

terdapat di dalam sistem, kemudian melihat proses dan data apa saja yang terlihat di dalam setiap elemen tersebut.

Adapun pada penelitian berorientasi objek dibutuhkan pemahaman yang baik mengenai karakteristik dan konsep-konsep dari orientasi objek itu sendiri. Menurut Dennis et al. terdapat beberapa karakteristik sederhana yang umum digunakan pada sistem berorientasi objek, yaitu (Dennis et al., 2015:19) :

1. Kelas (*Class*) dan Objek (*Object*)

   Kelas merupakan suatu *template* yang secara umum digunakan untuk mendefinisikan dan membuat *instances* atau objek. Objek merupakan hasil instansiasi dari suatu kelas atau dengan kata lain objek dapat berupa orang, tempat, atau benda di mana dari objek tersebut dapat diperoleh suatu informasi. Salah satu contohnya, apabila di dalam sebuah sistem untuk menunjukkan ruangan dokter, maka kelasnya adalah Dokter dan Pasien, sedangkan objeknya adalah Jim Maloney, Mary Walson, dan Theresa Marks, yang diasumsikan sebagai pasien yang telah terdaftar. Setiap objek dapat memiliki : (1) atribut (*attribute*) di mana berfungsi untuk mendeskripsikan suatu informasi mengenai objek, di mana atribut tersebut harus mempresentasikan objek tersebut. (2) status (*state*) berfungsi untuk mendefinisikan nilai dari atribut dan relasi dengan objek lain pada keadaan waktu tertentu. Dan (3) perilaku (*behavior*) berfungsi untuk mengspesifikasi apa yang akan objek kerjakan, istilah lainnya adalah metode (*method*). Setiap kelas juga dapat memiliki atribut dan metode, akan tetapi lebih cenderung kepada pemodelannya saja.

2. Metode (*Method*) dan *Messages*

   Metode berguna untuk mengimplementasikan perilaku dari objek atau suatu aksi yang dilakukan oleh objek. *Messages* merupakan informasi yang akan dikirimkan ke objek yang dipicu (*trigger*) oleh suatu metode atau bisa disebut juga sebagai pemanggilan fungsi atau prosedur dari satu objek ke objek yang lainnya.

3. Enkapsulasi (*Encapsulation*) dan Penyembunyian Informasi (*Information Hiding*)

   Enkapsulasi sederhananya merupakan aktivitas menggabungkan proses dan data ke dalam suatu entitas. Sedangkan penyembunyian informasi adalah suatu prinsip di

mana hanya informasi saja yang dibutuhkan pengguna untuk menggunakan modul perangkat lunak atau membuat pengguna tidak perlu memikirkan bagaimana modul dari fungsi-fungsi itu dapat bekerja. Pengguna hanya mengetahui cara menggunakan modulnya saja. Penyembunyian informasi itu sendiri dimaksudkan sebagai algoritma internal di dalam objek yang disembunyikan dari bagian-bagian sistem lainnya, sehingga apabila informasi atau pesan dari luar objek dikirimkan, maka akan memicu (*trigger*) algoritma internal di dalam objek tersebut.

4. Pewarisan (*Inheritance*)

   Pewarisan adalah suatu teknik di mana atribut-atribut dan metode-metode akan secara otomatis diwariskan dari kelas induk (*superclass*) ke kelas anaknya (*subclass*). Kelas induk (*superclass*) dan kelas anak (*subclass*) dapat dilihat berdasarkan hirarki dari kelasnya itu sendiri. Salah satu contoh, misalnya dalam suatu hirarki kelas terdapat kelas Dokter, Pasien dan Orang, adapun kelas Dokter dan Pasien yang merupakan kelas anak (*subclass*) dari kelas Orang. Sedangkan kelas induk (*superclass*) dari kelas Dokter dan Pasien adalah kelas orang. Adanya pewarisan, memudahkan untuk mendefinisikan sebuah kelas terutama untuk mendefinisikan atribut-atribut dan metode-metode yang sama dengan kelas induknya.

5. Polimerfisme (*Polymorphism*) dan *Dynamic Binding*

   Polimerfisme merupakan suatu teknik di mana ketika melakukan pemanggilan metode dapat diinterpretasikan berbeda oleh kelas-kelas dari objek yang berbeda. Salah satu contohnya, apabila objek Pelukis ingin menggambar suatu bentuk geometri pada sebuah kanvas, maka objek Pelukis cukup memanggil satu metode (*method*) *DrawGeometic* yang sama, kemudian untuk membedakan bentuk kotak, segitiga, atau lingkaran dapat dilihat dari informasi-informasi yang akan dikirimkannya. Hal ini menjelaskan bahwa yang membedakan fungsi dari setiap metodenya adalah informasi-informasi yang akan dikirimkan. *Dynamic Binding* adalah suatu teknik untuk mendukung polimerfisme di mana teknik tersebut dapat men*delay* pengetikan pada suatu objek sampai program tersebut dapat dijalankan.

Hal ini dapat membuat metode-metode yang dipanggil dari objek, tidak benar-benar dipilih oleh sistem sampai sistem tersebut dijalankan.

Adapun konsep-konsep dasar lainnya yang dijelaskan oleh Rosa dan Shalahuddin, diantaranya adalah (Rosa dan Shalahuddin, 2013:103-110) :

1. Abstraksi (*Abstraction*)

   Abstraksi adalah suatu prinsip untuk merepresentasikan objek yang kompleks agar menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan yang sedang bahas.

2. Antarmuka (*Interface*)

   Antarmuka sangat mirip dengan kelas, namun antarmuka ini tidak memiliki atribut dan memiliki metode yang dideklarasikan tanpa isi atau bersifat abstrak. Deklarasi metode pada sebuat antarmuka (*interface*) dapat diimplementasikan oleh kelas lain di mana kelas tersebut dapat mengimplementasikan lebih dari satu antarmuka (*interface*).

3. Pemanfaatan Kembali (*Reusability*)

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut. Salah satu contohnya, dalam sebuah sistem peminjaman buku diperlukan kelas Anggota, maka ketika membuat sistem penyewaan VCD, kelas anggota ini bisa digunakan kembali dengan adanya sedikit perubahan.

4. Generalisasi dan Spesialisasi

   Generalisasi dan spesialisasi menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Salah satu contoh kelas umumnya (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, bus, dan kereta.

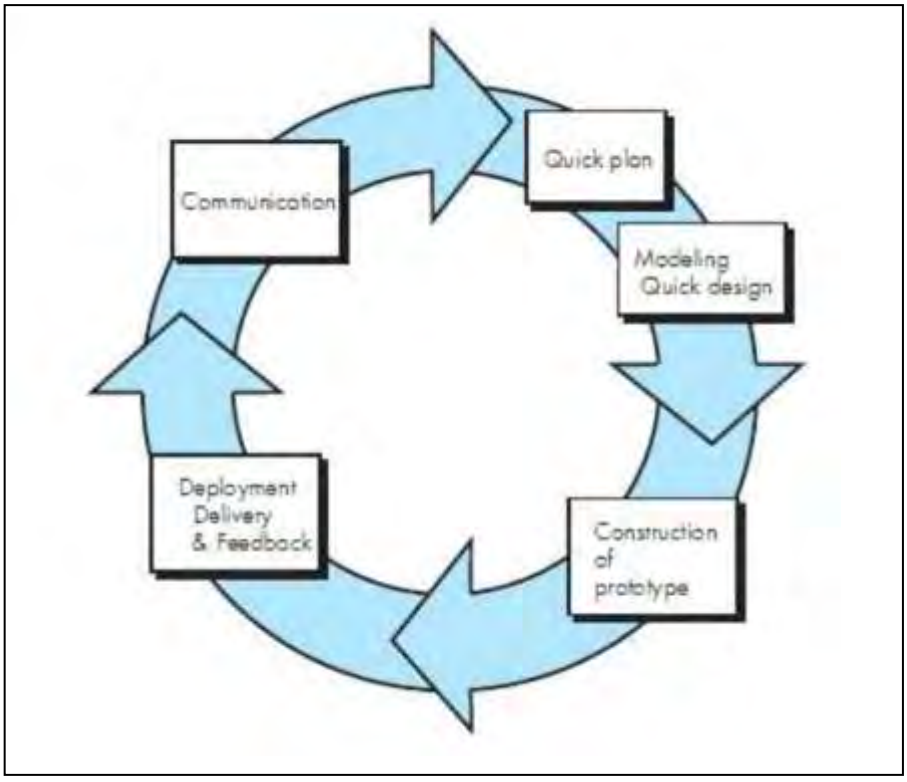
5. Kemasan (*Package*)

   Pakage merupakan sebuah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama dapat disimpan dalam *package* yang berbeda.

Berdasarkan uraian tersebut, maka dapat disimpulkan bahwa metodologi penelitian berorientasi objek adalah sebuah metodologi penelitian di mana objek-objek dapat dijadikan sebagai bagian utama di dalam sistem informasinya dan pada saat proses penelitiannya, setiap objek-objek akan didefinisikan berdasarkan informasi yang akan diperoleh dari sistem tersebut.

2.1.3 Model Pengembangan Sistem *Prototyping*

Model yang digunakan pada pengembangan sistem perangkat lunak ini adalah model pengembangan sistem *prototyping*. Model pengembangan *prototyping* merupakan salah satu dari model pengembangan, di mana dalam mengerjakan fase analisis, desain, dan implementasi dilakukan secara cepat dan berulang-ulang sampai sistem tersebut telah selesai. Setiap hasil pengerjaan akan melibatkan pengguna untuk memberikan komentar berkenaan dengan sistem tersebut, kemudian dari komentar-komentar tersebut digunakan untuk membuat *prototyping* berikutnya. Proses tersebut dilakukan secara terus menerus sampai sistem tersebut disetujui oleh pengguna. (Dennis et al., 2015:9-10)



Gambar 2.2
Model Pengembangan Sistem Prototyping
(Sumber : Pressman and Maxim, 2015:46)

Hal serupa juga dijelaskan oleh Pressman dan Maxim, di mana *prototyping* dimulai dari komunikasi dengan pengguna untuk mendefinisikan objektif-objektif dari sistem, kemudian dari objektif-objektif tersebut dibuat *prototyping*-nya secara cepat. Hasil *prototype* yang sudah di-*install* akan dievaluasi oleh pengguna, kemudian pengguna akan memberikan umpan balik yang nantinya akan menjadi perbaikan dari sistem tersebut. (Pressman and Maxim, 2015:45). Menurut Sommerville, *prototype* merupakan versi awal dari suatu sistem perangkat lunak di mana digunakan untuk mendemonstrasikan konsep, mencoba desain, dan mencari lebih jauh masalah-masalah dari sistem tersebut dan juga disertai dengan penyelesaiannya. Model *prototyping* juga dapat digunakan untuk mengantisipasi apabila terjadi perubahan pada fase spesifikasi kebutuhan atau desain. (Sommerville, 2016:62)

Dennis et al. juga menambahkan dalam bukunya bahwa model pengembangan *prototyping* memiliki kekurangan di mana dapat menimbulkan masalah apabila mengembangan sistem yang dikerjakan cukup kompleks. Hal tersebut dikarenakan isu-isu dan masalah-masalah yang tidak dapat ditemukan sampai benar-benar masuk ke dalam proses pengembangan. (Dennis et al., 2015:11)

Berdasarkan uraian di atas, maka alasan digunakan metode pengembangan sistem *prototyping* karena pada kasus perangkat lunak yang dikembangkan sering kali terjadi perubahan spesifikasi kebutuhan dan juga perangkat lunak yang dikembangkan merupakan perangkat lunak yang tidak kompleks atau sederhana. Sehingga tidak membutuhkan sebuah tim atau orang yang banyak dalam proses pengerjaannya.

2.1.4 Pemodelan Sistem Menggunakan UML

Sejak tahun 1995an, konsep objek menjadi popular di kalangan pengembang, akan tetapi hal ini juga menyebabkan metode dan notasi yang diimplementasi berbeda-beda oleh setiap pengembang. Oleh karena itu, di tahun yang sama, *Rational Software* bekerja sama dengan tiga industry perangkat lunak untuk bersama-sama membuat satu pendekatan pada pengembangan berorientasi objek yang nantinya akan dijadikan sebagai standar. UML (*Unified Modelling Language)* merupakan salah satu standar

pemodelan yang telah diakui pada tahun 1997 oleh OMG (*Object Management Group)* untuk pengembangan sistem berorientasi objek. UML memiliki tujuan untuk menyediakan kosa kata (*vocabulary*) yang umum pada istilah dalam orientasi objek dan juga teknik-teknik pendiagraman untuk membantu pemodelan sistem dari fase analisis sampai implementasi.

UML dijelaskan oleh Pressman dan Maxim dalam bukunya, di mana UML adalah "*a standard language for writing software blueprints. UML may be used to visualize, specific, construct, and document the artifcats of a software-intensive system"* (Pressman and Maxim, 2015:869). Berdasarkan kutipan tersebut dijelaskan bahwa UML adalah bahasa standar yang digunakan untuk menulis sketsa (*blueprint*) dari perangkat lunak dan biasanya UML digunakan untuk memvisualisasikan, menspesifikasikan, mengkonstruksi, dan mendokumentasi artifak-artifak (*artifacts*) pada sistem perangkat lunak.

Versi UML saat ini adalah versi 2.5.1, di mana mendefinisikan UML kedalam lima belas teknik pendiagraman untuk pemodelan sistem. Diagram-diagram tersebut dibagi menjadi dua kelompok utama, yaitu (1) untuk pemodelan struktur (*structure*) pada sistem dan (2) untuk pemodelan perilaku (*behavior*) pada sistem. Struktur diagram digunakan untuk merepresentasikan data dan relasi secara statis pada sistem informasi. Perilaku diagram digunakan oleh analis untuk menggambarkan relasi secara dinamis antar *instances* atau objek-objek yang merepresentasikan sistem informasi bisnis dan bisa juga digunakan untuk melakukan pemodelan perilaku melalui waktu hidup (*life time*) dari setiap objeknya. (Dennis et al., 2015:34)

Tabel 2.1
Rangkuman Diagram UML 2.5

No	Nama Diagram	Digunakan Untuk	Fase Primer
Struktur Diagram			
1	*Class*	Menggambarkan relasi model antar kelas pada sistem	Analisis, Desain
2	*Object*	Menggambarkan relasi model antar objek pada sistem	Analisis, Desain
3	*Package*	Mengelompokkan UML elemen lainnya secara bersama-sama sebagai bentuk dari *high-level* konstruksi	Analisis, Desain, Implementasi

No	Nama Diagram	Digunakan Untuk	Fase Primer
4	*Deployment*	Memperlihatkan arsitektur fisik pada sistem	Desain Fisik, Implementasi
5	*Component*	Menggambarkan relasi fisik antar komponen perangkat lunak lainnya	Desain Fisik, Implementasi
6	*Composite Structure Design*	Menggambarkan struktur internal pada suatu kelas	Analisis, Desain
7	*Profile*	Mengembangkan ekstensi dari UML itu sendiri	Tidak ada
Perilaku Diagram			
8	*Activity*	Menggambarkan alur bisnis yang tidak tergantung terhadap kelas di mana alur tersebut diambil dari aktivitas-aktivitas yang ada di *use-case*	Analisis, Desain
9	*Sequence*	Memodelkan perilaku dari objek-objek yang terdapat di *use-case* di mana lebih berfokus ke *time-based ordering* pada suatu aktivitas	Analisis, Desain
10	*Communication*	Memodelkan perilaku dari objek-objek yang terdapat di *use-case* di mana lebih berfokus pada komunikasi antar objek yang saling berkolaborasi pada suatu aktvitas	Analisis, Desain
11	*Interaction Overview*	Menggambarkan alur kontrol pada suatu proses	Analisis, Desain
12	*Timing*	Menggambarkan interaksi antar objek-objek dan perubahannya yang terlihat melalui satuan waktu	Analisis, Desain
13	*Behavioral State Machine*	Memeriksa perilaku pada satu kelas	Analisis, Desain
14	*Protocol State Machine*	Menggambarkan *dependencies* antar perbedaan *interface* pada suatu kelas	Analisis, Desain
15	*Use-Case*	Memperoleh spesifikasi bisnis pada sistem yang digambarkan melalui interaksi antar sistem dan lingkungan	Analisis

(Sumber : Dennis et al., 2015:35)

Setiap diagram-diagram yang digunakan, tergantung kebutuhan dari proses pengembangan sistemnya (Dennis et al., 2015:35). Adapun dalam pengembangan perangkat lunak yang digunakan, menggunakan empat diagram yaitu *use-case diagram,* kelas (*class*) *diagram, activity diagram,* dan *sequence diagram.*

2.1.4.1 *Use Case Diagram*

Use case mendeskripsikan suatu interaksi secara berurutan antara sistem dan aktor eksternal (pengguna yang berinteraksi langsung dengan sistem), di mana dalam *use case* tersebut menjelaskan suatu proses yang aktor perlu untuk selesaikan dan

biasanya ditandai ketika aktor dapat meraih suatu hasil atau nilai dari proses tersebut (Wiegers and Beatty, 2013:144). *Use case diagram* menyediakan representasi *high level* (fungsi-fungsi utama) dari spesifikasi perangkat lunak, yang didalamnya menjelaskan antara interaksi aktor dengan *use case* dan relasi antara *use case* dengan *use case.* (Wiegers and Beatty, 2013:148)

Hal ini juga dijelaskan oleh Broude dan Bernstein dalam bukunya bahwa *use case diagram* mendeskripsikan interaksi-interaksi secara berurutan antara pengguna dengan perangkat lunak dan juga menyediakan cerita bagaimana perangkat lunak tersebut digunakan (Broude and Bernstein, 2016:249-250). Setiap *use case* sering kali disertai dengan *use case scenario*, yang berisi spesifikasi detil dari *use case* tersebut. Menurut Broude dan Bernstein *use case scenario* merupakan suatu *use case* dokumen yang didalamnya terdiri dari beberapa tahapan secara berurutan di mana tujuannya untuk mendapatkan suatu hasil atau nilai dari *use case* tersebut. (Broude and Bernstein, 2016:250)

2.1.4.2 *Activity Diagram*

Menurut Wiegers dan Beatty, *activity diagram* digunakan untuk merepresentasikan aliran secara logis dalam suatu *use case* yang kompleks. Hal ini juga menjelaskan bahwa *activity diagram* tidak hanya menggambarkan aliran-aliran alternative lainnya yang tidak selalu dilewati untuk menyelesaikan *use case* tersebut. *Activity diagram* biasanya didasarkan atas *use case scenario* yang telah dibuat. (Wiegers dan Beatty, 2013:153)

Berbeda halnya menurut Broude dan Bernstein, di mana menjelaskan *activity diagram* sebagai model pengembangan dari *flowchart* yang dikhususkan untuk orientasi objek. Broude dan Bernstein juga menambahkan bahwa *activity diagram* selain dapat mendetailkan *use case*, juga dapat digunakan sebagai *flow control* dari suatu algoritma. (Broude and Bernstein, 2016:374-375)

2.1.4.3 Kelas (*Class*) *Diagram*

Class diagram digunakan untuk membantu memvisualisasikan desain pada suatu perangkat lunak di mana didalamnya mendeskripsikan kelas-kelas yang terdapat pada perangkat lunak, beserta dengan atribut, operasi, dan juga relasi antar kelas tersebut. Adapun salah satu keuntungan dengan menggunakan *class diagram* membantu *programmer* untuk membayangkan suatu perangkat lunak menjadi sekumpulan unit-unit. (Broude and Bernstein, 2016:368)

Menurut Nikiforova et.al dalam jurnalnya yang berjudul "*Role of UML Class Diagram in Object-Oriented Software Development", class diagram* disebut juga sebagai jembatan antara spesifikasi perangkat lunak dari segi pengguna dengan realisasi perangkat lunak dari segi pengembangan, di mana membutuhkan pemahaman yang baik dalam mengidentifikasi kelas pada objek-objek pada perangkat lunak tersebut dan juga kesesuaian notasi pada pemodelan diagram, yang nantinya akan digunakan dalam *coding* sistem. (Nikiforova el al., 2011)

2.1.4.4 *Sequence Diagram*

Sequence diagram digunakan untuk merepresentasikan secara visual *control flow* yang biasanya berguna untuk mendeskripsikan eksekusi-eksekusi (diambil dari *use case scenario*) yang melibatkan beberapa objek-objek (Broude and Bernstein, 2016:369). Hal ini juga membuat dalam pemodelan *sequence diagram* dibutuhkan pemahaman yang baik antara objek dan fungsi.

Hal ini juga dijelaskan oleh Sommerville dalam bukunya di mana *sequence diagram* merupakan diagram uml yang secara umum digunakan untuk memodelkan interaksi antara aktor dengan objek-objek satu sama lain yang terdapat dalam sistem tersebut (Sommerville, 2016:146). *Sequence diagram* juga berfungsi sebagai salah satu cara untuk meminimalisir kesalahan antara pendeskripsian objek-objek yang berperan sebagai aktor dengan objek-objek yang dijadikan model di dalam sistem. (Dennis et al., 2015:203)

2.1.5 *Computer-Aided Software Engineering* (CASE)

Menurut Dennis et al. dalam bukunya, *Computer-Aided Software Engineering* (CASE) diartikan sebagai "*a category of software that automates all or part of development process*" (Dennis et al., 2015:77). Hal ini juga menjelaskan bahwa *Computer-Aided Software Engineering* merupakan suatu kategori perangkat lunak di mana dapat mengotomatiskan semua atau sebagian dari proses pengembangan.

Hal ini juga dijelaskan oleh Rosa dan Shalahuddin dalam bukunya, yang menjelaskan bahwa CASE merupakan aplikasi atau perangkat lunak yang membantu pembuatan sebuah sistem perangkat lunak. (Rosa dan Shalahuddin, 2013:122)

CASE itu sendiri dapat dikelompokkan menjadi empat dimensi sebagai berikut (Rosa dan Shalahuddin, 2013:123-124) :

1. Pendukung alur hidup perangkat lunak (*life cycle support*)

   Dibagi menjadi dua kelompok yaitu:

   a. *Upper CASE*

      Upper CASE mendukung perencanaan strategis dan pengembangan perangkat lunak, misalnya aplikasi untuk membuat diagram UML. Salah satu contohnya adalah StarUML, ArgoUML, PoseidonCE*, Rational Rose, Visual Paradigm.*

   b. *Lower CASE*

      Lower CASE berfokus pada bagian akhir pengembangan perangkat lunak seperti implementasi perangkat lunak, perbaikan kesalahan perangkat lunak, pengujian, konstruksi dan integrasi perangkat lunak, perekayasaan perangkat lunak kembali (*reengineering*), atau *reverse engineering* (perbaikan rekayasa perangkat lunak diawali dari perangkat lunak yang sudah jadi).

2. Dimensi integrasi (*integration dimension*)

   Dimensi yang berfokus pada integrasi komponen perangkat lunak, seperti misalnya *framework*, integrasi lingkungan pengembangan perangkat lunak, dll.

3. Dimensi konstruksi (*construction dimension*)

   Dimensi yang berfokus pada proses konstruksi perangkat lunak, misalnya *reengineering* dan *reverse engineering.*

4. Dimensi CASE berbasis keilmuan (*knowledge-based CASE*)

   Dimensi yang berfokus pada perangkat lunak untuk membantu perangkat lunak memiliki basis keilmuan. Pada umumnya berupa perangkat lunak pemodelan simulasi proses bisnis yang diinginkan, misalnya untuk membangun sistem pendukung keputusan (*Decision Support System*) atau sistem pakar (*Expert System*).

Adapun dengan menggunakan CASE terdapat beberapa keuntungan yang dapat dimanfaatkan oleh pengembangan perangkat lunak, diantaranya: proses-proses pengembangan dapat diselesaikan dan diubah dengan cepat, dokumentasi pengembangan menjadi lebih tersentralisasi, dan informasi yang digambarkan melalui diagram-diagram menjadi lebih mudah untuk dimengerti. (Dennis et al., 2015:77)

2.2 *Computer Vision*

Berdasarkan Jan Erik Solem pada bukunya yang berjudul "*Programming Computer Vision with Python",* menjelaskan bahwa *computer vision* adalah ekstraksi informasi secara otomatis dari sebuah gambar. Informasi dapat berarti apa saja dari model 3D, posisi kamera, deteksi dan pengenalan objek untuk mengelompokkan dan mencari konten gambar. *Computer vision* dibangun untuk meniru fungsi dari penglihatan manusia (*human vision*), dengan menggunakan data dan pendekatan statistik ataupun geometri sebagai kunci dasar. *Computer vision* yang praktis terdiri atas tahap *programming, modeling* dan *mathematics* dalam memahami suatu gambar. Bagian matematika digunakan sebagai tahap untuk membantu dalam memahami suatu algoritma. (Solem, 2012:9)

Computer vision dapat didefinisikan sebagai bidang ilmiah yang mengekstrak informasi dari gambar digital, dimana tipe dari informasi yang diperoleh dari suatu gambar tersebut dapat bervariasi dari identifikasi, ruang pengukuran untuk navigasi, atau aplikasi *augmented reality*. Cara lain untuk mendefinisikan *computer vision* adalah melalui aplikasinya dengan membangun algoritma yang dapat memahami konten gambar dan menggunakannya untuk aplikasi lain.

Ada beberapa komponen yang diturunkan dari definisi *vision* atau penglihatan diantarannya adalah sebagai berikut (Krishna ,2017:17):

1. *Sensing Device*

   Perangkat penginderaan berfungsi menangkap sebanyak mungkin detail dari suatu gambar. Komponen ini dapat digambarkan seperti mata yang menangkap cahaya masuk melalui iris dan memproyeksikannya ke retina, dimana sel-sel khusus akan mengirimkan informasi ke otak melalui neuron. Kamera akan mengambil gambar dalam sebuah cara serupa dan mengirimkan informasi berupa *pixel* ke komputer. Di bagian ini, kamera lebih baik dari pada manusia karena kamera dapat melihat secara inframerah, melihat lebih jauh atau dengan lebih presisi.

2. *Interpreting Device*

   Perangkat penerjemah kemudian harus memperoleh informasi dan mengekstrak makna *sensing device*. Komponen ini dapat digambarkan seperti otak manusia yang memecahkan secara berganda langkah-langkah di berbagai wilayah objek. Teknologi *computer vision* masih tertinggal di belakang untuk mengikuti kinerja manusia pada komponen ini.

2.3 *Decision Tree*

Menurut Mihaela van der Schaar pada jurnalnya yang berjudul "*Classification and Regression Trees*", menjelaskan bahwa *decision tree* adalah suatu struktur yang diatur secara hierarkis, dengan masing-masing sampul atau *node* membelah ruang data menjadi potongan-potongan berdasarkan nilai yang telah ditetapkan. Ada beberapa istilah penting di dalam *decision tree* diantaranya adalah sebagai berikut (Van der Schaar, 2017:4):

1. *Parent*

   Induk atau *parent* dari simpul x adalah simpul pendahulu langsung atau yang sering dikenal sebagai *predecessor node.*

2. *Children*

   Anak-anak dari simpul x adalah penerus langsung dari x (*successors*), ekuivalen dari simpul-simpul yang memiliki x sebagai orang tua (*parent*).

3. *Root Node*

   Simpul akar atau *root node* adalah simpul atas dari pohon (*tree*), yaitu satu-satunya simpul yang tidak memiliki orang tua (*parent*).

4. *Leaf Node*

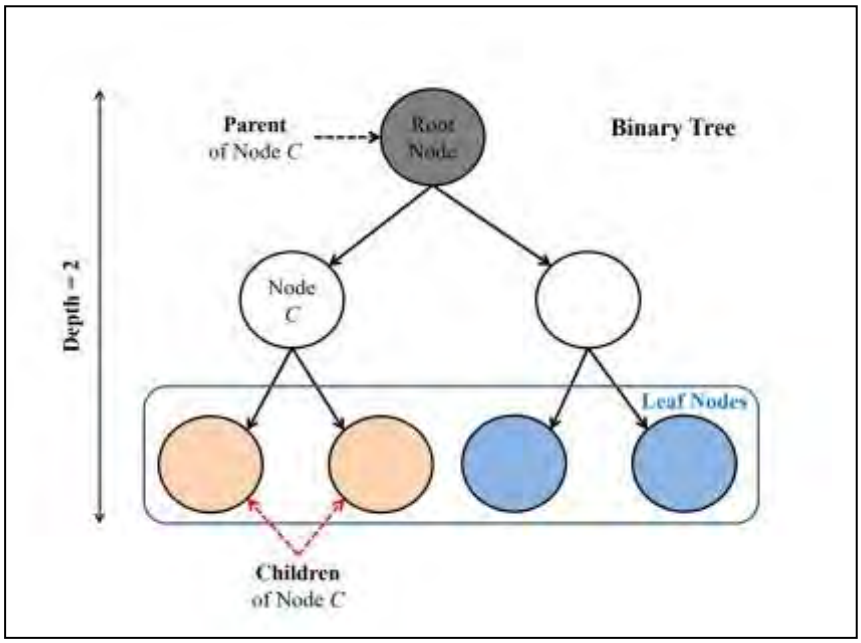
   Simpul daun atau *leaf node* adalah simpul yang tidak memiliki anak (*children*).

5. *K-ary Tree*

   Pohon K-ary adalah pohon dimana setiap simpul (kecuali simpul daun atau *leaf node*) memiliki anak-anak K. Biasanya berjalan dengan pohon biner (K = 2).

6. *Depth*

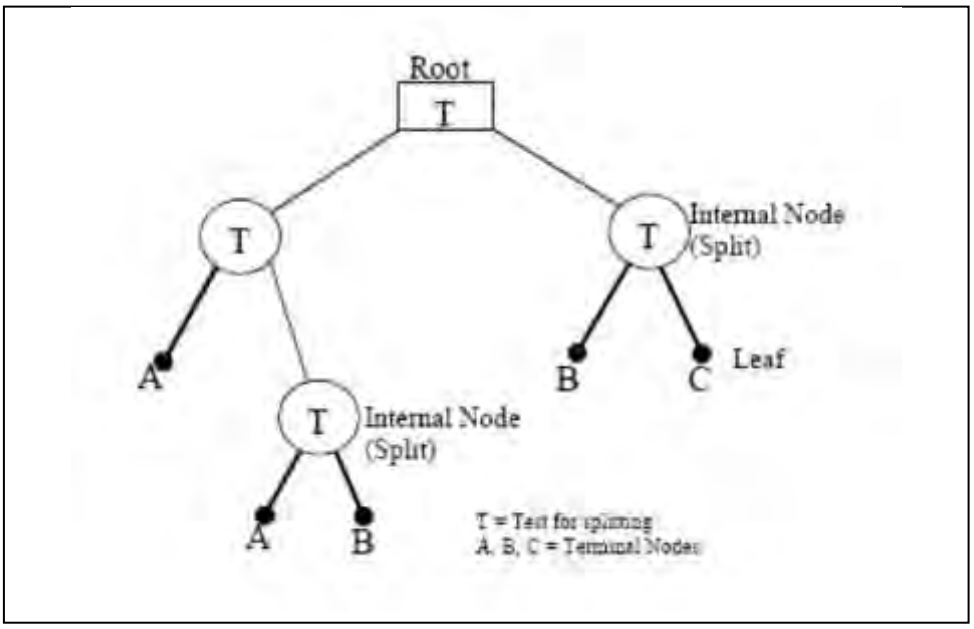
   Kedalaman (*depth*) pohon adalah panjang maksimal jalur dari akar simpul (*root node*) sampai simpul daun (*leaf node*).



Gambar 2.3
Decision Tree
(Sumber : Van der Schaar, 2017:5)

2.3.1 *Classification And Regression Tree*

Menurut Jake Morgan pada jurnalnya yang berjudul "*Classification and Regression Tree Analysis*", menjelaskan bahwa proses statistik di balik klasifikasi dan regresi (*classification and regression tree*) sangat mirip, tetapi kedua hal ini dapat dibedakan dengan sangat jelas. Ketika variabel respon yang diterima adalah numerik atau *continue*, untuk dapat memprediksi hasilnya dengan optimal dapat menggunakan proses pohon regresi (*regression tree*). Sedangkan, proses dari pohon klasifikasi atau *classification tree* adalah membagi data berdasarkan pada homogenitas, di mana pengelompokan didasarkan pada data yang serupa, untuk menyaring "*noise*" dan membuatnya lebih "*pure*" atau murni (konsep kriteria kemurnian). Dalam kondisi jika variabel respon tidak memiliki kelas, model dari regresi cocok untuk masing-masing independen variabel, mengisolasi variabel-variabel ini sebagai *node* sehingga dapat mengurangi kesalahan. (Morgan, 2014:4)



Gambar 2.4
The Structure of CART
(Sumber : Morgan, 2014:4)

Berdasarkan kutipan diatas, dijelaskan bahwa proses CART (*Classification and Regression Tree*) dapat mengelompokkan dan menyaring data berdasarkan variabel respon yang diterima sehingga menghasilkan data yang lebih akurat atau murni yang kemudian dapat digunakan sebagai salah satu metode pengambilan keputusan.

2.3.2 *Computational Consideration*

Untuk dapat menggunakan pohon keputusan (*decision tree*) terdapat beberapa pertimbangan komputasi yang harus dipahami terlebih dahulu. Pertimbangan komputasi itu sendiri dapat dikelompokkan menjadi dua jenis sebagai berikut (Van der Schaar, 2017:26) :

1. *Numerical Features*

   Variabel respon dapat membagi apa saja dengan ambang batas apa pun. Namun untuk kondisi yang diberikan, satu-satu nya titik perpecahan (*split points*) yang perlu dipertimbangkan adalah *n value* dalam data pelatihan. Jika diurutkan setiap kondisi dengan *n value* ini, dapat dengan cepat menghitung ketidakpastian metrik minat (*cross entropy* atau lainnya). Ini membutuhkan □(*dn* log *n*) kali.

2. *Categorical Features*

   Dengan asumsi *q* kategori yang berbeda, ada *2q-1-1* kemungkinan partisi biner yang dapat dipertimbangkan. Namun, hal-hal yang dapat disederhanakan dalam kasus klasifikasi biner (*binary classification*) atau regresi dapat menghasilkan perpecahan optimal (*optimal split)* untuk *cross entropy* dan *Gini* dengan hanya mempertimbangkan *q-1* kemungkinan perpecahan (*possible splits*).

2.3.3 Keuntungan dan Kerugian

Menurut Mihaela van der Schaar ada beberapa keuntungan dan kerugian pada saat menggunakan pohon keputusan (*decision tree*) diantaranya adalah sebagai berikut (Van der Schaar, 2017:27) :

1. *Advantages* (Keuntungan)

   Mudah ditafsirkan oleh manusia (jika pohonnya tidak terlalu besar). Efisien secara komputasi dalam menangani data numerik dan kategorikal. Bersifat parametrik sehingga detail atau *compact* tidak seperti *Nearest Neighborhood Classification,* metode ini tidak harus membawa contoh data pelatihan dalam membangun blok untuk berbagai metode *ensemble*.

2. *Disadvantages* (Kerugian)

   Teknik pelatihan yang bersifat heuristik. Menemukan partisi ruang yang meminimalkan kesalahan empiris atau *NP-hard.* Metode ini menggunakan pendekatan yang tergolong serakah dengan teori fondasi yang terbatas.

2.4 Android

Berisi beberapa penjelasan seputar Android mengenai pengertian, arsitektur, fitur-fitur didalamnya, Android version, dan Android SDK.

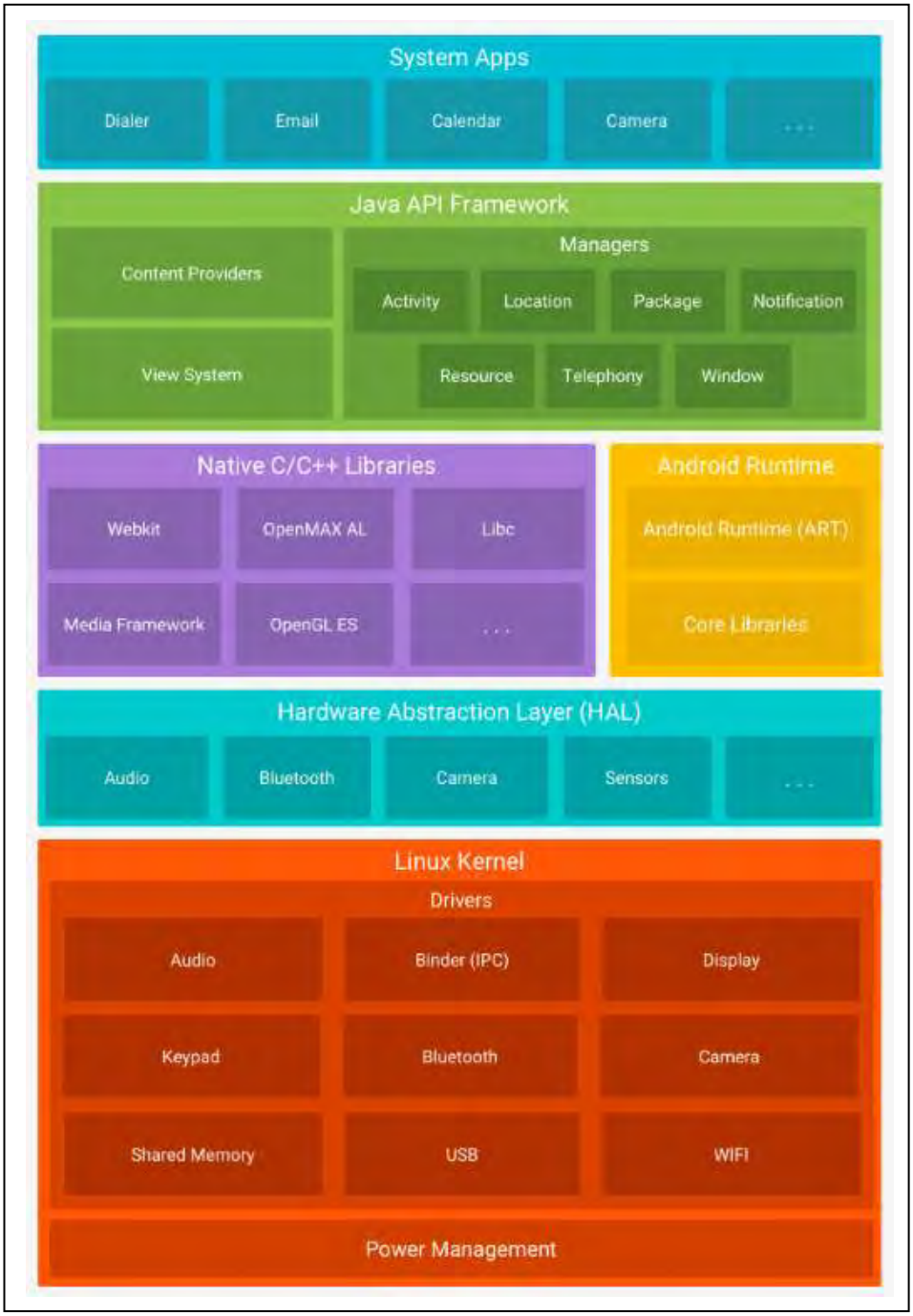
2.4.1 Pengertian Android

Berdasarkan Chryssa Aliferi pada bukunya yang berjudul "*Android Programming Cookbook*", menjelaskan bahwa Android adalah sistem operasi *open-source* yang bernama Android. Google telah membuat kode untuk semua "*low-level stuff*" serta *middleware* yang diperlukan untuk mendukung dalam menggunakan perangkat elektronik, dan memberikan Android secara gratis kepada siapa saja yang ingin menulis kode dan membangun sistem operasi. Bahkan ada kerangka aplikasi lengkap yang disertakan, sehingga aplikasi pihak ketiga dapat dibangun dan dipasang, kemudian tersedia bagi pengguna untuk berjalan sesuai keinginan. (Aliferi, 2016:1)

Hal ini juga dijelaskan oleh Wei-Meng Lee pada bukunya yang berjudul "*Beginning Android Application Development*", yang menjelaskan bahwa Android adalah sistem operasi seluler yang didasarkan pada versi modifikasi dari Linux. Awalnya dikembangkan oleh Android, Inc. Pada tahun 2005, sebagi bagian dari strategi untuk memasuki pasar mobile, Google membeli Android dan mengambil alih pekerjaan pengembangannya dan juga beserta dengan tim pengembangannya. (Lee, 2011:2)

2.4.2 Arsitektur Android

Untuk memahami cara kerja Android, lihat gambar 2.15, yang menunjukkan berbagai lapisan yang membentuk sistem operasi Android (OS).



Gambar 2.5
Arsitektur Android
(Sumber : https://developer.android.com/guide/platform)

Berdasarkan situs resmi dari *developer.android.com,* menjelaskan bahwa sistem operasi Android secara kasar dibagi menjadi enam bagian dalam lima lapisan utama, yaitu:

1. Linux *Kernel*

   Linux *kernel* adalah fondasi dasar platform dari Android. Sebagai contoh, *Android Runtime* (ART) bergantung pada kernel Linux untuk fungsionalitas dasar seperti threading dan pengelolaan memori tingkat rendah. Menggunakan kernel Linux memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan driver perangkat keras untuk kernel yang cukup dikenal.

2. *Hardware Abstraction Layer* (HAL)

   Hardware Abstraction Layer (HAL) memberikan antarmuka standar yang mengungkap kemampuan perangkat keras perangkat ke kerangka kerja API Java yang lebih tinggi. HAL terdiri atas beberapa modul pustaka, masing-masing menerapkan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Ketika API kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut.

3. Android *Runtime*

   Untuk perangkat yang menjalankan Android versi 5.0 (API level 21) atau lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap Android Runtime (ART). ART ditulis guna menjalankan beberapa mesin virtual pada perangkat bermemori rendah dengan mengeksekusi file DEX, format bytecode yang dirancang khusus untuk Android yang dioptimalkan untuk footprint memori minimal. Buat rantai aplikasi, misalnya Jack, mengumpulkan sumber Java ke bytecode DEX, yang dapat berjalan pada platform Android.

4. *Native C/C++ Libraries*

   Banyak komponen dan layanan sistem Android inti seperti ART dan HAL dibuat dari kode bawaan yang memerlukan pustaka bawaan yang tertulis dalam C dan C++. Platform Android memungkinkan kerangka kerja API Java meningkatkan fungsi beberapa pustaka bawaan pada aplikasi. Misalnya, dapat mengakses OpenGL ES

melalui kerangka kerja API OpenGL Java Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi.

5. *Java API Framework*

   Keseluruhan rangkaian fitur pada Android OS tersedia melalui API yang ditulis dalam bahasa Java. API ini membentuk elemen dasar yang harus buat aplikasi Android dengan menyederhanakan penggunaan ulang inti, komponen dan layanan sistem modular, yang mencakup berikut ini:

   a. Tampilan Sistem = yang kaya dan luas dapat digunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan browser web yang dapat disematkan.
   b. Pengelola Sumber Daya = memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file layout.
   c. Pengelola Notifikasi = yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status.
   d. Pengelola Aktivitas = yang mengelola siklus hidup aplikasi dan memberikan back-stack navigasi yang umum.
   e. Penyedia Materi = yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.

6. *System Apps*

   Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh pengembang atau *developer* dari aplikasi itu sendiri. Misalnya, jika aplikasi ingin mengirimkan pesan SMS, maka tidak perlu membangun fungsi tersebut sendiri. Karena bisa juga menjalankan aplikasi SMS mana saja yang telah diinstal guna mengirimkan pesan kepada penerima yang ingin cantumkan.

2.4.3 Fitur-Fitur Android

Android bersifat *open-source* dan tersedia secara bebas bagi para produsen untuk kustomisasi (membolehkan pengguna untuk bebas mengatur Android dalam

mengikuti gaya yang diinginkan), tidak ada konfigurasi perangkat keras dan perangkat lunak tetap. Namun, ada beberapa fitur-fitur yang dapat Android dukung diantaranya adalah sebagai berikut (Lee, 2011:3) :

1. *Storage*

   Penyimpanan dapat menggunakan SQLite, database relasional yang ringan, untuk penyimpanan data.

2. *Connectivity*

   Mendukung sambungan GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, *Bluetooth* (termasuk A2DP dan AVRCP), WiFi (*Wireless Fidelity*), LTE, dan WiMAX (Worldwide Interoperability for Microwave Access).

3. *Messaging*

   Mendukung fitur SMS (*Short Message Service*) dan MMS (*Multimedia Messaging Service*).

4. *Web Browser*

   Fitur yang didasarkan pada *open-source* WebKit, bersama dengan Chrome V8 JavaScript engine.

5. *Media Support*

   Termasuk dukungan atas beberapa media berikut: H.263, H.264 (dalam 3GP atau MP4 *container*), MPEG-4 SP, AMR, AMR-WB (dalam 3GP *container*), AAC, HE-AAC (dalam MP4 atau 3GP *container*), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, dan BMP.

6. *Hardware Support*

   Dukungan perangkat keras seperti: *Accelerometer Sensor*, kamera, kompas digital, *Proximity Sensor,* dan GPS (*Global Positioning System*).

7. *Multi-touch*

   Mendukung terhadap layar multi sentuh (*multi-tuch screen*).

8. *Multi-tasking*

   Mendukung terhadap aplikasi multi tugas (*multi-tasking applications*).

9. *Flash Support*

   Fitur dukungan ini mulai ada pada Android 2.3 yang mendukung Flash 10.1.

10. *Tethering*

    Mendukung dalam berbagi atas koneksi internet sebagai *hotspot* tanpa kabel (*wired/wireless hotspot*).

2.4.4 Android *Version*

Tabel 2.2
Sejarah Singkat dari Versi Android

Android Version	Release Date	Codename
-	Oktober 2008	BASE
1.1	Februari 2009	BASE_1_1
1.5	Mei 2009	CUPCAKE
1.6	September 2009	DONUT
2.0	November 2009	ECLAIR
2.2	Juni 2010	FROYO
2.3	November 2010	GINGERBREAD
3.0	Februari 2011	HONEYCOMB
4.0	Oktober 2011	ICE_CREAM_SANDWICH
4.1	Juni 2012	JELLY_BEAN
4.4	Oktober 2013	KITKAT
5.0	November 2014	LOLLIPOP
6.0	September 2015	MARSHMALLOW
7.0	Agustus 2016	NOUGAT
8.0	Agustus 2017	OREO
9.0	Agustus 2018	PIE
10.0	September 2019	Android 10

(Sumber : https://developer.android.com/reference/android/os/Build.VERSION_CODES)

2.4.5 Android SDK

Bagian penting berikutnya dari perangkat lunak yang harus diunduh tentunya adalah Android *Software Development Kit* (SDK). Android SDK berisi *debugger*, *libraries*, *sebuah emulator, documentation*, *sample code*, dan *tutorials.* (Lee, 2011:7)

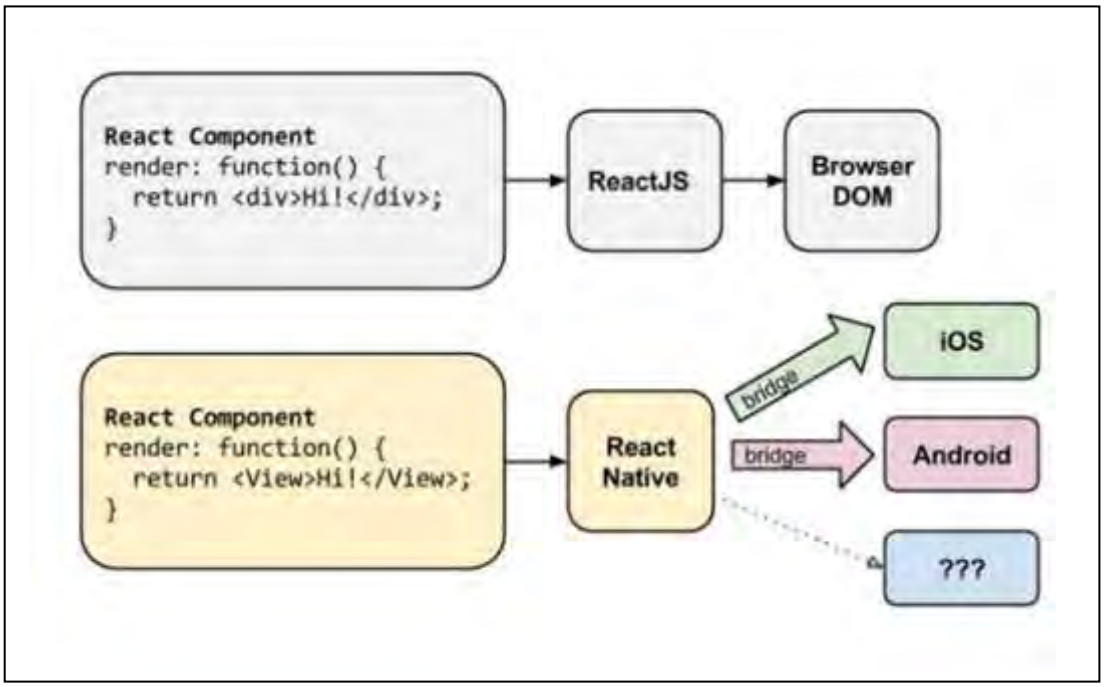
2.5 React

Menurut William Danielsson pada jurnalnya yang berjudul "*React Native Application Development*", menjelaskan bahwa React atau kadang sering disebut dengan React.js adalah kerangka kerja JavaScript yang dikembangkan oleh Facebook dan dirilis

sebagai sumber terbuka (*open-source*) pada tahun 2013 untuk membantu komunitas pengembangan dalam membangun antarmuka atau *interface*. React menyediakan tampilan atau *views* yang merupakan bagian dari pengembangan dengan paragdigma MVC (*Model – View – Controller*) dan sebagai kerangka kerja yang melayani V di MVC. Kekuatan sebenarnya dari React terletak pada bagaimana para pengembang atau *developer* membuat suatu kode program. React adalah suatu gaya pemrograman deklaratif menggambarkan apa yang harus dilakukan tetapi tidak bagaimana seharusnya dilakukan, sehingga menghasilkan jumlah kode program yang lebih sedikit dengan gaya pemrograman imperatif yang menggambarkan bagaimana melakukannya. (Dannielsson, 2016:8)

2.5.1 React Native

Pada konferensi React.js di tahun 2015, Facebook memperkenalkan kerangka kerja baru yaitu React Native, sebuah kerangka kerja yang akan merevolusi cara aplikasi seluler dibuat. Pertama kali React Native dirilis, hanya ada dukungan iOS tetapi sejak saat itu dukungan untuk Android telah ditambahkan dan masih dikembangkan. React Native berjalan dalam instansi yang disematkan dari JavaScriptCore (iOS) atau V8 (Android) di dalam aplikasi dan memberikan komponen spesifik platform tingkat lebih tinggi. Komponen JavaScript dideklarasikan dengan menggunakan seperangkat alat primitif bawaan yang didukung oleh iOS atau komponen Android. (Danielsson, 2016:10)

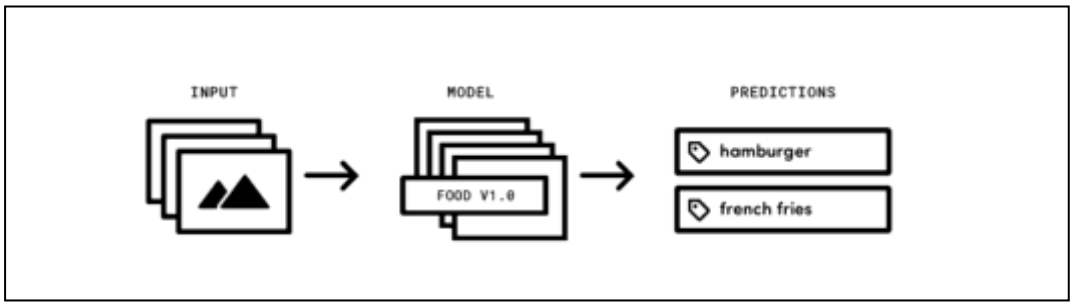


Gambar 2.6
Rendering in React and React Native
(Sumber : Danielsson, 2016:10)

2.5.2 Expo

Menurut situs resmi dari *expo.io*, menjelaskan bahwa Expo adalah sebuah kerangka kerja dan *platform* untuk aplikasi React universal. Expo merupakan seperangkat alat dan layanan yang dibangun di sekitar platform React Native dan *native* yang membantu para pengembang atau *developer* dalam mengembangkan, membangun, menyebarkan, dan beralih dengan cepat di iOS, Android, dan aplikasi berbasis Web dengan basis kode JavaScript atau TypeScript yang sama.

2.6 Clarifai



Gambar 2.15
Workflow of Clarifai API
(Sumber : https://docs.clarifai.com/)

Menurut situs resmi dari *clarifai.com*, menjelaskan bahwa Clarifai adalah sebuah perusahaan kecerdasan buatan (AI) yang berspesialisasi dalam *computer vision* dan menggunakan *machine learning* dan *deep neural networks* untuk mengidentifikasi dan menganalisis gambar dan video. Perusahaan ini menawarkan solusinya melalui API dan *mobile SDK*. Ini dari teknologi Clarifai didasarkan pada *convolutional neural networks*, yaitu proses yang memungkinkan komputer untuk belajar dari contoh data dan menarik kesimpulannya sendiri, memberikan kemampuan pada aplikasi untuk memprediksi tag atau nama yang benar pada gambar atau video yang akan dideteksi. Ada beberapa fitur-fitur yang dapat *clarifai* dukung diantaranya adalah sebagai berikut:

1. Request API

Gambar 2.7
Request Predict API of Clarifai
(Sumber : https://www.clarifai.com/models/general-image-recognition-model-aaa03c23b3724a16a56b629203edc62c)

Untuk dapat melakukan permintaan *Application Programming Interface* (*request Predict* API) pada clarifai, kita memerlukan API_KEY sebagai *authorization* (otorisasi). Hak otorisasi didapatkan pada saat melakukan pendaftaran pada *portal.clarifai.com/signup*. Model untuk *predict* API ini sendiri sangat beragam, seperti *apparel, color, demographics, face detection, food, general, nsfw,* dan lain sebagainya. Sebagai contoh untuk memanggil *predict* API dengan model ‘General’, cukup memasukkan input gambar dengan URL yang dapat diakses publik atau dengan mengirim byte gambar secara langsung.

2. Response API

Gambar 2.8
Response Predict API of Clarifai
(Sumber : https://www.clarifai.com/models/general-image-recognition-model-aaa03c23b3724a16a56b629203edc62c)

Predict API mengembalikan daftar konsep (seperti mobil, gadis, dan kebahagiaan) dengan skor probabilitas yang sesuai pada kemungkinan bahwa konsep-konsep ini hadir dalam gambar yang dikirimkan. Terdapat tiga komponen dari respon yang diberikan pada *predict* API, antara lain sebagai berikut (*clarifai.com/model/*):

a. *Model*

Keterangan model yang digunakan untuk membuat prediksi pada input API. Termasuk *id, name, created_at date, output info,* dan informasi pada versi model.

b. *Input*

Data gambar yang dikirimkan melalui model untuk membuat prediksi. Termasuk id dan data gambar.

c. *Data*

Data dikirimkan kembali dengan tanggapan, biasanya mencakup konsep yang terdeteksi dan nilai probabilitas yang sesuai. Kemudian data akan diambil dan diolah dengan menggunakan metode yang diterapkan agar menjadi suatu informasi mengenai suatu binatang yang tepat.

2.7 Visual Studio Code

Berdasarkan situs resmi dari *code.visualstudio.com,* menjelaskan bahwa Visual Studio Code adalah sumber editor kode yang ringan namun kuat yang dapat berjalan pada *desktop* dan tersedia untuk Windows, macOS dan Linux. Visual Studio Code muncul dengan dukungan bawaan untuk JavaScript, TypeScript dan Node.js yang memiliki lingkungan ekstensi yang banyak akan bahasa lain (seperti C++, C#, Java, Phyton, PHP, Go) dan *runtimes* (seperti .NET dan Unity).

Visual Studio Code adalah editor kode gratis, yang berjalan pada sistem operasi macOS Linux, dan Windows. Visual Studio Code ringan dan dapat dijalankan pada sebagian besar perangkat keras dan versi platform. Untuk sistem pembaharuan atau *update*, Visual Studio Code merilis versi baru setiap bulan dengan fitur baru dan perbaikan bug penting. Sebagian besar platform mendukung pembaruan otomatis dan para *developer* akan diminta untuk menginstal rilis baru ketika tersedia. Pengguna juga

dapat memeriksa pembaruan secara manual dengan menjalankan **Help > Check for Updates** pada Linux dan Windows sedangkan pada macOS dapat dengan menjalankan **Code > Check for Updates**. Beberapa ekstensi Visual Studio Code yang memungkinkan pihak ketiga dalam menambahkan dukungan tambahan diantaranya adalah sebagai berikut:

1. *Language* – C++, C#, Go, Java, Python
2. *Tools* – ESLint, JSHint, PowerShell
3. *Debuggers* – Chrome, PHP XDebug
4. *Keymaps* – Vim, Sublime Text, IntelliJ, Emacs, Atom, Visual Studio, Eclipse

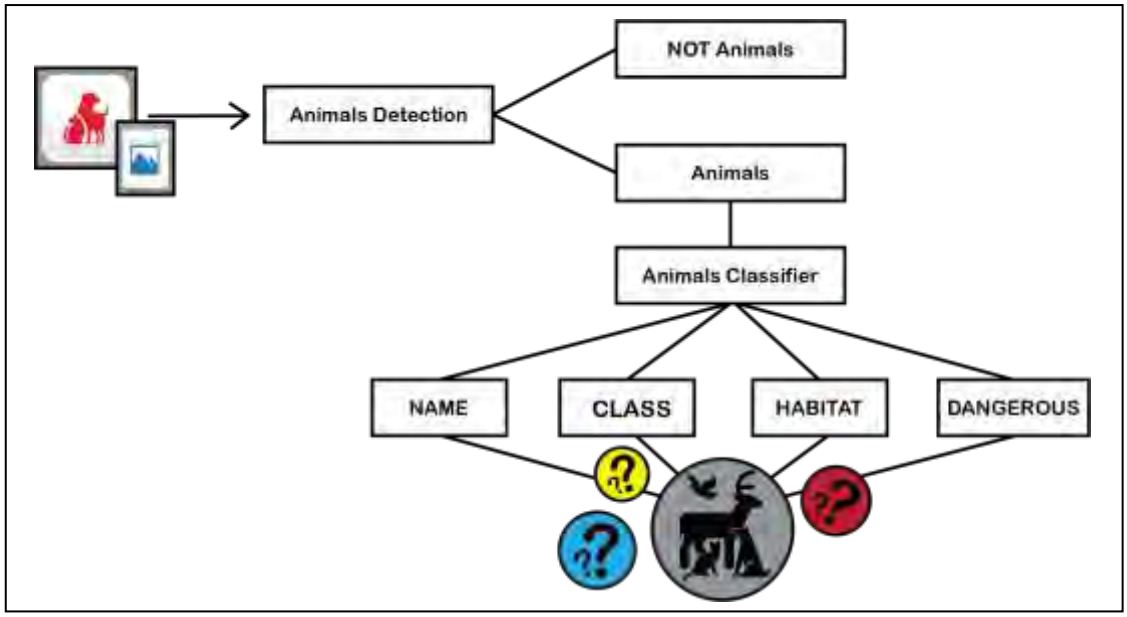
Ekstensi diintegrasikan ke dalam UI, perintah, dan sistem yang menjalankan tugas Visual Studio Code sehingga akan memudahkan *developer* dalam bekerja dengan teknologi yang berbeda melalui antarmuka bersama Visual Studio Code.

BAB III

ANALISIS DAN RANCANGAN SISTEM

3.1 Gambaran Umum Perangkat Lunak

Perangkat lunak pengenal suatu binatang ini nantinya akan dikembangkan sebagai perangkat lunak *object recognition* yang dapat dijalankan di *mobile*, dimana menyediakan fitur-fitur untuk mengenali suatu objek binatang dan memberikan beberapa informasi penting dari binatang tersebut, seperti: nama, *class*, habitat asli, dan tergolong binatang yang berbahaya bagi manusia atau tidak. Gambar 3.1 adalah gambaran umum dari perangkat lunak pengenal suatu binatang yang dirancang.



Gambar 3.1
Gambaran Umum Perangkat Lunak Pengenal Suatu Binatang

Pada gambar 3.1 menggambarkan perangkat lunak menerima masukkan gambar atau memperoleh suatu foto dari kamera pengguna, kemudian *animals detection* akan mendeteksi gambar tersebut apakah merupakan binatang (*animals*) atau bukan binatang (*not animals*). Apabila gambar tersebut merupakan binatang, perangkat lunak akan melakukan *animals classifier* pada gambar sebelumnya untuk mendapatkan beberapa informasi dari gambar tersebut, seperti: nama, *class,* habitat, dan *dangerous* (binatang yang berbahaya atau tidak). Kemudian informasi-informasi tersebut akan ditampilkan dan dapat disimpan oleh pengguna.

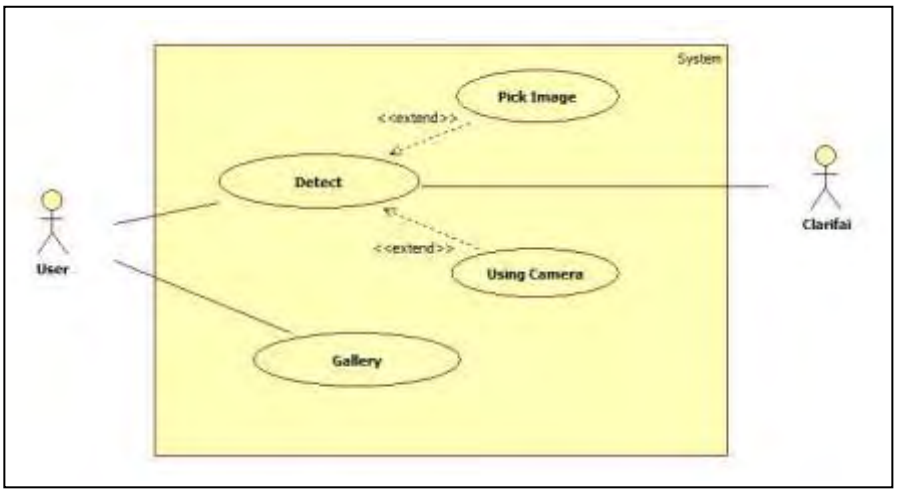
3.2 Spesifikasi Kebutuhan Perangkat Lunak

Perangkat lunak pengenal suatu binatang ini diharapkan dapat memenuhi spesifikasi kebutuhan antara lain adalah sebagai berikut ini:

1. Kebutuhan Fungsional
    a. Perangkat lunak dapat mendeteksi suatu binatang dengan menggunakan kamera atau gambar pada *smartphone* pengguna.
    b. Perangkat lunak dapat memberikan beberapa informasi mengenai suatu binatang yang dideteksi kepada pengguna, seperti: *name*, *class*, habitat asli, dan golongan binatang yang berbahaya atau tidak.
    c. Perangkat lunak dapat menyimpan hasil pengenalan suatu binatang yang dilakukan oleh pengguna.
    d. Perangkat lunak dapat menampilkan hasil pengenalan suatu binatang yang disimpan oleh pengguna sebelumnya.
2. Kebutuhan Non-fungsional
    a. Perangkat lunak dapat mengatur tingkat perbedaan antara objek binatang dengan objek yang lainnya.
    b. Perangkat lunak dapat mengatur tingkat keanekaragaman individu atau *class* pada suatu binatang yang dideteksi.

3.3 *Use Case Diagram*

Use case diagram perangkat lunak pengenal suatu binatang ditunjukkan pada gambar 3.2. Terdapat dua pilihan utama yang tersedia di dalam aplikasi. *Use case detect* merupakan fitur untuk mendeteksi dan mengenali suatu binatang melalui foto atau gambar yang akan diupload pada perangkat lunak. *Use ca*se *gallery* merupakan fitur untuk menampilkan seluruh hasil pengenalan suatu binatang yang telah disimpan oleh pengguna.



Gambar 3.2
Use Case Diagram Perangkat Lunak Pengenal Suatu Binatang

3.4 *Scenario Diagram*

Scenario diagram berfungsi untuk menggambarkan aksi aktor dan bagaimana sistem menanggapi terhadap aksi yang akan dilakukan oleh aktor tersebut. Berdasarkan *use case diagram* pada gambar 3.2, terdapat scenario *"Detect"* dan *"Gallery*".

3.4.1 *Scenario "Detect"*

Use Case : *Detect*

Actor : *User*

Pre-Condition : Pengguna memilih menu "*Detect"*

Description : Sistem melakukan proses pendeteksian dan pengenalan suatu binatang baik dengan menggunakan kamera pada *smartphone* secara langsung maupun dengan menggunakan gambar.

Tabel 3.1
Normal Scenario Use Case "Detect" Menggunakan Kamera

Actor	*System*	*Clarifai*
1. *Capture image*		
	2. Mengirimkan dan mengubah gambar menjadi objek *base64*	
		3. Merekam objek *base64*
		4. Mengirimkan hasil pemindaian objek *base64*

Actor	*System*	*Clarifai*
	5. Menentukan objek *base64* adalah binatang atau bukan	
6. Menerima notifikasi bahwa gambar adalah binatang		
	7. Pengenalan binatang (*Animals Classifier)*	
	8. Tampilkan deskripsi mengenai binatang	

Tabel 3.2
Alternatif Pertama *Scenario Use Case "Detect"* Menggunakan Kamera Menyimpan Koleksi Binatang

Actor	*System*	*Clarifai*
1. *Capture image*		
	2. Mengirimkan dan mengubah gambar menjadi objek *base64*	
		3. Merekam objek *base64*
		4. Mengirimkan hasil pemindaian objek *base64*
	5. Menentukan objek *base64* adalah binatang atau bukan	
6. Menerima notifikasi bahwa gambar adalah binatang		
	7. Pengenalan binatang (*Animals Classifier)*	
	8. Tampilkan deskripsi mengenai binatang	
9. Pilih tombol *save to gallery*		
	10. Menyimpan koleksi binatang	
11. Menerima notifikasi data koleksi berhasil di simpan		

Tabel 3.3
Alternatif Kedua *Scenario Use Case "Detect"* Menggunakan Kamera Gagal Mendeteksi Binatang

Actor	*System*	*Clarifai*
1. *Capture image*		
	2. Mengirimkan dan mengubah gambar menjadi objek *base64*	
		3. Merekam objek *base64*
		4. Mengirimkan hasil pemindaian objek *base64*

Actor	*System*	*Clarifai*
	5. Menentukan objek *base64* adalah binatang atau bukan	
6. Menerima notifikasi bahwa gambar tersebut bukan binatang		

Tabel 3.4
Alternatif Ketiga *Scenario Use Case "Detect"* Menggunakan Gambar

Actor	*System*	*Clarifai*
1. Pilih tombol *pick image*		
	2. Membuka album foto	
3. Memilih gambar		
	4. Mengirimkan dan mengubah gambar menjadi objek *base64*	
		5. Merekam objek *base64*
		6. Mengirimkan hasil pemindaian objek *base64*
	7. Menentukan objek *base64* adalah binatang atau bukan	
8. Menerima notifikasi bahwa gambar adalah binatang		
	9. Pengenalan binatang (*Animals Classifier)*	
	10. Tampilkan deskripsi mengenai binatang	

Tabel 3.5
Alternatif Keempat *Scenario Use Case "Detect"* Menggunakan Gambar Menyimpan Koleksi Binatang

Actor	*System*	*Clarifai*
1. Pilih tombol *pick image*		
	2. Membuka album foto	
3. Memilih gambar		
	4. Mengirimkan dan mengubah gambar menjadi objek *base64*	
		5. Merekam objek *base64*
		6. Mengirimkan hasil pemindaian objek *base64*
	7. Menentukan objek *base64* adalah binatang atau bukan	
8. Menerima notifikasi bahwa gambar adalah binatang		

Actor	*System*	*Clarifai*
	9. Pengenalan binatang (*Animals Classifier)*	
	10. Tampilkan deskripsi mengenai binatang	
11. Pilih tombol *save to gallery*		
	12. Menyimpan koleksi binatang	
13. Menerima notifikasi data koleksi berhasil di simpan		

Tabel 3.6
Alternatif Kelima *Scenario Use Case "Detect"* Menggunakan Gambar Gagal Mendeteksi Binatang

Actor	*System*	*Clarifai*
1. Pilih tombol *pick image*		
	2. Membuka album foto	
3. Memilih gambar		
	4. Mengirimkan dan mengubah gambar menjadi objek *base64*	
		5. Merekam objek *base64*
		6. Mengirimkan hasil pemindaian objek *base64*
	7. Menentukan objek *base64* adalah binatang atau bukan	
8. Menerima notifikasi bahwa gambar tersebut bukan binatang		

3.4.2 *Scenario "Gallery"*

Use Case : *Gallery*

Actor : *User*

Pre-Condition : Pengguna memilih menu "*Gallery"*

Description : Pengguna melihat daftar seluruh koleksi binatang yang tersimpan.

Tabel 3.7
Normal Scenario Use Case "Gallery"

Actor	*System*
1. Pilih tombol *gallery*	
	2. Tampilkan daftar seluruh koleksi binatang.
3. Pilih salah satu koleksi binatang	
	4. Tampilkan Informasi Binatang dari koleksi

Tabel 3.8
Alternatif Pertama *Scenario Use Case "Gallery"* Melakukan *Delete*

Actor	*System*
1. Pilih tombol *gallery*	
	2. Tampilkan daftar seluruh koleksi binatang.
3. Pilih salah satu koleksi binatang	
	4. Tampilkan Informasi Binatang dari koleksi
5. Pilih tombol "*Delete from gallery"*	
	6. Konfirmasi penghapusan
7. Pilih "Yes"	
	8. Tampilkan *Message Dialog* "Data berhasil dihapus"
	9. Tampilkan kembali daftar seluruh koleksi binatang.

Tabel 3.9
Alternatif Kedua *Scenario Use Case "Gallery"* Melakukan *Delete*
Batal Menghapus

Actor	*System*
1. Pilih tombol *gallery*	
	2. Tampilkan daftar seluruh koleksi binatang.
3. Pilih salah satu koleksi binatang	
	4. Tampilkan Informasi Binatang dari koleksi
5. Pilih tombol "*Delete from gallery"*	
	6. Konfirmasi penghapusan
7. Pilih "*Cancel*"	
	8. Tutup konfirmasi penghapusan

Tabel 3.10
Alternatif Ketiga *Scenario Use Case "Gallery"* Melakukan *Detect*

Actor	*System*
1. Pilih tombol *gallery*	
	2. Tampilkan daftar seluruh koleksi binatang.
3. Pilih salah satu koleksi binatang	
	4. Tampilkan Informasi Binatang dari koleksi
5. Pilih tombol "*pick image"* dan memilih gambar	
	6. Mendeteksi binatang (*Animals Detection*)
7. Menerima notifikasi bahwa gambar adalah binatang	
	8. Pengenalan binatang (*Animals Classifier*)
	9. Tampilkan deskripsi mengenai binatang

Tabel 3.11
Alternatif Keempat *Scenario Use Case "Gallery"* Melakukan *Detect*
Menyimpan Koleksi Binatang

Actor	*System*
1. Pilih tombol *gallery*	
	2. Tampilkan daftar seluruh koleksi binatang.
3. Pilih salah satu koleksi binatang	
	4. Tampilkan Informasi Binatang dari koleksi
5. Pilih tombol "*pick image"* dan memilih gambar	
	6. Mendeteksi binatang (*Animals Detection*)
7. Menerima notifikasi bahwa gambar adalah binatang	
	8. Pengenalan binatang (*Animals Classifier*)
	9. Tampilkan deskripsi mengenai binatang
10. Pilih tombol *save to gallery*	
	11. Menyimpan koleksi binatang
12. Menerima notifikasi data koleksi berhasil di simpan	

Tabel 3.12
Alternatif Kelima *Scenario Use Case "Gallery"* Melakukan *Detect*
Gagal Mendeteksi Binatang

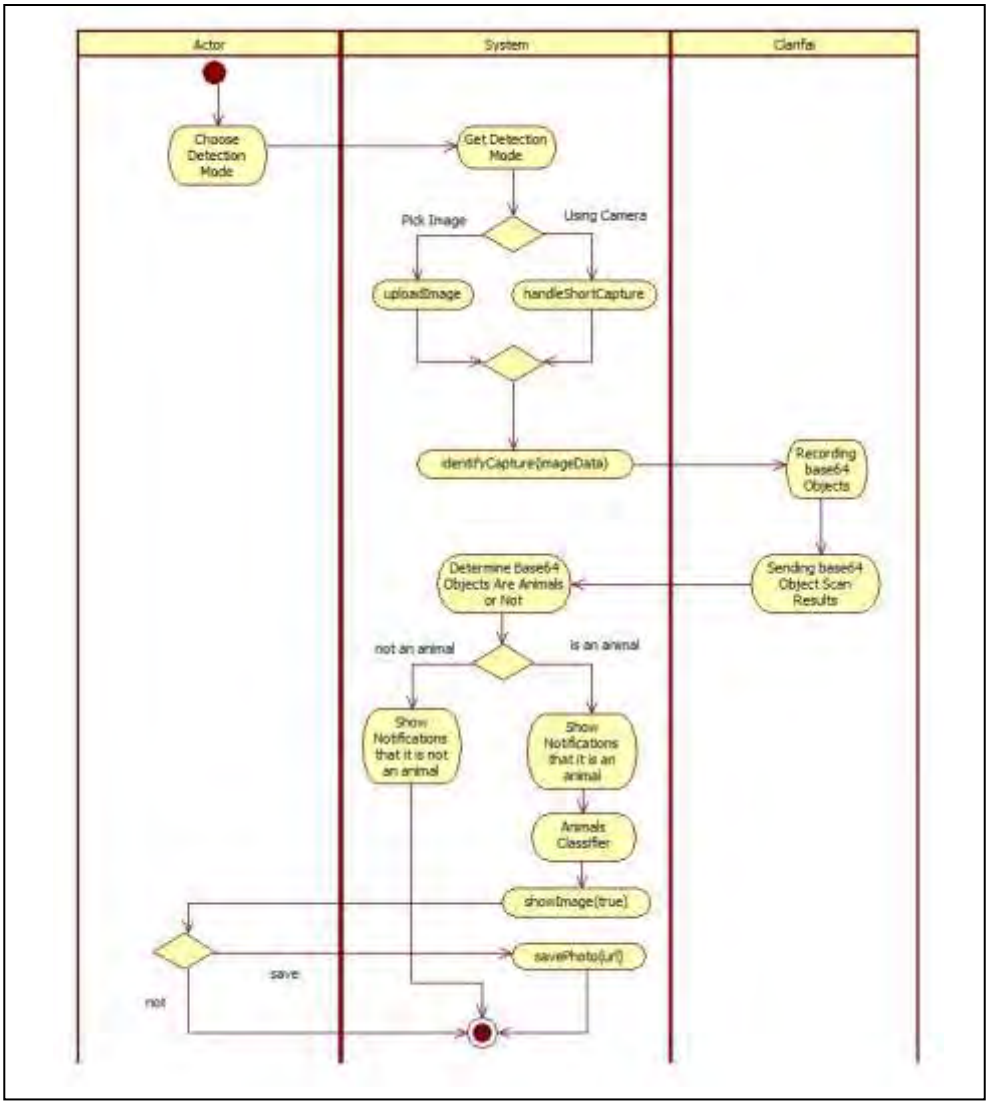
Actor	*System*
1. Pilih tombol *gallery*	
	2. Tampilkan daftar seluruh koleksi binatang.
3. Pilih salah satu koleksi binatang	
	4. Tampilkan Informasi Binatang dari koleksi
5. Pilih tombol "*pick image"* dan memilih gambar	
	6. Mendeteksi binatang (*Animals Detection*)
7. Menerima notifikasi bahwa gambar tersebut bukan binatang	

3.5 *Activity Diagram*

Activity diagram berfungsi untuk menggambarkan urutan kegiatan dari perangkat lunak yang bisa digunakan untuk menunjukkan arus kontrol perangkat lunak. Berdasarkan *use case diagram* pada gambar 3.2, terdapat *activity diagram "Detect"* dan *activity diagram "Gallery"*.

3.5.1 *Activity Diagram "Detect"*

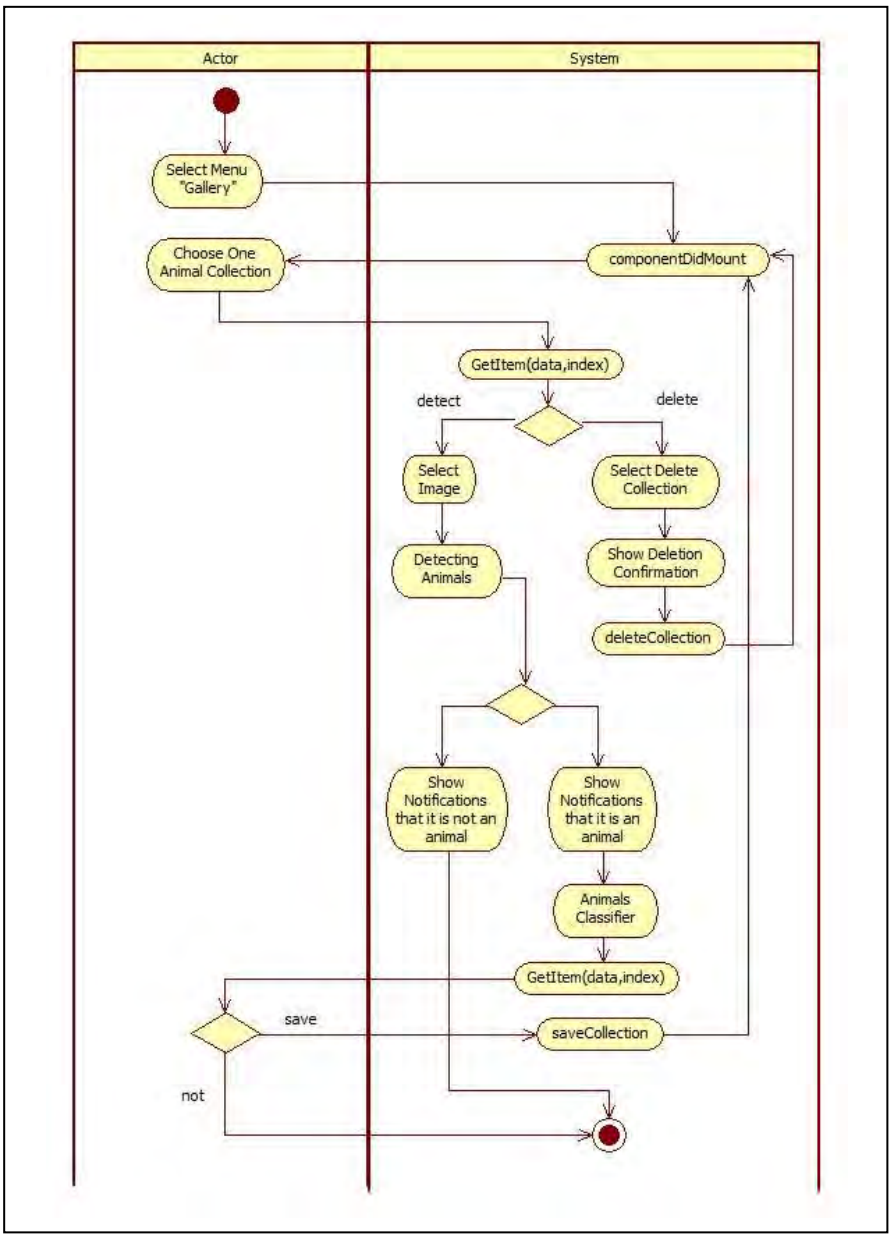
Activity diagram "Detect" ditunjukkan pada gambar 3.3. Pada saat perangkat lunak pengenal suatu binatang dijalankan, pengguna akan langsung diarahkan pada fitur *detect*, dimana perangkat lunak akan langsung mengaktifkan kamera *smartphone* untuk melakukan pendeteksian binatang. Selain menggunakan kamera *smartphone*, perangkat lunak pengenal suatu binatang ini juga menyediakan fitur *pick image*, dimana akan langsung mengarahkan pada album (*gallery*) *smartphone* sehingga pengguna dapat memilih gambar binatang apa yang hendak dideteksi dan dikenali.



Gambar 3.3
Activity Diagram "Detect"

3.5.2 *Activity Diagram "Gallery"*

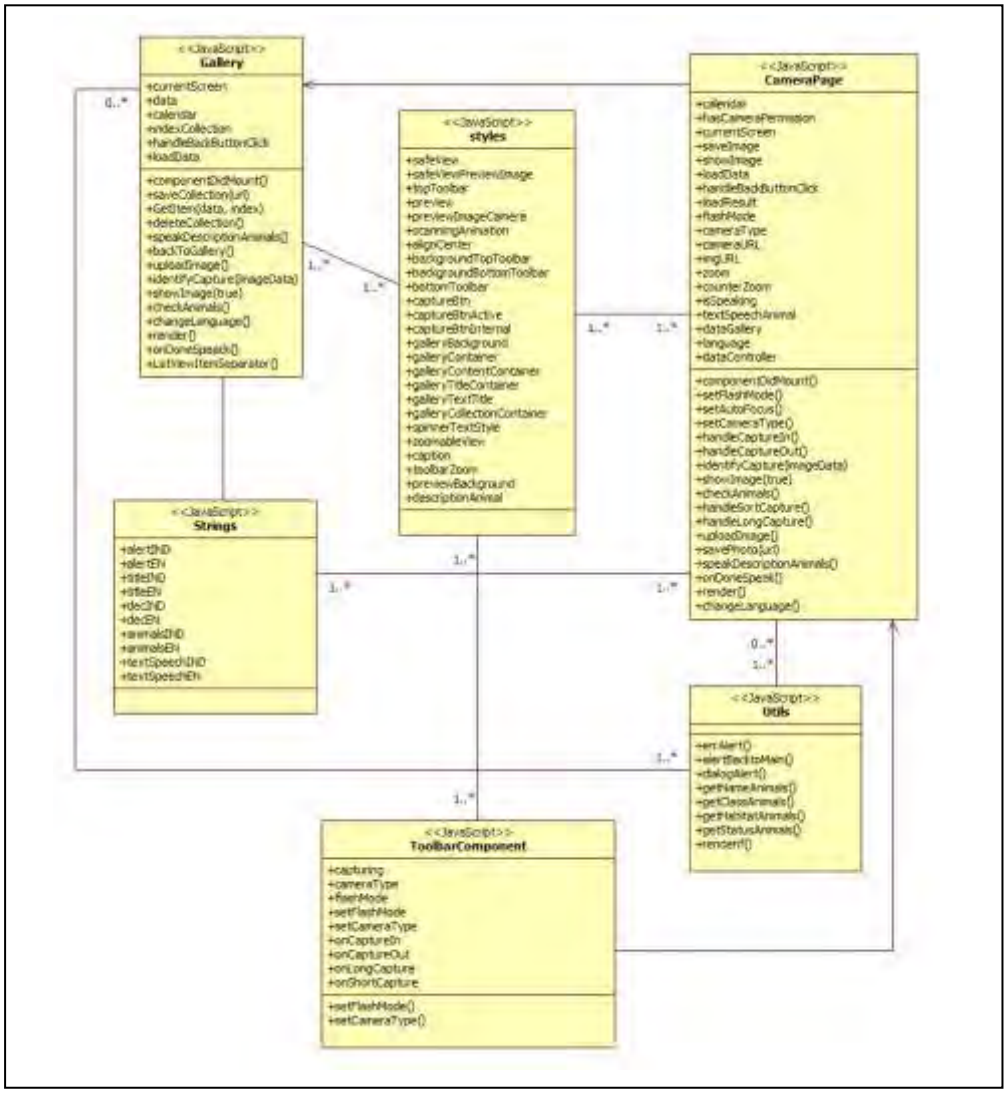
Activity diagram "Gallery" ditunjukkan pada gambar 3.4. Pada saat pengguna memilih fitur ini, perangkat lunak akan langsung menampilkan daftar seluruh koleksi binatang yang telah disimpan sebelumnya oleh pengguna, baik itu dengan menggunakan kamera ataupun dengan memasukkan gambar. Setelah itu, pengguna dapat melihat detail atau menghapus koleksi binatang tersebut pada daftar koleksi binatang (*gallery*).



Gambar 3.4
Activity Diagram "Gallery"

3.6 *Class Diagram*

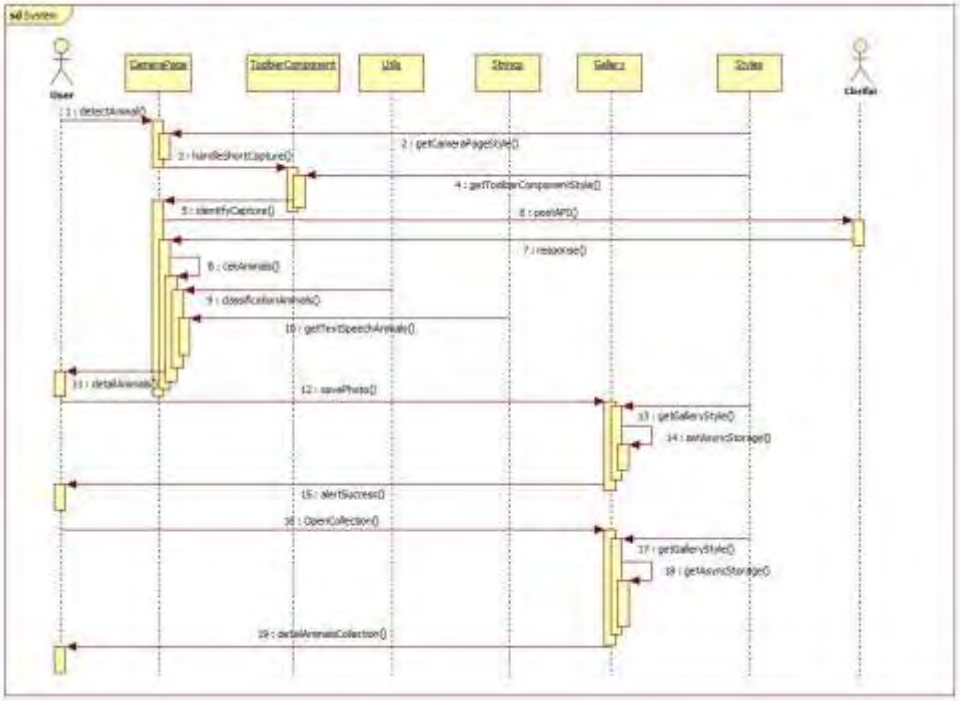
Class diagram berfungsi untuk menggambarkan kelas-kelas dan relasi antar kelas-kelas yang nantinya akan digunakan pada fase implementasi atau pembuatan. *Class diagram* dari perangkat lunak pengenal suatu binatang dapat digambarkan pada gambar 3.5. *Class diagram* pada perangkat lunak pengenal suatu binatang menggunakan kelas sebanyak 6 kelas, yang setiap kelasnya memiliki kegunaannya masing-masing.



Gambar 3.5
Class Diagram Perangkat Lunak Pengenal Suatu Binatang

3.7 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antara aktor dengan objek-objek dari *class diagra*m. *Sequence diagram* dari perangkat lunak pengenal suatu binatang dapat digambarkan pada gambar 3.6.

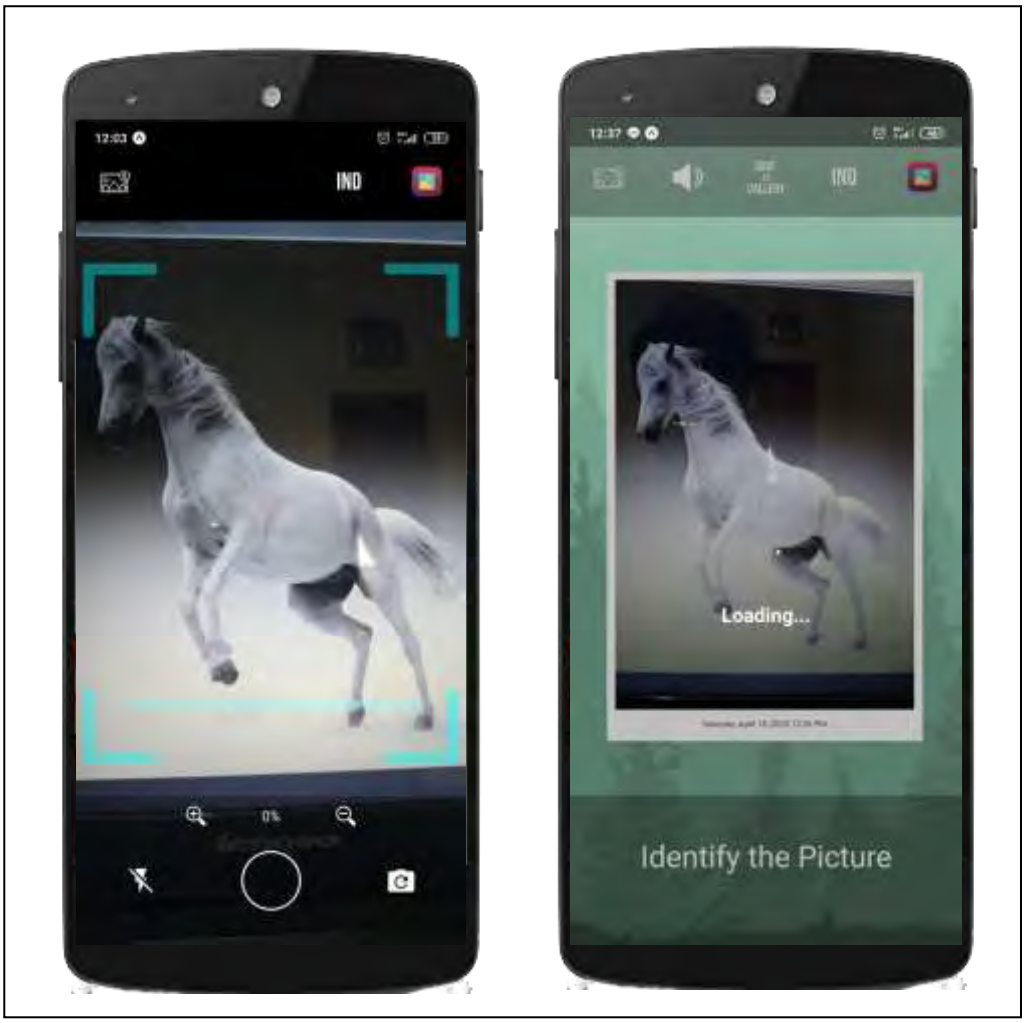


Gambar 3.6
Sequence Diagram Perangkat Lunak Pengenal Suatu Binatang

Aktor atau pengguna berinteraksi dengan objek *Camera Page* ketika aktor melakukan pendeteksian binatang dengan menggunakan kamera atau gambar. Kelas *Styles* merupakan kelas penyimpan seluruh ketentuan gaya atau model, dimana berfungsi untuk menetapkan ukuran *margin*, *font*, *padding*, dan *attribute* pada suatu tampilan. Selanjutnya objek *Camera Page* akan mengidentifikasi gambar dengan mengubah gambar menjadi objek *base64* untuk menerima *response* atau tanggapan dari *clarifai API*. Kelas *Utils* akan melakukan pengklasifikasian suatu binatang jika objek tersebut dianggap sebagai binatang. Kemudian untuk informasi binatang akan didapatkan dari objek *Strings* berdasarkan hasil dari proses pengklasifikasian tersebut. Pengguna juga berinteraksi dengan objek *Gallery* ketika aktor mengolah data hasil pengenalan suatu binatang seperti menyimpan atau menghapus koleksi pada *gallery*.

3.8 Rancangan Antarmuka

Rancangan antarmuka pada perangkat lunak pengenal suatu binatang ini dapat digambarkan dengan beberapa tampilan dibawah ini:

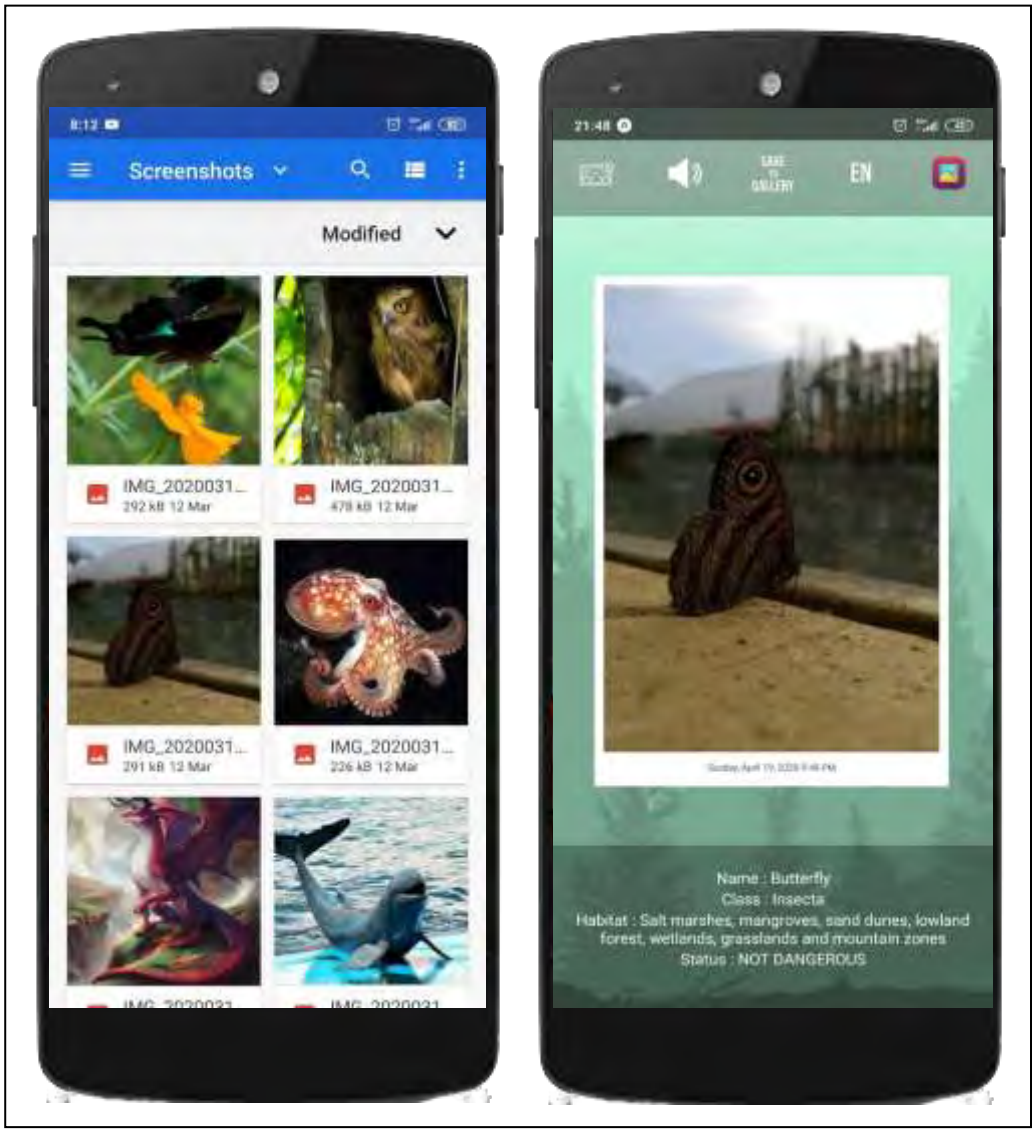


Gambar 3.7
Tampilan *Camera Page*

Gambar 3.6 adalah tampilan *camera page* atau halaman kamera yang akan muncul ketika perangkat lunak ini dijalankan. Pada halaman ini, pengguna dapat memilih beberapa fitur umum, yaitu: *pick image*, *language, gallery, zoom in, zoom out, flash, flip camera,* dan *capture*. Pada sebelah kanan adalah tampilan ketika sistem sedang melakukan proses pengenalan binatang melalui pengambilan foto atau memilih gambar.

Gambar 3.8
Tampilan *Gallery*

Gambar 3.7 adalah tampilan *gallery* yang menampilkan daftar seluruh koleksi binatang yang tersimpan. Pada sebelah kanan adalah tampilan *detail gallery* yang akan muncul ketika pengguna memilih salah satu koleksi binatang. Di dalam *detail gallery* terdapat informasi mengenai nama, *class*, habitat asli, deskripsi dari binatang tersebut dan beberapa menu yang dapat pengguna gunakan, seperti: tombol *pick image, speech, delete from gallery, language*, dan tombol *gallery.*



Gambar 3.9
Tampilan *Detect with Pick Image*

Gambar 3.8 adalah tampilan aplikasi jika menggunakan *mode* deteksi binatang dengan mengambil gambar *(pick image)*. Aplikasi akan secara langsung membuka album gambar yang tersimpan di dalam *smartphone* pengguna dan melakukan pengenalan suatu binatang pada gambar yang diambil atau dipilih.

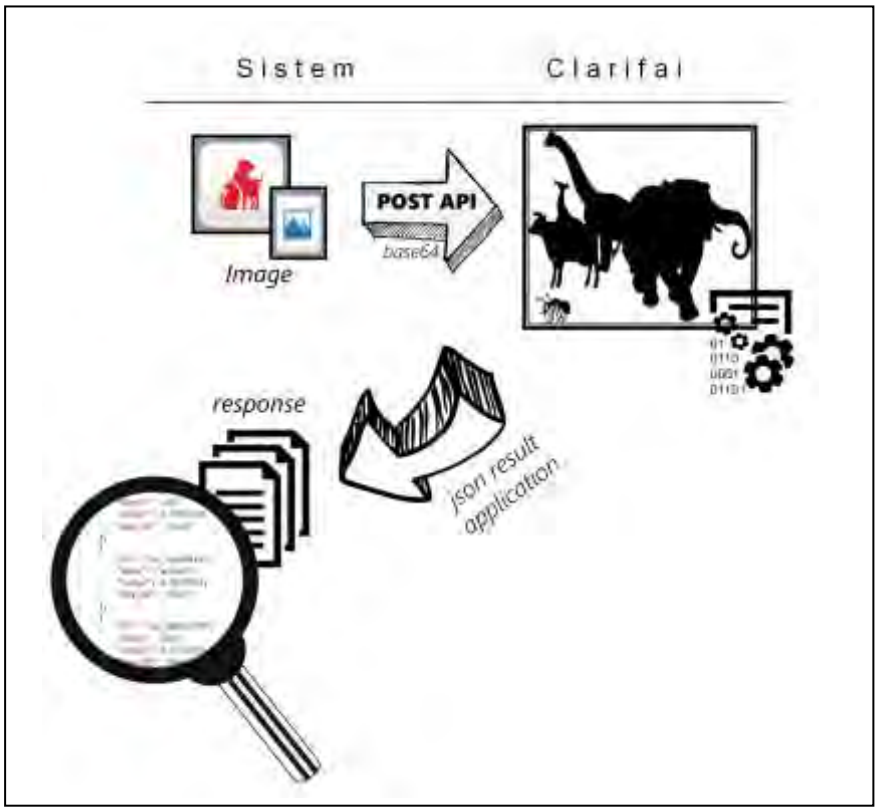
BAB IV

IMPLEMENTASI DAN PENGUJIAN PERANGKAT LUNAK

4.1 Implementasi

Pada tahap implementasi dilakukan pengumpulan data gambar dan foto pada perangkat lunak dengan membagi data tersebut ke dalam dua kategori utama, yaitu data yang merupakan binatang dan data yang bukan merupakan binatang. Pada pengumpulan data yang merupakan binatang, akan dikategorikan kembali menjadi data binatang yang berbahaya dan tidak berbahaya. Sedangkan pada pengumpulan data yang bukan merupakan binatang, akan digunakan sebagai data pengujian pada perangkat lunak. Setelah pengumpulan data berhasil dilakukan, tahap selanjutnya adalah tahap pembuatan perangkat lunak pengenal suatu binatang. Pada tahapan akhir akan dilakukan pengujian tampilan dan fungsi pada perangkat lunak pengenal suatu binatang.

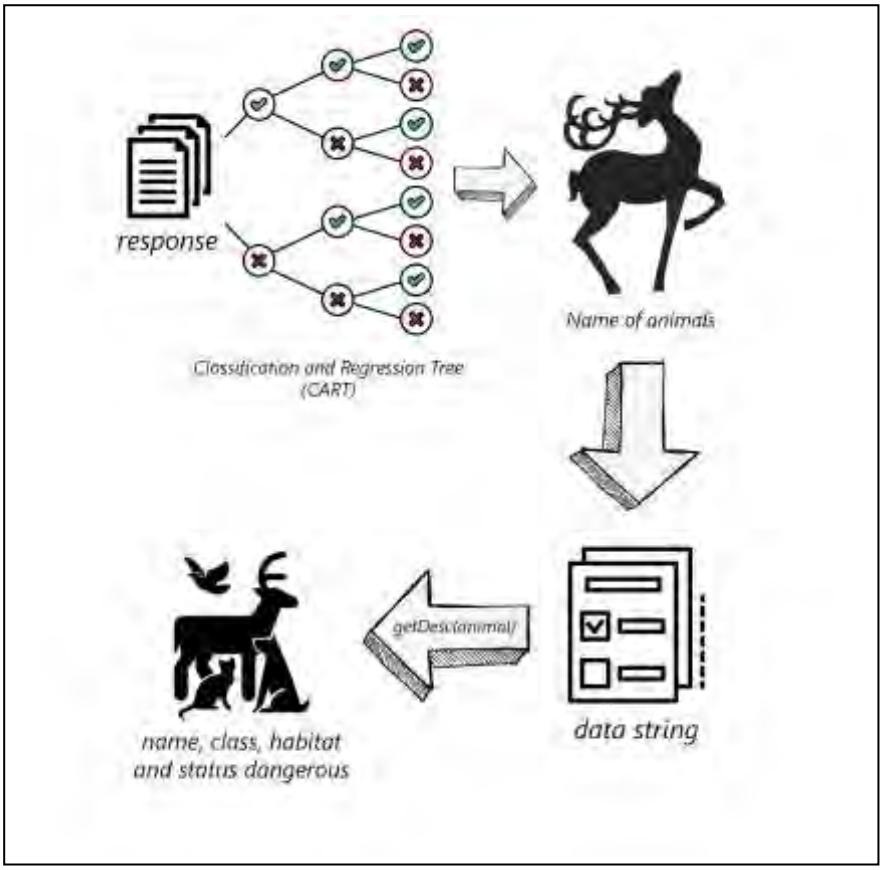
4.1.1 Pendeteksian



Gambar 4.1
Implementasi Proses Pendeteksian

Gambar 4.1 adalah proses pendeteksian pada perangkat lunak pengenal suatu binatang. Pendeteksian yang dilakukan pada perangkat lunak pengenal suatu binatang ini dengan menggunakan bantuan *clarifai API.* Dimana hasil gambar yang dimasukkan pengguna kedalam perangkat lunak pengenal suatu binatang akan diubah menjadi *base64* yang merupakan salah satu syarat pemanggilan *clarifai API* dengan bentuk model yang ditentukan (POST). Kemudian *clarifai API* akan mengirimkan tanggapan atau *response* berupa *array of object* yang berisikan hasil seluruh objek yang ada pada gambar tersebut. Dari hasil ini perangkat lunak pengenal suatu binatang melakukan pendeteksian binatang dengan cara mencari *response* atau tanggapan berupa *"animal"* dengan tingkat probabilitas diatas 0.5 yang berarti gambar tersebut adalah binatang.

4.1.2 Pengklasifikasian



Gambar 4.2
Implementasi Proses Pengklasifikasian

Penggambaran cara kerja pengklasifikasian pada perangkat lunak pengenal suatu binatang ini dapat dilihat pada gambar 4.2. Pengklasifikasian merupakan tahap

yang dilakukan setelah proses pendeteksian. Hasil dari pendeteksian kemudian akan dilanjutkan untuk menentukan suatu informasi mengenai nama, kelas, habitat, dan status binatang yang tergolong berbahaya atau tidak pada suatu binatang yang telah berhasil dideteksi. Cara kerja pengklasifikasian perangkat lunak pengenal suatu binatang ini dengan cara mencari seluruh ciri *response* atau tanggapan dengan tingkat probabilitas tertentu pada suatu binatang dengan melakukan analisis dari setiap *response* yang diberikan oleh *clarifai API*. Ciri *response* atau tanggapan suatu binatang tersebut kemudian akan dilanjutkan dengan proses *regression tree* untuk menghasilkan satu nama binatang yang akurat. Kemudian perangkat lunak pengenal suatu binatang akan mengambil informasi mengenai kelas, habitat, dan status binatang berbahaya atau tidak melalui data *string* yang tersimpan berdasarkan nama binatang tersebut.

4.2 Spesifikasi Kebutuhan Perangkat Keras

Spesifikasi kebutuhan perangkat keras yang digunakan pada saat implementasi dan pengujian perangkat lunak pengenal suatu binatang adalah sebagai berikut :

Tabel 4.1
Spesifikasi Kebutuhan Perangkat Keras saat Implementasi

No	Perangkat Keras	Spesifikasi
1.	Android version	10 QKQ1.190910.002
2.	RAM	4.00 GB
3.	CPU	Octa-core Max 2.2GHz
4.	Kernel version	builder@c4-miui-ota-bd10.bj
5.	Internal storage	64.00 GB

Tabel 4.2
Spesifikasi Kebutuhan Perangkat Keras saat Pengujian

No	Perangkat Keras	Spesifikasi
1.	Android version	10 QKQ1.190910.002
2.	RAM	4.00 GB
3.	CPU	Octa-core Max 2.2GHz
4.	Kernel version	builder@c4-miui-ota-bd10.bj
5.	Internal storage	64.00 GB

4.3 Pengujian Tampilan Antar Muka

Secara garis besar, tampilan perangkat lunak pengenal suatu binatang dapat dibagi menjadi dua bagian utama yaitu, tampilan untuk memfoto atau mengupload

gambar binatang, dan tampilan untuk melihat daftar koleksi binatang yang disimpan. Berikut merupakan rincian dari tampilan tersebut: tampilan *camera page*, dan tampilan *gallery.*

4.3.1 Tampilan *Camera Page*

Gambar 4.3 merupakan tampilan dari *camera page*. Tampilan *camera page* muncul setelah perangkat lunak dijalankan. Pada sebelah kanan adalah tampilan ketika sistem telah selesai melakukan proses pengenalan binatang melalui pengambilan foto atau mengupload gambar.

Gambar 4.3
Tampilan Pengujian *Camera Page*

Berdasarkan pada gambar 4.3, terdapat sepuluh komponen yang dapat dideskripsikan dari tampilan tersebut. Sepuluh komponen tersebut diantaranya:

1. Tombol *pick image,* merupakan tombol untuk mengunggah gambar dari *smartphone* pengguna dalam bentuk image picker.
2. Tombol *language*, merupakan tombol untuk merubah bahasa perangkat lunak.
3. Tombol *gallery*, merupakan tombol untuk membuka album koleksi binatang.

4. Tombol *zoom in*, merupakan tombol untuk memperbesar objek pada kamera.
5. Tombol *zoom out*, merupakan tombol untuk memperkecil objek pada kamera.
6. Tombol *flash*, merupakan tombol untuk menyalakan *flash* atau lampu kamera.
7. Tombol *flip*, merupakan tombol untuk merubah tipe kamera (kamera depan atau kamera belakang).
8. Tombol *capture*, merupakan tombol untuk mengambil foto dengan menggunakan kamera secara langsung.
9. Tombol *speech*, merupakan tombol untuk mendengarkan deskripsi binatang berdasarkan bahasa yang telah dipilih.
10. Tombol *save to gallery*, merupakan tombol untuk menyimpan hasil pengenalan binatang.

4.3.2 Tampilan *Gallery*

Gambar 4.4 merupakan tampilan *gallery*. Tampilan *gallery* muncul ketika pengguna memilih tombol *gallery* pada halaman *camera page*. Pada sebelah kanan adalah tampilan *detail gallery* yang akan muncul ketika pengguna memilih salah satu koleksi binatang yang tersimpan.

Gambar 4.4
Tampilan Pengujian *Gallery*

Berdasarkan pada gambar 4.4, terdapat sepuluh komponen yang dapat dideskripsikan dari tampilan tersebut. Sepuluh komponen tersebut diantaranya:

1. *Collection list,* merupakan daftar hasil pengenalan binatang yang disimpan oleh pengguna.
2. Tombol *pick image,* merupakan tombol untuk mengunggah gambar dari *smartphone* pengguna dalam bentuk image picker.
3. Tombol *speech*, merupakan tombol untuk mendengarkan deskripsi binatang berdasarkan bahasa yang telah dipilih.
4. Tombol *delete from gallery*, merupakan tombol untuk menghapus data koleksi.
5. Tombol *language*, merupakan tombol untuk merubah bahasa perangkat lunak.
6. Tombol *gallery*, merupakan tombol untuk membuka album koleksi binatang.

4.4 Pengujian Fungsi Perangkat Lunak

Pengujian fungsi perangkat lunak bertujuan untuk menemukan kesalahan-kesalahan atau kekurangan pada perangkat lunak pengenal suatu binatang ketika program dijalankan. Pengujian ini bertujuan untuk mengetahui apakah perangkat lunak pengenal suatu binatang sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak atau tidak. Pengujian perangkat lunak menggunakan metode *black box* yang berfokus pada persyaratan fungsional perangkat lunak.

4.4.1 Pengujian Fungsi Tampilan *Camera Page*

Tabel 4.3 merupakan pengujian fungsi pada tampilan *camera page* pada perangkat lunak pengenal suatu binatang. Pengujian fungsi pada tampilan *camera page* dilakukan dengan menguji setiap tombol fungsi yang ada dan mencoba salah satu beberapa fungsi komponen *user interface* seperti *single picker*, *message dialog*, dan *backhandle* dengan berbagai kondisi.

Tabel 4.3
Pengujian Fungsi Tampilan *Camera Page*

Data yang diuji	Hasil yang diharapkan	Hasil Pengujian	Status
Menekan tombol *pick image* (no 1 pada gambar 4.1)	Memunculkan *image picker*	Memunculkan *image picker*	*Valid*
Menekan tombol *language* (no 2 pada gambar 4.1)	Memunculkan *pop-up single picker* bahasa yang tersedia	Memunculkan *pop-up option button* bahasa yang tersedia	*Valid*
Menekan tombol *gallery* (no 3 pada gambar 4.1)	Masuk ke tampilan *gallery*	Masuk ke tampilan *gallery*	*Valid*
Menekan tombol *zoom in* (no 4 pada gambar 4.1)	Memperbesar objek pada kamera	Memperbesar objek pada kamera	*Valid*
Menekan tombol *zoom out* (no 5 pada gambar 4.1)	Memperkecil objek pada kamera	Memperkecil objek pada kamera	*Valid*
Menekan tombol *flash* (no 6 pada gambar 4.1)	Mengaktifkan lampu kamera	Mengaktifkan lampu kamera	*Valid*
Menekan tombol *flip* (no 7 pada gambar 4.1)	Merubah tipe kamera (kamera depan atau belakang)	Merubah tipe kamera (kamera depan atau belakang)	*Valid*
Menekan tombol *capture* (no 8 pada gambar 4.1)	Mengambil foto melalui kamera, Sistem melakukan proses pengenalan binatang	Mengambil foto melalui kamera, Sistem melakukan proses pengenalan binatang	*Valid*
Menekan tombol *speech* (no 9 pada gambar 4.1)	Menjalankan fitur suara untuk menjelaskan deskripsi binatang	Menjalankan fitur suara untuk menjelaskan deskripsi binatang	*Valid*
Menekan tombol *save to gallery* (no 10 pada gambar 4.1)	Menyimpan hasil pengenalan binatang	Menyimpan hasil pengenalan binatang	*Valid*
Mengupload gambar melalui *image picker*	Sistem melakukan proses pengenalan binatang	Sistem melakukan proses pengenalan binatang	*Valid*
Membuka *image picker*, Membatalkan pick gambar	Kembali ke halaman *camera page*	Kembali ke halaman *camera page*	*Valid*
Menekan tombol back *device*	Menampilkan *pop-up* konfirmasi “apakah ingin keluar dari aplikasi?”	Menampilkan *pop-up* konfirmasi “apakah ingin keluar dari aplikasi?”	*Valid*
Menekan tombol back *device,* Memilih *alert dialog “Ya”*	Aplikasi keluar	Aplikasi keluar	*Valid*

Data yang diuji	Hasil yang diharapkan	Hasil Pengujian	Status
Menekan tombol back *device,* Memilih *alert dialog "Tidak"*	Menutup *pop-up* konfirmasi dan aplikasi tetap berjalan	Menutup *pop-up* konfirmasi dan aplikasi tetap berjalan	*Valid*
Menekan tombol *language*, Memilih bahasa Indonesia	Sistem menerapkan bahasa Indonesia	Sistem menerapkan bahasa Indonesia	*Valid*
Menekan tombol *language*, Memilih bahasa Indonesia, Menekan tombol *speech*	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Indonesia	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Indonesia	*Valid*
Menekan tombol *language*, Memilih bahasa Inggris	Sistem menerapkan bahasa Inggris	Sistem menerapkan bahasa Inggris	*Valid*
Menekan tombol *language*, Memilih bahasa *speech*	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Inggris	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Inggris	*Valid*

4.4.2 Pengujian Fungsi Tampilan *Gallery*

Tabel 4.4 merupakan pengujian fungsi pada tampilan *gallery* pada perangkat lunak pengenal suatu binatang. Pengujian fungsi pada tampilan *gallery* dilakukan dengan menguji setiap tombol dan mencoba pilihan-pilihan pada daftar koleksi dengan berbagai kondisi.

Tabel 4.4
Pengujian Fungsi Tampilan *Gallery*

Data yang diuji	Hasil yang diharapkan	Hasil Pengujian	Status
Menekan tombol *pick image* (no 2 pada gambar 4.2)	Memunculkan *image picker*	Memunculkan *image picker*	*Valid*
Menekan tombol *speech* (no 3 pada gambar 4.2)	Menjalankan fitur suara untuk deskripsi binatang	Menjalankan fitur suara untuk deskripsi binatang	*Valid*
Menekan tombol *delete from gallery* (no 4 pada gambar 4.2)	Menghapus data koleksi yang dipilih dari daftar koleksi (no 1 pada gambar 4.2)	Menghapus data koleksi yang dipilih dari daftar koleksi (no 1 pada gambar 4.2)	*Valid*
Menekan tombol *language* (no 5 pada gambar 4.2)	Memunculkan *pop-up option button* bahasa yang tersedia	Memunculkan *pop-up option button* bahasa yang tersedia	*Valid*

Data yang diuji	Hasil yang diharapkan	Hasil Pengujian	Status
Menekan tombol *gallery* (no 6 pada gambar 4.2)	Kembali pada tampilan *gallery* (daftar koleksi)	Kembali pada tampilan *gallery* (daftar koleksi)	*Valid*
Mengupload gambar melalui *image picker*	Sistem melakukan proses pengenalan binatang	Sistem melakukan proses pengenalan binatang	*Valid*
Membuka *image picker*, Membatalkan upload gambar	Kembali ke halaman *detail gallery*	Kembali ke halaman *detail gallery*	*Valid*
Menekan tombol back *device*	Masuk ke tampilan *camera page*	Masuk ke tampilan *camera page*	*Valid*
Menekan tombol *language*, Memilih bahasa Indonesia	Sistem menerapkan bahasa Indonesia	Sistem menerapkan bahasa Indonesia	*Valid*
Menekan tombol *language*, Memilih bahasa Indonesia, Menekan tombol *speech*	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Indonesia	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Indonesia	*Valid*
Menekan tombol *language*, Memilih bahasa Inggris	Sistem menerapkan bahasa Inggris	Sistem menerapkan bahasa Inggris	*Valid*
Menekan tombol *language*, Memilih bahasa *speech*	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Inggris	Sistem menjelaskan deskripsi binatang dengan suara dalam bahasa Inggris	*Valid*

4.4.3 Pengujian Hasil Pengklasifikasian

Pada perangkat lunak pengenal suatu binatang ini digunakan sebanyak 2577 *datasets* atau *data example*. Setiap *datasets* atau *data example* mewakili gambar binatang-binatang yang dikumpulkan selama proses pengimplementasian. Pengujian ini dilakukan untuk mengetahui seberapa akurat hasil pengklasifikasian yang dilakukan oleh perangkat lunak dalam mengenal suatu binatang.

Tabel 4.5 merupakan pengujian hasil pengklasifikasian pada perangkat lunak pengenal suatu binatang. Terdapat sekitar 51 klasifikasi jenis binatang dengan tingkat keakuratan yang dapat dilihat sebagai berikut:

Tabel 4.5
Pengujian Tingkat Akurasi Hasil Klasifikasi Jenis Binatang

No	Binatang	Total Akurasi Benar	Total Akurasi Salah	*Data Example*	Persentase Akurasi
1.	*cat*	61	2	63	96.82%
2.	*dog*	51	3	54	94.44%
3.	*horse*	50	0	50	100%
4.	*snake*	49	1	50	98.00%
5.	*butterfly*	49	1	50	98.00%
6.	*rabbit*	48	2	50	96.00%
7.	*bee*	50	0	50	100%
8.	*frog*	50	0	50	100%
9.	*monkey*	50	0	50	100%
10.	*piglet*	50	0	50	100%
11.	*panda*	49	1	50	98.00%
12.	*rat*	48	2	50	96.00%
13.	*cow*	48	2	50	96.00%
14.	*goat*	50	0	50	100%
15.	*worm*	49	1	50	98.00%
16.	*crocodile*	50	0	50	100%
17.	*deer*	50	0	50	100%
18.	*kangaroo*	50	0	50	100%
19.	*elephant*	50	0	50	100%
20.	*chicken*	49	1	50	98.00%
21.	*giraffe*	50	0	50	100%
22.	*squirrel*	48	2	50	96.00%
23.	*turtle*	50	0	50	100%
24.	*crab*	50	0	50	100%
25.	*spider*	49	1	50	98.00%
26.	*bird*	53	0	53	100%
27.	*fish*	56	1	57	98.24%
28.	*duck*	46	4	50	92.00%
29.	*grasshopper*	47	3	50	94.00%
30.	*gorilla*	48	2	50	96.00%
31.	*wolf*	47	3	50	94.00%
32.	*snail*	50	0	50	100%
33.	*sheep*	49	1	50	98.00%
34.	*zebra*	50	0	50	100%
35.	*whale*	46	4	50	92.00%
36.	*shark*	48	2	50	96.00%
37.	*dolphin*	45	5	50	90.00%
38.	*lizard*	50	0	50	100%
39.	*dragonfly*	50	0	50	100%
40.	*jellyfish*	49	1	50	98.00%
41.	*lobster*	50	0	50	100%
42.	*leopard*	49	1	50	98.00%
43.	*tiger*	49	1	50	98.00%

No	Binatang	Total Akurasi Benar	Total Akurasi Salah	*Data Example*	Persentase Akurasi
44.	*flamingo*	50	0	50	100%
45.	*peacock*	48	2	50	96.00%
46.	*parrot*	47	3	50	94.00%
47.	*shrimp*	50	0	50	100%
48.	*beetle*	50	0	50	100%
49.	*caterpillar*	49	1	50	98.00%
50.	*hippopotamus*	50	0	50	100%
51.	*camel*	50	0	50	100%

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penulisan tugas akhir ini, dapat disimpulkan hal-hal sebagai berikut:

1. Proses pendeteksian binatang dengan menggunakan gambar, dapat diterapkan dengan bantuan *clarifai API,* mengubah gambar menjadi objek *base64* dan mengirimkan objek tersebut pada *clarifai* API untuk membuat *resource* baru di Representational State Transfer atau REST server (POST).
2. Metode *classification and regression tree* dalam melakukan pengklasifikasian binatang, dapat diterapkan dengan beberapa langkah sebagai berikut:
   a. Melakukan pengumpulan ciri suatu binatang terhadap *response* atau tanggapan dari *clarifai API*.
   b. Menempatkan setiap ciri *response* atau tanggapan suatu binatang menjadi *node* dan *children tree* beserta dengan tingkat probabilitas atas ciri *response* atau tanggapan tersebut.
   c. Menyimpulkan setiap *children tree* terakhir sebagai hasil keputusan yang akurat.

5.2 Saran

Beberapa saran untuk pengembangan tugas akhir ini adalah sebagai berikut:

1. Melakukan pengumpulan ciri suatu binatang terhadap *response* atau tanggapan dari *clarifai API* yang dilakukan oleh beberapa orang, agar ciri suatu binatang yang terkumpul dapat lebih banyak dan memakan waktu singkat.
2. Menggunakan *cloud database* misalnya *firebase* dalam proses pengklasifikasian akan membantu dalam pemutakhiran data. Penggunaan *cloud database* sangat berpengaruh atas pembaharuan informasi binatang tanpa harus me-*rebuild* atau mengupdate perangkat lunak terlebih dahulu.

3. Pengembangan selanjutnya dapat dikembangkan dengan menambah berbagai bahasa. Penambahan bahasa mempengaruhi cakupan *user* atau pengguna.
4. Penelitian selanjutnya dapat dikembangkan menggunakan metode klasifikasi lainnya seperti *dempster-shafer, simple additive weighting, naive bayes,* atau *k-nearest neighbors,* sehingga dapat dibandingkan antara metode tersebut yang lebih baik dalam melakukan proses pengklasifikasian.

DAFTAR PUSTAKA

Aliferi, Chryssa, 2016, "*Android Programming Cookbook"*, Exelixis Media P. C.

Braude, Michael, Michael E. Bernstein, 2016, "*Software Engineering Modern Approaches Second Edition"*, Waveland Press Inc.

Danielsson, William, 2016, "*React Native Application Development"*, Linkopings University.

Dennis, Barbara, et al., 2015, "*System Analysis and Design An Object-Oriented Approach with UML Fifth Edition"*, Jhon Wiley and Sons.

Krishna, Ranjay, 2017, "*Computer Vision: Foundations and Applications*", Standford University.

Lee, Wei-Meng, 2011, *"Beginning Android Application Development"*, Wiley Publishing Inc.

Morgan, Jake, 2014, "*Classification and Regression Tree Analysis*", Boston University.

Nikiforova, Janis, et al., 2011, "*Role of UML Class Diagram in Object-Oriented Software Development"*, Scientific Journal of Riga Technical University, Computer Science Applied Computer Systems.

Pressman, Bruce, Bruce R. Maxim, 2015, "*Software Engineering A Practitioner's Approach Eighth Edition"*, McGraw-Hill Education.

Rosa, Shalahuddin, M. Shalahuddin, 2013, "*Rekayasa Perangkat Lunak Terstruktur Dan Berorientasi Objek"*, Informatika Bandung.

Solem, Jan Erik, 2012, "*Programming Computer Vision with Python*", O'Reilly Media, Inc.

Sommerville, Ian, 2016, "*Software Engineering Tenth Edition"*, Pearson Education.

Van der Schaar, Mihaela, 2017, "*Classification and Regression Trees*"*,* Department of Engineering Science University of Oxford.

Wiegers, Joy, Joy Beatty, 2013, "*Software Requirements Third Edition"*, Microsoft Press.

https://developer.android.com/studio/intro, diakses tanggal 5 November 2018, pukul 13:49 WIB.

https://www.opencv.org, diakses tanggal 5 November 2018, pukul 14:42 WIB.

https://docs.expo.io/versions/v36.0.0/, diakses tanggal 31 Januari 2020, pukul 13:21 WIB.

https://www.clarifai.com/about, diakses tanggal 31 Januari 2020, pukul 13:36 WIB.

https://code.visualstudio.com/docs, diakses tanggal 3 Februari 2020, pukul 14:46 WIB.

https://docs.clarifai.com/, diakses tanggal 6 Februari 2020, pukul 14:08 WIB.

https://developer.android.com/guide/platform, diakses tanggal 20 Februari 2020, pukul 11:56 WIB.

https://developer.android.com/reference/android/os/Build.VERSION_CODES, diakses tanggal 20 Februari 2020, pukul 12:04 WIB.

https://www.clarifai.com/models/general-image-recognition-model-aaa03c23b3724a16a56b629203edc62c, diakses tanggal 10 Maret 2020, pukul 08:38 WIB.

LAMPIRAN

CameraPage.js

```
import React from 'react';
import { Alert, View, Text, Image, TouchableOpacity, SafeAreaView, BackHandler, Animated, AsyncStorage } from 'react-native';
import * as Permissions from 'expo-permissions';
import { Camera } from 'expo-camera';
import * as  MediaLibrary from 'expo-media-library';
import moment from 'moment';
import * as ImagePicker from 'expo-image-picker';
import Spinner from 'react-native-loading-spinner-overlay';
import * as Speech from 'expo-speech';
import ReactNativeZoomableView from '@dudigital/react-native-zoomable-view/src/ReactNativeZoomableView';
import RNFlash from 'react-native-flash';
import { SinglePickerMaterialDialog } from 'react-native-material-dialog';

import styles from './styles';
import Toolbar from './ToolbarComponent';
import Gallery from './Gallery';
import Utils from './Utils';
import Strings from '../constants/Strings';

export default class CameraPage extends React.Component {

   camera = null;
   constructor(props){
      super(props);
      this.animatedValue = new Animated.Value(0);
      this.value = 0;
      this.animatedValue.addListener(({ value }) => {
        this.value = value;
      });
      this.handleBackButtonClick = this.handleBackButtonClick.bind(this);
      this.state = {
         calendar: moment(Date.now()).format('LLLL'),
         currentScreen: "CameraPage",
         showImage: false,
         loadData: false,
         loadResult: true,
         captures: [],
         flashMode: Camera.Constants.FlashMode.off,
         capturing: null,
         cameraType: Camera.Constants.Type.back,
         hasCameraPermission: null,
         cameraURL: '',
         imgURL : '',
         zoom: 0,
         counterZoom: 0,
         inputRange: [0, 180],
         outputRange: ['0deg', '180deg'],
         isAnimal: false,
         nameAnimal: '',
         classAnimal: '',
         habitatAnimal: '',
```

```
            statusAnimal: '-',
            textSpeechAnimal: "Not animals",
            descriptionIdentify: 'Identify the Picture',
            dataGallery: [],
            isSpeaking: false,
            language: "EN",
            languageArr: ['EN', 'IND'],
            languagePicker: false,
            languageSelectedItem: undefined,
            dataController: "",
        };
    }

    setFlashMode = (flashMode) => this.setState({ flashMode });
    setAutoFocus = (autoFocus) => this.setState({ autoFocus });
    setCameraType = (cameraType) => this.setState({ cameraType });
    handleCaptureIn = () => this.setState({ capturing: true });

    async componentDidMount() {
        const languageDevice = await AsyncStorage.getItem('language');
        console.log("LanguageDevice : ", languageDevice);
        this.setState({
            language: languageDevice == "IND" ? "IND" : "EN",
            languageSelectedItem: this.state.languageArr.map((row, index) => ({ value: index,
label: row }))[0],
        })
        // Speech.speak(this.state.txtSpeak,{
        //     language: 'ind',
        //     pitch: 1,
        //     rate: 1,
        // })


        //await AsyncStorage.clear();

        this.setState({                dataGallery                :                JSON.parse(await
AsyncStorage.getItem("galleryStorage"))})
        if(!this.state.dataGallery){
            this.setState({
                dataGallery: [],
            })
        }
        BackHandler.addEventListener('hardwareBackPress', this.handleBackButtonClick);

        console.log('[CameraPage : ] Image exists : ')
        Permissions.askAsync(Permissions.CAMERA_ROLL)
        .then(status => {
            console.log("Camera Roll Permission: ", status.status);
            if (status.status != "granted") {
                throw new Error("Storage permissions disabled. Please enable them in your
settings.");
            }
        })
        .catch(error => {
            console.log(error);
            Utils.errAlert("Application Alert", error);
        });
```

```
        Permissions.askAsync(Permissions.AUDIO_RECORDING)
        .then(status => {
            console.log("Camera Roll Permission: ", status.status);
            if (status.status != "granted") {
                throw new Error("Storage permissions disabled. Please enable them in your
settings.");
            }
        })
        .catch(error => {
            console.log(error);
            Utils.errAlert("Application Alert", error);
        });
    };

    componentWillUnmount() {
        BackHandler.removeEventListener('hardwareBackPress',
this.handleBackButtonClick);
    }

    handleBackButtonClick() {
        Speech.stop();
        this.setState({isSpeaking: false,})
        if(this.state.showImage == false){
            Alert.alert(
                this.state.language == "EN"? Strings.CameraPage.tittleAlert6 :
Strings.CameraPage.tittleAlert6IND,
                this.state.language == "EN"? Strings.CameraPage.alert6 :
Strings.CameraPage.alert6IND,
                [
                  {text: 'Yes', onPress: () => BackHandler.exitApp()},
                  {text: 'No', onPress: () => console.log('No Pressed')},
                ],
                { cancelable: false }
            )
        }else{
            this.setState({
                isAnimal: false,
                nameAnimal: '',
                showImage: false,
                loadResult: true,
                inputRange: [0, 180],
                descriptionIdentify: 'Identify the Picture',
                outputRange: ['0deg', '180deg'],
            })
        }

        return true;
    }

    handleCaptureOut = () => {
        if (this.state.capturing)
            this.camera.stopRecording();
    };

    identifyCapture = (imageData) =>{
        const Clarifai = require('clarifai');

        const app = new Clarifai.App({
```

```
    apiKey: '22d8a802332443219479016d29810ab6'
});

app.models.predict(Clarifai.GENERAL_MODEL, {base64: imageData},{
    selectConcepts: [
        {name: 'cat'},
        {name: 'dog'},
        {name: 'horse'},
        {name: 'snake'},
        {name: 'butterfly'},
        {name: 'rabbit'},
        {name: 'bee'},
        {name: 'frog'},
        {name: 'monkey'},
        {name: 'piglet'},
        {name: 'panda'},
        {name: 'rat'},
        {name: 'cow'},
        {name: 'goat'},
        {name: 'worm'},
        {name: 'crocodile'},
        {name: 'deer'},
        {name: 'kangaroo'},
        {name: 'elephant'},
        {name: 'chicken'},
        {name: 'giraffe'},
        {name: 'squirrel'},
        {name: 'turtle'},
        {name: 'crab'},
        {name: 'spider'},
        {name: 'bird'},
        {name: 'fish'},
        {name: 'duck'},
        {name: 'grasshopper'},
        {name: 'gorilla'},
        {name: 'wolf'},
        {name: 'snail'},
        {name: 'sheep'},
        {name: 'zebra'},
        {name: 'whale'},
        {name: 'shark'},
        {name: 'dolphin'},
        {name: 'lizard'},
        {name: 'dragonfly'},
        {name: 'jellyfish'},
        {name: 'lobster'},
        {name: 'leopard'},
        {name: 'tiger'},
        {name: 'flamingo'},
        {name: 'peacock'},
        {name: 'parrot'},
        {name: 'shrimp'},
        {name: 'beetle'},
        {name: 'caterpillar'},
        {name: 'hippopotamus'},
        {name: 'camel'},
    ]
})
```

```
      .then(async (response) => {
        await this.checkAnimals(response);
      })
      .catch((err) => alert(err));
      this.setState({loadData: false})
    }


    checkAnimals = async (response) =>{
      if(response.outputs[0].data.concepts[0].value < 0.7){
        this.setState({
          dataController: this.state.language == "EN"? Strings.CameraPage.desc1 :
Strings.CameraPage.desc1IND,
          isAnimal: false,
          loadResult: false,
          isSpeaking: true,
          descriptionIdentify: this.state.language == "EN"? Strings.CameraPage.alert5:
Strings.CameraPage.alert5IND,
          nameAnimal: this.state.language == "EN"? Strings.CameraPage.desc1 :
Strings.CameraPage.desc1IND,
          textSpeechAnimal: this.state.language == "EN"? Strings.CameraPage.alert5:
Strings.CameraPage.alert5IND,
        })
        Speech.speak(this.state.textSpeechAnimal,{
          language: this.state.language == "EN" ? 'en' : 'ind',
          pitch: 1,
          rate: 1,
          onDone: () => this.onDoneSpeack(),
        })
        return Alert.alert(
          this.state.language == "EN"? Strings.CameraPage.tittleAlert5 :
Strings.CameraPage.tittleAlert5IND,
          this.state.language == "EN"? Strings.CameraPage.alert5:
Strings.CameraPage.alert5IND,
          [
            {text: 'OK', onPress: () => console.log('OK Pressed')},
            {text: 'Try Again', onPress: () => this.handleBackButtonClick()},
          ],
          { cancelable: false }
        )
      }else{
        this.setState({
          dataController: response.outputs[0].data.concepts[0].name,
          isAnimal: true,
          descriptionIdentify: "Identify the Picture",
          nameAnimal: await
Utils.getNameAnimals(response.outputs[0].data.concepts[0].name, this.state.language),
          classAnimal:
Utils.getClassAnimals(response.outputs[0].data.concepts[0].name, this.state.language),
          habitatAnimal:
Utils.getHabitatsAnimals(response.outputs[0].data.concepts[0].name,
this.state.language),
          statusAnimal:
Utils.getStatusAnimals(response.outputs[0].data.concepts[0].name, this.state.language),
        })
        await Utils.dialogAnimalAlert(this.state.language =="EN"?
Strings.CameraPage.tittleAlert1: Strings.CameraPage.tittleAlert1IND, this.state.language
```

```
==="EN"? Strings.CameraPage.alert1 +
Utils.getNameAnimals(response.outputs[0].data.concepts[0].name, this.state.language)
: Strings.CameraPage.alert1IND +
Utils.getNameAnimals(response.outputs[0].data.concepts[0].name, this.state.language));

        this.setState({
          isSpeaking: true,
          textSpeechAnimal:
Utils.getTextSpeechAnimals(this.state.nameAnimal.toLowerCase(), this.state.language),
        })

        Speech.speak(this.state.textSpeechAnimal,{
          language: this.state.language == "EN" ? 'en' : 'ind',
          pitch: 1,
          rate: 1,
          onDone: () => this.onDoneSpeack(),
        })
      }
      this.setState({
        loadResult: false,
      })
    }


    handleShortCapture = async () => {
      const photoData = await this.camera.takePictureAsync();
      this.setState({loadData: true, loadResult: true, capturing: false, captures:
[photoData, ...this.state.captures] })
      this.camera.takePictureAsync( {skipProcessing: true}).then( async
(capturephoto)=>{

        console.log("CAMERA TYPE : ", this.state.cameraType)
        if(this.state.cameraType == 1){
          this.setState({
            inputRange: [0, 180],
            outputRange: ['180deg', '360deg'],
          })
        }else{
          this.setState({
            inputRange: [0, 180],
            outputRange: ['0deg', '180deg'],
          })
        }

        const options = {
          base64: true
        };

        const data = await this.camera.takePictureAsync(options)
        this.identifyCapture(data.base64);

        this.setState({
          showImage: true,
          cameraURL: capturephoto.uri,
        })
      });
    };
```

```
  handleLongCapture = async () => {
    const videoData = await this.camera.recordAsync();
    this.setState({ capturing: false, captures: [videoData, ...this.state.captures] });
  };

  uploadImage = async () => {
    Speech.stop();
    this.setState({isSpeaking: false,})
    await Permissions.askAsync(Permissions.CAMERA_ROLL)
    .then(status => {
      console.log("Camera Roll Permission: ", status.status);
      if (status.status != "granted") {
        throw new Error("Storage permissions disabled. Please enable them in your
settings.");
      }
    })
    .then(_ => ImagePicker.launchImageLibraryAsync({ base64: true, allowsEditing:
false, aspect: [4,3] }))
    .then(result => {
      if (!result.cancelled) {
        this.setState({ imgURL: result.base64 });
        this.setState({
          isAnimal: false,
          loadResult: true,
          showImage: true,
          calendar: moment(Date.now()).format('LLLL'),
          cameraURL: result.uri,
        })
        this.identifyCapture(this.state.imgURL);
        // console.log("Image Link : ", result.uri);
      }
      return;
    })
    .catch(error => {
      console.log("[CameraPage] UploadImage - Error: ", error);
    });
  }

  savePhoto = async (url) => {
    Speech.stop();
    this.setState({isSpeaking: false,})
    const asset = MediaLibrary.createAssetAsync(url);
    await this.state.dataGallery.push({"url": url, "name": this.state.nameAnimal, "class":
this.state.classAnimal,
                    "habitat" : this.state.habitatAnimal, "status": this.state.statusAnimal,
"date": this.state.calendar,
                    "language" : this.state.language, "dataController" :
this.state.dataController});
    await AsyncStorage.setItem("galleryStorage",
JSON.stringify(this.state.dataGallery));
    Alert.alert(
      this.state.language == "EN"? Strings.CameraPage.tittleAlert2 :
Strings.CameraPage.tittleAlert2IND,
      this.state.language == "EN"? Strings.CameraPage.alert2 :
Strings.CameraPage.alert2IND,
      [
        {text: 'OK', onPress: () => console.log('OK Pressed')},
      ],
```

```
            { cancelable: false }
        )
    }

    flip_Animation = () => {
        if (this.value >= 90) {
            console.log("this.value >= 90 ")
            Animated.spring(this.animatedValue, {
                toValue: 0,
                tension: 10,
                friction: 8,
            }).start();
        } else {
            console.log("ELSE")
            Animated.spring(this.animatedValue, {
                toValue: 180,
                tension: 10,
                friction: 8,
            }).start();
        }
    };

    speakDescriptionAnimals = async () =>{
        this.setState({ isSpeaking : !this.state.isSpeaking})
        if(this.state.isSpeaking == false){
            await Speech.speak(this.state.textSpeechAnimal,{
                language: this.state.language == "EN" ? 'en' : 'ind',
                pitch: 1,
                rate: 1,
                onDone: () => this.onDoneSpeack(),
            })
        }else{
            Speech.stop();
        }
    }

    onDoneSpeack = () => {
        this.setState({ isSpeaking : false})
    }

    changeLanguage = () => {
        Speech.stop();
        this.setState({
            isSpeaking: false,
            languagePicker: true,
        })
    }

    closePickerLanguage = async () =>{
        this.setState({
            languagePicker: false,
            nameAnimal:          await          Utils.getNameAnimals(this.state.dataController,
this.state.language),
            classAnimal: Utils.getClassAnimals(this.state.dataController, this.state.language),
            habitatAnimal:                    Utils.getHabitatsAnimals(this.state.dataController,
this.state.language),
            statusAnimal:                      Utils.getStatusAnimals(this.state.dataController,
this.state.language),
```

```
        })
        console.log("Language Set by ", this.state.language);
        await AsyncStorage.setItem("language", this.state.language);
        console.log("Success");
        this.setState({
            textSpeechAnimal:
Utils.getTextSpeechAnimals(this.state.nameAnimal.toLowerCase(), this.state.language),
        })
    }

    render() {
        this.SetInterpolate = this.animatedValue.interpolate({
            inputRange: this.state.inputRange,
            outputRange: this.state.outputRange,
        });

        const Rotate_Y_AnimatedStyle = {
            transform: [{ rotateY: this.SetInterpolate }],
        };

        const { flashMode, cameraType, capturing } = this.state;

        const goToGallery = () => {
            Speech.stop()
            this.setState({isSpeaking: false, currentScreen: 'Gallery'})
        }

        const plusZoom = () =>{
            if(!(this.state.counterZoom >= 100)){
                this.setState({zoom : this.state.zoom + 0.1, counterZoom:
this.state.counterZoom + 10})
            }
        }

        const minusZoom = () =>{
            if(!(this.state.counterZoom <= 0)){
                this.setState({zoom : this.state.zoom - 0.1, counterZoom:
this.state.counterZoom - 10})
            }
        }

        if(this.state.currentScreen == "CameraPage" && this.state.showImage == false){
            return (
                <SafeAreaView style={styles.safeView}>
                    <View>
                        <Camera
                            zoom={this.state.zoom}
                            type={cameraType}
                            flashMode={flashMode}
                            style={styles.preview}
                            ref={camera => this.camera = camera}
                        />
                    </View>

                    <View style={styles.scanningAnimation}>
                        <Image style={{width: 378,height: 504, marginTop: 10}} source={
require('../assets/images/Scanning3.gif')}/>
                    </View>
```

```
            <View style={styles.topToolbar}>
              <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height:
50}} onPress={() => this.uploadImage()}>
                <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/localImage.png')}/>
              </TouchableOpacity>
              <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height:
50}} onPress={plusZoom}>
                {/* <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/loading.gif')}/> */}
              </TouchableOpacity>
              <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height:
50}} onPress={minusZoom}>
                {/* <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/loading.gif')}/> */}
              </TouchableOpacity>
              {
                Utils.renderIf(this.state.language == "EN",
                  <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50}} onPress={() => this.changeLanguage()}>
                    <Image style={{width: 30,height: 30, marginTop: 10}} source={
require('../assets/images/EN.png')}/>
                  </TouchableOpacity>
              )}

              {
                Utils.renderIf(this.state.language == "IND",
                  <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50}} onPress={() => this.changeLanguage()}>
                    <Image style={{width: 30,height: 30, marginTop: 10}} source={
require('../assets/images/IND.png')}/>
                  </TouchableOpacity>
              )}
              <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height:
50}} onPress={goToGallery}>
                <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/gallery.png')}/>
              </TouchableOpacity>
            </View>
            <View style={styles.backgroundBottomToolbar}>

            </View>
            <View style={styles.toolbarZoom}>
              <TouchableOpacity style={{flex: 2, alignItems: 'flex-end', width: 20, height:
20}} onPress={plusZoom}>
                <Image style={{width: 20,height: 20}}
source={require('../assets/images/plusZoom.png')}/>
              </TouchableOpacity>
              <View style={{flex: 2, alignItems: 'center', width: 20, height: 20}}>
                <Text style={{color: 'white'}}>{this.state.counterZoom + '%'}</Text>
              </View>
              <TouchableOpacity style={{flex: 2, alignItems: 'flex-start', width: 20, height:
20}} onPress={minusZoom}>
                <Image style={{width: 20,height: 20}}
source={require('../assets/images/minusZoom.png')}/>
              </TouchableOpacity>
            </View>
            <Toolbar
```

```
                    capturing={capturing}
                    flashMode={flashMode}
                    cameraType={cameraType}
                    setFlashMode={this.setFlashMode}
                    setCameraType={this.setCameraType}
                    onCaptureIn={this.handleCaptureIn}
                    onCaptureOut={this.handleCaptureOut}
                    onLongCapture={this.handleLongCapture}
                    onShortCapture={this.handleShortCapture}
                    />
                <SinglePickerMaterialDialog
                    title={Strings.CameraPage.text1}
                    colorAccent={"#2EA1DA"}
                    scrolled={false}
                    items={this.state.languageArr.map((row, index) => ({ value: index, label:
row }))}
                    visible={this.state.languagePicker}
                    selectedItem={this.state.languageSelectedItem}
                    onCancel={() => this.setState({ languagePicker: false })}
                    onOk={ async (result) => {
                        this.setState({
                            languageSelectedItem: result.selectedItem,
                            language: await result.selectedItem.label,
                        });
                        await this.closePickerLanguage();
                    }}
                />
                <Spinner
                    visible={this.state.loadData}
                    textContent={'Loading...'}
                    textStyle={styles.spinnerTextStyle}
                />

            </SafeAreaView>
        );
    }else if(this.state.currentScreen == "CameraPage" && this.state.showImage){
        return (
            <SafeAreaView style={styles.safeViewPreviewImage}>
                <Image
                    style={styles.previewBackground}
                    source={require('../assets/images/Background.jpg')}/>
                <View style={styles.backgroundTopToolbar}>

                </View>
                <View style={styles.topToolbar}>
                    <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height:
50}} onPress={() => this.uploadImage()}>
                        <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/localImage.png')}/>
                    </TouchableOpacity>
                    {
                        Utils.renderIf(!this.state.isSpeaking,
                            <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50}} onPress={() => this.speakDescriptionAnimals()} >
                                <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/speaker.png')}/>
                            </TouchableOpacity>
```

```
                )}
                {
                    Utils.renderIf(this.state.isSpeaking,
                        <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height: 50}} onPress={() => this.speakDescriptionAnimals()} >
                            <Image style={{width: 30,height: 30, marginTop: 10}} source={require('../assets/images/SpeakerON.gif')}/>
                        </TouchableOpacity>
                )}
                <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height: 50,}} onPress={() => this.savePhoto(this.state.cameraURL)}>
                    <Image style={{width: 40,height: 40, marginTop: 5}} source={ require('../assets/images/SaveCollection.png')}/>
                </TouchableOpacity>
                {
                    Utils.renderIf(this.state.language == "EN",
                        <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height: 50}} onPress={() => this.changeLanguage()}>
                            <Image style={{width: 30,height: 30, marginTop: 10}} source={ require('../assets/images/EN.png')}/>
                        </TouchableOpacity>
                )}
                {
                    Utils.renderIf(this.state.language == "IND",
                        <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height: 50}} onPress={() => this.changeLanguage()}>
                            <Image style={{width: 30,height: 30, marginTop: 10}} source={ require('../assets/images/IND.png')}/>
                        </TouchableOpacity>
                )}
                <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 50,height: 50}} onPress={goToGallery}>
                    <Image style={{width: 30,height: 30, marginTop: 10}} source={require('../assets/images/gallery.png')}/>
                </TouchableOpacity>
            </View>

            <ReactNativeZoomableView
                zoomEnabled={true}
                maxZoom={2}
                minZoom={0.5}
                zoomStep={0.25}
                initialZoom={0.9}
                bindToBorders={true}
                // onZoomAfter={this.logOutZoomState}
                style={styles.zoomableView}
                >
                <Animated.Image
                    style={[Rotate_Y_AnimatedStyle, styles.previewImageCamera]}
                    source={{uri: this.state.cameraURL}}
                />
                <Text style={styles.caption}>{this.state.calendar}</Text>

            </ReactNativeZoomableView>
            <View style={styles.backgroundBottomToolbar}>

            </View>
            <View style={styles.bottomToolbar}>
```

```
                {
                    Utils.renderIf(this.state.isAnimal,
                        <View style={styles.descriptionAnimal}>
                            <Text style={{color: 'white', textAlign: 'center'}}>{this.state.language
==          "EN"?          Strings.CameraPage.desc2          +          this.state.nameAnimal:
Strings.CameraPage.desc2IND + this.state.nameAnimal}</Text>
                            <Text style={{color: 'white', textAlign: 'center'}}>{this.state.language
==          "EN"?          Strings.CameraPage.desc3          +          this.state.classAnimal:
Strings.CameraPage.desc3IND + this.state.classAnimal}</Text>
                            <Text          style={{color:          'white',          textAlign:          'center'}}
numberOfLines={3}>{this.state.language          ==          "EN"?          Strings.CameraPage.desc4          +
this.state.habitatAnimal:                    Strings.CameraPage.desc4IND                    +
this.state.habitatAnimal}</Text>
                            <Text style={{color: 'white', textAlign: 'center'}}>{this.state.language
==          "EN"?          Strings.CameraPage.desc5          +          this.state.statusAnimal:
Strings.CameraPage.desc5IND + this.state.statusAnimal}</Text>
                        </View>
                    )}
                    {
                        Utils.renderIf(!this.state.isAnimal,
                            <View style={styles.descriptionAnimal}>
                                <Text  style={{marginTop:  20,color:  'white',  textAlign:  'center',
fontSize: 30}}>{this.state.descriptionIdentify}</Text>
                            </View>
                        )}
                </View>
                <SinglePickerMaterialDialog
                    title={Strings.CameraPage.text1}
                    colorAccent={"#2EA1DA"}
                    scrolled={false}
                    items={this.state.languageArr.map((row,  index)  =>  ({  value:  index,  label:
row }))}
                    visible={this.state.languagePicker}
                    selectedItem={this.state.languageSelectedItem}
                    onCancel={() => this.setState({ languagePicker: false })}
                    onOk={ async (result) => {
                        this.setState({
                            languageSelectedItem: result.selectedItem,
                            language: await result.selectedItem.label,
                        });
                        await this.closePickerLanguage();
                    }}
                />
                <Spinner
                    visible={this.state.loadResult}
                    textContent={'Loading...'}
                    textStyle={styles.spinnerTextStyle}
                />
            </SafeAreaView>
        );
    }else{
        return(
            <Gallery />
        );
    }
  };
};
```

Gallery.js

```
import React from 'react';
import {
   Text,
   View,
   Image,
   SafeAreaView,
   TouchableOpacity,
   AsyncStorage,
   FlatList,
   BackHandler,
   Animated,
   Alert,
   ScrollView,
} from 'react-native';
import * as Permissions from 'expo-permissions';
import ReactNativeZoomableView from '@dudigital/react-native-zoomable-
view/src/ReactNativeZoomableView';
import * as ImagePicker from 'expo-image-picker';
import moment from 'moment';
import Spinner from 'react-native-loading-spinner-overlay';
import * as Speech from 'expo-speech';
import * as  MediaLibrary from 'expo-media-library';
import { SinglePickerMaterialDialog } from 'react-native-material-dialog';

import styles from './styles';
import CameraPage from './CameraPage';
import Utils from './Utils';
import Strings from '../constants/Strings';

export default class Gallery extends React.Component {
   static navigationOptions = {
      title: '',
      header: null
   };

   constructor(props){
      super(props);
      this.animatedValue = new Animated.Value(0);
      this.value = 0;
      this.animatedValue.addListener(({ value }) => {
        this.value = value;
      });
      this.handleBackButtonClick = this.handleBackButtonClick.bind(this);
      this.state = {
         data:[],
         showImage: false,
         loadData: false,
         currentScreen: "Gallery",
         indexCollection: 99,
         urlAnimal: '',
         nameAnimal: '',
         habitatAnimal: '',
         statusAnimal: '',
         classAnimal: '',
         dateAnimal: '',
         imgURL : '',
```

```
            textSpeechAnimal: "Not animals",
            descriptionIdentify: 'This is not animal',
            inputRange: [0, 180],
            outputRange: ['0deg', '180deg'],
            statusIdentifyAnimals: false,
            isSpeaking: false,
            language: "EN",
            languageArr: ['EN', 'IND'],
            languagePicker: false,
            languageSelectedItem: undefined,
            dataController: "",
        }
    }

    async componentDidMount() {
        const languageDevice = await AsyncStorage.getItem('language');
        console.log("LanguageDevice : ", languageDevice);
        this.setState({
            language: languageDevice == "IND" ? "IND" : "EN",
        })
        this.setState({ data : JSON.parse(await AsyncStorage.getItem("galleryStorage"))})
        console.log("DATA GALLERY: ", this.state.data.length)

        BackHandler.addEventListener('hardwareBackPress', this.handleBackButtonClick);
    }

    componentWillUnmount() {
        BackHandler.removeEventListener('hardwareBackPress',
this.handleBackButtonClick);
    }

    handleBackButtonClick() {
        Speech.stop();
        this.setState({isSpeaking: false,})
        if(this.state.showImage){
            this.setState({showImage: false})
        }else{
            this.setState({currentScreen: 'CameraPage'})
        }
        return true;
    }

    ListViewItemSeparator = () => {
        return (
            <View style={{ height: 1.5, width: "100%", backgroundColor: "#000", }} />
        );
    }

    GetItem = async (data, index) =>{
        this.setState({
            dataController: data.dataController,
            showImage: true,
            nameAnimal: data.name,
            habitatAnimal: data.habitat,
            statusAnimal: data.status,
            classAnimal: data.class,
            dateAnimal: data.date,
            urlAnimal: data.url,
```

```
            indexCollection: index,
            isSpeaking: true,
            textSpeechAnimal:  await  Utils.getTextSpeechAnimals(data.name.toLowerCase(),
data.language),
        })
        Speech.speak(this.state.textSpeechAnimal,{
            language: data.language == "EN" ? 'en' : 'ind',
            pitch: 1,
            rate: 1,
            onDone: () => this.onDoneSpeack(),
        })
    }

    flip_Animation = () => {
        if (this.value >= 90) {
            console.log("this.value >= 90 ")
            Animated.spring(this.animatedValue, {
                toValue: 0,
                tension: 10,
                friction: 8,
            }).start();
        } else {
            console.log("ELSE")
            Animated.spring(this.animatedValue, {
                toValue: 180,
                tension: 10,
                friction: 8,
            }).start();
        }
    };

    backToGallery = () => {
        this.setState({showImage: false})
    }

    deleteData = async () => {
        await this.state.data.splice(this.state.indexCollection, 1);
        this.setState({
            showImage: false,
        })
        await AsyncStorage.setItem("galleryStorage", JSON.stringify(this.state.data));
        Alert.alert(
            this.state.language        ==        "EN"?        Strings.CameraPage.tittleAlert3        :
Strings.CameraPage.tittleAlert3IND,
            this.state.language        ==        "EN"?        Strings.CameraPage.alert3        :
Strings.CameraPage.alert3IND,
            [
              {text: 'OK', onPress: () => console.log('OK Pressed')},
            ],
            { cancelable: false }
        )
    }

    deleteCollection = () => {
        Speech.stop();
        this.setState({isSpeaking: false,})
        Alert.alert(
```

```
        this.state.language == "EN"? Strings.CameraPage.tittleAlert4 :
Strings.CameraPage.tittleAlert4IND,
        this.state.language == "EN"? Strings.CameraPage.alert4 :
Strings.CameraPage.alert4IND,
        [
          {text: 'Yes', onPress: () => this.deleteData()},
          {text: 'No', onPress: () => console.log('No Pressed')},
        ],
        { cancelable: false }
      )
  }

  uploadImage = async () => {
    Speech.stop();
    this.setState({isSpeaking: false,})
    await Permissions.askAsync(Permissions.CAMERA_ROLL)
    .then(status => {
      console.log("Camera Roll Permission: ", status.status);
      if (status.status != "granted") {
        throw new Error("Storage permissions disabled. Please enable them in your
settings.");
      }
    })
    .then(_ => ImagePicker.launchImageLibraryAsync({ base64: true, allowsEditing:
false, aspect: [4,3] }))
    .then(result => {
      if (!result.cancelled) {
        this.setState({ imgURL: result.base64 });
        this.setState({
          statusIdentifyAnimals: true,
          loadData: true,
          nameAnimal: this.state.language == "EN"? Strings.CameraPage.desc1 :
Strings.CameraPage.desc1IND,
          statusAnimal: "-",
          descriptionIdentify: "Identify the Picture",
          showImage: true,
          dateAnimal: moment(Date.now()).format('LLLL'),
          urlAnimal: result.uri,
        })
        this.identifyCapture(this.state.imgURL);
        // console.log("Image Link : ", result.uri);
      }
      return;
    })
    .catch(error => {
      console.log("[CameraPage] UploadImage - Error: ", error);
    });
  }

  identifyCapture = (imageData) =>{
    const Clarifai = require('clarifai');

    const app = new Clarifai.App({
      apiKey: '22d8a802332443219479016d29810ab6'
    });

    app.models.predict(Clarifai.GENERAL_MODEL, {base64: imageData},{
      selectConcepts: [
```

```
            {name: 'cat'},
            {name: 'dog'},
            {name: 'horse'},
            {name: 'snake'},
            {name: 'butterfly'},
            {name: 'rabbit'},
            {name: 'bee'},
            {name: 'frog'},
            {name: 'monkey'},
            {name: 'piglet'},
            {name: 'panda'},
            {name: 'rat'},
            {name: 'cow'},
            {name: 'goat'},
            {name: 'worm'},
            {name: 'crocodile'},
            {name: 'deer'},
            {name: 'kangaroo'},
            {name: 'elephant'},
            {name: 'chicken'},
            {name: 'giraffe'},
            {name: 'squirrel'},
            {name: 'turtle'},
            {name: 'crab'},
            {name: 'spider'},
            {name: 'bird'},
            {name: 'fish'},
            {name: 'duck'},
            {name: 'grasshopper'},
            {name: 'gorilla'},
            {name: 'wolf'},
            {name: 'snail'},
            {name: 'sheep'},
            {name: 'zebra'},
            {name: 'whale'},
            {name: 'shark'},
            {name: 'dolphin'},
            {name: 'lizard'},
            {name: 'dragonfly'},
            {name: 'jellyfish'},
            {name: 'lobster'},
            {name: 'leopard'},
            {name: 'tiger'},
            {name: 'flamingo'},
            {name: 'peacock'},
            {name: 'parrot'},
            {name: 'shrimp'},
            {name: 'beetle'},
            {name: 'caterpillar'},
            {name: 'hippopotamus'},
            {name: 'camel'},
        ]
    })
    .then(async (response) => {
        await this.checkAnimals(response);
    })
    .catch((err) => alert(err));
}
```

```
  speakDescriptionAnimals = async () =>{
    speakDescriptionAnimals = async () =>{
      this.setState({ isSpeaking : !this.state.isSpeaking})
      if(this.state.isSpeaking == false){
        await Speech.speak(this.state.textSpeechAnimal,{
          language: this.state.language == "EN" ? 'en' : 'ind',
          pitch: 1,
          rate: 1,
          onDone: () => this.onDoneSpeack(),
        })
      }else{
        Speech.stop();
      }
    }
  }

  onDoneSpeack = () => {
    this.setState({ isSpeaking : false})
  }

  checkAnimals = async (response) =>{
    if(response.outputs[0].data.concepts[0].value < 0.5){
      this.setState({
        dataController: this.state.language == "EN"? Strings.CameraPage.desc1 :
Strings.CameraPage.desc1IND,
        loadData: false,
        isAnimal: false,
        isSpeaking: true,
        descriptionIdentify: this.state.language == "EN"? Strings.CameraPage.alert5:
Strings.CameraPage.alert5IND,
        nameAnimal: this.state.language == "EN"? Strings.CameraPage.desc1 :
Strings.CameraPage.desc1IND,
        textSpeechAnimal: this.state.language == "EN"? Strings.CameraPage.alert5:
Strings.CameraPage.alert5IND,
      })
      Speech.speak(this.state.textSpeechAnimal,{
        language: this.state.language == "EN" ? 'en' : 'ind',
        pitch: 1,
        rate: 1,
        onDone: () => this.onDoneSpeack(),
      })
      return Alert.alert(
        this.state.language == "EN"? Strings.CameraPage.tittleAlert5 :
Strings.CameraPage.tittleAlert5IND,
        this.state.language == "EN"? Strings.CameraPage.alert5:
Strings.CameraPage.alert5IND,
        [
          {text: 'OK', onPress: () => console.log('OK Pressed')},
          {text: 'Try Again', onPress: () => this.handleBackButtonClick()},
        ],
        { cancelable: false }
      )
    }else{
      this.setState({
        dataController: response.outputs[0].data.concepts[0].name,
        isAnimal: true,
        descriptionIdentify: "Identify the Picture",
```

```
            nameAnimal:                                                                    await
Utils.getNameAnimals(response.outputs[0].data.concepts[0].name, this.state.language),
            classAnimal:
Utils.getClassAnimals(response.outputs[0].data.concepts[0].name, this.state.language),
            habitatAnimal:
Utils.getHabitatsAnimals(response.outputs[0].data.concepts[0].name,
this.state.language),
            statusAnimal:
Utils.getStatusAnimals(response.outputs[0].data.concepts[0].name, this.state.language),
          })
          await                Utils.dialogAnimalAlert(this.state.language                ==="EN"?
Strings.CameraPage.tittleAlert1:  Strings.CameraPage.tittleAlert1IND,  this.state.language
==="EN"?                              Strings.CameraPage.alert1                              +
Utils.getNameAnimals(response.outputs[0].data.concepts[0].name, this.state.language)
            :                         Strings.CameraPage.alert1IND                           +
Utils.getNameAnimals(response.outputs[0].data.concepts[0].name, this.state.language));

          this.setState({
            isSpeaking: true,
            textSpeechAnimal:
Utils.getTextSpeechAnimals(this.state.nameAnimal.toLowerCase(), this.state.language),
          })
          Speech.speak(this.state.textSpeechAnimal,{
            language: this.state.language == "EN" ? 'en' : 'ind',
            pitch: 1,
            rate: 1,
            onDone: () => this.onDoneSpeack(),
          })
        }
        this.setState({
          loadData: false,
        })
    }

    saveCollection = async (url) => {
        Speech.stop();
        this.setState({isSpeaking: false,})
        const asset = MediaLibrary.createAssetAsync(url);
        await this.state.dataGallery.push({"url": url, "name": this.state.nameAnimal, "class":
this.state.classAnimal,
                              "habitat" : this.state.habitatAnimal, "status": this.state.statusAnimal,
"date": this.state.calendar,
                              "language"      :      this.state.language,      "dataController"      :
this.state.dataController});
        await                                              AsyncStorage.setItem("galleryStorage",
JSON.stringify(this.state.dataGallery));
        Alert.alert(
          this.state.language       ==       "EN"?       Strings.CameraPage.tittleAlert2       :
Strings.CameraPage.tittleAlert2IND,
          this.state.language       ==       "EN"?       Strings.CameraPage.alert2       :
Strings.CameraPage.alert2IND,
          [
            {text: 'OK', onPress: () => console.log('OK Pressed')},
          ],
          { cancelable: false }
        )
    }
```

```
  speakDescriptionAnimals = async () =>{
    this.setState({ isSpeaking : true})
    await Speech.speak(this.state.textSpeechAnimal,{
      language: this.state.language == "EN" ? 'en' : 'ind',
      pitch: 1,
      rate: 1,
      onDone: () => this.onDoneSpeack(),
    })
  }

  onDoneSpeack = () => {
    this.setState({ isSpeaking : false})
  }

  changeLanguage = () => {
    Speech.stop();
    this.setState({
      isSpeaking: false,
      languagePicker: true,
    })
  }

  closePickerLanguage = async () =>{
    this.setState({
      languagePicker: false,
      nameAnimal: await Utils.getNameAnimals(this.state.dataController,
this.state.language),
      classAnimal: Utils.getClassAnimals(this.state.dataController, this.state.language),
      habitatAnimal: Utils.getHabitatsAnimals(this.state.dataController,
this.state.language),
      statusAnimal: Utils.getStatusAnimals(this.state.dataController,
this.state.language),
    })
    console.log("Language Set by ", this.state.language);
    await AsyncStorage.setItem("language", this.state.language);
    console.log("Success");
    this.setState({
      textSpeechAnimal:
Utils.getTextSpeechAnimals(this.state.nameAnimal.toLowerCase(), this.state.language),
    })
  }

  render() {
    this.SetInterpolate = this.animatedValue.interpolate({
      inputRange: this.state.inputRange,
      outputRange: this.state.outputRange,
    });

    const Rotate_Y_AnimatedStyle = {
      transform: [{ rotateY: this.SetInterpolate }],
    };

    const goToGallery = () => {
      this.setState({currentScreen: 'Gallery'})
    }

    if(this.state.currentScreen == "Gallery"  && this.state.showImage == false){
      return (
```

```
            <SafeAreaView style={{flex:1}}>
               <Image
                  style={styles.galleryBackground}
                  source={require('../assets/images/Background.jpg')}/>
               <View style={styles.galleryContainerBackground}>

               </View>
               <View style={styles.galleryContainer}>
                  <View style={styles.galleryContentContainer}>
                     <View style={styles.galleryTitleContainer}>
                        <Text style={[styles.galleryTextTitle, {fontSize: 30}]}>MY
ANIMALS</Text>
                        <Text style={[styles.galleryTextTitle, {fontSize:
18}]}>COLLECTION</Text>
                     </View>
                     <View style={styles.galleryCollectionContainer}>
                        {
                           Utils.renderIf(this.state.data.length == 0,
                              <View style={{flexDirection: 'row'}}>
                                 <View style={{flex:4, margin: 5, width: 70, height: 70,
alignItems: 'center', justifyContent: 'center'}}>
                                    <Text style={{textAlign: 'center', fontSize: 20, }}>{"There is
no data collection!"}</Text>
                                 </View>
                              </View>
                        )}
                        {
                           Utils.renderIf(this.state.data.length,
                              <FlatList
                                 style={{flex:1, marginBottom: 70}}
                                 data={this.state.data}
                                 ItemSeparatorComponent={this.ListViewItemSeparator}
                                 enableEmptySections={true}
                                 keyExtractor={(item, index) => index.toString()}
                                 renderItem={({ item, index }) => (
                                    <TouchableOpacity
                                       onPress={ this.GetItem.bind(this, item, index) }
                                       >
                                       <View style={{flexDirection: 'row'}}>
                                          <View style={{flex:1, margin: 5}}>
                                             <Image style={{width: 70,height: 80}} source={{uri:
item.url}}/>
                                          </View>
                                          <View style={{flex:3, margin: 5}}>
                                             <Text style={{textAlign: 'center'}}>{"Collection #" +
(index + 1)}</Text>
                                             <Text style={{textAlign: 'left'}}>{"Nama : " +
item.name}</Text>
                                             <Text style={{textAlign: 'left'}}>{"Date : " +
item.date.substring(item.date.indexOf(",")+1)}</Text>
                                             <Text style={{textAlign: 'left'}}>{"Status : " +
item.status}</Text>
                                          </View>
                                       </View>
                                    </TouchableOpacity>
                                 )}
                              />
                           )
```

```
                    }
                  </View>
                </View>
              </View>
            </SafeAreaView>
          );
        }else if(this.state.currentScreen == "Gallery" && this.state.showImage){
          return (
            <SafeAreaView style={styles.safeViewPreviewImage}>
              <Image
                style={styles.previewBackground}
                source={require('../assets/images/Background.jpg')}/>
              <View style={styles.backgroundTopToolbar}>

              </View>
              <View style={styles.topToolbar}>
                <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 30, height:
30}} onPress={() => this.uploadImage()}>
                  <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/localImage.png')}/>
                </TouchableOpacity>
                {
                  Utils.renderIf(!this.state.isSpeaking,
                    <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50}} onPress={() => this.speakDescriptionAnimals()} >
                      <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/speaker.png')}/>
                    </TouchableOpacity>

                )}
                {
                  Utils.renderIf(this.state.isSpeaking,
                    <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50}} onPress={() => this.speakDescriptionAnimals()} >
                      <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/SpeakerON.gif')}/>
                    </TouchableOpacity>
                )}
                {
                  Utils.renderIf(this.state.statusIdentifyAnimals,
                    <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50,}} onPress={() => this.saveCollection(this.state.urlAnimal)}>
                      <Image style={{width: 40,height: 40, marginTop: 5}} source={
require('../assets/images/SaveCollection.png')}/>
                    </TouchableOpacity>
                )}
                {
                  Utils.renderIf(!this.state.statusIdentifyAnimals,
                    <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50,}} onPress={() => this.deleteCollection()}>
                      <Image style={{width: 40,height: 40, marginTop: 5}} source={
require('../assets/images/Delete.png')}/>
                    </TouchableOpacity>
                )}
                {
                  Utils.renderIf(this.state.language == "EN",
                    <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50}} onPress={() => this.changeLanguage()}>
```

```
                        <Image style={{width: 30,height: 30, marginTop: 10}} source={
require('../assets/images/EN.png')}/>
                    </TouchableOpacity>
                )}
                {
                    Utils.renderIf(this.state.language == "IND",
                        <TouchableOpacity style={{flex: 1, alignItems: 'center', width:
50,height: 50}} onPress={() => this.changeLanguage()}>
                            <Image style={{width: 30,height: 30, marginTop: 10}} source={
require('../assets/images/IND.png')}/>
                        </TouchableOpacity>
                )}
                <TouchableOpacity style={{flex: 1, alignItems: 'center', width: 30, height:
30}} onPress={() => this.backToGallery()}>
                    <Image style={{width: 30,height: 30, marginTop: 10}}
source={require('../assets/images/gallery.png')}/>
                </TouchableOpacity>
            </View>

            <ReactNativeZoomableView
                zoomEnabled={true}
                maxZoom={2}
                minZoom={0.5}
                zoomStep={0.25}
                initialZoom={0.9}
                bindToBorders={true}
                style={styles.zoomableView}
                >
                    <Animated.Image
                        style={[Rotate_Y_AnimatedStyle, styles.previewImageCamera]}
                        source={{uri: this.state.urlAnimal}}
                    />
                    <Text style={styles.caption}>{this.state.dateAnimal}</Text>

            </ReactNativeZoomableView>
            <View style={styles.backgroundBottomToolbar}>

            </View>
            <View style={styles.bottomToolbar}>
                {
                    Utils.renderIf(this.state.nameAnimal != "Not Animals" ||
this.state.nameAnimal != "Bukan Binatang",
                        <View style={styles.descriptionAnimal}>
                            <Text style={{color: 'white', textAlign: 'center'}}>{"Name : " +
this.state.nameAnimal}</Text>
                            <Text style={{color: 'white', textAlign: 'center'}}>{"Class : " +
this.state.classAnimal}</Text>
                            <Text style={{color: 'white', textAlign: 'center'}}
numberOfLines={3}>{"Habitat : " + this.state.habitatAnimal}</Text>
                            <Text style={{color: 'white', textAlign: 'center'}}>{"Status : " +
this.state.statusAnimal}</Text>
                        </View>
                )}
                {
                    Utils.renderIf(this.state.nameAnimal == "Not Animals" ||
this.state.nameAnimal == "Bukan Binatang",
                        <View style={styles.descriptionAnimal}>
```

```
                        <Text style={{marginTop: 20,color: 'white', textAlign: 'center', fontSize: 30}}>{this.state.descriptionIdentify}</Text>
                    </View>
                )}
            </View>
            <SinglePickerMaterialDialog
                title={Strings.CameraPage.text1}
                colorAccent={"#2EA1DA"}
                scrolled={false}
                items={this.state.languageArr.map((row, index) => ({ value: index, label: row }))}
                visible={this.state.languagePicker}
                selectedItem={this.state.languageSelectedItem}
                onCancel={() => this.setState({ languagePicker: false })}
                onOk={ async (result) => {
                    this.setState({
                        languageSelectedItem: result.selectedItem,
                        language: await result.selectedItem.label,
                    });
                    await this.closePickerLanguage();
                }}
            />
            <Spinner
                visible={this.state.loadData}
                textContent={'Loading...'}
                textStyle={styles.spinnerTextStyle}
            />
        </SafeAreaView>
      );
    }else{
      return(
        <CameraPage/>
      );
    }
  }
}
```

ToolbarComponent.js

```
import React from 'react';
import { Camera } from 'expo-camera';
import { Ionicons } from '@expo/vector-icons';
import { Col, Row, Grid } from "react-native-easy-grid";
import { View, TouchableWithoutFeedback, TouchableOpacity} from 'react-native';
import RNFlash from 'react-native-flash';

import styles from './styles';

const { FlashMode: CameraFlashModes, Type: CameraTypes} = Camera.Constants;

export default ({
  capturing = false,
  cameraType = CameraTypes.back,
  flashMode = CameraFlashModes.off,
  setFlashMode, setCameraType,
  onCaptureIn, onCaptureOut, onLongCapture, onShortCapture,
}) => (
```

```
   <Grid style={styles.bottomToolbar}>
      <Row>
         <Col style={styles.alignCenter}>
            <TouchableOpacity onPress={() => setFlashMode(
               flashMode === CameraFlashModes.on ? CameraFlashModes.off :
CameraFlashModes.on
            )}>
               <Ionicons
                  name={flashMode == CameraFlashModes.on ? "md-flash" : 'md-flash-off'}
                  color="white"
                  size={30}
               />
            </TouchableOpacity>
         </Col>
         <Col size={2} style={styles.alignCenter}>
            <TouchableWithoutFeedback
               onPressIn={onCaptureIn}
               onPressOut={onCaptureOut}
               onLongPress={onLongCapture}
               onPress={onShortCapture}>
               <View style={[styles.captureBtn, capturing && styles.captureBtnActive]}>
                  {capturing && <View style={styles.captureBtnInternal} />}
               </View>
            </TouchableWithoutFeedback>
         </Col>
         <Col style={styles.alignCenter}>
            <TouchableOpacity onPress={() => setCameraType(
               cameraType === CameraTypes.back ? CameraTypes.front :
CameraTypes.back
            )}>
               <Ionicons
                  name="md-reverse-camera"
                  color="white"
                  size={30}
               />
            </TouchableOpacity>
         </Col>
      </Row>
   </Grid>
);
```

Utils.js

```
import React from 'react';
import Moment, { lang } from 'moment';
import {
   Alert,
   AsyncStorage,
   BackHandler,
} from 'react-native';

import Strings from '../constants/Strings';

export default class Utils extends React.Component
{
```

```
    componentDidMount() {
        BackHandler.addEventListener('hardwareBackPress', this.handleBackButtonClick);
    }

    componentWillUnmount() {
        BackHandler.removeEventListener('hardwareBackPress',
this.handleBackButtonClick);
    }

    handleBackButtonClick() {
        return true;
    }

    static errAlert = (_title, _msg) => {
        return Alert.alert(
            _title,
            _msg,
            [
              {text: 'OK', onPress: () => console.log('OK Pressed') },
            ],
            { cancelable: false }
        )
    }

    static alertBackToMain = (_msg, _props) => {
        return Alert.alert(
            "Application Alert",
            _msg,
            [
              {text: 'OK', onPress: () => _props.navigation.popToTop() },
            ],
            { cancelable: false }
        )
    }

    static dialogAnimalAlert = (_title, _msg) => {
        return Alert.alert(
            _title,
            _msg,
            [
              {text: 'OK', onPress: () => console.log('OK Pressed')},
            ],
            { cancelable: false }
        )
    }


    static getNameAnimals = (animals, language) =>
    {
        if(language == "EN"){
            if(animals == 'cat'){
                return Strings.Cat.Name;
            }else if(animals == 'dog'){
                return Strings.Dog.Name;
            }else if(animals == 'horse'){
                return Strings.Horse.Name;
            }else if(animals == 'snake'){
                return Strings.Snake.Name;
```

```
}else if(animals == 'butterfly'){
   return Strings.Butterfly.Name;
}else if(animals == 'rabbit'){
   return Strings.Rabbit.Name;
}else if(animals == 'bee'){
   return Strings.Bee.Name;
}else if(animals == 'frog'){
   return Strings.Frog.Name;
}else if(animals == 'monkey'){
   return Strings.Monkey.Name;
}else if(animals == 'piglet'){
   return Strings.Piglet.Name;
}else if(animals == 'panda'){
   return Strings.Panda.Name;
}else if(animals == 'rat'){
   return Strings.Rat.Name;
}else if(animals == 'cow'){
   return Strings.Cow.Name;
}else if(animals == 'goat'){
   return Strings.Goat.Name;
}else if(animals == 'worm'){
   return Strings.Worm.Name;
}else if(animals == 'crocodile'){
   return Strings.Crocodile.Name;
}else if(animals == 'deer'){
   return Strings.Deer.Name;
}else if(animals == 'kangaroo'){
   return Strings.Kangaroo.Name;
}else if(animals == 'elephant'){
   return Strings.Elephant.Name;
}else if(animals == 'chicken'){
   return Strings.Chicken.Name;
}else if(animals == 'giraffe'){
   return Strings.Giraffe.Name;
}else if(animals == 'bird'){
   return Strings.Bird.Name;
}else if(animals == 'squirrel'){
   return Strings.Squirrel.Name;
}else if(animals == 'turtle'){
   return Strings.Turtle.Name;
}else if(animals == 'crab'){
   return Strings.Crab.Name;
}else if(animals == 'spider'){
   return Strings.Spider.Name;
}else if(animals == 'bird'){
   return Strings.Bird.Name;
}else if(animals == 'fish'){
   return Strings.Fish.Name;
}else if(animals == 'duck'){
   return Strings.Duck.Name;
}else if(animals == 'grasshopper'){
   return Strings.Grasshopper.Name;
}else if(animals == 'gorilla'){
   return Strings.Gorilla.Name;
}else if(animals == 'wolf'){
   return Strings.Wolf.Name;
}else if(animals == 'snail'){
   return Strings.Snail.Name;
```

```
    }else if(animals == 'sheep'){
        return Strings.Sheep.Name;
    }else if(animals == 'zebra'){
        return Strings.Zebra.Name;
    }else if(animals == 'whale'){
        return Strings.Whale.Name;
    }else if(animals == 'shark'){
        return Strings.Shark.Name;
    }else if(animals == 'dolphin'){
        return Strings.Dolphin.Name;
    }else if(animals == 'lizard'){
        return Strings.Lizard.Name;
    }else if(animals == 'dragonfly'){
        return Strings.Dragonfly.Name;
    }else if(animals == 'jellyfish'){
        return Strings.Jellyfish.Name;
    }else if(animals == 'lobster'){
        return Strings.Lobster.Name;
    }else if(animals == 'leopard'){
        return Strings.Leopard.Name;
    }else if(animals == 'tiger'){
        return Strings.Tiger.Name;
    }else if(animals == 'flamingo'){
        return Strings.Flamingo.Name;
    }else if(animals == 'peacock'){
        return Strings.Peacock.Name;
    }else if(animals == 'parrot'){
        return Strings.Parrot.Name;
    }else if(animals == 'shrimp'){
        return Strings.Shrimp.Name;
    }else if(animals == 'beetle'){
        return Strings.Beetle.Name;
    }else if(animals == 'caterpillar'){
        return Strings.Caterpillar.Name;
    }else if(animals == 'hippopotamus'){
        return Strings.Hippopotamus.Name;
    }else if(animals == 'camel'){
        return Strings.Camel.Name;
    }else{
        return "Undefined";
    }
}else if(language == "IND"){
    if(animals == 'cat'){
        return Strings.CatIND.Name;
    }else if(animals == 'dog'){
        return Strings.DogIND.Name;
    }else if(animals == 'horse'){
        return Strings.HorseIND.Name;
    }else if(animals == 'snake'){
        return Strings.SnakeIND.Name;
    }else if(animals == 'butterfly'){
        return Strings.ButterflyIND.Name;
    }else if(animals == 'rabbit'){
        return Strings.RabbitIND.Name;
    }else if(animals == 'bee'){
        return Strings.BeeIND.Name;
    }else if(animals == 'frog'){
        return Strings.FrogIND.Name;
```

```
}else if(animals == 'monkey'){
   return Strings.MonkeyIND.Name;
}else if(animals == 'piglet'){
   return Strings.PigletIND.Name;
}else if(animals == 'panda'){
   return Strings.PandaIND.Name;
}else if(animals == 'rat'){
   return Strings.RatIND.Name;
}else if(animals == 'cow'){
   return Strings.CowIND.Name;
}else if(animals == 'goat'){
   return Strings.GoatIND.Name;
}else if(animals == 'worm'){
   return Strings.WormIND.Name;
}else if(animals == 'crocodile'){
   return Strings.CrocodileIND.Name;
}else if(animals == 'deer'){
   return Strings.DeerIND.Name;
}else if(animals == 'kangaroo'){
   return Strings.KangarooIND.Name;
}else if(animals == 'elephant'){
   return Strings.ElephantIND.Name;
}else if(animals == 'chicken'){
   return Strings.ChickenIND.Name;
}else if(animals == 'giraffe'){
   return Strings.GiraffeIND.Name;
}else if(animals == 'bird'){
   return Strings.BirdIND.Name;
}else if(animals == 'squirrel'){
   return Strings.SquirrelIND.Name;
}else if(animals == 'turtle'){
   return Strings.TurtleIND.Name;
}else if(animals == 'crab'){
   return Strings.CrabIND.Name;
}else if(animals == 'spider'){
   return Strings.SpiderIND.Name;
}else if(animals == 'bird'){
   return Strings.BirdIND.Name;
}else if(animals == 'fish'){
   return Strings.FishIND.Name;
}else if(animals == 'duck'){
   return Strings.DuckIND.Name;
}else if(animals == 'grasshopper'){
   return Strings.GrasshopperIND.Name;
}else if(animals == 'gorilla'){
   return Strings.GorillaIND.Name;
}else if(animals == 'wolf'){
   return Strings.WolfIND.Name;
}else if(animals == 'snail'){
   return Strings.SnailIND.Name;
}else if(animals == 'sheep'){
   return Strings.SheepIND.Name;
}else if(animals == 'zebra'){
   return Strings.ZebraIND.Name;
}else if(animals == 'whale'){
   return Strings.WhaleIND.Name;
}else if(animals == 'shark'){
   return Strings.SharkIND.Name;
```

```
        }else if(animals == 'dolphin'){
            return Strings.DolphinIND.Name;
        }else if(animals == 'lizard'){
            return Strings.LizardIND.Name;
        }else if(animals == 'dragonfly'){
            return Strings.DragonflyIND.Name;
        }else if(animals == 'jellyfish'){
            return Strings.JellyfishIND.Name;
        }else if(animals == 'lobster'){
            return Strings.LobsterIND.Name;
        }else if(animals == 'leopard'){
            return Strings.LeopardIND.Name;
        }else if(animals == 'tiger'){
            return Strings.TigerIND.Name;
        }else if(animals == 'flamingo'){
            return Strings.FlamingoIND.Name;
        }else if(animals == 'peacock'){
            return Strings.PeacockIND.Name;
        }else if(animals == 'parrot'){
            return Strings.ParrotIND.Name;
        }else if(animals == 'shrimp'){
            return Strings.ShrimpIND.Name;
        }else if(animals == 'beetle'){
            return Strings.BeetleIND.Name;
        }else if(animals == 'caterpillar'){
            return Strings.CaterpillarIND.Name;
        }else if(animals == 'hippopotamus'){
            return Strings.HippopotamusIND.Name;
        }else if(animals == 'camel'){
            return Strings.CamelIND.Name;
        }
        else{
            return "Tidak ditemukan";
        }
    }
}

static getClassAnimals = (animals, language) =>
{
    if(language == "EN"){
        if(animals == 'cat'){
            return Strings.Cat.Class;
        }else if(animals == 'dog'){
            return Strings.Dog.Class;
        }else if(animals == 'horse'){
            return Strings.Horse.Class;
        }else if(animals == 'snake'){
            return Strings.Snake.Class;
        }else if(animals == 'butterfly'){
            return Strings.Butterfly.Class;
        }else if(animals == 'rabbit'){
            return Strings.Rabbit.Class;
        }else if(animals == 'bee'){
            return Strings.Bee.Class;
        }else if(animals == 'frog'){
            return Strings.Frog.Class;
        }else if(animals == 'monkey'){
            return Strings.Monkey.Class;
```

```
}else if(animals == 'piglet'){
    return Strings.Piglet.Class;
}else if(animals == 'panda'){
    return Strings.Panda.Class;
}else if(animals == 'rat'){
    return Strings.Rat.Class;
}else if(animals == 'cow'){
    return Strings.Cow.Class;
}else if(animals == 'goat'){
    return Strings.Goat.Class;
}else if(animals == 'worm'){
    return Strings.Worm.Class;
}else if(animals == 'crocodile'){
    return Strings.Crocodile.Class;
}else if(animals == 'deer'){
    return Strings.Deer.Class;
}else if(animals == 'kangaroo'){
    return Strings.Kangaroo.Class;
}else if(animals == 'elephant'){
    return Strings.Elephant.Class;
}else if(animals == 'chicken'){
    return Strings.Chicken.Class;
}else if(animals == 'giraffe'){
    return Strings.Giraffe.Class;
}else if(animals == 'bird'){
    return Strings.Bird.Class;
}else if(animals == 'squirrel'){
    return Strings.Squirrel.Class;
}else if(animals == 'turtle'){
    return Strings.Turtle.Class;
}else if(animals == 'crab'){
    return Strings.Crab.Class;
}else if(animals == 'spider'){
    return Strings.Spider.Class;
}else if(animals == 'bird'){
    return Strings.Bird.Class;
}else if(animals == 'fish'){
    return Strings.Fish.Class;
}else if(animals == 'duck'){
    return Strings.Duck.Class;
}else if(animals == 'grasshopper'){
    return Strings.Grasshopper.Class;
}else if(animals == 'gorilla'){
    return Strings.Gorilla.Class;
}else if(animals == 'wolf'){
    return Strings.Wolf.Class;
}else if(animals == 'snail'){
    return Strings.Snail.Class;
}else if(animals == 'sheep'){
    return Strings.Sheep.Class;
}else if(animals == 'zebra'){
    return Strings.Zebra.Class;
}else if(animals == 'whale'){
    return Strings.Whale.Class;
}else if(animals == 'shark'){
    return Strings.Shark.Class;
}else if(animals == 'dolphin'){
    return Strings.Dolphin.Class;
```

```
    }else if(animals == 'lizard'){
        return Strings.Lizard.Class;
    }else if(animals == 'dragonfly'){
        return Strings.Dragonfly.Class;
    }else if(animals == 'jellyfish'){
        return Strings.Jellyfish.Class;
    }else if(animals == 'lobster'){
        return Strings.Lobster.Class;
    }else if(animals == 'leopard'){
        return Strings.Leopard.Class;
    }else if(animals == 'tiger'){
        return Strings.Tiger.Class;
    }else if(animals == 'flamingo'){
        return Strings.Flamingo.Class;
    }else if(animals == 'peacock'){
        return Strings.Peacock.Class;
    }else if(animals == 'parrot'){
        return Strings.Parrot.Class;
    }else if(animals == 'shrimp'){
        return Strings.Shrimp.Class;
    }else if(animals == 'beetle'){
        return Strings.Beetle.Class;
    }else if(animals == 'caterpillar'){
        return Strings.Caterpillar.Class;
    }else if(animals == 'hippopotamus'){
        return Strings.Hippopotamus.Class;
    }else if(animals == 'camel'){
        return Strings.Camel.Class;
    }
    else{
        return "Undefined";
    }
}else if(language == "IND"){
    if(animals == 'cat'){
        return Strings.CatIND.Class;
    }else if(animals == 'dog'){
        return Strings.DogIND.Class;
    }else if(animals == 'horse'){
        return Strings.HorseIND.Class;
    }else if(animals == 'snake'){
        return Strings.SnakeIND.Class;
    }else if(animals == 'butterfly'){
        return Strings.ButterflyIND.Class;
    }else if(animals == 'rabbit'){
        return Strings.RabbitIND.Class;
    }else if(animals == 'bee'){
        return Strings.BeeIND.Class;
    }else if(animals == 'frog'){
        return Strings.FrogIND.Class;
    }else if(animals == 'monkey'){
        return Strings.MonkeyIND.Class;
    }else if(animals == 'piglet'){
        return Strings.PigletIND.Class;
    }else if(animals == 'panda'){
        return Strings.PandaIND.Class;
    }else if(animals == 'rat'){
        return Strings.RatIND.Class;
    }else if(animals == 'cow'){
```

```
    return Strings.CowIND.Class;
}else if(animals == 'goat'){
    return Strings.GoatIND.Class;
}else if(animals == 'worm'){
    return Strings.WormIND.Class;
}else if(animals == 'crocodile'){
    return Strings.CrocodileIND.Class;
}else if(animals == 'deer'){
    return Strings.DeerIND.Class;
}else if(animals == 'kangaroo'){
    return Strings.KangarooIND.Class;
}else if(animals == 'elephant'){
    return Strings.ElephantIND.Class;
}else if(animals == 'chicken'){
    return Strings.ChickenIND.Class;
}else if(animals == 'giraffe'){
    return Strings.GiraffeIND.Class;
}else if(animals == 'bird'){
    return Strings.BirdIND.Class;
}else if(animals == 'squirrel'){
    return Strings.SquirrelIND.Class;
}else if(animals == 'turtle'){
    return Strings.TurtleIND.Class;
}else if(animals == 'crab'){
    return Strings.CrabIND.Class;
}else if(animals == 'spider'){
    return Strings.SpiderIND.Class;
}else if(animals == 'bird'){
    return Strings.BirdIND.Class;
}else if(animals == 'fish'){
    return Strings.FishIND.Class;
}else if(animals == 'duck'){
    return Strings.DuckIND.Class;
}else if(animals == 'grasshopper'){
    return Strings.GrasshopperIND.Class;
}else if(animals == 'gorilla'){
    return Strings.GorillaIND.Class;
}else if(animals == 'wolf'){
    return Strings.WolfIND.Class;
}else if(animals == 'snail'){
    return Strings.SnailIND.Class;
}else if(animals == 'sheep'){
    return Strings.SheepIND.Class;
}else if(animals == 'zebra'){
    return Strings.ZebraIND.Class;
}else if(animals == 'whale'){
    return Strings.WhaleIND.Class;
}else if(animals == 'shark'){
    return Strings.SharkIND.Class;
}else if(animals == 'dolphin'){
    return Strings.DolphinIND.Class;
}else if(animals == 'lizard'){
    return Strings.LizardIND.Class;
}else if(animals == 'dragonfly'){
    return Strings.DragonflyIND.Class;
}else if(animals == 'jellyfish'){
    return Strings.JellyfishIND.Class;
}else if(animals == 'lobster'){
```

```
            return Strings.LobsterIND.Class;
        }else if(animals == 'leopard'){
            return Strings.LeopardIND.Class;
        }else if(animals == 'tiger'){
            return Strings.TigerIND.Class;
        }else if(animals == 'flamingo'){
            return Strings.FlamingoIND.Class;
        }else if(animals == 'peacock'){
            return Strings.PeacockIND.Class;
        }else if(animals == 'parrot'){
            return Strings.ParrotIND.Class;
        }else if(animals == 'shrimp'){
            return Strings.ShrimpIND.Class;
        }else if(animals == 'beetle'){
            return Strings.BeetleIND.Class;
        }else if(animals == 'caterpillar'){
            return Strings.CaterpillarIND.Class;
        }else if(animals == 'hippopotamus'){
            return Strings.HippopotamusIND.Class;
        }else if(animals == 'camel'){
            return Strings.CamelIND.Class;
        }
        else{
            return "Tidak ditemukan";
        }
    }

}

static getHabitatsAnimals = (animals, language) =>
{
    if(language == "EN"){
        if(animals == 'cat'){
            return Strings.Cat.Habitat;
        }else if(animals == 'dog'){
            return Strings.Dog.Habitat;
        }else if(animals == 'horse'){
            return Strings.Horse.Habitat;
        }else if(animals == 'snake'){
            return Strings.Snake.Habitat;
        }else if(animals == 'butterfly'){
            return Strings.Butterfly.Habitat;
        }else if(animals == 'rabbit'){
            return Strings.Rabbit.Habitat;
        }else if(animals == 'bee'){
            return Strings.Bee.Habitat;
        }else if(animals == 'frog'){
            return Strings.Frog.Habitat;
        }else if(animals == 'monkey'){
            return Strings.Monkey.Habitat;
        }else if(animals == 'piglet'){
            return Strings.Piglet.Habitat;
        }else if(animals == 'panda'){
            return Strings.Panda.Habitat;
        }else if(animals == 'rat'){
            return Strings.Rat.Habitat;
        }else if(animals == 'cow'){
            return Strings.Cow.Habitat;
```

```
}else if(animals == 'goat'){
   return Strings.Goat.Habitat;
}else if(animals == 'worm'){
   return Strings.Worm.Habitat;
}else if(animals == 'crocodile'){
   return Strings.Crocodile.Habitat;
}else if(animals == 'deer'){
   return Strings.Deer.Habitat;
}else if(animals == 'kangaroo'){
   return Strings.Kangaroo.Habitat;
}else if(animals == 'elephant'){
   return Strings.Elephant.Habitat;
}else if(animals == 'chicken'){
   return Strings.Chicken.Habitat;
}else if(animals == 'giraffe'){
   return Strings.Giraffe.Habitat;
}else if(animals == 'bird'){
   return Strings.Bird.Habitat;
}else if(animals == 'squirrel'){
   return Strings.Squirrel.Habitat;
}else if(animals == 'turtle'){
   return Strings.Turtle.Habitat;
}else if(animals == 'crab'){
   return Strings.Crab.Habitat;
}else if(animals == 'spider'){
   return Strings.Spider.Habitat;
}else if(animals == 'bird'){
   return Strings.Bird.Habitat;
}else if(animals == 'fish'){
   return Strings.Fish.Habitat;
}else if(animals == 'duck'){
   return Strings.Duck.Habitat;
}else if(animals == 'grasshopper'){
   return Strings.Grasshopper.Habitat;
}else if(animals == 'gorilla'){
   return Strings.Gorilla.Habitat;
}else if(animals == 'wolf'){
   return Strings.Wolf.Habitat;
}else if(animals == 'snail'){
   return Strings.Snail.Habitat;
}else if(animals == 'sheep'){
   return Strings.Sheep.Habitat;
}else if(animals == 'zebra'){
   return Strings.Zebra.Habitat;
}else if(animals == 'whale'){
   return Strings.Whale.Habitat;
}else if(animals == 'shark'){
   return Strings.Shark.Habitat;
}else if(animals == 'dolphin'){
   return Strings.Dolphin.Habitat;
}else if(animals == 'lizard'){
   return Strings.Lizard.Habitat;
}else if(animals == 'dragonfly'){
   return Strings.Dragonfly.Habitat;
}else if(animals == 'jellyfish'){
   return Strings.Jellyfish.Habitat;
}else if(animals == 'lobster'){
   return Strings.Lobster.Habitat;
```

```
    }else if(animals == 'leopard'){
        return Strings.Leopard.Habitat;
    }else if(animals == 'tiger'){
        return Strings.Tiger.Habitat;
    }else if(animals == 'flamingo'){
        return Strings.Flamingo.Habitat;
    }else if(animals == 'peacock'){
        return Strings.Peacock.Habitat;
    }else if(animals == 'parrot'){
        return Strings.Parrot.Habitat;
    }else if(animals == 'shrimp'){
        return Strings.Shrimp.Habitat;
    }else if(animals == 'beetle'){
        return Strings.Beetle.Habitat;
    }else if(animals == 'caterpillar'){
        return Strings.Caterpillar.Habitat;
    }else if(animals == 'hippopotamus'){
        return Strings.Hippopotamus.Habitat;
    }else if(animals == 'camel'){
        return Strings.Camel.Habitat;
    }
    else{
        return "Undefined";
    }
}else if(language == "IND"){
    if(animals == 'cat'){
        return Strings.CatIND.Habitat;
    }else if(animals == 'dog'){
        return Strings.DogIND.Habitat;
    }else if(animals == 'horse'){
        return Strings.HorseIND.Habitat;
    }else if(animals == 'snake'){
        return Strings.SnakeIND.Habitat;
    }else if(animals == 'butterfly'){
        return Strings.ButterflyIND.Habitat;
    }else if(animals == 'rabbit'){
        return Strings.RabbitIND.Habitat;
    }else if(animals == 'bee'){
        return Strings.BeeIND.Habitat;
    }else if(animals == 'frog'){
        return Strings.FrogIND.Habitat;
    }else if(animals == 'monkey'){
        return Strings.MonkeyIND.Habitat;
    }else if(animals == 'piglet'){
        return Strings.PigletIND.Habitat;
    }else if(animals == 'panda'){
        return Strings.PandaIND.Habitat;
    }else if(animals == 'rat'){
        return Strings.RatIND.Habitat;
    }else if(animals == 'cow'){
        return Strings.CowIND.Habitat;
    }else if(animals == 'goat'){
        return Strings.GoatIND.Habitat;
    }else if(animals == 'worm'){
        return Strings.WormIND.Habitat;
    }else if(animals == 'crocodile'){
        return Strings.CrocodileIND.Habitat;
    }else if(animals == 'deer'){
```

```
    return Strings.DeerIND.Habitat;
}else if(animals == 'kangaroo'){
    return Strings.KangarooIND.Habitat;
}else if(animals == 'elephant'){
    return Strings.ElephantIND.Habitat;
}else if(animals == 'chicken'){
    return Strings.ChickenIND.Habitat;
}else if(animals == 'giraffe'){
    return Strings.GiraffeIND.Habitat;
}else if(animals == 'bird'){
    return Strings.BirdIND.Habitat;
}else if(animals == 'squirrel'){
    return Strings.SquirrelIND.Habitat;
}else if(animals == 'turtle'){
    return Strings.TurtleIND.Habitat;
}else if(animals == 'crab'){
    return Strings.CrabIND.Habitat;
}else if(animals == 'spider'){
    return Strings.SpiderIND.Habitat;
}else if(animals == 'bird'){
    return Strings.BirdIND.Habitat;
}else if(animals == 'fish'){
    return Strings.FishIND.Habitat;
}else if(animals == 'duck'){
    return Strings.DuckIND.Habitat;
}else if(animals == 'grasshopper'){
    return Strings.GrasshopperIND.Habitat;
}else if(animals == 'gorilla'){
    return Strings.GorillaIND.Habitat;
}else if(animals == 'wolf'){
    return Strings.WolfIND.Habitat;
}else if(animals == 'snail'){
    return Strings.SnailIND.Habitat;
}else if(animals == 'sheep'){
    return Strings.SheepIND.Habitat;
}else if(animals == 'zebra'){
    return Strings.ZebraIND.Habitat;
}else if(animals == 'whale'){
    return Strings.WhaleIND.Habitat;
}else if(animals == 'shark'){
    return Strings.SharkIND.Habitat;
}else if(animals == 'dolphin'){
    return Strings.DolphinIND.Habitat;
}else if(animals == 'lizard'){
    return Strings.LizardIND.Habitat;
}else if(animals == 'dragonfly'){
    return Strings.DragonflyIND.Habitat;
}else if(animals == 'jellyfish'){
    return Strings.JellyfishIND.Habitat;
}else if(animals == 'lobster'){
    return Strings.LobsterIND.Habitat;
}else if(animals == 'leopard'){
    return Strings.LeopardIND.Habitat;
}else if(animals == 'tiger'){
    return Strings.TigerIND.Habitat;
}else if(animals == 'flamingo'){
    return Strings.FlamingoIND.Habitat;
}else if(animals == 'peacock'){
```

```
            return Strings.PeacockIND.Habitat;
        }else if(animals == 'parrot'){
            return Strings.ParrotIND.Habitat;
        }else if(animals == 'shrimp'){
            return Strings.ShrimpIND.Habitat;
        }else if(animals == 'beetle'){
            return Strings.BeetleIND.Habitat;
        }else if(animals == 'caterpillar'){
            return Strings.CaterpillarIND.Habitat;
        }else if(animals == 'hippopotamus'){
            return Strings.HippopotamusIND.Habitat;
        }else if(animals == 'camel'){
            return Strings.CamelIND.Habitat;
        }
        else{
            return "Tidak ditemukan";
        }
    }

}

static getStatusAnimals = (animals, language) =>
{
    if(language == "EN"){
        if(animals == 'cat'){
            return Strings.Cat.Status;
        }else if(animals == 'dog'){
            return Strings.Dog.Status;
        }else if(animals == 'horse'){
            return Strings.Horse.Status;
        }else if(animals == 'snake'){
            return Strings.Snake.Status;
        }else if(animals == 'butterfly'){
            return Strings.Butterfly.Status;
        }else if(animals == 'rabbit'){
            return Strings.Rabbit.Status;
        }else if(animals == 'bee'){
            return Strings.Bee.Status;
        }else if(animals == 'frog'){
            return Strings.Frog.Status;
        }else if(animals == 'monkey'){
            return Strings.Monkey.Status;
        }else if(animals == 'piglet'){
            return Strings.Piglet.Status;
        }else if(animals == 'panda'){
            return Strings.Panda.Status;
        }else if(animals == 'rat'){
            return Strings.Rat.Status;
        }else if(animals == 'cow'){
            return Strings.Cow.Status;
        }else if(animals == 'goat'){
            return Strings.Goat.Status;
        }else if(animals == 'worm'){
            return Strings.Worm.Status;
        }else if(animals == 'crocodile'){
            return Strings.Crocodile.Status;
        }else if(animals == 'deer'){
            return Strings.Deer.Status;
```

```
}else if(animals == 'kangaroo'){
   return Strings.Kangaroo.Status;
}else if(animals == 'elephant'){
   return Strings.Elephant.Status;
}else if(animals == 'chicken'){
   return Strings.Chicken.Status;
}else if(animals == 'giraffe'){
   return Strings.Giraffe.Status;
}else if(animals == 'bird'){
   return Strings.Bird.Status;
}else if(animals == 'squirrel'){
   return Strings.Squirrel.Status;
}else if(animals == 'turtle'){
   return Strings.Turtle.Status;
}else if(animals == 'crab'){
   return Strings.Crab.Status;
}else if(animals == 'spider'){
   return Strings.Spider.Status;
}else if(animals == 'bird'){
   return Strings.Bird.Status;
}else if(animals == 'fish'){
   return Strings.Fish.Status;
}else if(animals == 'duck'){
   return Strings.Duck.Status;
}else if(animals == 'grasshopper'){
   return Strings.Grasshopper.Status;
}else if(animals == 'gorilla'){
   return Strings.Gorilla.Status;
}else if(animals == 'wolf'){
   return Strings.Wolf.Status;
}else if(animals == 'snail'){
   return Strings.Snail.Status;
}else if(animals == 'sheep'){
   return Strings.Sheep.Status;
}else if(animals == 'zebra'){
   return Strings.Zebra.Status;
}else if(animals == 'whale'){
   return Strings.Whale.Status;
}else if(animals == 'shark'){
   return Strings.Shark.Status;
}else if(animals == 'dolphin'){
   return Strings.Dolphin.Status;
}else if(animals == 'lizard'){
   return Strings.Lizard.Status;
}else if(animals == 'dragonfly'){
   return Strings.Dragonfly.Status;
}else if(animals == 'jellyfish'){
   return Strings.Jellyfish.Status;
}else if(animals == 'lobster'){
   return Strings.Lobster.Status;
}else if(animals == 'leopard'){
   return Strings.Leopard.Status;
}else if(animals == 'tiger'){
   return Strings.Tiger.Status;
}else if(animals == 'flamingo'){
   return Strings.Flamingo.Status;
}else if(animals == 'peacock'){
   return Strings.Peacock.Status;
```

```
    }else if(animals == 'parrot'){
        return Strings.Parrot.Status;
    }else if(animals == 'shrimp'){
        return Strings.Shrimp.Status;
    }else if(animals == 'beetle'){
        return Strings.Beetle.Status;
    }else if(animals == 'caterpillar'){
        return Strings.Caterpillar.Status;
    }else if(animals == 'hippopotamus'){
        return Strings.Hippopotamus.Status;
    }else if(animals == 'camel'){
        return Strings.Camel.Status;
    }
    else{
        return "Undefined";
    }
}else if(language == "IND"){
    if(animals == 'cat'){
        return Strings.CatIND.Status;
    }else if(animals == 'dog'){
        return Strings.DogIND.Status;
    }else if(animals == 'horse'){
        return Strings.HorseIND.Status;
    }else if(animals == 'snake'){
        return Strings.SnakeIND.Status;
    }else if(animals == 'butterfly'){
        return Strings.ButterflyIND.Status;
    }else if(animals == 'rabbit'){
        return Strings.RabbitIND.Status;
    }else if(animals == 'bee'){
        return Strings.BeeIND.Status;
    }else if(animals == 'frog'){
        return Strings.FrogIND.Status;
    }else if(animals == 'dolphin'){
        return Strings.DolphinIND.Status;
    }else if(animals == 'monkey'){
        return Strings.MonkeyIND.Status;
    }else if(animals == 'piglet'){
        return Strings.PigletIND.Status;
    }else if(animals == 'panda'){
        return Strings.PandaIND.Status;
    }else if(animals == 'rat'){
        return Strings.RatIND.Status;
    }else if(animals == 'cow'){
        return Strings.CowIND.Status;
    }else if(animals == 'goat'){
        return Strings.GoatIND.Status;
    }else if(animals == 'worm'){
        return Strings.WormIND.Status;
    }else if(animals == 'crocodile'){
        return Strings.CrocodileIND.Status;
    }else if(animals == 'deer'){
        return Strings.DeerIND.Status;
    }else if(animals == 'kangaroo'){
        return Strings.KangarooIND.Status;
    }else if(animals == 'elephant'){
        return Strings.ElephantIND.Status;
    }else if(animals == 'chicken'){
```

```
    return Strings.ChickenIND.Status;
}else if(animals == 'giraffe'){
    return Strings.GiraffeIND.Status;
}else if(animals == 'bird'){
    return Strings.BirdIND.Status;
}else if(animals == 'squirrel'){
    return Strings.SquirrelIND.Status;
}else if(animals == 'turtle'){
    return Strings.TurtleIND.Status;
}else if(animals == 'duck'){
    return Strings.DuckIND.Status;
}else if(animals == 'crab'){
    return Strings.CrabIND.Status;
}else if(animals == 'spider'){
    return Strings.SpiderIND.Status;
}else if(animals == 'bird'){
    return Strings.BirdIND.Status;
}else if(animals == 'fish'){
    return Strings.FishIND.Status;
}else if(animals == 'duck'){
    return Strings.DuckIND.Status;
}else if(animals == 'grasshopper'){
    return Strings.GrasshopperIND.Status;
}else if(animals == 'gorilla'){
    return Strings.GorillaIND.Status;
}else if(animals == 'wolf'){
    return Strings.WolfIND.Status;
}else if(animals == 'snail'){
    return Strings.SnailIND.Status;
}else if(animals == 'sheep'){
    return Strings.SheepIND.Status;
}else if(animals == 'zebra'){
    return Strings.ZebraIND.Status;
}else if(animals == 'whale'){
    return Strings.WhaleIND.Status;
}else if(animals == 'shark'){
    return Strings.SharkIND.Status;
}else if(animals == 'dolphin'){
    return Strings.DolphinIND.Status;
}else if(animals == 'lizard'){
    return Strings.LizardIND.Status;
}else if(animals == 'dragonfly'){
    return Strings.DragonflyIND.Status;
}else if(animals == 'jellyfish'){
    return Strings.JellyfishIND.Status;
}else if(animals == 'lobster'){
    return Strings.LobsterIND.Status;
}else if(animals == 'leopard'){
    return Strings.LeopardIND.Status;
}else if(animals == 'tiger'){
    return Strings.TigerIND.Status;
}else if(animals == 'flamingo'){
    return Strings.FlamingoIND.Status;
}else if(animals == 'peacock'){
    return Strings.PeacockIND.Status;
}else if(animals == 'parrot'){
    return Strings.ParrotIND.Status;
}else if(animals == 'shrimp'){
```

```
            return Strings.ShrimpIND.Status;
        }else if(animals == 'beetle'){
            return Strings.BeetleIND.Status;
        }else if(animals == 'caterpillar'){
            return Strings.CaterpillarIND.Status;
        }else if(animals == 'hippopotamus'){
            return Strings.HippopotamusIND.Status;
        }else if(animals == 'camel'){
            return Strings.CamelIND.Status;
        }
        else{
            return "Tidak ditemukan";
        }
    }

}

static getTextSpeechAnimals = (animals, language) =>
{
    if(language == "EN"){
        if(animals == 'cat'){
            return Strings.Cat.textSpeech;
        }else if(animals == 'dog'){
            return Strings.Dog.textSpeech;
        }else if(animals == 'horse'){
            return Strings.Horse.textSpeech;
        }else if(animals == 'snake'){
            return Strings.Snake.textSpeech;
        }else if(animals == 'butterfly'){
            return Strings.Butterfly.textSpeech;
        }else if(animals == 'rabbit'){
            return Strings.Rabbit.textSpeech;
        }else if(animals == 'bee'){
            return Strings.Bee.textSpeech;
        }else if(animals == 'frog'){
            return Strings.Frog.textSpeech;
        }else if(animals == 'monkey'){
            return Strings.Monkey.textSpeech;
        }else if(animals == 'piglet'){
            return Strings.Piglet.textSpeech;
        }else if(animals == 'panda'){
            return Strings.Panda.textSpeech;
        }else if(animals == 'rat'){
            return Strings.Rat.textSpeech;
        }else if(animals == 'cow'){
            return Strings.Cow.textSpeech;
        }else if(animals == 'goat'){
            return Strings.Goat.textSpeech;
        }else if(animals == 'worm'){
            return Strings.Worm.textSpeech;
        }else if(animals == 'crocodile'){
            return Strings.Crocodile.textSpeech;
        }else if(animals == 'deer'){
            return Strings.Deer.textSpeech;
        }else if(animals == 'kangaroo'){
            return Strings.Kangaroo.textSpeech;
        }else if(animals == 'elephant'){
            return Strings.Elephant.textSpeech;
```

```
}else if(animals == 'chicken'){
   return Strings.Chicken.textSpeech;
}else if(animals == 'giraffe'){
   return Strings.Giraffe.textSpeech;
}else if(animals == 'bird'){
   return Strings.Bird.textSpeech;
}else if(animals == 'squirrel'){
   return Strings.Squirrel.textSpeech;
}else if(animals == 'turtle'){
   return Strings.Turtle.textSpeech;
}else if(animals == 'crab'){
   return Strings.Crab.textSpeech;
}else if(animals == 'spider'){
   return Strings.Spider.textSpeech;
}else if(animals == 'bird'){
   return Strings.Bird.textSpeech;
}else if(animals == 'fish'){
   return Strings.Fish.textSpeech;
}else if(animals == 'duck'){
   return Strings.Duck.textSpeech;
}else if(animals == 'grasshopper'){
   return Strings.Grasshopper.textSpeech;
}else if(animals == 'gorilla'){
   return Strings.Gorilla.textSpeech;
}else if(animals == 'wolf'){
   return Strings.Wolf.textSpeech;
}else if(animals == 'snail'){
   return Strings.Snail.textSpeech;
}else if(animals == 'sheep'){
   return Strings.Sheep.textSpeech;
}else if(animals == 'zebra'){
   return Strings.Zebra.textSpeech;
}else if(animals == 'whale'){
   return Strings.Whale.textSpeech;
}else if(animals == 'shark'){
   return Strings.Shark.textSpeech;
}else if(animals == 'dolphin'){
   return Strings.Dolphin.textSpeech;
}else if(animals == 'lizard'){
   return Strings.Lizard.textSpeech;
}else if(animals == 'dragonfly'){
   return Strings.Dragonfly.textSpeech;
}else if(animals == 'jellyfish'){
   return Strings.Jellyfish.textSpeech;
}else if(animals == 'lobster'){
   return Strings.Lobster.textSpeech;
}else if(animals == 'leopard'){
   return Strings.Leopard.textSpeech;
}else if(animals == 'tiger'){
   return Strings.Tiger.textSpeech;
}else if(animals == 'flamingo'){
   return Strings.Flamingo.textSpeech;
}else if(animals == 'peacock'){
   return Strings.Peacock.textSpeech;
}else if(animals == 'parrot'){
   return Strings.Parrot.textSpeech;
}else if(animals == 'shrimp'){
   return Strings.Shrimp.textSpeech;
```

```
    }else if(animals == 'beetle'){
        return Strings.Beetle.textSpeech;
    }else if(animals == 'caterpillar'){
        return Strings.Caterpillar.textSpeech;
    }else if(animals == 'hippopotamus'){
        return Strings.Hippopotamus.textSpeech;
    }else if(animals == 'camel'){
        return Strings.Camel.textSpeech;
    }
    else{
        return "Undefined";
    }
}else if(language == "IND"){
    if(animals == 'kucing'){
        return Strings.CatIND.textSpeech;
    }else if(animals == 'anjing'){
        return Strings.DogIND.textSpeech;
    }else if(animals == 'kuda'){
        return Strings.HorseIND.textSpeech;
    }else if(animals == 'ular'){
        return Strings.SnakeIND.textSpeech;
    }else if(animals == 'kupu-kupu'){
        return Strings.ButterflyIND.textSpeech;
    }else if(animals == 'kelinci'){
        return Strings.RabbitIND.textSpeech;
    }else if(animals == 'lebah'){
        return Strings.BeeIND.textSpeech;
    }else if(animals == 'katak'){
        return Strings.FrogIND.textSpeech;
    }else if(animals == 'monyet'){
        return Strings.MonkeyIND.textSpeech;
    }else if(animals == 'babi'){
        return Strings.PigletIND.textSpeech;
    }else if(animals == 'panda'){
        return Strings.PandaIND.textSpeech;
    }else if(animals == 'tikus'){
        return Strings.RatIND.textSpeech;
    }else if(animals == 'sapi'){
        return Strings.CowIND.textSpeech;
    }else if(animals == 'kambing'){
        return Strings.GoatIND.textSpeech;
    }else if(animals == 'cacing'){
        return Strings.WormIND.textSpeech;
    }else if(animals == 'buaya'){
        return Strings.CrocodileIND.textSpeech;
    }else if(animals == 'rusa'){
        return Strings.DeerIND.textSpeech;
    }else if(animals == 'kangguru'){
        return Strings.KangarooIND.textSpeech;
    }else if(animals == 'gajah'){
        return Strings.ElephantIND.textSpeech;
    }else if(animals == 'ayam'){
        return Strings.ChickenIND.textSpeech;
    }else if(animals == 'jerapah'){
        return Strings.GiraffeIND.textSpeech;
    }else if(animals == 'burung'){
        return Strings.BirdIND.textSpeech;
    }else if(animals == 'tupai'){
```

```
    return Strings.SquirrelIND.textSpeech;
}else if(animals == 'penyu atau kura-kura'){
    return Strings.TurtleIND.textSpeech;
}else if(animals == 'kepiting'){
    return Strings.CrabIND.textSpeech;
}else if(animals == 'laba-laba'){
    return Strings.SpiderIND.textSpeech;
}else if(animals == 'burung'){
    return Strings.BirdIND.textSpeech;
}else if(animals == 'ikan'){
    return Strings.FishIND.textSpeech;
}else if(animals == 'bebek'){
    return Strings.DuckIND.textSpeech;
}else if(animals == 'belalang'){
    return Strings.GrasshopperIND.textSpeech;
}else if(animals == 'gorila'){
    return Strings.GorillaIND.textSpeech;
}else if(animals == 'serigala'){
    return Strings.WolfIND.textSpeech;
}else if(animals == 'siput'){
    return Strings.SnailIND.textSpeech;
}else if(animals == 'domba'){
    return Strings.SheepIND.textSpeech;
}else if(animals == 'zebra'){
    return Strings.ZebraIND.textSpeech;
}else if(animals == 'paus'){
    return Strings.WhaleIND.textSpeech;
}else if(animals == 'hiu'){
    return Strings.SharkIND.textSpeech;
}else if(animals == 'lumba-lumba'){
    return Strings.DolphinIND.textSpeech;
}else if(animals == 'kadal'){
    return Strings.LizardIND.textSpeech;
}else if(animals == 'capung'){
    return Strings.DragonflyIND.textSpeech;
}else if(animals == 'ubur-ubur'){
    return Strings.JellyfishIND.textSpeech;
}else if(animals == 'lobster'){
    return Strings.LobsterIND.textSpeech;
}else if(animals == 'macan tutul'){
    return Strings.LeopardIND.textSpeech;
}else if(animals == 'harimau'){
    return Strings.TigerIND.textSpeech;
}else if(animals == 'flamingo'){
    return Strings.FlamingoIND.textSpeech;
}else if(animals == 'merak'){
    return Strings.PeacockIND.textSpeech;
}else if(animals == 'burung beo'){
    return Strings.ParrotIND.textSpeech;
}else if(animals == 'udang'){
    return Strings.ShrimpIND.textSpeech;
}else if(animals == 'kumbang'){
    return Strings.BeetleIND.textSpeech;
}else if(animals == 'ulat'){
    return Strings.CaterpillarIND.textSpeech;
}else if(animals == 'kuda nill'){
    return Strings.HippopotamusIND.textSpeech;
}else if(animals == 'unta'){
```

```
            return Strings.CamelIND.textSpeech;
          }
          else{
            return "Tidak ditemukan";
          }
        }

    }

    static renderIf = (condition, content) =>
    {
        if(condition){
            return content;
        }
        else{
            return null;
        }
    }
}
```

Strings.js

```
export default {
    CameraPage: {
        text1: 'Select your langauge',
        tittleAlert1: "Announcement",
        tittleAlert1IND: "Pemberitahuan",
        alert1: "It's animal !! This is a ",
        alert1IND: "Binatang !! Binatang ini adalah ",
        tittleAlert2: "Success",
        tittleAlert2IND: "Berhasil",
        alert2: "Successfully saved in gallery!",
        alert2IND: "Berhasil di simpan di galeri!",
        tittleAlert3: "Success",
        tittleAlert3IND: "Berhasil",
        alert3: "Data successfully deleted!",
        alert3IND: "Data berhasil dihapus!",
        tittleAlert4: "Announcement",
        tittleAlert4IND: "Pemberitahuan",
        alert4: "Are you sure want to delete this collection?",
        alert4IND: "Apakah Anda yakin ingin menghapus koleksi ini?",
        tittleAlert5: "Announcement",
        tittleAlert5IND: "Pemberitahuan",
        alert5: "This is not animal",
        alert5IND: "Ini bukan binatang",
        tittleAlert6: "Announcement",
        tittleAlert6IND: "Pemberitahuan",
        alert6: "Are you sure want to exit?",
        alert6IND: "Anda yakin ingin keluar?",
        alert7: "This animals is Undefined",
        alert7IND: "Binatang ini tidak terdefinisikan",

        desc1: "Not Animals",
        desc1IND: "Bukan Binatang",
        desc2: "Name : ",
        desc2IND: "Nama : ",
        desc3: "Class : ",
        desc3IND: "Kelas : ",
```

```
    desc4: "Habitat : ",
    desc4IND: "Habitat : ",
    desc5: "Status : ",
    desc5IND: "Status : ",
  },
  Cat:{
    Name: 'Cat',
    Class: 'Mammalia',
    Habitat: 'Savannas, tropical rain forests, and boreal forest',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Cat. Cats are mammals. They live in Savannas, tropical rain forests, and boreal forests. Cats are harmless animals and not dangerous or low.',
  },
  CatIND:{
    Name: 'Kucing',
    Class: 'Mamalia',
    Habitat: 'Savana, hutan hujan tropis, dan hutan boreal',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kucing. Kucing adalah mamalia. Mereka tinggal di Savana, hutan hujan tropis, dan hutan boreal. Kucing adalah hewan yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Dog:{
    Name: 'Dog',
    Class: 'Mammalia',
    Habitat: 'Prairies, deserts, grasslands, forests, rain forests, coastal regions and arctic zones',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Dog. Dogs are mammals. They live in Prairies, deserts, grasslands, forests, rain forests, coastal regions and arctic zones. Dogs are harmless animals and not dangerous or low.',
  },
  DogIND:{
    Name: 'Anjing',
    Class: 'Mamalia',
    Habitat: 'Padang rumput, gurun, hutan, hutan hujan, wilayah pesisir dan zona Arktik',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Anjing. Anjing adalah mamalia. Mereka tinggal di padang rumput, gurun, hutan, hutan hujan, daerah pesisir dan zona Arktik. Anjing adalah hewan yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Horse:{
    Name: 'Horse',
    Class: 'Mammalia',
    Habitat: 'Utilize grasslands, meadows, mountains, foothill',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Horse. Horse are mammals. They live in Utilize grasslands, meadows, mountains, and foothill. Horses are wild animals and medium dangerous.',
  },
  HorseIND:{
    Name: 'Kuda',
    Class: 'Mamalia',
    Habitat: 'Padang rumput yang luas, gunung, dan bukit di kaki gunung',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kuda. Kuda adalah mamalia. Mereka tinggal di padang rumput yang luas, gunung, dan bukit di kaki gunung. Kuda adalah binatang liar dan tingkat berbahaya nya sedang',
  },
```

```
  Snake:{
    Name: 'Snake',
    Class: 'Reptilia',
    Habitat: 'Forests, prairies, and deserts. Others live in water environments',
    Status: 'DANGEROUS',
    textSpeech: 'Snake. Snakes are reptiles. They live in Forests, prairies, and deserts. Others live in water environments. Snake are wild animals and very dangerous.',
  },
  SnakeIND:{
    Name: 'Ular',
    Class: 'Reptil',
    Habitat: 'Hutan, padang rumput, dan gurun. Ada Spesies lain hidup di lingkungan air juga',
    Status: 'BERBAHAYA',
    textSpeech: 'Ular. Ular adalah reptil. Mereka tinggal di Hutan, padang rumput, dan gurun. Ada Spesies lain hidup di lingkungan air juga. Ular adalah binatang liar dan sangat berbahaya.',
  },
  Butterfly:{
    Name: 'Butterfly',
    Class: 'Insecta',
    Habitat: 'Salt marshes, mangroves, sand dunes, lowland forest, wetlands, grasslands and mountain zones',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Butterfly. Butterflies are insects. They live in Salt marshes, mangroves, sand dunes, lowland forest, wetlands, grasslands and mountain zones. Butterflies are harmless insects and not dangerous or low.',
  },
  ButterflyIND:{
    Name: 'Kupu-Kupu',
    Class: 'Serangga',
    Habitat: 'Rawa-rawa, daerah bakau, bukit pasir, hutan dataran rendah, lahan basah, padang rumput, dan zona gunung',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kupu-kupu. Kupu-kupu adalah serangga. Mereka hidup di rawa-rawa, daerah bakau, bukit pasir, hutan dataran rendah, lahan basah, padang rumput, dan zona pegunungan. Kupu-kupu adalah serangga yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Rabbit:{
    Name: 'Rabbit',
    Class: 'Mammalia',
    Habitat: 'Meadows, woods, forests, grasslands, deserts and wetlands',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Rabbit. Rabbit are Mammalia. They live in Meadows, woods, forests, grasslands, deserts and wetlands. Rabbits are harmless mammals and not dangerous or low.',
  },
  RabbitIND:{
    Name: 'Kelinci',
    Class: 'Mamalia',
    Habitat: 'Padang rumput, pepohonan, hutan, padang rumput, gurun dan lahan basah',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kelinci. Kelinci adalah mamalia. Mereka hidup di padang rumput, pepohonan, hutan, padang rumput, gurun dan lahan basah. Kelinci adalah hewan yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
```

```
  Bee:{
    Name: 'Bee',
    Class: 'Insecta',
    Habitat: 'Gardens, woodlands, orchards, meadows and other areas where flowering plants are abundant',
    Status: 'DANGEROUS',
    textSpeech: 'Bee. Bees are insects. They live in Gardens, woodlands, orchards, meadows and other areas where flowering plants are abundant. Bees are wild insects and very dangerous',
  },
  BeeIND:{
    Name: 'Lebah',
    Class: 'Serangga',
    Habitat: 'Kebun, hutan, bunga, padang rumput dan daerah lain di mana tanaman berbunga berlimpah',
    Status: 'BERBAHAYA',
    textSpeech: 'Lebah. Lebah adalah serangga. Mereka hidup di Kebun, hutan, bunga, padang rumput dan daerah lain di mana tanaman berbunga berlimpah. Lebah adalah serangga liar dan tergolong berbahaya.',
  },
  Frog:{
    Name: 'Frog',
    Class: 'Amphibia',
    Habitat: 'Ponds, lakes, streams, creeks or rivers',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Frog. Frogs are amphibians. They live in Ponds, lakes, streams, creeks or rivers. Frogs are harmless amphibians and not dangerous',
  },
  FrogIND:{
    Name: 'Katak',
    Class: 'Amfibi',
    Habitat: 'Kolam, danau, sungai, anak sungai atau sungai kecil',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Katak. Katak adalah Amfibi. Mereka hidup di Kolam, danau, sungai, anak sungai atau sungai kecil. Katak adalah amfibi yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Dolphin:{
    Name: 'Dolphin',
    Class: 'Mammalia',
    Habitat: 'Inhabit all of the oceans and some important rivers. Most of them live in saltwater, but a few species thrive in freshwater. ',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Dolphin. Dolphins are mammals. They live Inhabit all of the oceans and some important rivers. Most of them live in saltwater, but a few species thrive in freshwater. Dolphins are harmless mammals and not dangerous',
  },
  DolphinIND:{
    Name: 'Lumba-Lumba',
    Class: 'Mamalia',
    Habitat: 'Mendiami semua lautan dan beberapa sungai penting. Sebagian besar dari mereka hidup di air asin, tetapi beberapa spesies tumbuh subur di air tawar',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Lumba-Lumba. Lumba-Lumba adalah Mamalia. Mereka Mendiami semua lautan dan beberapa sungai penting. Sebagian besar dari mereka hidup di air asin, tetapi beberapa spesies tumbuh subur di air tawar. Lumba-lumba adalah mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
```

```
  Monkey:{
    Name: 'Monkey',
    Class: 'Mammalia',
    Habitat: 'Forests, grasslands, high plains, and mountain habitats',
    Status: 'DANGEROUS',
    textSpeech: 'Monkey. Monkeys are mammals. They live in Forests, grasslands, high plains, and mountain habitats. Monkeys are wild mammals and very dangerous',
  },
  MonkeyIND:{
    Name: 'Monyet',
    Class: 'Mamalia',
    Habitat: 'Hutan, padang rumput, dataran tinggi, dan habitat gunung',
    Status: 'BERBAHAYA',
    textSpeech: 'Monyet. Monyet adalah Mamalia. Mereka hidup di Hutan, padang rumput, dataran tinggi, dan habitat gunung. Monyet adalah mamalia liar yang berbahaya.',
  },
  Piglet:{
    Name: 'Piglet',
    Class: 'Mammalia',
    Habitat: 'Rainforests, scrubby secondary forests, mangroves, swamps, and grasslands.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Piglet. Piglets are mammals. They live in Rainforests, scrubby secondary forests, mangroves, swamps, and grasslands. Piglet are harmless mammals and not dangerous',
  },
  PigletIND:{
    Name: 'Babi',
    Class: 'Mamalia',
    Habitat: 'Hutan hujan, hutan lebat, hutan bakau, rawa, dan padang rumput',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Babi. Babi adalah Mamalia. Mereka hidup di Hutan hujan, hutan lebat, hutan bakau, rawa, dan padang rumput. Babi adalah mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Panda:{
    Name: 'Panda',
    Class: 'Mammalia',
    Habitat: 'Mountainous regions of central China, in Sichuan, Shaanxi and Gansu provinces, according to the National Zoo.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Panda. Pandas are mammals. They live in Mountainous regions of central China, in Sichuan, Shaanxi and Gansu provinces, according to the National Zoo. Panda are harmless mammals and not dangerous',
  },
  PandaIND:{
    Name: 'Panda',
    Class: 'Mamalia',
    Habitat: 'Daerah pegunungan di Cina tengah, di provinsi Sichuan, Shaanxi, dan Gansu, dapat ditemukan juga di Kebun Binatang Nasional.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Panda. Panda adalah Mamalia. Mereka hidup di Daerah pegunungan di Cina tengah, di provinsi Sichuan, Shaanxi, dan Gansu, dapat ditemukan juga di Kebun Binatang Nasional. Panda adalah mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Rat:{
```

```
    Name: 'Rat',
    Class: 'Mammalia',
    Habitat: 'Ports, woodlands, dumps, sewers, barns, sheds, basements, attics, cities, suburbs, and more.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Rat. Rats are mammals. They live in ports, woodlands, dumps, sewers, barns, sheds, basements, attics, cities, suburbs, and more. Rats are wild mammals and not dangerous',
  },
  RatIND:{
    Name: 'Tikus',
    Class: 'Mamalia',
    Habitat: 'Pelabuhan, hutan, tempat pembuangan, selokan, lumbung, gudang, ruang bawah tanah, loteng, kota, pinggiran kota, dan banyak lagi',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Tikus. Tikus adalah Mamalia. Mereka hidup di pelabuhan, hutan, tempat pembuangan, selokan, lumbung, gudang, ruang bawah tanah, loteng, kota, pinggiran kota, dan banyak lagi. Tikus adalah mamalia liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Cow:{
    Name: 'Cow',
    Class: 'Mammalia',
    Habitat: 'Grasslands, forests, pastures, and parts of the world',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Cow. Cows are mammals. They live in Grasslands, forests, pastures, and parts of the world. Cows are harmless mammals and not dangerous',
  },
  CowIND:{
    Name: 'Sapi',
    Class: 'Mamalia',
    Habitat: 'Padang rumput, hutan, padang rumput domestik, dan dapat ditemukan di berbagai belahan dunia',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Sapi. Sapi adalah Mamalia. Mereka hidup di Padang rumput, hutan, padang rumput domestik, dan dapat ditemukan di berbagai belahan dunia. Sapi adalah mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Goat:{
    Name: 'Goat',
    Class: 'Mammalia',
    Habitat: 'Grasslands, forests, pastures, and parts of the world',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Goats. Goats are mammals. They live in Grasslands, forests, pastures, and parts of the world. Goats are harmless mammals and not dangerous',
  },
  GoatIND:{
    Name: 'Kambing',
    Class: 'Mamalia',
    Habitat: 'Padang rumput, hutan, padang rumput domestik, dan dapat ditemukan di berbagai belahan dunia',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kambing. Kambing adalah Mamalia. Mereka hidup di Padang rumput, hutan, padang rumput domestik, dan dapat ditemukan di berbagai belahan dunia. Kambing adalah mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Worm:{
    Name: 'Worm',
```

```
    Class: 'Clitellata',
    Habitat: 'Moist soil, mud, shores of lakes or swamps.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Worm. Worms are Clitellata. They live in moist soil, mud, shores of lakes or swamps. Worms are harmless Clitellata and not dangerous',
  },
  WormIND:{
    Name: 'Cacing',
    Class: 'Clitellata',
    Habitat: 'Tanah yang lembab, lumpur, tepi danau atau rawa',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Cacing. Cacing adalah Clitellata. Mereka hidup di Tanah yang lembab, lumpur, tepi danau atau rawa. Cacing adalah Clitellata yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Crocodile:{
    Name: 'Crocodile',
    Class: 'Reptilia',
    Habitat: 'Rivers, lakes, wetlands and sometimes in brackish water.',
    Status: 'DANGEROUS',
    textSpeech: 'Crocodile. Crocodiles are reptiles. They live in Rivers, lakes, wetlands and sometimes in brackish water. Crocodiles are wild animals and very dangerous',
  },
  CrocodileIND:{
    Name: 'Buaya',
    Class: 'Reptil',
    Habitat: 'Sungai, danau, lahan basah, dan terkadang ada di rawa.',
    Status: 'BERBAHAYA',
    textSpeech: 'Buaya. Buaya adalah Reptil. Mereka hidup di Sungai, danau, lahan basah, dan terkadang ada di rawa. Buaya adalah binatang liar yang sangat buas dan berbahaya.',
  },
  Deer:{
    Name: 'Deer',
    Class: 'Mammalia',
    Habitat: 'Tropical rainforests, tundra, mountains, hidden forests, and swampy areas',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Deer. Deers are mammals. They live in Tropical rainforests, tundra, mountains, hidden forests, swampy areas. Deers are wild animals and not dangerous',
  },
  DeerIND:{
    Name: 'Rusa',
    Class: 'Mamalia',
    Habitat: 'Hutan hujan tropis, tundra, gunung, hutan tersembunyi, dan daerah rawa.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Rusa. Rusa adalah Mamalia. Mereka hidup di Hutan hujan tropis, tundra, gunung, hutan tersembunyi, daerah rawa. Rusa adalah binatang liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Kangaroo:{
    Name: 'Kangaroo',
    Class: 'Mammalia',
    Habitat: 'Wooded forests, savannahs, scrublands, deserts, grassy plains, and more in Australian.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Kangaroo. Kangaroos are mammals. They live in Wooded forests, savannahs, scrublands, deserts, grassy plains, and more in Australian. Kangaroos are harmless mammals and not dangerous',
```

```
  },
  KangarooIND:{
    Name: 'Kangguru',
    Class: 'Mamalia',
    Habitat: 'Hutan berkayu, sabana, semak belukar, gurun, dataran berumput, dan banyak lagi di Australia.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kangguru. Kangguru adalah Mamalia. Mereka hidup di Hutan berkayu, sabana, semak belukar, gurun, dataran berumput, dan banyak lagi di Australia. Kangguru adalah mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Elephant:{
    Name: 'Elephant',
    Class: 'Mammalia',
    Habitat: 'Savannas, grasslands, and forests, but they also live in deserts, swamps, and mountains.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Elephant. Elephants are mammals. They live in savannas, grasslands, and forests, but they also live in deserts, swamps, and mountains. Elephants are harmless mammals and not dangerous',
  },
  ElephantIND:{
    Name: 'Gajah',
    Class: 'Mamalia',
    Habitat: 'Hutan sabana, padang rumput, dan hutan, tetapi mereka juga hidup di gurun, rawa, dan gunung',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Gajah. Gajah adalah Mamalia. Mereka hidup di Hutan sabana, padang rumput, dan hutan, tetapi mereka juga hidup di gurun, rawa, dan gunung. Gajah adalah mamalia liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Chicken:{
    Name: 'Chicken',
    Class: 'Aves',
    Habitat: 'Utilize farms and backyards',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Chicken. Chickens are aves. They live in Utilize farms and backyards. Chickens are harmless aves and not dangerous',
  },
  ChickenIND:{
    Name: 'Ayam',
    Class: 'Aves',
    Habitat: 'Lahan pertanian dan halaman belakang perternakan',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Ayam. Ayam adalah Aves. Mereka hidup di Lahan pertanian dan halaman belakang perternakan. Ayam adalah binatang aves yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Giraffe:{
    Name: 'Giraffe',
    Class: 'Mammalia',
    Habitat: 'Savannas, grasslands or open woodlands',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Giraffe. Giraffes are mammals. They live in savannas, grasslands or open woodlands. Giraffes are harmless mammals and not dangerous',
  },
  GiraffeIND:{
    Name: 'Jerapah',
```

```
    Class: 'Mamalia',
    Habitat: 'Hutan sabana, padang rumput atau hutan terbuka',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Jerapah. Jerapah adalah mamalia. Mereka hidup di Hutan sabana, padang rumput atau hutan terbuka. Jerapah adalah mamalia liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Bird:{
    Name: 'Bird',
    Class: 'Aves',
    Habitat: 'Forests, estuaries, swamps and almost all over the world',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Bird. Birds are aves. They live in forests, estuaries, swamps and almost all over the world. Birds are harmless aves and not dangerous',
  },
  BirdIND:{
    Name: 'Burung',
    Class: 'Aves',
    Habitat: 'Hutan, muara, rawa-rawa dan hampir terdapat diseluruh dunia',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Burung. Burung adalah aves. Mereka hidup di hutan, muara, rawa-rawa dan hampir terdapat diseluruh dunia. Burung adalah aves liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Squirrel:{
    Name: 'Squirrel',
    Class: 'Mammalia',
    Habitat: 'Almost every habitat, from tropical rainforest to semiarid desert.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Squirrel. Squirrels are mammals. They live in Almost every habitat, from tropical rainforest to semiarid desert. Squirrels are harmless mammals and not dangerous',
  },
  SquirrelIND:{
    Name: 'Tupai',
    Class: 'Mamalia',
    Habitat: 'Hampir ada di setiap habitat, dari hutan hujan tropis hingga gurun semi kering.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Tupai. Tupai adalah mamalia. Mereka hidup di Hampir setiap habitat, dari hutan hujan tropis hingga gurun semi kering. Tupai adalah mamalia liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Turtle:{
    Name: 'Turtle',
    Class: 'Reptilia',
    Habitat: 'Almost every ocean basin throughout the world, nesting on tropical and subtropical beaches.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Turtle. Turtles are reptiles. They live in almost every ocean basin throughout the world, nesting on tropical and subtropical beaches. Turtles are harmless reptiles and not dangerous',
  },
  TurtleIND:{
    Name: 'Penyu atau Kura-Kura',
    Class: 'Reptil',
    Habitat: 'Hampir setiap cekungan laut di seluruh dunia, bersarang di pantai tropis dan subtropis.',
```

```
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Penyu atau Kura-Kura adalah reptil. Mereka hidup di hampir setiap cekungan laut di seluruh dunia, bersarang di pantai tropis dan subtropis. Penyu atau Kura-Kura adalah reptil yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Duck:{
    Name: 'Duck',
    Class: 'Aves',
    Habitat: 'Wetlands, marshes, ponds, rivers, lakes and oceans.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Duck. Ducks are Aves. They live in wetlands, marshes, ponds, rivers, lakes and oceans. Ducks are harmless aves and not dangerous',
  },
  DuckIND:{
    Name: 'Bebek',
    Class: 'Aves',
    Habitat: 'Lahan basah, rawa-rawa, kolam, sungai, danau, dan lautan.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Bebek. Bebek adalah aves. Mereka hidup di Lahan basah, rawa-rawa, kolam, sungai, danau, dan lautan. Bebek adalah aves yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Crab:{
    Name: 'Crab',
    Class: 'Malacostraca',
    Habitat: 'Just about any body of water, including salt water and fresh water, and some can survive on land.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Crab. Crabs are Malacostraca. They live in Just about any body of water, including salt water and fresh water, and some can survive on land. Crabs are harmless Malacostraca and not dangerous',
  },
  CrabIND:{
    Name: 'Kepiting',
    Class: 'Malacostraca',
    Habitat: 'Hampir terdapat di semua badan air, termasuk air asin dan air tawar, dan beberapa dapat bertahan hidup di darat.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kepiting. Kepiting adalah Malacostraca. Mereka hidup di Hampir terdapat di semua badan air, termasuk air asin dan air tawar, dan beberapa dapat bertahan hidup di darat. Kepiting adalah Malacostraca yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Spider:{
    Name: 'Spider',
    Class: 'Arachnida',
    Habitat: 'Everywhere ranges from desert to rainforest to backyard and everything in between.',
    Status: 'DANGEROUS',
    textSpeech: 'Spider. Spiders are Arachnida. They live in Everywhere ranges from desert to rainforest to backyard and everything in between. Spiders are Arachnida which are medium dangerous',
  },
  SpiderIND:{
    Name: 'Laba-laba',
    Class: 'Arachnida',
    Habitat: 'Di mana-mana mulai dari gurun ke hutan hujan ke halaman belakang dan segala sesuatu di antaranya',
```

```
    Status: 'BERBAHAYA',
    textSpeech: 'Laba-Laba. Laba-Laba adalah Arachnida. Mereka hidup Di mana-mana
mulai dari gurun ke hutan hujan ke halaman belakang dan segala sesuatu di antaranya.
Laba-Laba adalah Arachnida yang cukup berbahaya bagi beberapa spesies tertentu.',
  },


  Bird:{
    Name: 'Bird',
    Class: 'Aves',
    Habitat: 'On beaches, estuaries, marshes and almost all over the world',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Bird. Birds are Aves. They live On beaches, estuaries, marshes and
almost all over the world. Birds are harmless aves and not dangerous',
  },
  BirdIND:{
    Name: 'Burung',
    Class: 'Aves',
    Habitat: 'Di pantai, muara, rawa-rawa dan hampir di seluruh dunia.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Burung. Burung adalah aves. Mereka hidup Di pantai, muara, rawa-
rawa dan hampir di seluruh dunia. Burung adalah makhluk hidup terbang yang tidak
berbahaya atau tingkat berbahaya nya rendah.',
  },
  Fish:{
    Name: 'Fish',
    Class: 'Agnatha, Chondrichthyes, or Osteichthyes ',
    Habitat: 'Can be found in all waters, swamps, and even mud.',
    Status: 'Depends on species',
    textSpeech: 'Fish. Fish are aquatic living things. They Can be found in all waters,
swamps, and even mud. Fish are very diverse and very dangerous for certain species',
  },
  FishIND:{
    Name: 'Ikan',
    Class: 'Agnatha, Chondrichthyes, atau Osteichthyes',
    Habitat: 'Dapat ditemukan di seluruh perairan, rawa, bahkan lumpur.',
    Status: 'Tergantung Pada Spesies',
    textSpeech: 'Ikan. Ikan adalah makhluk hidup air. Mereka Dapat ditemukan di
seluruh perairan, rawa, bahkan lumpur. Ikan sangat beragam dan sangat berbahaya
untuk spesies tertentu.',
  },
  Grasshopper:{
    Name: 'Grasshopper',
    Class: 'Insecta',
    Habitat: 'Grass and other low plants, though some species live in forests or jungles.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Grasshopper. Grasshopper is an Insect. They live in Grass and other
low plants, though some species live in forests or jungles. Grasshopper is a harmless
pest insect and not dangerous',
  },
  GrasshopperIND:{
    Name: 'Belalang',
    Class: 'Serangga',
    Habitat: 'Rumput dan tanaman rendah lainnya, meskipun beberapa spesies hidup di
hutan atau hutan.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Belalang. Belalang adalah serangga. Mereka hidup Di Rumput dan
tanaman rendah lainnya, meskipun beberapa spesies hidup di hutan atau hutan.
```

```
Belalang adalah serangga hama yang tidak berbahaya atau tingkat berbahaya nya rendah',
  },
  Gorilla:{
    Name: 'Gorilla',
    Class: 'Mammalia',
    Habitat: 'Dense old-growth forests, areas along the edges of forests with high concentrations of low vegetation, mountain and sub-mountain forests, swamp forests and savannah forests',
    Status: 'DANGEROUS',
    textSpeech: 'Gorilla. Gorillas are Mammals. They live in dense old-growth forests, areas along the edges of forests with high concentrations of low vegetation, mountain and sub-mountain forests, swamp forests and savannah forests. Gorillas are mammals that are classified as dangerous',
  },
  GorillaIND:{
    Name: 'Gorila',
    Class: 'Mamalia',
    Habitat: 'Hutan tua yang lebat, area di sepanjang tepi hutan dengan konsentrasi vegetasi rendah, hutan pegunungan dan sub-gunung yang tinggi, hutan rawa dan hutan sabana',
    Status: 'BERBAHAYA',
    textSpeech: 'Gorila. Gorila adalah mamalia. Mereka hidup Di Hutan tua yang lebat, area di sepanjang tepi hutan dengan konsentrasi vegetasi rendah, hutan pegunungan dan sub-gunung yang tinggi, hutan rawa dan hutan sabana. Gorila adalah mamalia liar yang berbahaya.',
  },
  Wolf:{
    Name: 'Wolf',
    Class: 'Mammalia',
    Habitat: 'Deserts, open places with a lot of grass, forests.',
    Status: 'DANGEROUS',
    textSpeech: 'Wolf. Wolves are mammals. They live in Deserts, open places with a lot of grass, forests. Wolves are wild mammals that are very dangerous',
  },
  WolfIND:{
    Name: 'Serigala',
    Class: 'Mamalia',
    Habitat: 'Gurun, tempat terbuka dengan banyak rumput, hutan.',
    Status: 'BERBAHAYA',
    textSpeech: 'Serigala. Serigala adalah mamalia. Mereka hidup Di Gurun, tempat terbuka dengan banyak rumput, hutan. Serigala adalah mamalia liar yang sangat berbahaya',
  },
  Snail:{
    Name: 'Snail',
    Class: 'Gastropoda',
    Habitat: 'Dwell in areas of high altitude, mountainous regions, hot and cold places',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Snail. Snails are Gastropoda. They dwell in areas of high altitude, mountainous regions, hot and cold places. Snails are harmless Gastropoda and not dangerous',
  },
  SnailIND:{
    Name: 'Siput',
    Class: 'Gastropoda',
    Habitat: 'Tinggal di daerah dataran tinggi, daerah pegunungan, tempat panas dan dingin',
```

```
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Siput. Siput adalah Gastropoda. Mereka Tinggal di daerah dataran tinggi, daerah pegunungan, tempat panas dan dingin. Siput adalah Gastropoda liar yang tidak berbahay atau tingkat berbahaya nya rendah',
  },
  Sheep:{
    Name: 'Sheep',
    Class: 'Mammalia',
    Habitat: 'Farmland, grassland, pasture, and other similar habitats with plenty of grass and plants to feed on.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Sheep. Sheep are Mammals. They live in farmland, grassland, pasture, and other similar habitats with plenty of grass and plants to feed on. Sheep are harmless mammals and not dangerous',
  },
  SheepIND:{
    Name: 'Domba',
    Class: 'Mamalia',
    Habitat: 'Tanah pertanian, padang rumput, padang rumput, dan habitat serupa lainnya dengan banyak rumput dan tanaman untuk dimakan',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Domba. Domba adalah mamalia. Mereka hidup Di tanah pertanian, padang rumput, padang rumput, dan habitat serupa lainnya dengan banyak rumput dan tanaman untuk dimakan. Domba adalah mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah',
  },
  Zebra:{
    Name: 'Zebra',
    Class: 'Mammalia',
    Habitat: 'The treeless grasslands and woodlands of eastern and southern Africa',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Zebra. Zebras are mammals. They live in the treeless grasslands and woodlands of eastern and southern Africa. Zebras are harmless mammals and not dangerous',
  },
  ZebraIND:{
    Name: 'Zebra',
    Class: 'Mamalia',
    Habitat: 'Padang rumput dan hutan tanpa pohon di Afrika timur dan selatan',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Zebra. Zebra adalah mamalia. Mereka hidup Di padang rumput dan hutan tanpa pohon di Afrika timur dan selatan. Zebra adalah mamalia liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Whale:{
    Name: 'Whale',
    Class: 'Mammalia',
    Habitat: 'Living in large waters including oceans',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Whale. Whales are Mammals. They Living in large waters including oceans. Whales are harmless large marine mammals and not dangerous',
  },
  WhaleIND:{
    Name: 'Paus',
    Class: 'Mamalia',
    Habitat: 'Di mana-mana mulai dari gurun ke hutan hujan ke halaman belakang dan segala sesuatu di antaranya',
    Status: 'TIDAK BERBAHAYA',
```

textSpeech: 'Paus. Ikan paus adalah mamalia. Mereka hidup Di perairan besar termasuk samudra. Paus adalah mamalia besar di laut yang tidak berbahaya atau tingkat berbahaya nya rendah.',
},
Shark:{
Name: 'Shark',
Class: 'Chondrichthyes',
Habitat: 'Inhabit shallow, coastal regions, while others live in deep waters, on the ocean floor and in the open ocean.',
Status: 'DANGEROUS',
textSpeech: 'Shark. Sharks are Chondrichthyes. They live inhabit shallow, coastal regions, while others live in deep waters, on the ocean floor and in the open ocean. Sharks are dangerous Chondrichthyes',
},
SharkIND:{
Name: 'Hiu',
Class: 'Chondrichthyes',
Habitat: 'Perairan dangkal, wilayah pesisir, sementara yang lain hidup di perairan dalam, di dasar lautan dan di lautan terbuka.',
Status: 'BERBAHAYA',
textSpeech: 'Hiu. Ikan Hiu adalah Chondrichthyes. Mereka hidup Di Perairan dangkal, wilayah pesisir, sementara yang lain hidup di perairan dalam, di dasar lautan dan di lautan terbuka. Ikan Hiu adalah Chondrichthyes yang sangat berbahaya',
},
Lizard:{
Name: 'Lizard',
Class: 'Reptilia',
Habitat: 'Deserts, forests, prairies, marshes, and rocky areas. Most lizards live on the ground or in trees.',
Status: 'NOT DANGEROUS',
textSpeech: 'Lizard. lizard is a reptile. They live in Deserts, forests, prairies, marshes, and rocky areas. Most lizards live on the ground or in trees. Lizard is a reptile that is not dangerous for some species',
},
LizardIND:{
Name: 'Kadal',
Class: 'Reptil',
Habitat: 'Gurun, hutan, padang rumput, rawa-rawa, dan daerah berbatu. Kebanyakan kadal hidup di tanah atau di pohon.',
Status: 'TIDAK BERBAHAYA',
textSpeech: 'Kadal. Kadal adalah Reptil. Mereka hidup Di Gurun, hutan, padang rumput, rawa-rawa, dan daerah berbatu. Kebanyakan kadal hidup di tanah atau di pohon. Kadal adalah reptil yang tidak berbahaya untuk sebagian jenis',
},
Dragonfly:{
Name: 'Dragonfly',
Class: 'Insecta',
Habitat: 'Around water such as lakes, ponds, streams and wetlands',
Status: 'NOT DANGEROUS',
textSpeech: 'Dragonfly. Dragonflies are insects. They live Around water such as lakes, ponds, streams and wetlands. Dragonflies are harmless Insects and not dangerous',
},
DragonflyIND:{
Name: 'Capung',
Class: 'Serangga',
Habitat: 'Di sekitar air seperti danau, kolam, aliran dan lahan basah',
Status: 'TIDAK BERBAHAYA',

```
    textSpeech: 'Capung. Capung adalah serangga. Mereka hidup Di sekitar air seperti danau, kolam, aliran dan lahan basah. Capung adalah serangga liar yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Jellyfish:{
    Name: 'Jellyfish',
    Class: 'Scyphozoa',
    Habitat: 'Any Ocean, cold arctic waters and others live in the warm tropical waters.',
    Status: 'DANGEROUS',
    textSpeech: 'Jellyfish. Jellyfish are Scyphozoa. They live in Any Ocean, cold arctic waters and others live in the warm tropical waters. Jellyfish are dangerous Scyphozoa',
  },
  JellyfishIND:{
    Name: 'Ubur-ubur',
    Class: 'Scyphozoa',
    Habitat: 'Semua samudra, perairan arktik dingin, dan lainnya hidup di perairan tropis yang hangat',
    Status: 'BERBAHAYA',
    textSpeech: 'Ubur-ubur. Ubur-ubur adalah scyphozoa. Mereka hidup Di Semua samudra, perairan arktik dingin, dan lainnya hidup di perairan tropis yang hangat. Laba-Laba adalah Arachnida yang cukup berbahaya bagi beberapa spesies tertentu.',
  },
  Lobster:{
    Name: 'Lobster',
    Class: 'Malacostraca',
    Habitat: 'Muddy, sandy, rocky substrates, burrows, rocky crevices, or underwater caves.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Lobster. Lobsters are Malacostraca. They live in Muddy, sandy, rocky substrates, burrows, rocky crevices, or underwater caves.  Lobsters are Malacostraca and not dangerous',
  },
  LobsterIND:{
    Name: 'Lobster',
    Class: 'Malacostraca',
    Habitat: 'Tanah berlumpur, berpasir, berbatu, lubang, celah berbatu, atau gua bawah air.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Lobster. Lobster adalah Malacostraca. Mereka hidup Di Tanah berlumpur, berpasir, berbatu, lubang, celah berbatu, atau gua bawah air. Lobster adalah Malacostraca yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Leopard:{
    Name: 'Leopard',
    Class: 'Mammalia',
    Habitat: 'Rainforests, deserts, woodlands, grassland savannas, forests, mountain habitats, coastal scrubs, shrub lands and swampy areas',
    Status: 'DANGEROUS',
    textSpeech: 'Leopard. Leopards are Mammals. They live in rainforests, deserts, woodlands, grassland savannas, forests, mountain habitats, coastal scrubs, shrub lands and swampy areas. Leopards are wild animals and very dangerous',
  },
  Leopard:{
    Name: 'Macan Tutul',
    Class: 'Mamalia',
    Habitat: 'Hutan hujan, gurun, padang rumput padang rumput, hutan, habitat gunung, semak-semak pantai, tanah semak dan daerah berawa',
    Status: 'BERBAHAYA',
```

textSpeech: 'Macan Tutul. Macan tutul adalah mamalia. Mereka hidup Di hutan hujan, gurun, hutan, padang rumput padang rumput, habitat gunung, semak-semak pantai, tanah semak dan daerah berawa. Macan tutul adalah mamalia liar yang sangat berbahaya',
},
Tiger:{
Name: 'Tiger',
Class: 'Mammalia',
Habitat: 'Tropical forests, evergreen forests, woodlands and mangrove swamps to grasslands, savannah and rocky country',
Status: 'DANGEROUS',
textSpeech: 'Tiger. Tigers are Mammalia. They live in tropical forests, evergreen forests, woodlands and mangrove swamps to grasslands, savannah and rocky country. Tigers are wild animals and very dangerous',
},
TigerIND:{
Name: 'Harimau',
Class: 'Mamalia',
Habitat: 'Hutan tropis, hutan hijau, hutan dan rawa-rawa bakau ke padang rumput, sabana dan negara berbatu',
Status: 'BERBAHAYA',
textSpeech: 'Harimau. Harimau adalah mamalia. Mereka hidup Di hutan tropis, hutan hijau, hutan dan rawa-rawa bakau ke padang rumput, sabana dan negara berbatu. Harimau adalah binatang buas yang sangat berbahaya',
},
Flamingo:{
Name: 'Flamingo',
Class: 'Aves',
Habitat: 'Mangrove swamps, tidal flats, and sandy islands in the intertidal zone',
Status: 'NOT DANGEROUS',
textSpeech: 'Flamingo. Flamingos are Aves. They live in mangrove swamps, tidal flats, and sandy islands in the intertidal zone. Flamingos are harmless aves and not dangerous',
},
FlamingoIND:{
Name: 'Flamingo',
Class: 'Aves',
Habitat: 'Rawa-rawa bakau, dataran pasang surut, dan pulau-pulau berpasir di zona pasang surut',
Status: 'TIDAK BERBAHAYA',
textSpeech: 'Flamingo. Flamingo adalah Aves. Mereka hidup Di Rawa-rawa bakau, dataran pasang surut, dan pulau-pulau berpasir di zona pasang surut. Flamingo adalah binatang Aves liar yang tidak berbahaya',
},
Peacock:{
Name: 'Peacock',
Class: 'Aves',
Habitat: 'Bushlands and rainforests. Many will nest on the ground while some will roost in trees.',
Status: 'NOT DANGEROUS',
textSpeech: 'Peacock. Peacos are Aves. They live in Bushlands and rainforests. Many will nest on the ground while some will roost in trees. Peacocks are harmless aves and not dangerous',
},
PeacockIND:{
Name: 'Merak',
Class: 'Aves',

```
    Habitat: 'Lahan hutan dan hutan hujan. Banyak yang akan bersarang di tanah sementara beberapa akan bertengger di pohon.',
    Status: 'Tidak BERBAHAYA',
    textSpeech: 'Merak. Burung Merak adalah Aves. Mereka hidup Di Lahan hutan dan hutan hujan. Banyak yang akan bersarang di tanah sementara beberapa akan bertengger di pohon. Burung merak adalah binatang Aves liar yang tidak berbahaya.',
  },
  Parrot:{
    Name: 'Parrot',
    Class: 'Aves',
    Habitat: 'Woodlands, rainforests, palm forests, savannas, grasslands, desert edges, scrubland, and more.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Parrot. Parrots are Aves. They live in Bushlands and rainforests. Many will nest on the ground while some will roost in trees. Parrots are harmless aves and not dangerous',
  },
  ParrotIND:{
    Name: 'Burung Beo',
    Class: 'Aves',
    Habitat: 'Hutan, hutan hujan, hutan palem, sabana, padang rumput, tepi gurun, semak belukar, dan banyak lagi.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Burung Beo. Burung Beo adalah Aves. Mereka hidup Di hutan, hutan hujan, hutan palem, sabana, padang rumput, tepi gurun, semak belukar, dan banyak lagi. Burung Beo adalah binatang Aves liar yang tidak berbahaya.',
  },
  Shrimp:{
    Name: 'Shrimp',
    Class: 'Malacostraca',
    Habitat: 'Near the seafloor of most coasts and estuaries, as well as in rivers and lakes.',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Shrimp. Shrimps are Malacostraca. They live near the seafloor of most coasts and estuaries, as well as in rivers and lakes. Shrimps are harmless Malacostraca and not dangerous',
  },
  ShrimpIND:{
    Name: 'Udang',
    Class: 'Malacostraca',
    Habitat: 'Dekat dasar laut dari sebagian besar pantai dan muara, juga di sungai dan danau.',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Udang. Udang adalah Malacostraca. Mereka hidup Di Dekat dasar laut dari sebagian besar pantai dan muara, juga di sungai dan danau. Udang adalah binatang Malacostraca yang tidak berbahaya atau tingkat berbahaya nya rendah',
  },
  Beetle:{
    Name: 'Beetle',
    Class: 'Insecta',
    Habitat: 'Dry soil, moist, hot, cold but also deep in the ground, in trees, inside fruit, inside seeds, inside leaves, inside dead wood, inside live plants and in carcasses',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Beetle. Beetles are Insects. They live in Dry soil, moist, hot, cold but also deep in the ground, in trees, inside fruit, inside seeds, inside leaves, inside dead wood, inside live plants and in carcasses. Beetles are harmless insects and not dangerous.',
  },
```

```
  BeetleIND:{
    Name: 'Kumbang',
    Class: 'Serangga',
    Habitat: 'Tanah kering, lembab, panas, dingin tetapi juga jauh di dalam tanah, di pohon, di dalam buah, di dalam biji, di dalam daun, di dalam kayu mati, di dalam tanaman hidup dan di bangkai',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Kumbang. Kumbang adalah serangga. Mereka hidup Di Tanah kering, lembab, panas, dingin tetapi juga jauh di dalam tanah, di pohon, di dalam buah, di dalam biji, di dalam daun, di dalam kayu mati, di dalam tanaman hidup dan di bangkai. Kumbang adalah serangga yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Caterpillar:{
    Name: 'Caterpillar',
    Class: 'Insecta',
    Habitat: 'Almost everywhere from sandy beaches to meadows to mountain forests, worldwide. There are even caterpillars in some Arctic areas',
    Status: 'NOT DANGEROUS',
    textSpeech: 'Caterpillar. Caterpillars are insects. They live in almost everywhere from sandy beaches to meadows to mountain forests, worldwide. There are even caterpillars in some Arctic areas. Caterpillar are harmless insects and not dangerous.',
  },
  CaterpillarIND:{
    Name: 'Ulat',
    Class: 'Serangga',
    Habitat: 'Hampir di mana-mana dari pantai berpasir ke padang rumput ke hutan gunung, di seluruh dunia. Bahkan ada ulat bulu di beberapa daerah Arktik',
    Status: 'TIDAK BERBAHAYA',
    textSpeech: 'Ulat. Ulat adalah Serangga. Mereka hidup hampir di mana-mana dari pantai berpasir ke padang rumput ke hutan gunung, di seluruh dunia. Bahkan ada ulat bulu di beberapa daerah Arktik. Ulat adalah serangga kecil yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
  Hippopotamus:{
    Name: 'Hippopotamus',
    Class: 'Mammalia',
    Habitat: 'Rivers, swamps, lakes and sometimes can be found on land.',
    Status: 'DANGEROUS',
    textSpeech: 'Hippopotamus. Hippopotamus are Mammals. They live in Rivers, swamps, lakes and sometimes can be found on land. Hippopotamus are wild animals and very dangerous',
  },
  HippopotamusIND:{
    Name: 'Kuda Nil',
    Class: 'Mamalia',
    Habitat: 'Sungai, rawa, danau dan kadang dapat ditemukan di daratan.',
    Status: 'BERBAHAYA',
    textSpeech: 'Kuda Nil. Kuda Nil adalah mamalia. Mereka hidup Di sungai, rawa, danau dan kadang dapat ditemukan di daratan. Kuda Nil adalah binatang liar yang sangat buas dan agresif',
  },
  Camel:{
    Name: 'Camel',
    Class: 'Mammalia',
    Habitat: 'Deserts, where it is hot and dry. Camels have adapted and found ways to help them survive in deserts.',
    Status: 'NOT DANGEROUS',
```

```
    textSpeech: 'Camel. Camels are Mammals. They live in Deserts, where it is hot and dry. Camels have adapted and found ways to help them survive in deserts. Camels are harmless mammals and not dangerous',
  },
  CamelIND:{
    Name: 'Unta',
    Class: 'Mamalia',
    Habitat: 'gurun, di mana panas dan kering. Unta telah beradaptasi dan menemukan cara untuk membantu mereka bertahan hidup di gurun.',
    Status: 'Tidak BERBAHAYA',
    textSpeech: 'Unta. Unta adalah mamalia. Mereka hidup Di gurun, di tempat panas dan kering. Unta adalah binatang mamalia yang tidak berbahaya atau tingkat berbahaya nya rendah.',
  },
};
```

styles.js

```
import { StyleSheet, Dimensions } from 'react-native';
const { width: winWidth, height: winHeight } = Dimensions.get('window');
export default StyleSheet.create({
  safeView:{
    flex: 1,
    backgroundColor: 'black'
  },
  safeViewPreviewImage:{
    flex: 1,
    backgroundColor: 'gray'
  },
  topToolbar:{
    flex: 1,
    position: 'absolute',
    zIndex: 2,
    flexDirection: 'row',
    marginTop: 35,
  },
  preview: {
    width: 560,
    height: 747,
    position: 'absolute',
    left: 0,
    top: 100,
    right: 0,
    bottom: 0,
  },
  previewImageCamera: {
    flex: 1,
    width: '100%',
    height: '100%',
    marginBottom: 10,
  },
  scanningAnimation: {
    position: 'absolute',
    alignItems: 'center',
    justifyContent: 'center',
    left: 0,
    top: 0,
```

```
    opacity: 0.4,
    right: 0,
    bottom: 50,
},
alignCenter: {
    flex: 1,
    marginTop: 20,
    alignItems: 'center',
    justifyContent: 'center',
},
backgroundTopToolbar: {
    backgroundColor: 'black',
    opacity: 0.3,
    zIndex: 2,
    width: winWidth,
    position: 'absolute',
    height: 100,
    top: 0,
},
backgroundBottomToolbar: {
    backgroundColor: 'black',
    opacity: 0.3,
    width: winWidth,
    position: 'absolute',
    height: 150,
    bottom: 0,
},
bottomToolbar: {
    width: winWidth,
    position: 'absolute',
    height: 150,
    bottom: 0,
},
captureBtn: {
    width: 60,
    height: 60,
    borderWidth: 2,
    borderRadius: 60,
    borderColor: "#FFFFFF",
},
captureBtnActive: {
    width: 80,
    height: 80,
},
captureBtnInternal: {
    width: 76,
    height: 76,
    borderWidth: 2,
    borderRadius: 76,
    backgroundColor: "red",
    borderColor: "transparent",
},
galleryBackground:{
    height: winHeight + 100,
    width: winWidth,
    position: 'absolute',
    zIndex: 1,
},
```

```
galleryContainerBackground:{
    margin: 20,
    marginTop: 70,
    height: winHeight + 100,
    zIndex: 0,
    backgroundColor: 'white',
    opacity: 0.5,
},
galleryContainer:{
    position: 'absolute',
    margin: 20,
    marginTop: 70,
    height: winHeight + 100,
    width: winWidth - 40,
    zIndex: 1,
    opacity: 1,
    flex:1,
},
galleryContentContainer:{
    margin: 10,
    height: winHeight + 100,
    zIndex: 1,
    borderColor: 'black',
    borderWidth: 3,
    opacity: 1,
    padding: 10,
    flex:1,
},
galleryTitleContainer:{
    justifyContent: 'center',
    marginBottom: 20,
},
galleryTextTitle:{
    textAlign: 'center',
    fontWeight: 'bold',
},
galleryCollectionContainer:{
    margin: 10,
    borderColor: 'black',
    borderWidth: 2,
    flex:1,
},
spinnerTextStyle: {
    color: '#FFF'
},
zoomableView: {
  padding: 10,
  backgroundColor: '#fff',
  width: 390,
  height: 580,
  position: 'absolute',
  left: 0,
  top: 100,
  right: 0,
  bottom: 0,
},
caption: {
    fontSize: 10,
```

```
        marginBottom: 5,
        color: '#444',
        alignSelf: 'center',
    },
    toolbarZoom:{
        flex: 1,
        flexDirection: "row",
        width: winWidth,
        position: 'absolute',
        height: 20,
        bottom: 120,
    },
    previewBackground:{
        height: winHeight + 100,
        width: winWidth,
        position: 'absolute',
        zIndex: 0,
    },
    descriptionAnimal:{
        padding: 20,
        alignItems: 'center',
        justifyContent: 'center',
    }
});
```