BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori adalah penjelasan mengenai sumber-sumber atau kajian dari teori-teori yang mendukung dan mendasari aplikasi yang akan dibangun. Landasan teori yang mendasari pembangunan aplikasi KYC (*Know Your Customer*) Digital untuk Mencegah Penipuan pada *Fintech Lending* Memanfaatkan API Clafifai dan BlinkID Android SDK, meliputi pembahasan *Fintech Lending*, Prinsip KYC, Aplikasi, Android, Api Clarifai, BlinkID, Java, Algoritma C4.5, UML, dan MySQL.

2.1.1 *Fintech Lending*

Fintech lending dalam bahasa indonesia berarti layanan pinjam meminjam uang berbasis teknologi informasi, yang selanjutnya dalam Peraturan Otoritas Jasa Keuangan Nomor 77/POJK.01/2016 tentang Pinjam Meminjam Uang Berbasis Teknologi Informasi, menjelaskan bahwa pengertian layanan pinjam meminjam uang berbasis teknologi informasi adalah layanan jasa keuangan untuk mempertemukan pemberi pinjaman dengan penerima pinjaman dalam rangka melakukan perjanjian pinjam meminjam dalam mata uang rupiah secara langsung melalui sistem elektronik dengan menggunakan jaringan internet.

Dari pengertian di atas, dapat diuraikan unsur-unsur yang membangun sistem *fintech lending* adalah sebagai berikut:

2.1.1.1 Sistem Elektronik

Sistem elektronik adalah serangkaian perangkat dan prosedur elektronik yang berfungsi mempersiapkan, mengumpulkan, mengolah, menganalisis, menyimpan, menampilkan, mengumumkan, mengirimkan, dan/atau menyebarkan informasi elektronik di bidang layanan jasa keuangan.

2.1.1.2 Teknologi Informasi

Teknologi informasi dalah suatu teknik untuk mengumpulkan, menyiapkan, menyimpan, memproses, mengumumkan, menganalisis, dan/atau menyebarkan informasi di bidang layanan jasa keuangan.

2.1.1.3 Penyelenggara Layanan Pinjam Meminjam Uang Berbasis Teknologi Informasi

Penyelenggara adalah badan hukum Indonesia yang menyediakan, mengelola, dan mengoperasikan Layanan Pinjam Meminjam Uang Berbasis Teknologi Informasi. Badan hukum penyelenggara berbentuk:

1. Perseroan terbatas; atau
2. Koperasi.

2.1.1.4 Penerima Pinjaman

Penerima pinjaman adalah orang dan/atau badan hukum yang mempunyai utang karena perjanjian Layanan Pinjam Meminjam Uang Berbasis Teknologi Informasi. Penerima pinjaman harus berasal dan berdomisili di wilayah hukum Negara Kesatuan Republik Indonesia. Penerima pinjaman terdiri dari:

1. Orang perseorangan warga negara Indonesia; atau
2. Badan hukum Indonesia.

2.1.1.5 Pemberi Pinjaman

Pemberi pinjaman adalah orang, badan hukum, dan/atau badan usaha yang mempunyai piutang karena perjanjian Layanan Pinjam Meminjam Uang Berbasis Teknologi Informasi. Pemberi pinjaman dapat berasal dari dalam dan/atau luar negeri. Pemberi pinjaman terdiri dari:

1. Orang perseorangan warga negara Indonesia;
2. Orang perseorangan warga negara asing;
3. Badan hukum Indonesia/asing;
4. Badan usaha Indonesia/asing; dan/atau
5. Lembaga internasional.

2.1.1.6 Pengguna Layanan Pinjam Meminjam Uang Berbasis Teknologi Informasi

Pengguna adalah pemberi pinjaman dan penerima pinjaman yang menggunakan layanan pinjam meminjam uang berbasis teknologi informasi.

2.1.1.7 Dokumen Elektronik

Perjanjian penyelenggaraan Layanan Pinjam Meminjam Uang Berbasis Teknologi Informasi Dokumen elektronik adalah setiap informasi elektronik yang dibuat, diteruskan, dikirimkan, diterima, atau disimpan dalam bentuk analog, digital, elektromagnetik, optikal, atau sejenisnya, yang dapat dilihat, ditampilkan, dan/atau didengar melalui komputer atau sistem elektronik termasuk tetapi tidak terbatas pada tulisan, suara, gambar, peta perancangan, foto atau sejenisnya, huruf, tanda, angka, kode akses, simbol atau perforasi yang memiliki makna ayau arti atau dapat dipahami oleh orang yang mampu memahaminya sebagaimana dimaksud dalam Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik.

2.1.2 Prinsip KYC

Know Your Customer Principle (KYCP) yang selanjutnya disebut dengan prinsip KYC adalah prinsip yang diterapkan bank untuk mengetahui identitas nasabah, memantau kegiatan transaksi nasabah, termasuk pelaporan transaksi yang mencurigakan dan sudah menjadi kewajiban bank untuk menerapkannya [3].

Prinsip KYC dirasa perlu diterapkan pada perusahaan *fintech*, mengingat fungsi *fintech* menyangkut salah satu fungsi bank sebagai penerima pinjaman. Tentunya fungsi pemberi pinjaman ini dapat disalahgunakan oleh penerima pinjaman, atau dalam dunia perbankan disebut sebagai nasabah, sebagai sarana penipuan dan/atau penggelapan uang. Dengan demikian, guna mencegah terjadinya upaya penipuan atau penggelapan uang, diperlukanlah prinsip mengenal pengguna jasa atau KYC sebagaimana yang tertuang dalam Undang-

Undang Nomor 8 Tahun 2010 tentang Pencegahan dan Pemberantasan Tindak Pidana Pencucian Uang.

Lebih lanjut, proses KYC yang dijelaskan dalam Undang-Undang Nomor 8 Tahun 2010 tentang Pencegahan dan Pemberantasan Tindak Pidana Pencucian Uang Pasal 18 Ayat 5 disebutkan bahwa prinsip mengenal pengguna jasa harus memuat identifikasi pengguna jasa, verifikasi pengguna jasa, dan pemantauan pengguna jasa.

Identifikasi pengguna jasa atau penerima pinjaman merupakan proses utama dan pertama dalam penerapan prinsip KYC. Dimana sebelum melakukan hubungan usaha dengan penerima pinjaman, *fintech* wajib meminta informasi mengenai identitas calon penerima pinjaman, maksud dan tujuan hubungan usaha yang akan dilakukan calon penerima pinjaman dengan perusahaan, serta informasi lain yang memungkinkan perusahaan untuk dapat mengetahui profil calon penerima pinjaman. Dalam Peraturan Bank Indonesia Nomor 3/10/PBI/2001 Tentang Penerapan Prinsip Mengenal Nasabah (*Know Your Customer Principles*) Pasal 5, identifikasi penerima pinjaman/Nasabah bagi nasabah perorangan sekurang-kurangnya terdiri dari:

1) Identitas Nasabah yang memuat:
   (i) Nama;
   (ii) Alamat tinggal tetap;
   (iii) Tempat dan tanggal lahir;
   (iv) Kewarganegaraan;
1) Keterangan mengenai pekerjaan;
2) Spesimen tanda tangan; dan
3) Keterangan mengenai sumber dana dan tujuan penggunaan dana

Proses selanjutnya dalam KYC adalah verifikasi pengguna jasa. Prinsip KYC dalam perbankan dilakukan secara langsung (*face to face*), namun mengingat bahwa *fintech* merupakan perusahanaan pinjam meminjam secara online, maka prinsip KYC yang digunakan adalah *E-KYC* atau *Electronic Know Your Customer*, dimana dalam proses verifikasi diselesaikan secara *online* dan

real time dengan otoritas langsung dari pelanggannya tanpa proses tatap muka langsung. Dalam penerapan E-KYC bagi setiap perusahaan *fintech lending* bisa beragam. Mulai dari panggilan video, mengirimkan foto wajah, memanfaatkan data kependudukan lewat KTP elektronik, hingga menggunakan tanda tangan digital.

Untuk proses terakhir dalam penerapan prinsip KYC adalah pemantauan pengguna jasa atau penerima pinjaman. Hal ini dapat berupa:

1) Menatausahakan dokumen-dokumen penerima pinjaman dalam jangka waktu sekurang-kurangnya 5 tahun.
2) Melakukan pengkinian data jika terdapat perubahan terhadap dokumen-dokumen penerima pinjaman.
3) Memiliki sistem informasi yang dapat mengidentifikasi, menganalisa, memantau dan menyediakan laporan secara efektif mengenai karakteristik transaksi yang dilakukan oleh penerima pinjaman.

2.1.3 Aplikasi

Aplikasi berasal dari kata *application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju [4]

Menurut Sri Widianti pengertian aplikasi merupakan sebuah *software* (perangkat lunak) yang bertugas sebagai *front end* pada sebuah sistem yang dipakai untuk mengelola berbagai macam data sehingga menjadi sebuah informasi yang bermanfaat untuk penggunaannya dan juga sistem yang berkaitan [5]. Pengertian lainnya menurut Harip Santoso, aplikasi merupakan sebuah kelompok file (*class, form, report*) yang ditujukan sebagai pengeksekusi aktivitas tertentu yang saling berkaitan[6].

Berdasarkan beberapa pengertian aplikasi yang telah dijabarkan di atas, maka dapat ditarik kesimpulan bahwa aplikasi adalah suatu program yang melaksanakan atau mengeksekusi suatu fungsi atau aktivitas tertentu bagi pengguna atau aplikasi lain. Dalam penelitian ini dibuat suatu aplikasi KYC

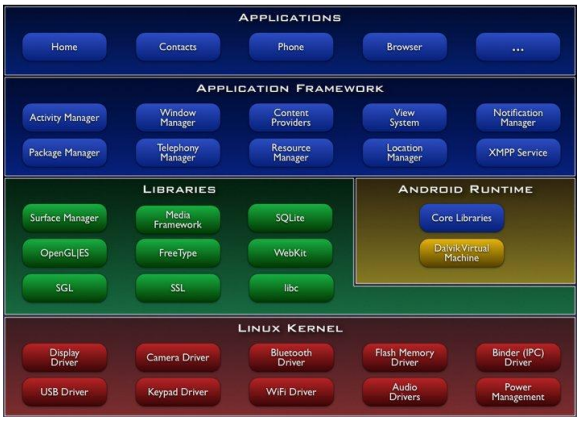
dengan tujuan agar dapat memudahkan proses klasifikasi nasabah penerima pinjaman dimana pengujian-pengujian yang dilakukan dieksekusi oleh berbagai fungsi seperti fungsi deteksi wajah, pindai KTP, dan algoritma C4.4, hal ini sesuai dengan definisi dan fungsi aplikasi.

2.1.4 Android

Android adalah *software* besutan Google yang mencakup sistem operasi atau *middleware*, dan aplikasi kunci yang berbasis Linux dan dirancang untuk perangkat mobile seperti telepon seluler, *smartphone*, dan komputer *tablet* [7]. Android awalnya dikembangkan oleh Android, Inc., kemudian dibeli Google pada tahun 2005 dan dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya *Open Handset Alliace*, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ada dua jenis distributor sistem operasi Android, yaitu yang mendapat dukungan penuh dari *Google* atau *Google Mail Service* (GMS) dan kedua adalah yang bebas distribusinya tanpa dukungan langsung dari Google atau dikenal sebagai *Open Handset Distribution* (OHD) [8].

2.1.4.1 Arsitektur Android

Secara garis umum arsitektur android dapat digambarkan sebagai berikut:



Sumber Gambar: Radliya, Nizar Rabbi. 2015. Pemrograman Mobile (Android). Unikom.

Gambar 2.1 Arsitektur Android

Berdasarkan sumber [8], penjelasan dari gambar arsitektur di atas dapat diuraikan berikut ini:

1. Applications dan Widgets

   Application dan widgets ini adalah layer penghubung antara pengguna dengan aplikasi. Dengan proses pertama *download*, kemudian instalasi dan penggunaan. Di layer terdapat aplikasi inti seperti *email*, program SMS, kalender, peta, *browser*, kontak, android market, pemutar video, dll. Semua aplikasi dibangun di atas *application layer* atau pada *library* API yang sama. Applicarion layer ini berjalan di dalam android *runtime*, menggunakan *class* dan layanan yang dibuat dari *application framework*.

2. Application Framework

   Application framewok adalah layer pengembangan/pembangunan dari aplikasi yang dapat dijalankan pada sistem operasi android. Dimana pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background* dan sebagainya guna membangun aplikasi yang bagus dan inovatif. Application framework menyediakan berbagai class yang digunakan untuk membuat aplikasi android. Termasuk berhubungan dengan akses *hardware* dan manajemen *user-interface* dan manajemen memori.

3. *Libraries*

   Libraries adalah *layer* tempat fitur-fitur android berada. Dimana para pembuat aplikasi dapat mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas kenel yang didalam kenel yang didalamnya terdapat kumpulan library C dan C++ seperti:

   a. *Library* media untuk pemutaran media audio dan video.
   b. *Library* untuk manajemen tampilan.

c. *Library Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
d. *Library* SQLite untuk dukungan *database*.
e. *Library* LiveWebcore mencakup *modern web browser* dengan engine embedded web view.
f. *Library* SSL dan WebKit untuk integrasi *web browser* dan keamanan internet.

4. Android Runtime

   Android runtime terdiri dari *Core Libraries* dan *Dalvik Virtual Machine* (DVM). Core libraries adalah serangkaian *library* java yang terdiri dari berbagai macam fungsi dasar pada bahasa pemrograman Java, sedangkan DVM merupakan *Java Virtual Machine* yang secara khusus dijalankan pada android.

5. Linux Kernel

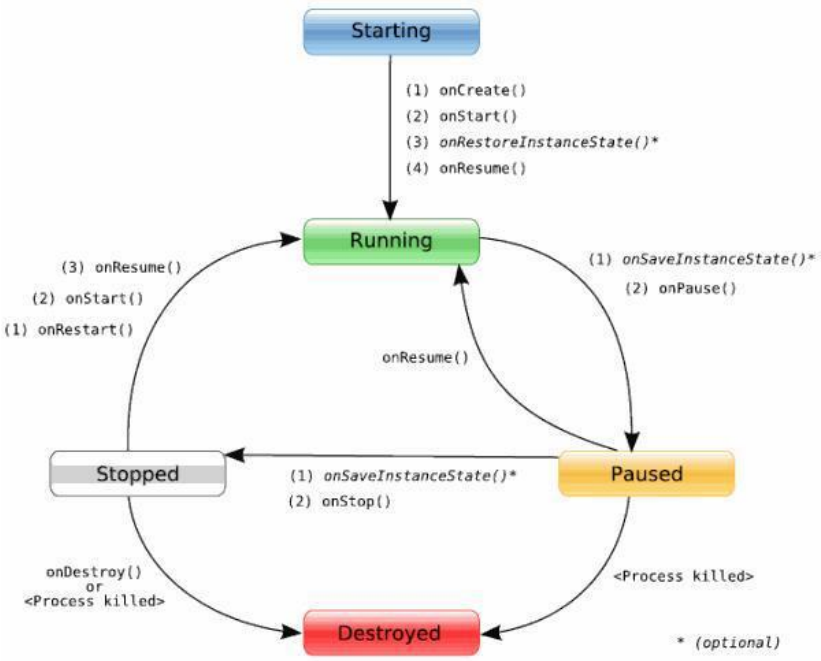
   Linur kernel adalah layer tempat sistem operasi android berada. Berisi file-file sistem yang mengatur sistem processing, memory, resource, drivers, dan sistem-sistem lainnya. Linux kernel yang digunakan android adalah linux kernel 2.6.

2.1.4.2 Komponen Android

Android memiliki beberapa jenis komponen didalamnya, dalam sumber [8] hal ini akan dijelaskan pada penjelasan berikut:

1. Activity

   Activity adalah kelas yang mempunyai siklus yang mewakili setiap *user intrerface*. Dimana sebuah aplikasi dapat terdiri dari satu atau lebih activity yang diproses dalam Linux. Alur *activity lifecycle* digambarkan di bawah ini:



Sumber Gambar: Radliya, Nizar Rabbi. 2015. Pemrograman Mobile (Android). Unikom.

Gambar 2.2 Activity Lifecycle

Dalam gambar di atas dapat dilihat beberapa status yang merupakan hasil kontrol dimana akan muncul pesan karena adanya perubahan status melalui method onXX(). Berikut penjelasan setiap status:

a. onCreate(Bundle), dipanggil saat pertama kali aplikasi dijalankan. Status ini digunakan untuk deklarasi cariabel atau membuat user interface.
b. onStart(), mengindikasikan *activity* yang ditampilkan ke pengguna (*user*).
c. onResume(), dipanggil saat aplikasi mulai berinteraksi dengan pengguna. Disini cocok untuk meletakkan animasi ataupun musik.
d. onPause(), dipanggil saat aplikasi yang dijalankan kembali ke halaman sebelumnya tau biasanya karena ada *activity* baru yang

dijalankan. Disini cocok untuk meletakkan algoritma penyimpanan (*save*).

e. onStop, dipanggil saat aplikasi berjalan di belakang layar dalam waktu cukup lama.
f. onRestart(), activity kembali menampilkan user interface setelah status stop.
g. onDestroy(), dipanggil saat aplikasi benar-benar berhenti.
h. onSaveInstanceState(Bundle), method ini mengizinkan activity untuk menyimpan setiap status intance.
i. onResotreInstanceState (Bundle), dipanggil saat activity kembali menginisialisasi dari status sebelumnya yang disimpan oleh onSaveInstanceState(Bundle).

2. Service

   Service tidak memiliki user interface, namun berjalan di belakang layar. Misalnya music player, sebuah activity digunakan untuk memilih lagu kemudian diputar/dimainkan. Agar music player bisa berjalan di belakang aplikasi lain maka harus menggunakan service.

3. Content Provider

   Content provider menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu activity. Dalam arti lain, content provider digunakan dalam aplikasi untuk sharing data (berhubungan dengan database).

4. Resource

   Resource digunakan untuk menyimpan file-file non-coding yang diperlukan pada sebuah aplikasi misalnya file icon, gambar, audio, video atau yang lain.

5. Views

   Digunakan untuk membangun antarmuka pengguna untuk komponen *activity* yang akan digunakan.

6. Notification

   Digunakan untuk mekanisme sinyal ke pengguna secara insidentil sesuai dengan waktu yang telah ditentukan. Contohnya pemberitahuan dalam bentuk getar atau lampu ketika zona waktu berubah, baterai *low*, dan lain-lain.

2.1.4.3 Android SDK

Android SDK (*Software Development Kit*) adalah *tools* API (Application Programming Interface) yang dipergunakan untuk mulai mengembangkan aplikasi pada platform android menggunakan bahasa pemrograman Java. Beberapa fitur android yang paling penting adalah:

1. *Messaging* (Pesan), SMS dan MMS yang tersedia dalam bentuk threaded pesan termasuk pesan teks.
2. *Connectivity* (Konektivitas) dalam hal ini termasuk GSM?EDGE, CDMA, EV-DO, UMTS, Bluetooth, dan Wi-Fi.
3. *Handset Layout.* Grafis yang dioptimalkan dan didukung oleh libraries grafis 2D, grafis 3D berdasarkan spesifikasi open GL ES 1.0 dan tata letak smartphone.
4. *Web Browser* berdasarkan pada *engine open source* Web Kit.
5. *Storage* (Penyimpanan), dalam hal ini software SQLite digunakan untuk penyimpanan data.
6. *Locksreen*. Untuk membuka layar (*lockscreen*) tersedia beberapa pilihan yaitu menggeser (*slide*), PIN, Password, pendeteksi wajah yang cukup keren dan canggih bahkan pemindai sidik jari.
7. Framework aplikasi yang mendukung penggantian komponen dan reusable.
8. Mesin virtual dalvik dioptimalkan untuk perangkat mobile..
9. Media support yang mendukung audio, video, dan gampar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM Telephony (tergantung hardware).
10. Kamera, GPS, kompas dan Accelerator (tergantung hardware).

11. Lingkungan development yang lengkap dan kaya, termask perangkat emulator, tools untuk debuging, profil dan kinerja memori, dan plug in untuk IDE Eclipse.

Alasan mengapa penelitian ini menggunakan aplikasi android adalah karena:

1. Market Share

   Adanya *android market* membuat aplikasi yang dibuat dapat digunakan dan diinstall oleh pengguna tanpa perlu mencarinya di internet. Hal ini tentunya akan sangat bermanfaat bagi perusahaan *fintech* selaku pengguna aplikasi.
2. Time to Market

   Adanya Android APIs (*Application Programming Interface*) dapat memberikan kemudahan untuk membangun aplikasi dengan mudah tanpa waktu yang begitu lama.
3. Open Platform

   Sistem operasi android merupakan platform terbuka yang memungkinkan perkembangan *market* dengan sangat cepat, karena aplikasi dapat dibuat dan dijual.
4. Cross Compatibility

   Android dapat berjalan di berbagai perangkat dengan ukuran dan resolusi yang disesuaikan dengan perangkat handphone pengguna.
5. Mashup Capability

   Android memungkinkan *developer* untuk mengkombinasikan atau menggabungkan dua atau lebih fitur dalam satu aplikasi.

2.1.5 API Clarifai

API Clarifai adalah produk platform yang dirilis oleh Clarifai Inc. – perusahaan kecerdasan buatan yang berspesialisasi dalam visi komputer dan menggunakan pembelajaran mesin dan jaringan saraf dalam mengidentifikasi dan menganalisis gambar dan video – pada tahun 2016. Inti dari teknologi clarifai

didasarkan pada jaringan saraf convolutional sehingga menjadi proses yang memungkinkan komputer untuk belajar dari contoh data dan menarik kesimpulannya sendiri, memberikan kemampuan untuk memprediksi suatu gambar atau video.

Aplikasi yang dibangun memanfaatkan API Clarifai karena platform ini mencakup kemampuan untuk memoderasi konten, melakukan pencarian visual, kesamaan visual, dan mengatur koleksi media. Selain itu, platform juga memiliki model pengenalan yang dibangun dengan kemampuan mengidentifikasi serangkaian konsep tertentu termasuk objek, ide, dan emosi dari suatu gambar atau video. Dengan kata lain, API Clarifai menawarkan pengenalan gambar dan video sebagai layanan dengan memanfaatkan teknologi kecerdasan buatan untuk mengenali konten visual dari gambar dan video yang diunggah oleh penggunanya.

2.1.5.1 Alur Kerja API Clarifai

Alur kerja API Clarifai ditunjukkan pada gambar berikut ini:

Sumber Gambar: https://clarifai.com/developer/guide/

Gambar 2.3 Alur Kerja API Clarifai

Alur kerja API Clarifai dimulai dari pengguna yang menginput/mengunggah konten visual (gambar atau video) kemudian sistem API Clarifai akan memodelkannya sesuai dengan input yang diberi dan mengidentifikasi konsep sesuai dengan hasil pemodelan yang kemudian dihasilkan suatu prediksi hasil kerja sistem kepada pengguna. Contohnya pada gambar 2.3 input yang diberikan adalah menu sarapan pengguna, kemudian sistem membaca input dan memodelkannya ke dalam model makanan, setelah itu sistem memberikan output berupa prediksi hasil identifikasi konsep dari model makanan tersebut kepada

pengguna bahwa prediksinya adalah menu sarapan pengguna terdiri dari hamburger dan french fries dengan probabilitas akurat hamburger 0,996 dan french fries 0,923.

2.1.5.2 Model API Clarifai

Model pengenalan gambar yang dibangun oleh API Clarifai terdiri dari berbagai macam sesuai dengan kebutuhan spesifikasi pengguna, yaitu:

1. Apparel
2. Celebrity
3. Color
4. Demographics
5. Face Detection
6. Face Embedding
7. Food
8. General
9. General Embedding
10. Moderation
11. NSFW
12. Texture and Patterns
13. Travel
14. Wedding

2.1.6 BlinkID

BlinkID adalah salah satu produk dengan memanfaatkan teknologi OCR seluler yang memungkinkan pengembangan aplikasi seluler dalam memecahkan masalah pengenalan teks menggunakan kamera perangkat seluler. BlinkID dibuat dan dikembangkan oleh Microblink yaitu perusahaan R&D yang mengembangkan teknologi visi komputer yang dioptimalkan untuk pemrosesan realtime di perangkat seluler.

BlinkID sebagai teknologi OCR dibuat dengan tujuan untuk menghilangkan kebutuhan entri data manual pada perangkat seluler dan

menggantinya dengan pemindaian melalui kamera perangkat seluler, sehingga meningkatkan pengalaman dan keterlibatan pengguna. BlinkID memungkinkan pengambilan data dalam berbagai kasus penggunaan seperti pembayaran seluler, onboarding pengguna baru dalam aplikasi, KYC, pelacakan pengeluaran, validasi tiket, check-in di bandara dan hotel, keamanan, aktivasi kartu loyalitas, pendaftaran kartu SIM, TOP UP, dan pendaftaran voting, serta kasus lainnya.

BlinkID memungkinkan pemindaian lebih dari 100 dokumen identitas internasional termasuk lisensi mengemudi, kartu identitas nasional, paspor dan lainnya. BlinkID mampu mengekstraksi informasi hasil pemindaian dari semua dokumen yang mengandung:

1. MRZ dan MRTD

   MRZ singkatan untuk Marchine Readable Zone, sedangkan MRTD yaitu mengacu pada dokumen yang dapat dibaca oleh mesin. MRZ adalah format dokumen yang berisi data pemegang dokumen dalam bentuk yang dapat di baca secara visual dan dikodekan dengan pengenalan karakter optik. MRZ dapat ditemukan di sebagian besar paspor dan dokumen identitas di seluruh dunia (KTP).

2. Barcode PDF417

   Selain dokumen yang mengandung MRZ dan MRTD, BlinkID juga dapat memindai dokumen yang mengandung barcode PDF417. PDF417 adalah format barcode dua dimensi yang paling umum digunakan karena merupakan cara yang efisien dan aman untuk menyimpan data dalam jumlah besar. Biasanya PDF417 digunakan pada sejumlah besar identitas dan komen perjalanan di seluruh dunia seperti SIM dan KTP.

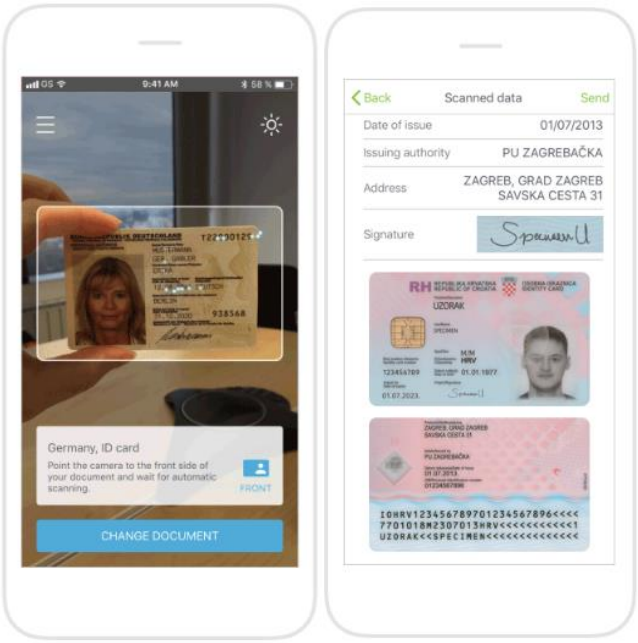
3. Dokumen Lainnya

   BlinkID juga mendukung ekstraksi data dari dokumen apapun yang memuat teks biasa dengan mengaktifkan template OCR khusus.

BlinkID dengan segala kecanggihan yang memanfaatkan teknologi OCR menyediakan berbagai fitur-fitur unggulan, antara lain:

1. Melakukan pemindaian secara cepat dan nyata.
2. Bekerja offline, secara lokal di perangkat yang tidak mempunyai koneksi internet.
3. Akurasi tinggi dalam mengekstraksi semua data.
4. Untuk beberapa negara, pemindaian kedua sisi kartu identitas didikung dengan perbandingan tambahan data dari depan dan belakang.
5. Ekstraksi berupa gambar dokumen seperti yang ada pada KTP dan tanda tangan pun tersedia.

   Berikut ini adalah contoh tampilan penggunaan scan BlinkID pada KTP:



Sumber Gambar: https://microblink.com/products/blinkid

Gambar 2.4 Contoh Tampilan Penggunaan BlinkID

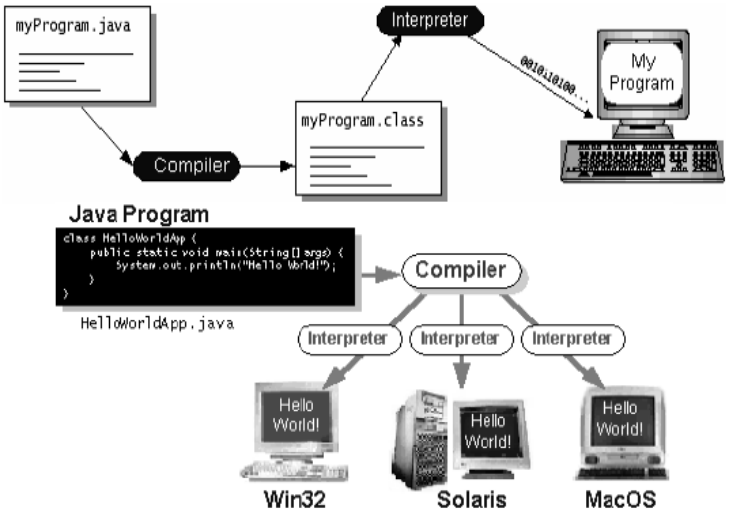
Aplikasi yang dibangun memanfaatkan BlinkID agar pengecekan identitas nasabah tidak harus dilakukan dengan entri data secara manual melainkan menggunakan pemindaian foto yang diupload dari galeri atau diambil langsung dengan kamera. Hal ini tentunya memberikan kemudahan dan mengatasi efektifitas serta efisiensi dalam entri data.

2.1.7 Java

Java adalah bahasa pemrogramana berorientasi objek murni yang dibuat berdasarkan kemampuan-kemampuan terbaik bahasa pemrograman objek sebelumnya (C++, Ada, Simula). Java diperkenalkan oleh James Gosling, seorang developer dari Sun Microsystem pada tahun 1991 [9].

Pengertian java menurut Sun Microsystem adalah sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *stand alone* ataupun pada lingkungan jaringan [10]. Java memiliki karakteristik yang membedakannya dengan bahasa pemrograman lainnya, antara lain adalah sederhana (*simple*), berorientasi objek (*object oriented*), terdistribusi (*distributed*), *interpreted*, *Robust*, Aman (*secure*), *architecture neutral*, *portable*, *performance*, *multithreaded*, dan dinamis.

Bahasa java bersifat case sensitive, hal ini berarti diperlukan kehati-hatian dalam penggunaan huruf besar dan kecil. Selain itu penulisan *source code* program tidak harus memperhatikan bentuk tertentu, sehingga bisa dituliskan semua baris *source code* dalam satu baris dengan tanda titik koma (;), atau menuliskan tiap kata dalam satu baris tersendiri. Cara kerja java dapat digambarkan sebagai berikut:



Sumber Gambar: Noviyanto, ST. Pemrograman Berorientasi Objek

Gambar 2.5 Cara Kerja Java

Java 2 adalah generasi kedua dari platform Java. Macam-macam Java 2 Software Developer Kit (J2SDK) antara lain J2SE (Jaca 2 Standard Edition), J2EE (Java 2 Enterprise Edition), dan J2ME (Java 2 Micro Edition). Adapun platform java terdiri dari Java Virtual Machine (Java VM) dan Java Application Programming Interface (Java API).

Adapun sintaks dalam bahasa pemrograman Java ditunjukkan sebagai berikut:

```
public class HelloWorld {
public static void main(String[] args) {
     System.out.println("Hello, World");
}
}
```

Menurut Warno dalam jurnalnya berjudul "Pembelajaran Pemrograman Bahasa Java dan Arti *Keyword*" [9], dalam menyusun struktur pemrograman dengan menggunakan Java, ditemui beberapa kelemahan dan keuntungan. Kelemahan menggunakan Java yaitu:

1. Mudah didekompilasi. Dekompilasi adalah proses membalikkan suatu kode jadi menjadi kode sumber. Hal ini dikarenakan kode jadi java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Dengan demikian ini membuat algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak.
2. Penggunaan memori yang banyak. Pengunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang semakin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

3. Implementasi J2ME tidak global. Contohnya, J2ME untuk Motorola dnegan J2ME untuk Sony Ericson tidak sama. Berbeda lagi J2ME untuk Nokia. Setiap produk selalu mempunyai modul tersendiri yang dinilai aneh penerapannya dan harus di-*compile* dengan modul yang berbeda-beda.

Hal yang membuat Java menjadi bahasa pemrograman dalam membangun aplikasi ini adalah dikarenakan Java memiliki beberapa keunggulan berikut ini:

1. Platform Independent. Hal ini berarti Java – baik source program maupun hasil kompilasinya – sama sekali tidak bergantung kepada sistem operasi dan platform yang digunakan.
2. Sederhana dan Berorientasi Objek. Konsep dasar dari teknologi Java dibuat dengan sederhana agar dapat dimengerti dengan mudah, dan programmer dapat segera menghasilkan sesuatu sedini mungkin. Java juga dipastikan bermula dari bahasa pemrograman dasar yang sudah ada sebelumnya dengan membuang berbagai fitur yang rumit dan membingungkan dari bahasa pemrograman sebelumnya itu.
3. Automatic Garbage Collection. Hal ini artinya pengumpulan sampah otomatis, dimana Java memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung. Fitu ini dapat membersihkan objek yang tidak terpakai dari memori.
4. Mengurangi *pointer* aritmetik. Hal ini berarti pengaksesan lokasi memori secara langsung dilakukan dengan menggunakan *pointer* yang memungkinkan program untuk melakukan suatu tindakan yang tidak seharusnya atau tidak boleh dilakukan. Untuk mengurangi dan menghilangkan kemungkinan seperti ini, penggunaan *pointer* pada Java telah dibatasi dengan menggunakan *reference*.
5. Library yang lengkap. Dimana Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program-program yang disertakan dalam pemrograman Java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya.

Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

6. Cocok untuk membangun program yang besar. Hal ini dikarenakan Java bersifat OOP (*Object Oriented Programming*) atau pemrograman berorientasi objek yang artinya semua aspek yang terdapat di Java adalah objek. Bahkan dapat dikatakan bahwa Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Dimana semua tipe data diturunkan dari kelas dasar yang disebut objek. Hal ini akhirnya sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah, dan terorganisir. Kelebihan ini menjadikan Java sebagia salah satu bahasa pemrograman termudah, bahkan untuk fungsi-fungsi yang *advance* seperti komunikasi antara komputer sekalipun.

2.1.8 Algoritma C4.5

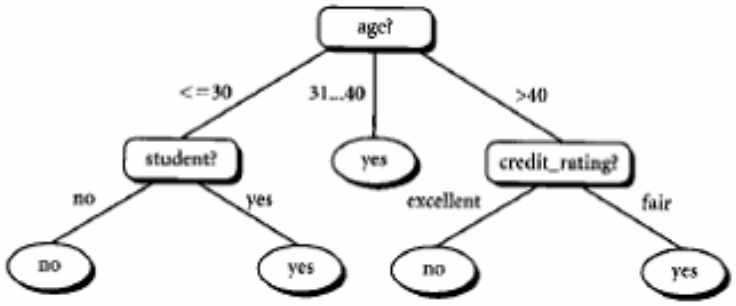
Algoritma C4.5 diciptakan oleh J.Rose Quinlan dan merupakan pengembangan dari algoritma ID3. Algoritma C4.5 merupakan algoritma yang sangat populer dan digunakan oleh banyak peneliti di dunia, hal ini dijelaskan oleh Xindong Wu dan Vipin Kumar dalam bukunya yang berjudul “The Top Ten Algorithms in Data Mining” [11].

Dalam jurnal berjudul “Klasifikasi Nasabah Menggunakan Algoritma C4.5 Sebagai Dasar Pemberian Kredit” yang ditulis oleh Larissa Navia Rani, pengertian Algortima C4.5 adalah salah satu teknik *decision tree* yang sering digunakan, yang menghasilkan beberapa aturan dan sebuah pohon keputusan dengan tujuan untuk meningkatkan keakuratan dari prediksi yang sedang dilakukan [12].

Adapun pohon keputusan adalah metode klasifikasi dimana model prediksi menggunakan struktur pohon atau struktur berhirarki. Konsep dari pohon

keputusan adalah mengubah data menjadi pohon keputusan dan aturan-aturan keputusan. Pohon tersebut juga memperlihatkan faktor-faktor kemungkinan/probabilitas yang akan mempengatuhi alternatif-alternatif keputusan sebagai pemecahan masalah yang dapat diambil, disertai dengan estimasi hasil akhir yang akan didapat bila mengambil alternatif keputusan tersebut [13]

Manfaat utama dari penggunaan pohon keputusan menurut sumber yang sama [13] adalah kemampuannya untuk mem-*break down* proses pengambilan keputusan yang kompleks menjadi lebih simpel sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan. Selain itu, pohon keputusan juga berguna untuk mengeksplorasi data, menemukan hubungan tersembunyi antara sejumlah calon variabel input dengan sebuah variabel target. Dari contoh pada Gambar 2.6, setiap percabangan pada pohon keputusan menyatakan kondisi yang harus dipenuhi dan tiap ujung pohon menyatakan kelas data.



Sumber: Pramudiono, 2008.

Gambar 2.6 Contoh Model Pohon Keputusan

Secara umum algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut [14]:

a. Pilih atribut sebagai akar
b. Buat cabang untuk tiap-tiap nilai
c. Bagi kasus dalam cabang
d. Ulangi proses untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama.

Sebuah objek yang diklasifikasikan dalam pohon keputusan harus dites nilai *entropy*-nya. *Entropy* adalah ukuran dari teori informasi yang dapat mengetahui karakteristik dari *impuryt* dan *homogenity* dari kumpulan data. Dari nilai *entropy* tersebut kemudian dihitung nilai *information gain* (IG) masing-masing atribut. *Entropy* (S) merupakan jumlah bit yang diperkirakan dibutuhkan untuk dapat mengekstrak suatu kelas (+ atau -) dari sejumlah data acak pada ruang sampel S. *Entropy* dapat dikatakan sebagai kebutuhan bit untuk menyatakan suatu kelas. Semakin kecil nilai *Entropy* maka akan semakin sering *entropy* digunakan dalam mengekstrak suatu kelas. *Entropy* digunakan untuk mengukur ketidakaslian sistem informasi atau disebut dengan *processing system* [15].

Besarnya *entropy* pada ruang sampel S didefinisikan secara matematis dengan persamaan (1) berikut:

$$Entropy(S) = \sum_{i=1}^{n} - pi \times \log_2 pi \qquad \text{..... (1)}$$

Sumber : Kusrini dan Luthfi (2009)

Dengan:

S = Himpunan Kasus

n = Jumlah partisi S

pi = Proporsi dari Si terhadap S

Selanjutnya untuk mengklasifikasi objek dengan pohon keputusan diperlukan pengetahuan tentang *information gain*. *Information gain* adalah salah satu *attribute selection measure* yang digunakan untuk memilih *test attribute* tiap *node* pada pohon. Atribut dengan informasi gain tertinggi dipilih sebagai test atribut dari suatu node [16].

Dengan kata lain, untuk memilih atribut sebagai akar pada algoritma C4.5 didasarkan pada nilai gain tertinggi dari atribut-atribut yang ada. Gain (S,A) merupakan perolehan informasi dari atribut A relatif terhadap output data S. Perolehan informasi didapat dari output data atau *variable dependent* S yang dikelompokkan berdasarkan atribut A, dinotasikan dengan gain (S,A). Untuk menghitung gain digunakan rumus seperti yang tertera pada rumus (2) berikut ini:

$$Gain\ (S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|Si|}{|S|} \times Entropy(Si)$$

Sumber: Kusrini dan Luthfi (2009)

Dengan:

S = Himpunan Kasus

A = Atribut

n = Jumlah partisi atribut A

|Si| = Jumlah kasus pada partisi ke i

|S| = Jumlah kasus dalam S

Dalam pembangunan aplikasi ini menggunakan proses klasifikasi kriteria nasabah penerima pinjaman menerapkan Algoritma C4.5 karena algoritma ini memiliki keakuratan yang tinggi dimana apabila data sample untuk dianalisis bertambah maka keakuratan akan semakin tinggi. Hal ini berbeda dengan analisis lainnya yang mana bila semakin banyak data sample yang dianalisis maka proses analisis klasifikasi akan semakin lama, namun dengan algoritma C4.5 justru akan semakin akurat.

2.1.9 UML (Unified Modeling Language)

UML atau singkatan dari *Unified Modeling Languade* pertama kali diperkenalkan pada tahun 1980-an oleh *Object Management Group*, yaitu sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP. UML kemudian dikembangkan sebagai suatu alat untuk menganalisis dan mendesain masalah dengan berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. UML merupakan dasar bagi *design tools* yang berorientasi objek pada IBM.

Menurut Whitten & Bentley (2007:371), *Unified Modeling Language (UML)* adalah sekumpulan konversi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek, dimana UML menyediakan 13 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case Diagram* menggambarkan interaksi antara sistem internal, sistem eksternal, dan *user*. Dalam hal ini berarti *Use Case Diagram* mampu menjelaskan secara grafik siapa pengguna sistem dan bagaimana pengguna dapat berinteraksi dengan sistem.
2. *Activity Diagram* menggambarkan alur *sequential* dari aktivitas sebuah proses bisnis atau *Use Case*. Dan dapat digunakan sebagai model logika yang digunakan oleh sistem.
3. *Class Diagram* menggambarkan struktur objek sistem. Hal ini menunjukkan kelas yang menjadi komponen penyusun sistem, serta hubungan antar kelas.
4. *Object Diagram* hampir sama dengan *Class Diagram* namun disini sebagai model yang menggambarkan instansi objek yang sebenarnya beserta nilai atributnya.
5. *State Machine Diagram* adalah model yang menggambarkan perilaku objek di dalam sistem terhadap suatu kejadian (*event*) selama masa hidupnya.
6. *Composite Structure Diagram* menguraikan struktur internal, komponen, atau *Use Case* dari suatu kelas.
7. *Sequence Diagram* menggambarkan bagaimana objek berinteraksi melalui pengiriman pesan (*message*) dalam pengeksekusian sebuah *Use Case* atau operasi tertentu.
8. *Communication Diagram* atau yang biasa disebut juga dengan *Collaboration Diagram*, hampir sama dengan *Sequence Diaram*. Namun disini lebih difokuskan pada pemilihan waktu atau urutan pesan. *Communication Diagram* juga berfokus pada penyusunan struktur objek dalam bentuk jaringan.
9. *Interaction Overview Diagram* adalah diagram yang mengkombinasikan *Activity Diagram* dan *Sequence Diagram* dengan tujuan untuk menunjukkan bagaimana objek berorientasi dalam setiap aktivitas *Use Case*.

10. *Timing Diagram* adalah diagram interaksi bentuk lain yang menitikberatkan pada batasan pemilihan waktu dalam keadaan satu objek atau kumpulan objek yang berubah. Diagram ini sangat berguna untuk mendesain *embedded software* di banyak perangkat.
11. *Component Diagram* menggambarkan penyusunan kode *programming* yang dibagi menjadi beberapa komponen dan menjelaskan bagaimana komponen tersebut berinteraksi.
12. *Deployment Diagram* menggambarkan konfigurasi dari komponen *software* dalam arsitektur fisik dari "simpul-simpul" sistem *hardware*.
13. *Package Diagram* menggambarkan bagaimana kelas/konstruksi dari UML lain disusun dalam bentuk paket (berkaitan dengan paket *Java* atau *C++* dan *namespaces* dari *.NET*) dan ketergantungannya antar paket.

2.1.10 MySQL

MySQL adalah sebuah program database server yang mampu menerima dan mengirimkan datanya sangat cepat, *multiuser* serta menggunakan perintah dasar SQL (*Structured Query Language*) yang bersifat *free*. Dalam hal ini maksudnya MySQL dapat bebas digunakan sebagai database untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya. MySQL pertama kali diperkenalkan oleh seorang *programmer* database bernama Michael Widenius. MySQL juga merupakan program yang dapat mengakses suatu database MySQL yang berposisi sebagai server, yang berarti program kita berposisi sebagai client. [17]

Berdasarkan uraian diatas maka dapat disimpulkan bahwa MySQL adalah sebuah database yang dapat digunakan sebagai client maupun server. Database MySQL itu sendiri merupakan suatu perangkat lunak database yang berbentuk database relasional atau disebut *Relational Database Management System* (RDBMS) yang menggunakan suatu bahasa permintaan yang bernama SQL (*Structured Query Language*).

SQL (*Structured Query Language*) yaitu sebuah bahasa permintaan database yang terstruktur. Bahasa SQL ini dibuat sebagai bahasa yang dapat

merelasikan beberapa tabel dalam database maupun merelasikan antar database. SQL sendiri dibagi menjadi tiga bentuk Query, yaitu [17]:

2.1.10.1 DDL (*Data Definition Language*)

DDL (*Data Definition Language*) adalah suatu metode Query yang berguna untuk mendefinisikan data pada sebuah Database, Query yang dimiliki DDL adalah:

1. CREATE : Digunakan untuk membuat Database dan Tabel.
2. DROP : Digunakan untuk menghapus Database dan Tabel.
3. ALTER : Digunakan untuk melakukan perubahan struktur tabel yang telah dibuat, bisa berupa penambahan field (Add), penggantian nama field (Change), penamaan kembali field (Rename), dan penghapusan field (Drop).

2.1.10.2 DML (*Data Manipulation Language*)

DML (*Data Manipulation Language*) adalah suatu metode Query yang dapat digunakan untuk melakukan pemanipulasian database yang telah dibuat. Dalam hal ini berarti DML hanya dapat digunakan apabila DDL telah terjadi. Query yang dimiliki DML adalah:

1. INSERT : Digunakan untuk memasukkan data pada Tabel Database.
2. UPDATE : Digunakan untuk merubah data yang ada pada Tabel Database.
3. DELETE : Digunakan untuk menghapus data pada Tabel Database.

2.1.10.3 DCL (*Data Control Language*)

DCL (*Data Control Language*) adalah suatu metode Query SQL yang digunakan untuk memberikan hak otoritas untuk mengakses database, mengalokasikan space, mendefinisikan space dan mengaudit penggunaan database. Query yang dimiliki DCL adalah:

1. GRANT : Digunakan untuk mengizinkan *user* mengakses Tabel dalam Database.
2. REVOKE GRANT : Digunakan untuk membatalkan izin hak *user* yang ditetapkan oleh perintah.
3. COMMIT : Digunakan untuk menetapkan penyimpanan Database.
4. ROLLBACK: Digunakan untuk membatalkan penyimpanan Database.