BAB II

TINJAUAN PUSTAKA

2.1 PT. Inti Bharu Mas

PT. Inti Bharu Mas merupakan perusahaan berbentuk "Perseroan Terbatas" yang dimiliki swasta, menjalankan usaha dibidang distribusi barang. Perusahaan distributor terkemuka di Provinsi Lampung ini didirikan pada tahun 1989. PT Inti Bharu Mas berpusat di daerah Campang Raya dengan alamat jalan Tembesu Raya No. 9 Kelurahan Campang Raya, Bandar Lampung. Dan memiliki cabang di Pringsewu Lampung dan di Provinsi Sumatera Selatan (Palembang). PT Inti Bharu Mas ini menjalin mitra kerja yang baik dengan beberapa Principle atau kurun waktu yang relative panjang dan memiliki jangkauan pemasaran dan service hampir seluruh provinsi Lampung. Beberapa mitra kerja PT Kongguan Biscuit Indonesia, PT monde Biscuit, PT Mayora Indah,Tbk, PT Nissin, PT Unicarm Indonesia dan PT Unilever Indonesia.

2.2 Barcode

Pada tahun 1932, Wallace Flint membuat sistem pemeriksaan barang di perusahaan retail. Awalnya, teknologi kode batang dikendalikan oleh perusahaan retail, lalu diikuti oleh perusahaan industri. Lalu pada tahun 1948, pemilik toko makanan lokal meminta *Drexel Institute of Technology di Philadelphia*, untuk membuat sistem pembacaan informasi produk selama c*heckout* secara otomatis (Siswanto & Siswanto, 2018). *Barcode* adalah sekumpulan kode untuk mendefinisikan huruf dan angka yang terdiri dari kombinasi garis dengan pengaturan jarak yang berbeda-beda. Aturan tersebut merupakan metode untuk dapat memasukkan data ke dalam komputer. Informasi pada *barcode* berisi enkripsi dari sejumlah digit angka. Saat *barcode* tersebut di scan dengan alat *barcode scanner*, maka kode tersebut secara otomatis terhubung ke data barang yang sudah disimpan dalam database. Hasil dari pemindaian tersebut berisi data-data dari berbagai produk seperti nama

vendor, nama produk, harga dan data lainnya sesuai dengan apa yang sudah dimasukkan pada database(Sudarma et al., n.d.).

Gambar 1 Barcode Sumber : (Yudha, Sudarma, & Mertasana, 2017)

2.3 Algoritma *Knuth Morris Pratt*

Algoritma *Knuth Morris Pratt* merupakan algoritma pencocokan pola dan dapat digunakan dalam sistem *barcode* untuk mengetahui informasi produk hasil pembacaan *barcode*. Algoritma KMP mirip dengan algoritma *brute force*, namun algoritma ini bergeser dengan lebih pintar. Algoritma ini menggunakan fungsi pinggiran (*border function*) yang berguna untuk menentukan seberapa banyak pola digeser ketika ditemukan ketidakcocokan antara teks dan pola. Sehingga pada algoritma KMP, pergeseran tidak selalu hanya satu karakter saja seperti pada algoritma *brute force*. Oleh karena itu, waktu pencarian pada algoritma KMP berkurang secara signifikan dibandingkan algoritma *brute force* (Ghassani, 2015). Algoritma *Knuth Morris Pratt* dikembangkan oleh D. E. Knuth, bersama dengan J. H. Morris dan V. R. Pratt. Algoritma *Knuth-Morris-Prat* merupakan pengembangan dari algoritma pencarian *string* sebelumnya, yaitu algoritma *Brute Force*. Algoritma *Brute Force* merupakan algoritma dasar yang paling sederhana dalam menyelesaikan persoalan pencocokan string yang melakukan pencarian pada setiap posisi di dalam teks antara 0 dan n-m, dimana n adalah panjang teks/banyaknya nama file yang tersimpan di komputer dan m adalah panjang karakter dari suatu pattern kata yang akan dicari .Perhitungan penggeseran pada algoritma ini adalah sebagai berikut, bila terjadi ketidakcocokkan pada saat pattern sejajar dengan teks[i..i+n-1], dapat menganggap ketidakcocokan pertama terjadi di antara teks[i+j] dan pattern[j], dengan 0<j<n. Berarti, teks[i..i+j1]=pattern[0..j-1] dan a=teks[i+j] tidak sama dengan b=pattern[j]. Ketika kita menggeser, sangat

beralasan bila ada sebuah awalan v dari pattern akan sama dengan sebagian akhiran u dari sebagian teks. Sehingga kita bisa menggeser pattern agar awalan v tersebut sejajar dengan akhiran dari u (Rossaria & Susilo, 2015).

Secara sistematis, langkah-langkah yang dilakukan algoritma *Knuth Morris Pratt* pada saat mencocokkan *string*:

1) Algoritma *Knuth Morris Pratt* mulai mencocokkan *pattern* atau pola susunan kata yang dijadikan sebagai contoh pada awal teks.
2) Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern atau pola susunan kata yang dijadikan sebagai contoh dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi :
    a. Karakter di pattern atau pola susunan kata yang dijadikan sebagai contoh dan di teks yang dibandingkan tidak cocok (*mismatch*).

    b. Semua karakter di *pattern* atau pola susunan kata yang dijadikan sebagai contoh cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3) Algoritma kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

2.4 Distributor

Kata distributor berasal dari Bahasa Inggris *distribute* yang artinya menyalurkan. Jadi, dalam istilah ekonomi distributor diartikan sebagai penyalur produk dari produsen ke konsumen. Dalam Kamus Besar Bahasa Indonesia (KBBI), distributor didefinisikan sebagai orang atau badan yang bertugas mendistribusikan barang (dagangan) atau yang disebut penyalur. Yang dimaksud distributor adalah orang yang melakukan kegiatan distribusi. Adapun kegiatan distribusi adalah kegiatan penyaluran produk dari produsen kepada konsumen(Zaenuddin et al., n.d.).

2.5 Aplikasi

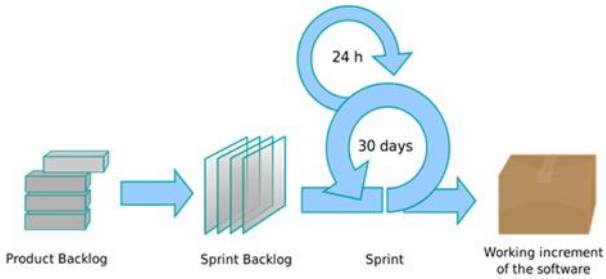
Aplikasi adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas seperti sistem perniagaan, *game,* pelayanan masyarakat, periklanan dan hampir semua proses kegiatan ((Rohayah et al., n.d.). Aplikasi sering juga disebut sebagai perangkat lunak, merupakan program komputer yang isi instruksinya dapat diubah dengan mudah (Syani et al. 2020). Dari pengertian diatas, dapat disimpulkan bahwa aplikasi adalah suatu perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan oleh pengguna.

2.6 Aplikasi *Mobile*

Aplikasi *Mobile* adalah perangkat lunak yang berjalan pada perangkat mobile seperti smartphone atau tablet PC. Aplikasi *Mobile* juga dikenal sebagai aplikasi yang dapat diunduh dan memiliki fungsi tertentu sehingga menambah fungsionalitas dari perangkat mobile itu sendiri. Untuk mendapatkan *mobile application* yang diinginkan, user dapat mengunduhnya melalui situs tertentu sesuai dengan sistem operasi yang dimiliki. Google Play Store dan Appstore merupakan beberapa contoh dari situs yang menyediakan beragam aplikasi bagi pengguna Android dan iOS untuk mengunduh aplikasi yang diinginkan (Zulfadhilah et al., n.d.).

2.7 Metode Pengembangan Perangkat Lunak

Metode pengembangan yang digunakan dalam pembuatan aplikasi ini yaitu dengan menggunakan metode *scrum.*



Gambar 2 Metode *Scrum*

Model *scrum* merupakan metode pengembangan perangkat lunak secara cepat (*agile*). Model *scrum* memiliki beberapa kelebihan seperti, mampu mentransformasikan proses bisnis yang sulit menjadi mudah dikembangkan, dengan model *scrum* mampu memonitoring dan mengontrol aktivitas pada proses pengembangan sistem (Kuswinanti et al., 2021) Metode pengembangan sistem mengacu pada metode *Scrum* yang merupakan sebuah metode yang mudah dikontrol, fleksibel, memuat strategi pengembangan menyeluruh dimana seluruh tim bekerja sebagai satu unit untuk mencapai goal yang sama (Warkim et al., 2020)

Proses pengembangan menggunakan metode scrum terdapat empat tahapan pengembangan yaitu : ((1) *product backlog*, (2) *sprint Backlog*, (3) *sprint log*, (4) *Increment*. Adapun kerangka kerja dalam model scrum terdiri dari:

a. *Product Backlog*

   Tahap *Product Backlog* lebih mengarah kepada pengumpulan kebutuhan, pembaruan, pemeliharaan, dan deskripsi singkat tentang fungsi-fungsi yang dinginkan pada saat aplikasi akan dibangun.

b. *Sprint Backlog*

   Tahap Sprint backlog dilakukan untuk sebuah proses pemenuhan kebutuhan sesuai dengan yang dinginkan pada proses *Product Backlog* sebelumnya.

c. *Sprint Log*

Tahap *Sprint Log* merupakan proses dimana paparan aplikasi dalam bentuk sebuah prototype dan pemaparan dalam bentuk hal teknis baik berupa tools yang dibutuhkan untuk mengembangkan aplikasi

d. *Increment*

Tahap *Increment* merupakan hasil akhir dari tahap *Product Backlog* yang telah selesai dikembangkan pada saat tahapan *Sprint Log*. Pada tahap ini, diharapkan tahapan Increment telah selesai dilakukan sehingga mampu untuk digunakan sesuai dengan yang dinginkan.

2.8 Android Studio

Android Studio merupakan sebuah *Software Tools Integrated Development Environment* (IDE) untuk *platform Android*. *Android Studio* diluncurkan pada tanggal 16 mei 2013 pada konferensi *Google I/O* oleh produk manajer *Google, Ellie Powers*. Android Studio bersifat free di bawah *Apache License 2.0*. *Android Studio* berbasiskan *JetBrains' intellij IDEA*, *studio* didesain khusus untuk *Android Development* (Sonatha et al., 2019)

Android studio memiliki fitur yang baik untuk melakukan pengembangan aplikasi. Berikut beberapa fitur yang disediakan dalam *Android Studio*:

1. *Instant run* secara dramatis mempercepat siklus pengeditan, pembuatan, dan penjalanan membuat pekerjaan "tetap mengalir".
2. *Android Emulator* yang cepat dan memungkinkan untuk mengubah ukuran secara dinamis.
3. *Gradle, android studio* menawarkan otomatisasi pembuatan aplikasi berkinerja tinggi, pengelolaan dependensi yang tangguh, dan konfigurasi yang dapat disesuaikan.
4. *Android studio* menyediakan lingkungan yang menyatu untuk mengembangkan aplikasi ponsel dan *tablet android, android wear, android tv, dan android auto.*
5. Memulai proyek dengan template kode untuk pola atau mengimpor sample kode Google dari *GitHub*.

2.9 Firebase Realtime Database

Firebase Realtime Database merupakan *cloud database*. Data disimpan dalam format *JSON* dan disinkronkan secara *realtime* ke setiap klien yang terhubung. Ketika membangun aplikasi *hybrid* lintas *platform*, seperti *Android* dan iOS maka semua klien berbagi satu *instance Realtime Database* dan secara otomatis menerima pembaruan dengan data tertentu. *Firebase Realtime Database* adalah basis data NoSQL dan karena itu memiliki optimalisasi dan fungsionalitas yang berbeda dibandingkan dengan basis data relasional. Membuat *database Firebase* bisa melalui import file JSON ke konsol *Firebase*, atau dapat juga dibuat langsung melalui halaman konsol *Realtime Database* secara manual (Ilhami, 2017).

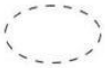
2.10 *UML* (*Unified Modeling Language*)

Unified Modeling Language (UML) adalah sebuah bahasa yang berdasarkan gambar untuk menvisualisasikan menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis objek. UML bukanlah merupakan bahasa pemrograman tetapi model-model yang tercipta berhubungan langsung dengan berbagai macam bahasa pemrograman, sehingga memungkinkan melakukan pemetaan (*mapping*) langsung dari model-model yang dibuat dengan UML degan bahasa-bahasa pemrograman berorientasi obyek, seperti Java (Fitri Setianda et al., 2020).

2.10.1 *Use case* Diagram

Use case diagram adalah pemodelan untuk menggambarkan perilaku sistem yang akan dibuat pada sebuah aplikasi(Heriyanto, Y. 2018).. *Use case* diagram memiliki beberapa *symbol* sebagai berikut :

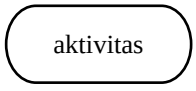
Tabel 1. Simbol *Usecase* Diagram

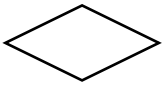
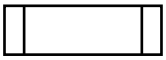
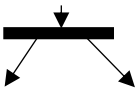
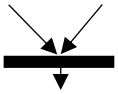
SIMBOL	NAMA	KETERANGAN
	*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
	Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
	Include	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
	Extend	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
	*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

2.10.2 *Activity* Diagram

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Perlu diperhatikan bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem(Heriyanto, Y. 2018). Berikut adalah simbol-simbol yang ada pada diagram diagram aktivitas:

Tabel 2. Simbol-simbol *activity* diagram

Simbol	Deskripsi
Status awal 	Status awal dari diagram aktivitas untuk mengawali proses aktivitas sistem.
Aktivitas 	Aktivitas pada sistem biasanya diawali dengan kata kerja.

Percabangan/*decision* 	Asosiasi percabangan ada jika pilihan aktivitas lebih dari satu.
Penggabungan/*join* 	Asosiasi penggabungan dilakukan jika ada lebih dari satu aktivitas dan digabungkan menjadi satu.
Status akhir 	Status akhir dari sistem yang merupakan akhir dari suatu sistem aktivitas.
Database 	Symbol ini untuk menunjukan penyimpanan pada database sistem.
Fork, 	Menunjukkan aktivitas yang dilakukan secara parallel.
Join 	Menunjukkan aktivitas yang digabungkan.

2.10.3 *Class* Diagram

Class diagram merupakan alat bantu untuk menentukan langkah-langkah kerja yang akan dilakukan oleh pemogram di mulai dari proses pengumpulan data, sampe pembentukan tabel sesuai dengan permasalahan yang ditangani. *Class* diagram ini terlebih dahulu dirancang dalam mendukung rancangan pengolahan data elektronis supaya dapat berjalan dengan baik, dan dengan relasi yang baik akan di peroleh gambaran umum sistem yang akan di persiapkan (Purwati & Rahardi, 2018). Berikut adalah simbol yang ada pada *class* diagram:

Tabel 3 Simbol-simbol *class* diagram

SIMBOL CLASS DIAGRAM

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

2.11 Black Box Testing

Black box testing merupakan pengujian untuk mengetahui fungsi perangkat lunak yang telah berjalan sesuai dengan kebutuhannya (Argani & Taraka, n.d.) Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan pengeluaran perangkat lunak sesuai dengan spesifikasi yang dibutuhkan (Misdi D, Dalis S 2018).

2.12 Penelitian Terkait

Dalam penyusunan skripsi ini, peneliti terinspirasi dan mereferensi dari penelitian penelitian sebelumnya yang berkaitan dengan skripsi ini. Daftar penelitian terkait ialah sebagai berikut :

1. Pembuatan Aplikasi Pencatatan *Stock* Dengan Menggunakan *Barcode* Pada Android.

   (Wibisono et al., n.d.) Menguraikan bahwa *Stock* opname merupakan penghitungan dan penyesuaian stok barang dan aset yang dimiliki oleh toko atau perusahaan di gudang atau etalase dengan data stok yang terdapat pada database sistem perusahaan. Selama *stock* opname dilakukan, kegiatan masuk dan keluarnya barang tidak dapat dilakukan. Hal ini menyebabkan perusahaan tidak teratur dalam melakukan *stock* opname. Selain itu kesalahan pencatatan yang dilakukan oleh pegawai juga rawan terjadi jika barang yang dicatat cukup banyak. Berdasarkan latar belakang masalah tersebut, maka dibuatlah aplikasi untuk melakukan pencatatan stock dengan menggunakan *barcode*. Aplikasi ini dibuat pada *mobile device* berbasis Android karena selain kegunaannya sebagai telepon genggam, *mobile device* juga memiliki fasilitas yang modern seperti kamera dan koneksi internet menggunakan wifi sehingga dapat pula dijadikan sebagai *barcode reader* yang dapat terhubung dengan server perusahaan. Hasil dari aplikasi yang telah dibuat antara lain, dapat melakukan download data *stock* barang dan aset dari *server* tergantung otorisasi, dapat memindai *barcode* dengan menggunakan kamera yang terdapat pada *mobile device* untuk melakukan pencataan barang, dapat melakukan pencatatan jumlah stok barang dan aset pada periode tertentu,dan terdapat laporan pencatatan *stock* barang dan aset pada periode tertentu.

2. Perancangan Aplikasi Sistem *Inventory* Barang *Menggunakan Barcode Scanner* Berbasis *Android.*

   (Sudarma et al., n.d.). Menguraikan bahwa Penelitian ini bertujuan untuk merancang aplikasi sistem *inventory* barang yang menggunakan *barcode scanner* di perangkat *Android*. Perancangan aplikasi ini menggunakan bahasa *Basic* dalam pembuatan aplikasi desktop dan bahasa Java pada aplikasi Android. *Database* yang digunakan adalah

database MySql yang disimpan pada server *online* sehingga dapat diakses oleh aplikasi sistem *inventory* di Android secara mobile. Proses pertukaran data antara *Android* dengan *database* menggunakan web *service* untuk mempermudah distribusi data. Hasil dari pembuatan aplikasi ini yaitu dapat mengecek status informasi ketersediaan stok barang melalui smartphone *Android* hanya dengan memindai barcode yang tertera pada barang. Secara tidak langsung aplikasi ini juga dapat meningkatkan pelayan kepada konsumen toko.

3. Penerapan Algoritma *Knuth Morris Pratt* Pada Aplikasi Pencarian Berkas *Shipment* Berbasis Web (Studi Kasus Di Pt Yec Semarang).
   (Khasanah & Nurul, n.d.) Menguraikan bahwa PT. Yudhanusa Ekspresindo Caraka memiliki banyak data transaksi yang digunakan dalam penyelesaian suatu shipment ekspor maupun impor. Sistem yang selama ini berjalan dalam pencarian berkas masih manual dengan mencari berkas dalam bentuk fisik pada lemari berkas file atau mencari pdf *attachment* yang terdapat di email. Dengan sistem pencarian berkas dikembangkan dengan bahasa pemograman ASP.net MVC dan database SQLite dengan menerapkan algoritma *knuth morris pratt*. Dari hasil uji efektifitas, dapat disimpulkan bahwa efektifitas sistem yang baru jauh lebih besar dari pada sistem yang lama, yaitu sebesar 85,00%. Sedangkan untuk sistem yang lama hanya sebesar 45,00%. Ini menunjukkan bahwa sistem yang baru sudah lebih baik dan efektif. Aplikasi pencarian berkas *shipment* memberikan informasi yang tepat guna dan lebih cepat. Hal ini didukung dengan hasil dari 3 responden yang memberikan nilai rata-rata 3,5 yang berarti sistem ini dapat memberikan hasil rekomendasi yang tepat bagi konsumen.

4. Implementasi Algoritma Pencocokan String *Knuth Morris Pratt* Dalam Aplikasi Pencarian Dokumen *Digital* Berbasis *Android.*
   (Rossaria & Susilo, 2015). Menguraikan bahwa Pada penelitian ini dibangun sebuah aplikasi yang bertujuan untuk mencari dokumen yang

berasal dari *Android* dan mengaplikasikannya dengan menggunakan algoritma pencocokan *string* sebagai salah satu cara untuk menemukan dokumen *digital* yang terdapat pada Android. aplikasi ini dibangun berbasis Android dengan menggunakan algoritma Pencocokan string *Knuth Morris Pratt* sebagai algoritma dalam aplikasi pencarian dokumen, dan dibangun dengan menggunakan bahasa pemrograman JAVA dengan *IDE ECLIPSE JUNO*, analisis perancangan sistem ini menggunakan *Unified Modeling Language* (UML). Dapat disimpulkan bahwa aplikasi ini dapat melakukan pencarian dokumen digital yang terdapat dalam *Android* dengan menggunakan algoritma *Knuth Morris Pratt*. Hasil pencarian yang ditampilkan berupa dokumen-dokumen yang tersedia dalam *Android* dan informasi mengenai jumlah dari dokumen yang tersedia dalam Android tersebut, serta menunjukkan bahwa algoritma *Knuth Morris Pratt* bisa digunakan dalam aplikasi pencarian dokumen pada *Android*.

5. Perancangan Aplikasi Pendeteksi Kesalahan Perintah Sql *Query* Menggunakan Algoritma *Knuth Morris Pratt*. (Ginting & Napitupulu, 2018) Menguraikan bahwa Aplikasi *Database* Server kini banyak dimanfaatkan oleh lembaga atau institusi untuk berbagai kepentingan. Kebanyakan dari penguna melakukan pendeteksian kesalahan pengetikan dengan cara manual sehingga kurang efektif. Permasalahan yang sering terjadi yaitu pada penulisan atau pengetikan dikarenakan sensitifnya bahasa pemograman tersebut. Misalanya huruf besar dan kecil, titik dan koma, atau tanda baca lainnya. Selain dari masalah tersebut pengguna dari aplikasi MySQL pada umumnya menggunakan tools untuk membantu dalam pembuatan database ataupun dalam proses manipulasi sehingga struktur *Query* dari SQL sering dilupakan. Algoritma *Knuth Morris Pratt* merupakan algoritma yang digunakan untuk melakukan proses pencocokan string. Algoritma ini merupakan jenis *Exact String Matching* Algorithm yang merupakan pencocokan string secara tepat dengan susunan karakter dalam string yang

dicocokkan memiliki jumlah maupun urutan karakter dalam string yang sama. Algoritma *Knuth Morris Pratt* merupakan bagian dalam proses pencarian string yang memegang peranan penting untuk mendeteksi kesalahan menjadi lebih cepat dan efisien. Untuk mengetahui kesalahan pengetikan perintah SQL dengan menggunakan aplikasi pendeteksi kesalahan perintah SQL maka penugguna terlebih dahulu melakukan penginputan file SQL yang berisikan struktur database. Setelah itu pengguna juga mengetikkan *Query* SQL yang akan diperiksa. Melalui langkah tersebut maka dilakukan perbandingan antara *Query* yang diinput oleh pengguna dengan struktur SQL