# BAB III

# ANALISA DAN PERANCANGAN

## III.1. Analisa Permasalahan

Seiring dengan perkembangan teknologi saat ini serta untuk meningkatkan efesiensi kerja dan waktu, maka masih banyak sistem yang ada saat ini yang masih manual, yang harus diganti dengan sistem yang lebih baik lagi, yaitu dengan membangun sistem yang terkomputerisasi.

Aplikasi kamus digital ini dibuat atas dasar kebutuhan ilmu pengetahuan akan istilah dalam dunia komputer terutama dalam bidang teknologi informasi. Dari hasil pengamatan penulis mengenai analisis sistem pencarian arti, sering mendapat kendala dalam pencariannya. Hal ini terjadi karena istilah tersebut disajikan dalam bentuk buku berupa kamus istilah yang dalam pencariannya membutuhkan banyak waktu dan tampilan yang membosankan. Dengan demikian penulis mencoba membangun sebuah program aplikasi kamus digital komputer yang dapat mempermudah, mempercepat pencarian makna dari istilah-istilah tersebut.

Aplikasi kamus digital ini dirancang dengan menggunakan tiga metoda pencarian, yaitu :

1. *Match case*

Dalam metoda ini, akan dicari kata yang sama persis dengan kata yang dimasukkan oleh pengguna. Contoh : jika pengguna memasukkan kata ‘apa kabar’( yang diantara kata apa dan kata kabar dibatasi hanya oleh satu spasi ),

maka program akan mencari dalam basis data kata ‘apa kabar’( dengan satu spasi di tengah kata), perbedaan jumlah spasi di tengah kata dalam metoda ini akan sangat berpengaruh.

2. *Large case*

   Pada metoda *large case*, akan dicari kata yang didepannya mengandung kata yang dimasukkan oleh pengguna. Contoh : jika pengguna memasukkan kata ‘abad’, maka program akan mencari dalam basis data kata yang didepannya terkandung kata ‘abad’ ( abad%) . Sehingga bila kata ‘abad’ tidak terdapat dalam basis data, namun terdapat kata ‘abadi’ dan kata ‘abadilah’, maka kedua kata ini yang akan ditampilkan dalam hasil pencarian

3. *Wider case*

   Metoda ini akan mencari kata yang mengandung kata yang dimasukkan oleh pengguna ( baik itu didepan, ditengah maupun dibelakang ). Contoh : jika pengguna memasukkan kata ‘abad’, maka program akan mencari kata yang mengandung kata ‘abad’ (%abad%). Sehingga bila kata ‘abad’ tidak terdapat dalam basis data, namun terdapat kata ‘perabadian’ dan ‘abadilah’, maka kedua kata itulah yang akan ditampilkan dalam hasil pencarian.

## III.2. Perancangan Sistem

Pengembangan sebuah perangkat lunak bertujuan untuk menghasilkan perangkat lunak yang dapat memenuhi kebutuhan *user*. Setiap pengembangan sebuah sistem perangkat lunak memerlukan adanya dokumentasi terhadap kebutuhan-kebutuhan *user* agar tujuan tersebut tercapai. Tahap analisis berorientasi obyek dilakukan dengan membuat alur bisnis sistem yang kemudian diurutkan dalam sebuah daftar kebutuhan. Tahap selanjutnya adalah memodelkan kebutuhan *user* ke dalam *use case diagram. Use case diagram* bertujuan untuk menggambarkan kebutuhan-kebutuhan fungsional yang harus disediakan oleh sistem agar dapat memecahkan permasalahan yang dihadapi *user*.

### III.2.1. Analisa Algoritma *Boyer-Moore*

Algoritma *Boyer-Moore* adalah salah satu algoritma untuk mencari suatu *pattern* di dalam teks, dibuat oleh R.M Boyer dan J.S Moore. Ide utama algoritma ini adalah mencari *pattern* dengan melakukan pembandingan karakter mulai dari karakter paling kanan dari *pattern* yang dicari. Dengan mengunakan algoritma ini, secara rata-rata proses pencarian akan menjadi lebih cepat jika dibandingakan dengan algoritma lainnya. alasan melakukan pencocokan dari kanan (posisi terakhir *pattern* yang dicari) ditunjukan dalam contoh sederhana berikut :

pattern : RESTORAN

string : AKU AREP MANGAN BAKSO SAPI NENG RESTORAN.

Sebelum mulai pencarian, algoritma BM ini perlu menghitung nilai pergeseran dari *pattern* yg akan dicari. jadi pasti akan ada 2 tabel pergeseran(OH dan MH).

Nanti pada saat mulai mencari, jika terjadi ketidakcocokan karakter, algoritma akan memilih salah satu nilai pergeseran yang akan dilakukan. nilai pergeseran yang dipilih adalah yang paling maksimal, supaya pencarian menjadi lebih cepat.

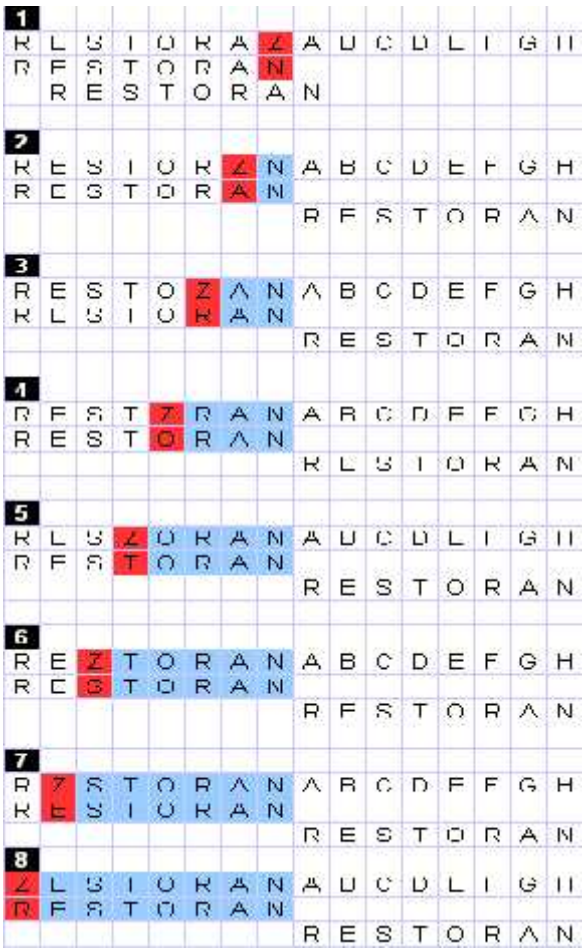
**Tabel III.1** ***Occurence Heuristic***

| R | E | S | T | O | R | A | N |
|---|---|---|---|---|---|---|---|
| 2 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Tabel III.2** ***Match Heuristic***

| R | E | S | T | O | R | A | N |
|---|---|---|---|---|---|---|---|
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 1 |

untuk setiap langkah, baris pertama adalah potongan *string*, dan baris kedua adalah *pattern* yg dicari. Jadi untuk MH, point utamanya adalah “pergeseran dilakukan berdasarkan posisi ketidakcocokan karakter yang terjadi”, maksudnya untuk menghitung MH, kita perlu tahu pada posisi ke berapa terjadi ketidakcocokan. posisi ketidakcocokan itulah yang akan menentukan besar pergeseran. berbeda dengan OH yang menentukan nilai pergeseran berdasarkan karakter apa yang menyebabkan tidak cocok.



Cara perhitungannya adalah sebagai berikut.

langkah-1

untuk ketidakcocokan karakter pada posisi 8, karakter “N” maka nilai pergeserannya selalu 1

langkah-2

jika karakter “N” sudah cocok, tetapi karakter pada posisi 7 (sebelum “N”) bukan “A” maka geser sebanyak 8 posisi, sehingga posisi *string* melewati teks.karena sudah pasti “RESTORZN” bukan “RESTORAN”

langkah-3

jika karakter “AN” sudah cocok, tetapi karakter pada posisi 6 (sebelum “AN”) bukan “R” maka geser sebanyak 8 posisi, sehingga posisi string melewati teks.karena sudah pasti “RESTOZAN” bukan “RESTORAN”

langkah-4

jika karakter “RAN” sudah cocok, tetapi karakter pada posisi 5 (sebelum “RAN”) bukan “O” maka geser sebanyak 8 posisi, sehingga posisi string melewati teks.karena sudah pasti “RESTZORAN” bukan “RESTORAN”

langkah-5

jika karakter “ORAN” sudah cocok, tetapi karakter pada posisi 4 (sebelum “ORAN”) bukan “T” maka geser sebanyak 8 posisi, sehingga posisi *string* melewati teks.karena sudah pasti “RESZORAN” bukan “RESTORAN”

langkah-6

jika karakter “TORAN” sudah cocok, tetapi karakter pada posisi 3 (sebelum “TORAN”) bukan “S” maka geser sebanyak 8 posisi, sehingga posisi string melewati teks.karena sudah pasti “REZTORAN” bukan “RESTORAN”

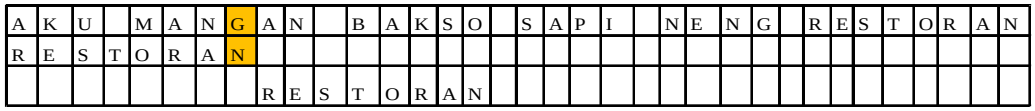
langkah-7

jika karakter “STORAN” sudah cocok, tetapi karakter pada posisi 2 (sebelum “STORAN”) bukan “E” maka geser sebanyak 8 posisi, sehingga posisi *string* melewati teks.karena sudah pasti “RZSTORAN” bukan “RESTORAN”

langkah-8

jika karakter “ESTORAN” sudah cocok, tetapi karakter pada posisi 1 (sebelum “ESTORAN”) bukan “R” maka geser sebanyak 8 posisi, sehingga posisi *string* melewati teks.karena sudah pasti “ZESTORAN” bukan “RESTORAN”

Selanjutnya proses pencarian *pattern* “RESTORAN” di dalam *string* langkah-1




-”G” tidak cocok dengan “N” selanjutnya algoritma membandingkan nilai pergeseran dari 2 tabel.

-tabel OH : karakter “G” nilai pergeserannya = 8 belum ada karakter yang cocok

-tabel MH : ketidakcocokan pada posisi 8 (karakter “N”) nilai pergeserannya = 1

-sehingga geser *string* sebesar 8 posisi (nilai maksimal dari kedua tabel pergeseran)

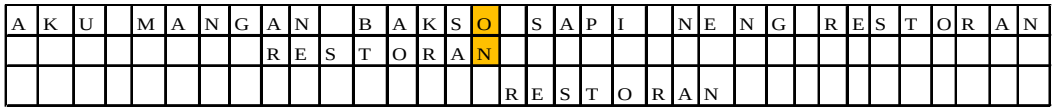
dapat dilihat perbedaan karakter dari 2 tabel, tabel OH melihat “karakter APA yang menyebabkan ketidakcocokan” sedangkan tabel MH melihat “karakter pada posisi keberapa yang menyebabkan ketidakcocokan”

tabel OH : karakter yang menyebabkan ketidakcocokan adalah “G” nilai pergeseran untuk karakter “G” adalah 8.

tabel MH : karakter yang menyebabkan ketidakcocokan adalah karakter pada posisi 8, dan nilai pergeseran apabila terjadi ketidakcocokan pada posisi 8 adalah

sebesar 1. seanjutnya 8>1 kan? maka daripada cuma menggeser *pattern* sebanyak 1 karakter, lebih baik jika melakukan pergeseran sebesar 8 karakter.

langkah-2



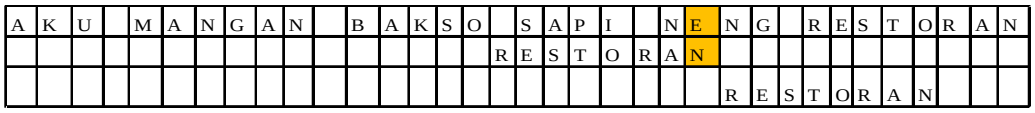

-”O” tidak cocok dengan “N”

-tabel OH : karakter “O” nilai pergeserannya = 3 belum ada karakter yang cocok

-tabel MH : ketidakcocokan pada posisi 8 (karakter “N”) nilai pergeserannya = 1

-sehingga geser *string* sebesar 8 posisi (nilai maksimal dari kedua tabel pergeseran)

langkah-3



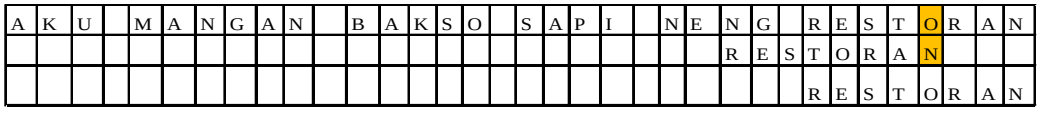

-”E” tidak cocok dengan “N”

-tabel OH : karakter “E” nilai pergeserannya = 6 belum ada karakter yang cocok

-tabel MH : ketidakcocokan pada posisi 8 (karakter “N”) nilai pergeserannya = 1

-sehingga geser *string* sebesar 8 posisi (nilai maksimal dari kedua tabel pergeseran)

langkah-4

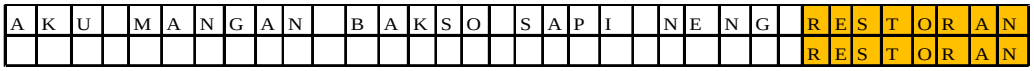



-”O” tidak cocok dengan “N”

-tabel OH : karakter “O” nilai pergeserannya = 3 belum ada karakter yang cocok

-tabel MH : ketidakcocokan pada posisi 8 (karakter “N”) nilai pergeserannya = 1

-sehingga geser *string* sebesar 3 posisi (nilai maksimal dari kedua tabel pergeseran)




-sehingga geser *string* sebesar 3 posisi (nilai maksimal dari kedua tabel pergeseran) setelah digeser 3 karakter, *pattern* ditemukan.

Algoritma *boyer-moore* adalah salah satu algoritma pencarian *string* yang paling efisien yang digunakan untuk berbagai aplikasi saat ini. kelebihan algoritma ini dibandingkan algoritma yang lain adalah algoritma ini dapat melakukan lompatan pengecekan jika terjadi ketidakcocokan karakter antara *pattern* dan teks. Dengan metode ini, jumlah pengecekan karakter yang diperlukan akan menjadi lebih sedikit dan waktu pencarian akan menjadi lebih cepat.

### III.2.2. Daftar Kebutuhan

Daftar kebutuhan merupakan daftar yang menguraikan kebutuhan-kebutuhan pengguna yang harus disediakan oleh perangkat lunak baik kebutuhan fungsional maupun non fungsional. Sebelum dapat membuat daftar kebutuhan, perlu dirumuskan terlebih dahulu aktor-aktor yang menggunakan sistem ini. Tabel dibawah ini akan memperlihatkan aktor-aktor beserta penjelasannya yang merupakan hasil dari proses identifikasi aktor.

**Tabel III.3 Deskripsi Aktor**

| No | *User* | Penjelasan |
|---|---|---|
| 1. | *User* | Aktor yang menggunakan aplikasi kamus tanpa proses *login* |
| 2. | *Admin* | Aktor yang telah melakukan login pada sistem, bertugas untuk memanipulasi data dan memiliki hak-hak tertentu sesuai kapasitas dan keperluannya. Termasuk perbaikan sistem apabila terjadi gangguan pada sistem |

**Tabel III.4 Daftar Kebutuhan fungsional**

| No | *Use case* | Kebutuhan |
|---|---|---|
| 1. | *Login* admin | Sistem harus memberikan fasilitas untuk *login* untuk administrator untuk memudahkan manajemen sistem. |
| 2. | *Logout* admin | Sistem harus menyediakan fasilitas untuk *logout* agar administrator dapat keluar dari sistem. |
| 3. | Menambah data kamus | Sistem harus menyediakan fasilitas untuk menambah referensi kata pada kamus |
| 4. | Menghapus data kamus | Sistem harus menyediakan fasilitas untuk menghapus data |
| 5. | Mengubah data kamus | Sistem harus menyediakan fasilitas untuk mengubah data |
| 6. | Menentukan pencari kata | Sistem harus menyediakan fasilitas untuk mencari kata |

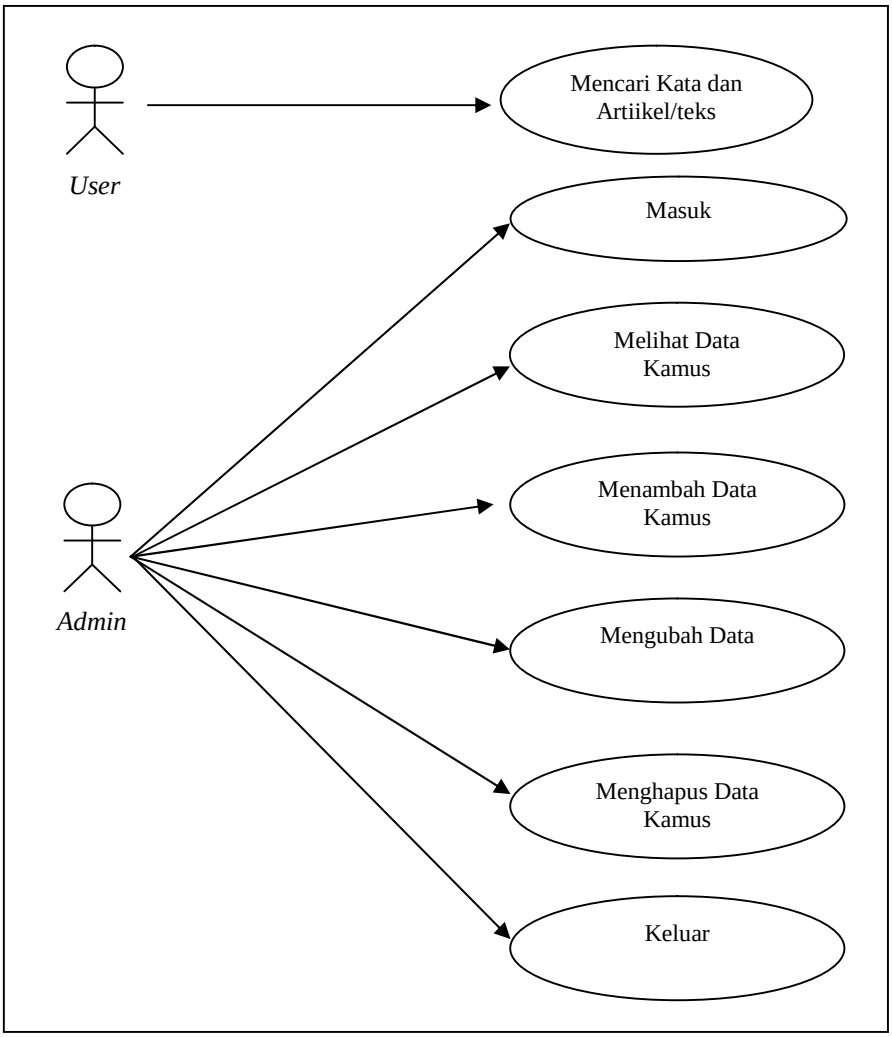
**Tabel III.5 Daftar Kebutuhan non fungsional**

| No | Kebutuhan |
|---|---|
| 1. | Sistem di kembangkan dengan menggunakan bahasa pemrograman *Visual Basic.Net* 2008 dan MySQL sebagai *database* nya |

## III.3. Desain Sistem

### III.3.1. *Use Case Diagram*

*Use case diagram* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar kegiatan, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Berikut ini adalah *use case diagram* dari aplikasi kamus :

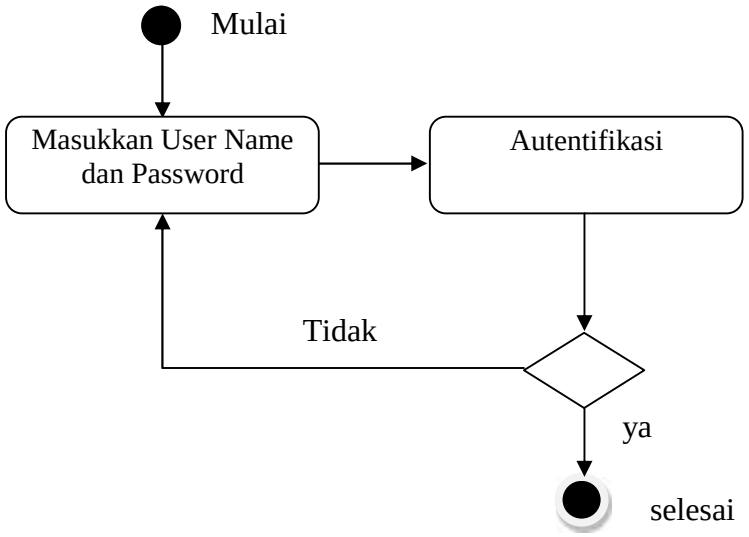



**Gambar III.1 *Use Case Diagram* Aplikasi Kamus**

### III.3.2. *Activity Diagram*

*Activity diagram* adalah teknik untuk mendiskusikan logika *prosedural*, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* banyak mempunyai peran seperti halnya *flowchart*, akan tetapi *flowchart* berbeda dengan *Activity diagram*. Berikut ini adalah *activity diagram* yang digunakan dalam merancang program aplikasi Kamus.

1. *Activity Diagram* untuk Login




**Gambar III.2. *Activity* Diagram *Login***

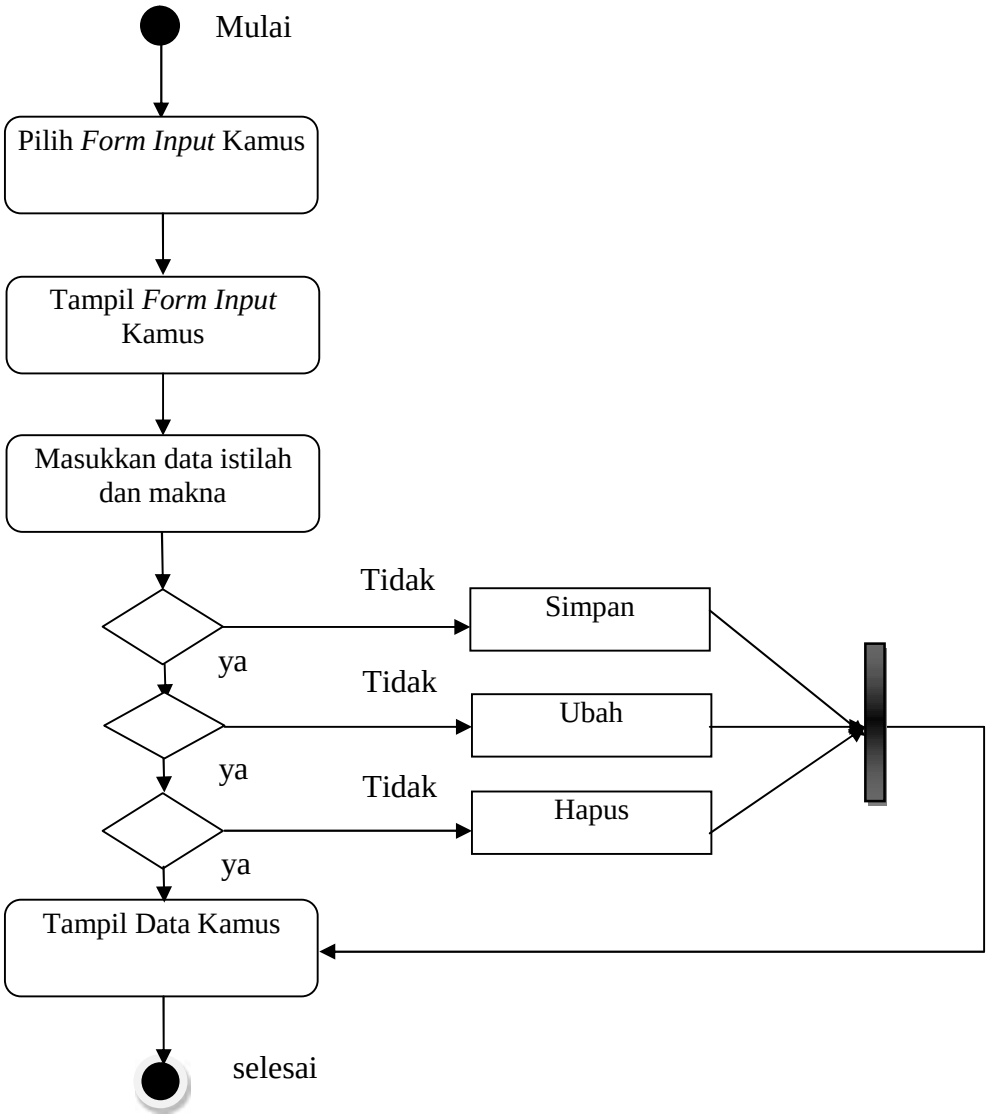
Proses *login* :

a. Masukkan *UserName* dan *Password*

b. Sistem akan melakukan autentifikasi data user

- Jika ya, *user* dapat menggunakan sistem
- Jika tidak, *user* tidak dapat menggunakan sistem

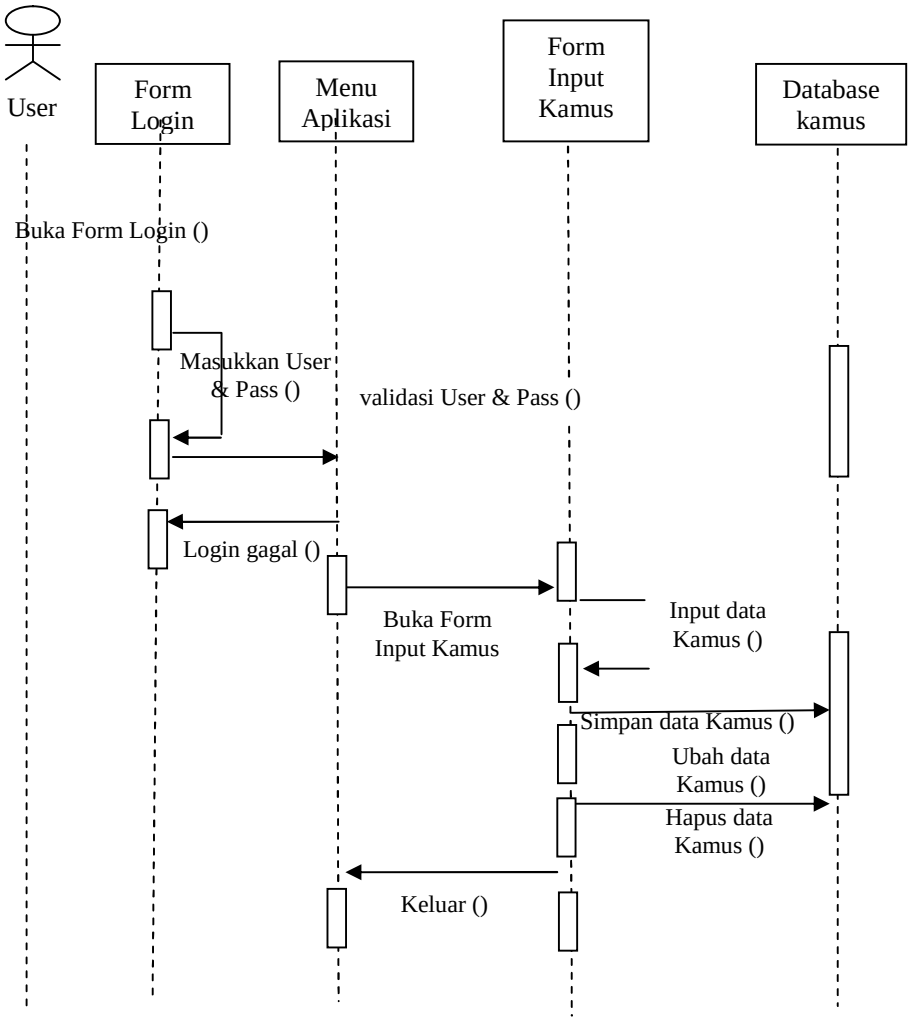
2. *Diagram* Data Input Kamus




**Gambar III.3. *Activity Diagram* Data *Input* Kamus**

### III.3.3. *Sequence diagram*

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan disekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu di dalam *use case*. Berikut ini adalah *sequence diagram* sistem pengolahan nilai siswa :



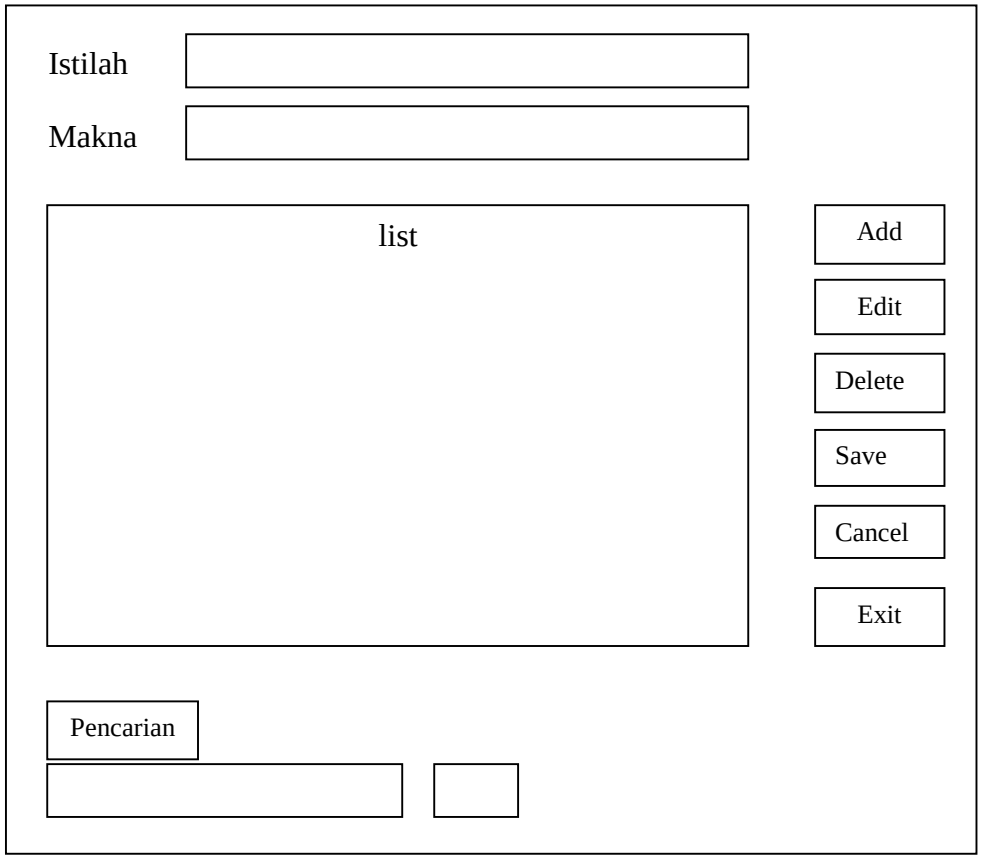

**Gambar III.4. *Sequence Diagram* Mengolah Data Input Kamus**

### III.3.4. Desain *Inputan*

Pada *desain input* ini akan di *desain* jenis-jenis *Form* yang terdiri dari rancangan inputan dari sistem yang dibangun. Adapun gambar-gambarnya dapat dilihat pada gambar di bawah ini :

1. *Form Input* Kamus

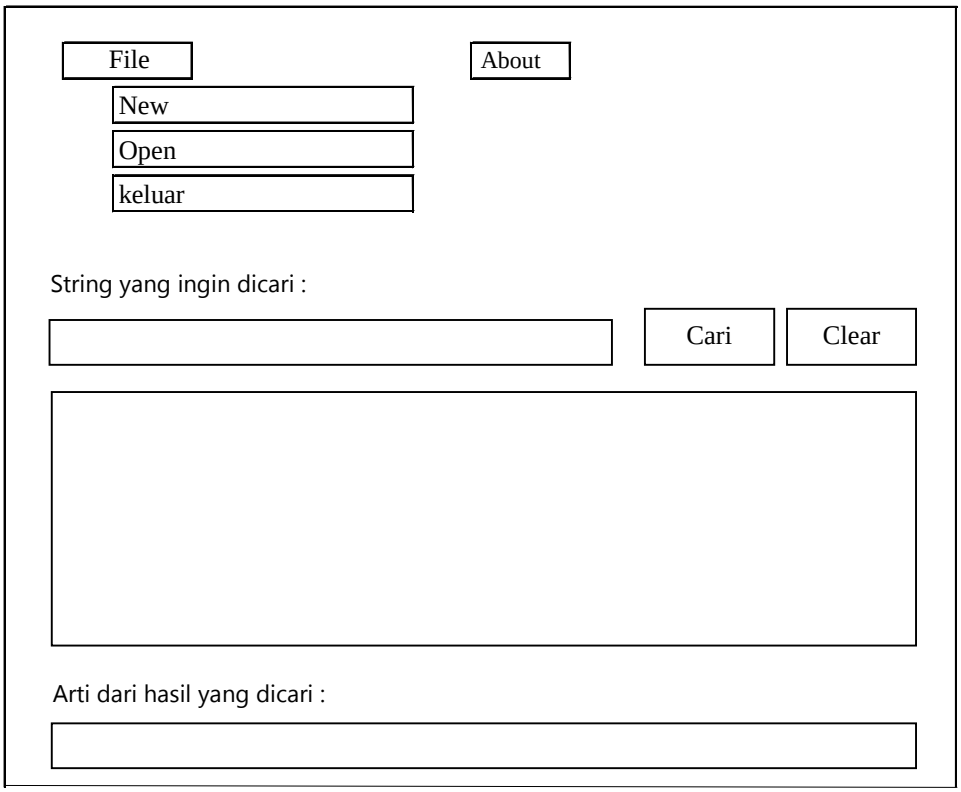
   *Form Input* digunakan untuk menambah kata-kata baru, menghapus, serta mengedit kata-kata yang ada dalam *database*. *Form* ini juga mempunyai 1 buah *listview* yang berfungsi untuk menampilkan isi dari *database*, atau menampilkan kata-kata yang sudah masuk dalam *database*.




**Gambar III.5. *Desain Input* Data**

2. *Form* Pencari Artikel

*Form* pencari artikel ini merupakan *form* untuk mencari suatu kata dalam artikel. *Form* ini merupakan implementasi dari algoritma *boyer moore*.

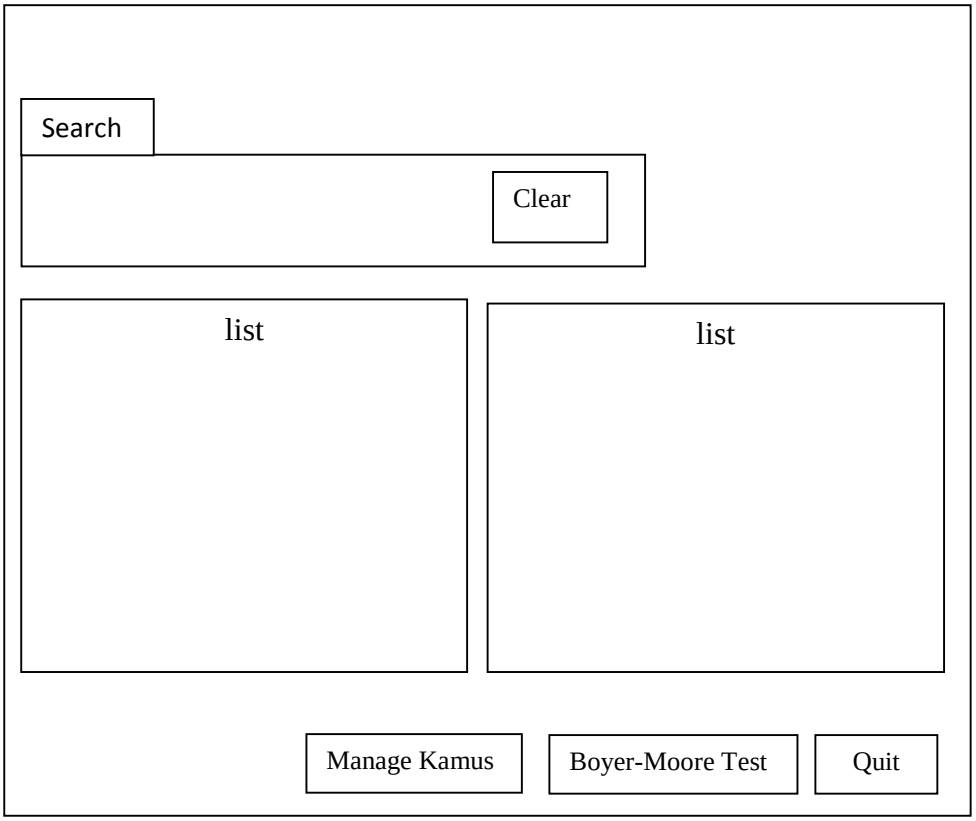



**Gambar III.6. *Desain Form* Artikel**

### III.3.5. Desain *Output*

Perancangan *output* adalah perancangan *form* yang digunakan sebagai hasil dari kamus tersebut. Dalam *form* ini kita bisa melihat istilah-istilah yang telah di *input* melalui *form Input*, serta *searching* dalam pencarian istilah. Untuk mencari istilah di kamus ini, ketikkan istilah pada *textbox* pada kotak pencarian sebelah kanan, kemudian akan keluar *output*-nya secara langsung pada *list view*.

Jika tidak terdapat istilah di dalam *database* maka tidak akan keluar *output* apapun.




**Gambar III.7. Layar Utama *Aplikasi* Kamus**

### III.4. Rancangan *Database*

*Database* adalah sekumpulan data yang terdiri dari suatu table yang saling berhubungan. Fungsi dari suatu *database* adalah untuk menampung beberapa tabel dan *query* yang digunakan sebagai sumber pengolahan data. Untuk membuat dan mengakses *database* dapat mempergunakan beberapa cara, tetapi yang di rancang dalam aplikasi ini adalah database dengan *File IO (File Input Output)*

karena dapat membuat tampilan database sesuai dengan keinginan perancang. *Database* ini didukung oleh form yang menampilkan data melalui kontrol *list view*, *kontrol list view* merupakan sebuah kontrol yang berfungsi melakukan pencetakan ke layar.

*Database* yang saya gunakan dalam perancangan ini adalah kamusdb.myi dari *mysql server*. Dalam *database* ini terdapat 1 tabel yang digunakan untuk penyimpanan istilah-istilah yang akan dimasukkan oleh pengguna. Berikut adalah tampilan rancangan tabel yang dipergunakan, yaitu tabel kamus

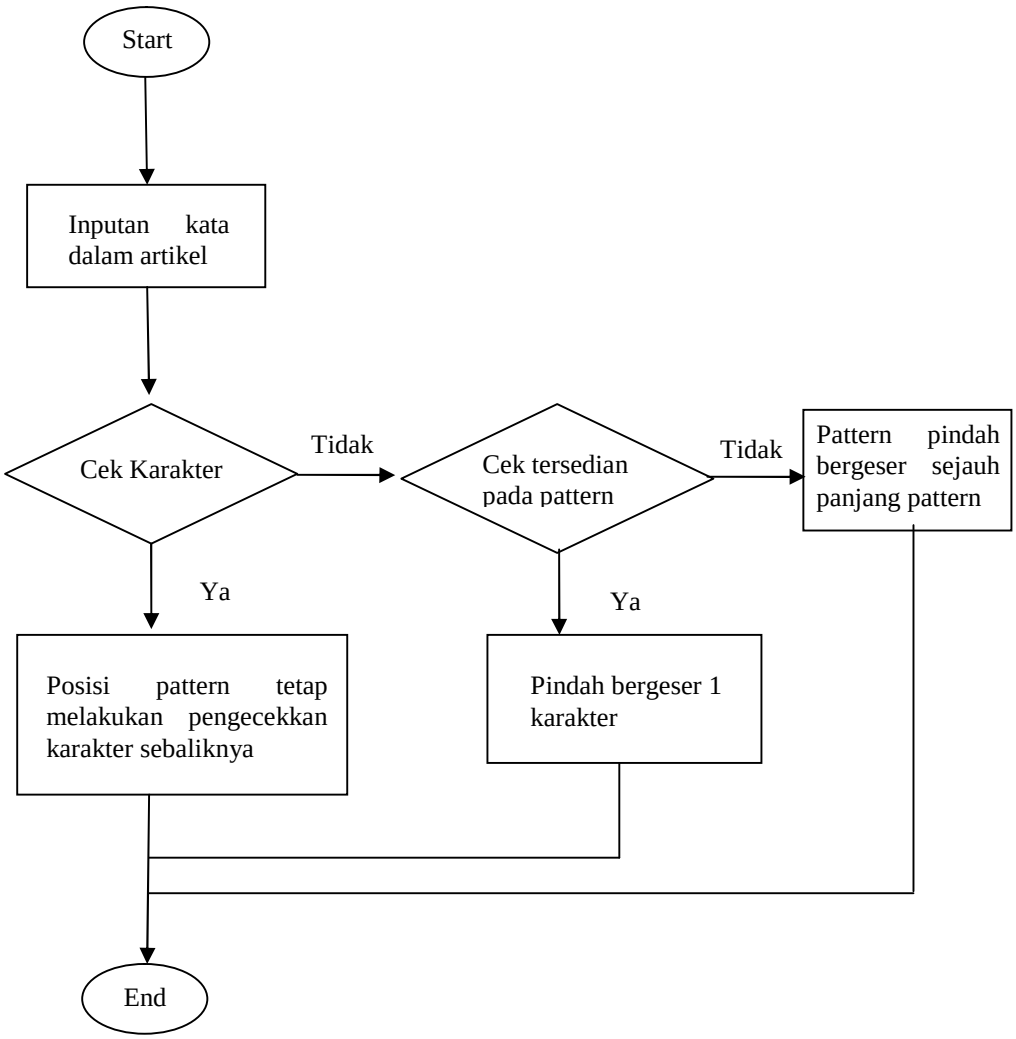
**Tabel III.6 Kamus**

| ID | Istilah | Makna |
|---|---|---|
| 1. | Abot | Berat |
| 2. | Abang | merah |

.

### III.5. *Flowchart* Pembuatan Program

*Flowchart* merupakan bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. *Flowchart* menjelaskan tentang urutan-urutan dari prosedur yang ada di dalam sistem dengan menggunakan simbol-simbol.




**Gambar III.8 *Flowchart* Metode *Boyer Moore***

Penjelasan dari *flowchart* di atas adalah sebagai berikut :

Pertama kita akan membuka aplikasi kamus dan memilih *form* pencari artikel, kemudian kita masukan satu kata yang ingin kita cari dari artikel tersebut. Setelah itu sistem akan melakukan pengecekkan dalam *database*. Jika ada akan melakukan pengecekakan karakter sebaliknya ataupun Jika tidak ada maka akan dicari ketersediaan *pattern*, jika ada akan melakukan pergeseran 1 karakter dan jika tidak ditemukan akan melakukan pergeseran sejauh panjang *pattern* dan memperoleh hasil.