BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas implementasi berdasarkan metodologi dan perancangan yang telah diuraikan pada bab sebelumnya. Implementasi yang akan dibahas pada bab ini terdiri dari lingkungan implementasi, implementasi program, implementasi antarmuka, dan implementasi uji coba.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan diuraikan pada sub bab ini meliputi lingkungan perangkat keras dan lingkungan perangkat lunak yang digunakan sebagai pengembangan perangkat lunak menggunakan algoritma *Rabin Karp* untuk implementasi *Question Answering System*.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan aplikasi implementasi *Question Answering System* menggunakan algoritma *Rabin Karp* menggunakan *personal computer* dengan spesifikasi :

1. Prosesor Intel Core 2 Duo @ 2 Ghz
2. Memory 2048 MB RAM
3. Monitor 14.1”
4. Harddisk dengan kapasitas 250 GB
5. Keyboard
6. Mouse

4.1.2 Lingkungan Implementasi Perangkat Lunak

Adapun perangkat lunak yang digunakan dalam pembuatan aplikasi implementasi *Question Answering System* menggunakan algoritma *Rabin Karp* meliputi :

1. Sistem Operasi *Windows 7* Ultimate
2. XAMPP 1.7.3
3. PHP Version 5.3.1
4. Apache 2.2.14
5. MySQL 5.1.41
6. Microsoft Office 2007
7. Enterprise Architect 8.0.860
8. Macromedia Dreamweaver 8
9. Mozilla Firefox 20.0.1

4.2 Implementasi Program

4.2.1 Kelas dan Fungsi

Pada tahap implementasi sistem, untuk membuat program aplikasi digunakan sistem pemrograman yang berorientasikan kepada objek. Sebagai dasar dari modularitas dan struktur dalam pemrograman berorientasi objek implementasinya didefinisikan dalam sebuah kelas (*class*). Semua data dan fungsi di dalam program dibungkus dalam kelas-kelas ini. Kelas-kelas tersebut terdiri dari kelas Dokumen, kelas *Preprocessing*, kelas *Rabin Karp* dan kelas *Rule Based*. Berikut penjabaran kelas-kelas tersebut :

1. Kelas Dokumen

Kelas Dokumen digunakan untuk searching dokumen berdasarkan *keyword* pertanyaan. Dokumen-dokumen ini disimpan dalam *database*, untuk mengambil data tersebut digunakan beberapa fungsi, yaitu :

a. getDokumen()

   Fungsi untuk mendapatkan dokumen yang relevan berdasarkan *keyword* pencarian yang diinputkan. *Output* yang dikeluarkan dari fungsi ini berupa *array* yang berisi kalimat asli dan kalimat yang telah di*tagging*.

b. getStopword()

Fungsi untuk mendapatkan daftar kata *stopword* (kata-kata yang tidak penting) yang tersimpan dalam *database*. Daftar kata *stopword* ini nantinya digunakan untuk proses *filtering*.

c. getKtdasar()

Fungsi untuk mendapatkan daftar kata dasar bahasa Indonesia yang tersimpan dalam *database*. Daftar kata dasar ini nantinya digunakan untuk proses *stemming*.

d. getKtkerja()

Fungsi untuk mendapatkan daftar kata kerja yang tersimpan dalam *database*. Daftar kata kerja ini nantinya digunakan untuk proses pembobotan pada metode *Rule Based*.

2. Kelas Preprocessing

Kelas preprocessing digunakan untuk mengurangi *noise* pada data agar lebih terstruktur. Proses-proses yang dilakukan pada *preprocessing* ini adalah *case folding*, *tokenizing*, *filtering*, dan *stemming*. Berikut

a. tokenizingSubstring()

Pada fungsi ini dilakukan proses mengubah semua huruf menjadi bentuk huruf kecilnya (*lowercase*), kemudian dilakukan penguraian kalimat menjadi kata-kata. Karakter selain huruf dihilangkan dan *dianggap delimiter*. Adapun *delimiter* yang didaftarkan pada sistem ini adalah untuk tanda baca '.', ',', "", """, '-', '/', '{', '}', '+', '_', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '?', '>', '<', '[', ']', '|', '`', '~', ';', ':', '=', '\\', "\n", "\r". *Delimiter* juga dianggap sebagai pemisah kata.

b. filtering()

Fungsi ini digunakan untuk membuang kata - kata yang kurang penting berdasarkan daftar *stopword*. Bila kata *stopword* ditemukan maka kata dihilangkan, tetapi bila kata yang dicari tidak ditemukan dalam *database* maka kata dijadikan kata penting.

c. stemming()

Fungsi ini digunakan untuk mengubah tiap kata dari kalimat menjadi *root word* atau kata dasarnya.

3. Kelas Rabin Karp

Kelas Rabin Karp digunakan untuk penghitungan K-Gram, *Hashing*, dan pembobotan heuristik Rabin Karp. Berikut fungsi yang terdapat dalam kelas ini :

a. kgramParsing()

Fungsi ini digunakan untuk *parsing* memecah kata berdasarkan K-Gram yang diinputkan.

b. hashing()

Fungsi ini digunakan untuk *hashing* kata berdasarkan K-Gram yang telah diproses sebelumnya.

c. tresholdHeuristik()

Fungsi ini digunakan untuk menghitung nilai *threshold*.

d. countMatch()

Fungsi ini digunakan untuk menghitung nilai *count match*.

e. nilaiSM()

Fungsi ini digunakan untuk menghitung nilai kata kueri pertanyaan yang terdapat dalam kandidat jawaban.

f. nilaiORD()

Fungsi ini digunakan untuk menghitung nilai urutan kata yang terdapat dalam kandidat jawaban.

g. ruleApa()

Fungsi ini digunakan untuk memberikan jawaban berdasarkan aturan pada *rule* Apa.

h. ruleSiapa()

Fungsi ini digunakan untuk memberikan jawaban berdasarkan aturan pada *rule* Siapa.

i. ruleBerapa()

Fungsi ini digunakan untuk memberikan jawaban berdasarkan aturan pada *rule* Berapa.

j. ruleKapan()

Fungsi ini digunakan untuk memberikan jawaban berdasarkan aturan pada *rule* Kapan.

k. ruleDimana()

Fungsi ini digunakan untuk memberikan jawaban berdasarkan aturan pada *rule* Dimana.

4. Kelas Rule Based

Kelas Rule Based digunakan untuk penghitungan *wordmatch* dan pembobotan aturan Rule Based. Berikut fungsi yang terdapat dalam kelas ini :

a. wordmatch()

Fungsi ini digunakan untuk memberikan skoring kata yang cocok dengan *keyword* dengan skor yang lebih tinggi untuk kata kerja.

b. ruleApa()

Fungsi ini digunakan untuk memberikan skoring jawaban berdasarkan aturan pada *rule* Apa.

c. ruleSiapa()

Fungsi ini digunakan untuk memberikan skoring jawaban berdasarkan aturan pada *rule* Siapa.

d. ruleBerapa()

Fungsi ini digunakan untuk memberikan skoring jawaban berdasarkan aturan pada *rule* Berapa.

e. ruleKapan()

Fungsi ini digunakan untuk memberikan skoring jawaban berdasarkan aturan pada *rule* Kapan.

f. ruleDimana()

Fungsi ini digunakan untuk memberikan skoring jawaban berdasarkan aturan pada *rule* Dimana.

4.2.2 Tahapan Pemrosesan

1. Pada pembobotan heuristik Rabin Karp

Setelah user memasukkan kalimat pertanyaan pada sistem langkah pertama yang dilakukan adalah memecahnya menjadi kata tanya dan *keyword*. Kata tanya digunakan untuk mengidentifikasi tipe dari pertanyaan tersebut sehingga dapat ditentukan *named entity* yang akan dicari untuk dapat menemukan kandidat jawaban. *Keyword* digunakan untuk mencari dokumen yang relevan dapat dilihat pada kode sumber 4.1.

```
function getDokumen($search){

      $query = mysql_query("SELECT * FROM document WHERE $search");
      while ($row = @mysql_fetch_array($query)){
            $result['dokumen_asli'][] = trim($row['dokumen_asli']);
            $result['dokumen_tagging'][] =
trim($row['dokumen_tagging']);
      }

      return $result;

}
```

Kode Sumber 4.1 *Source code* proses cari dokumen

Sumber : [Implementasi]

Setelah dokumen teks berhasil didapatkan oleh sistem selanjutnya dilakukan tahap *preprocessing* awal yaitu *case folding* dan *tokenizing* yang dapat dilihat pada kode sumber 4.2, mengubah semua huruf menjadi huruf kecil semua menggunakan dan menghilangkan tanda bacanya. Menghilangkan *delimiter* tanda baca dan menggantinya menjadi spasi, kemudian memecah string dalam bentuk kata.

```
function tokenizingSubstring ($string)
{
   /* Proses tokenizing/parsing dan penghilangan karakter selain
huruf
      output -> array per-kata
   */
      $a=0;
      $string     = strtolower($string); // casefolding
      $delimiter  = array('.',',','"',"'",'-
','/','{','}','+','_','!','@','#','$','%','^','&','*','(',')','?','>
','<','[',']','|','`','~',';',':','=','\\',"\n","\r"); //tanda baca
      $teksTemp   = str_replace($delimiter," ",$string);
//menghilangkan tanda baca diganti spasi

      $temp       = explode(" ",trim($teksTemp)); //
tokenizing/parsing

            for($i=0;$i<count($temp);$i++){
               // Pengecekan jika ada kalimat yang tidak standar
misal: pergi.kemana
               if($temp[$i]=='')
               unset($temp[$i]);
            }

      // Pengisian nilai array
      foreach ($temp as $word):
         if($word!=''){
            $result[$a] = trim($word);
            $a++;
         }

      endforeach;

      return $result;
}
```

Kode Sumber 4.2 *Source code* proses *case folding* dan *tokenizing*

Sumber : [Implementasi]

Proses selanjutnya adalah *filtering* untuk menghilangkan kata-kata yang tergolong *stopwords.* Kemudian kata-kata yang tidak termasuk dalam *stopwords* akan dilakukan proses *stemming*. Proses *filtering* dapat dilihat pada kode sumber 4.3

```
function filtering($string){
   /* Menghilangkan kata tidak penting stopword (dari database)
        output -> array => (array kata2 penting dan kalimat unik
yang digabung tanpa spasi)
      */

   $query = mysql_query("SELECT * FROM m_stopword");
   while ($row = @mysql_fetch_array($query)){
         $stopword[] = trim($row['stopword']);
   }

         if($string)
                  $string = array_diff($string, $stopword);

   $result['array_kata'] = $string; //menyimpan dlm array kata2
penting saja (tp index array-nya masi belum urut)
   //$result = $string;

            // Menngabungkan kata2 menjadi kalimat tanpa spasi
            foreach($string as $kata):

      $result['kalimat_unik']=$result['kalimat_unik'].trim($kata);
            endforeach;

   return $result;

}
```

Kode Sumber 4.3 *Source code* proses *filtering*

Sumber : [Implementasi]

Setelah proses *preprocessing* terhadap pertanyaan dan dokumen, langkah selanjutnya adalah mencari kandidat jawaban (*passages*) yang sesuai berdasarkan *rule* untuk pertanyaan Apa, Kapan, Siapa, Berapa, dan Dimana. Pencarian dilakukan dengan melakukan penamaan entitas untuk proses perolehan kandidat jawaban sesuai dengan jenis pertanyaan. *Named entity* yang digunakan terdiri dari *DESCRIPTION*, *NAME*, *ORGANIZATION*, *LOCATION*, *CURRENCY*, *DATE*, *TIME* dan *NUMBER*. Proses pencarian *rule* dapat dilihat pada kode sumber 4.4

```
function ruleApa($arrDokumen){

      $needleStart =  "<description>";
      $needleEnd =  "</description>";
```

```
        for($i=0;$i<count($arrDokumen);$i++)
        {
                $result = substr($arrDokumen[$i],
strpos($arrDokumen[$i], $needleStart)+strlen($needleStart));
                $result = substr($result, 0, strpos($result,
$needleEnd));
                if($result)
                {
                        $hasil[] = $arrDokumen[$i];
                }
        }

        return $hasil;
}
```

Kode Sumber 4.4 *Source code* proses pencarian kandidat berdasarkan rule

Sumber : [Implementasi]

Parsing untuk masing-masing kandidat jawaban dibagi ke dalam K-Gram dengan jumlah K-Gram yang telah ditentukan oleh *user*. Proses *parsing* K-Gram dapat dilihat pada kode sumber 4.5

```
function kgramParsing($string, $kgram){
        /* Fungsi untuk memparsing kalimat menjadi substring-substring
sebanyak k-gram
                output -> array(substring)
        */
        $kgramAwal  = $kgram;
        $jum        = strlen($string);
        $temp       = 0;
        $kgram_temp = 0;
        $batas      = $kgramAwal-1;

        for($i=0;$i<$jum-$batas;$i++)
        {
                // Menyimpan substring yang telah di bagi per k-gram
kedalam array
                $res[] = substr($string,$i,$kgramAwal);
                $temp  = $kgram;
                $kgram = $kgram+1;
        }

        return $res;
}
```

Kode Sumber 4.5 *Source code* proses *parsing* K-Gram

Hasil *parsing* K-Gram selanjutnya dilakukan *hashing*, dengan nilai *modulo* ditentukan oleh *user*. Nilai *modulo* menentukan panjang hasil *hashing*. Proses *parsing* K-Gram dapat dilihat pada kode sumber 4.6

```
function hashing($kata,$posisi){

      $basis = 10;
      $modulo = 101;
      $n = 0;

      $jum_huruf = strlen($kata);
      $result['string'] = $kata;
      $result['posisi'] = $posisi;

            for($i=$jum_huruf-1;$i>=0;$i--){

              $temp                =
pow($basis,$i)*ord($kata[$n]);//$this->basis*ord($kata[$n]); //rumus
hashing
              $result['hash_n'][] = $temp; //dibutuhkan u/ yg rabin
jika hashnya sama dlakukan pngecekan lg

              $result['hash']      = $result['hash']+$temp;
              $result['ketemu']    = false;
              $n++;

            }

      $result['hash_mod'] = $result['hash']%$modulo;


      return $result;
}
```

Kode Sumber 4.6 *Source code* proses *hashing*

Sumber : [Implementasi]

Setelah proses Rabin-Karp selesai, kemudian dilakukan perhitungan pembobotan heuristik. Pembobotan dilakukan dengan mengikuti tahapan yang terdapat pada jurnal Ballesteros & Xiaoyan-Li (2007) yang dijadikan acuan dalam penelitian ini, pembobotan *passages* terdiri dari :

1. Pembobotan *passages* dari proses *wordmatch* sesuai *threshold* diproses pada fungsi `tresholdHeuristik()`
2. Nilai *count match* dihitung pada fungsi `countMatch()`
3. Pembobotan *passages* berdasarkan jumlah kata cocok yang terdapat dalam *passages* diproses pada fungsi `nilaiSM()`
4. Pembobotan *passages* berdasarkan urutan kata yang cocok diproses pada fungsi `nilaiORD()`
5. Pembobotan berdasarkan hasil dari *wordmatch* dibagi dengan jumlah kata dari masing-masing *passage* disimpan dalam variabel nilai W.

```
function tresholdHeuristik($query, $match){

	$t = 0;

	// Jika count_queri kurang dari 4, t=count_queri
	if ($query<4)
	{
		$t = $query;
	}

	// Jika count_queri antara 4 dan 8, t=count_queri/2.0+1.0
	elseif ($query>=4 && $query<=8) {
		$t = ($query/2)+1;
	}


	// Jika lebih besar dari 8, t=count_queri/3.0+2.0
	elseif ($query>8) {
		$t = ($query/3)+2;
	}

	return $t;
}
```

Kode Sumber 4.7 *Source code* proses perhitungan *threshold*

Sumber : [Implementasi]

```
function countMatch($treshold, $match){

	if ( $match < $treshold )
	{
		$countMatch = 0;
```

```
        }
        else
        {
                $countMatch = $match;
        }

        return $countMatch;
}
```

Kode Sumber 4.8 *Source code* proses perhitungan *count match*

Sumber : [Implementasi]

```
function nilaiSM($teks1, $teks2){

        $txt1       = strtolower($teks1);
        $txt2       = strtolower($teks2);
        $smValue    = 0;

        $temp = explode(" ",trim($txt1));

        for($i=0;$i<count($temp);$i++)
        {
                if(strpos($txt2,$temp[$i]))
                {
                        $smValue = 1;
                }
        }

        return $smValue;
}
```

Kode Sumber 4.9 *Source code* proses perhitungan nilai SM

Sumber : [Implementasi]

```
function nilaiORD($teks1, $teks2){

        $txt1       = strtolower($teks1);
        $txt2       = strtolower($teks2);

        $posisi = strpos($txt2,$txt1);

        if($posisi === false)
        {
                $ordValue = 0;
        }
        else
        {
```

```
            $ordValue = 1;
        }

        return $ordValue;
}
```

Kode Sumber 4.10 *Source code* proses perhitungan nilai ORD

Sumber : [Implementasi]

2. Pada pembobotan Rule Based

Langkah pertama dalam metode ini diawali dengan pengguna memasukkan kueri berupa kalimat tanya. Dalam proses pencarian dokumen yang relevan langkahnya sama dengan pencarian dokumen pada metode Rabin Karp. Setiap kalimat dokumen dan kalimat kueri sama-sama masuk ke dalam proses *preprocessing* yang akan menghasilkan *token-token*. *Token-token* pada setiap kalimat dokumen dan kalimat kueri dibandingkan dalam proses *WordMatch*, kemudian masuk ke dalam *rule* sesuai dengan tipe kueri yang diberikan.

```
function getKtkerja(){

        $query = mysql_query("SELECT * FROM kamus_kata WHERE
keterangan='v'");
        while ($row = @mysql_fetch_array($query)){
              $result[] = trim($row['kata_dasar']);
        }
        return $result;
}
```

Kode Sumber 4.11 *Source code* proses menyimpan kata kerja

Sumber : [Implementasi]

```
public function wordmatch ($question, $sentence, $kata_kerja)
{
        $que  = strtolower($question);
        $sen  = strtolower($sentence);
        $skor = 0;

        $temp = explode(" ",trim($que));

        for($i=0;$i<count($temp);$i++)
```

```
	{
		if(strpos($sen,$temp[$i]))
		{
			if(in_array($temp[$i],$kata_kerja)){
				$skor = $skor + 6;
			}
			else
			{
				$skor = $skor + 3;
			}
		}
	}

	return $skor;
}
```

Kode Sumber 4.12 *Source code* proses perhitungan *wordmatch*

Sumber : [Implementasi]

```
function ruleApa($question, $sentence){

	$que  = strtolower($question);
	$sen  = strtolower($sentence);
	$skor = 0;

	// If contains (S, {"ialah", "adalah"}) Then Score += 4
	if( strpos($sen,"ialah") || strpos($sen,"adalah") )
	{
		$skor = $skor + 4;
	}

	return $skor;
}
```

Kode Sumber 4.13 *Source code* proses perhitungan *rule* apa

Sumber : [Implementasi]

```
function ruleDimana($question, $sentence){

	$que  = strtolower($question);
	$sen  = strtolower($sentence);
	$skor = 0;

	// If contains (S, keterangan tempat) ThenScore += 6

	$needleStart =  "<location>";
```

```
        $needleEnd =  "</location>";

        $result = substr($sen, strpos($sen,
$needleStart)+strlen($needleStart));
        $result = substr($result, 0, strpos($result, $needleEnd));
        if($result)
        {
                $skor = $skor + 6;
        }

        return $skor;

}
```

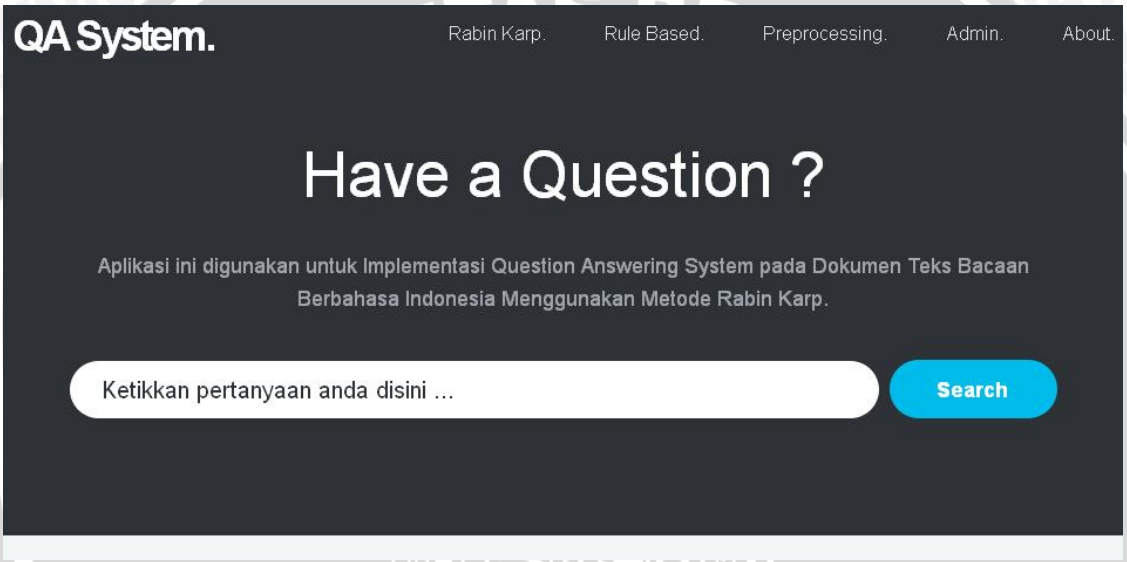
Kode Sumber 4.14 *Source code* proses perhitungan *rule* dimana

Sumber : [Implementasi]

4.3 Implementasi *User Interface*

Implementasi merupakan proses penerapan rancangan ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Implementasi sistem membuat dan menerapkan sistem secara utuh baik dari sisi perangkat keras maupun perangkat lunaknya. Sesuai dengan metodologi dan perancangan yang dijelaskan pada bab 3, pada bagian ini akan dijelaskan tentang implementasi *user interface* (antar muka) berdasarkan perancangan yang telah dilakukan sebelumnya. Implementasi *user interface* pada penelitian *Question Answering system* ini diterapkan pada aplikasi berbasiskan web menggunakan bahasa pemrograman. Gambar implementasi user interface ditunjukan pada gambar-gambar berikut :

Pada halaman utama sistem, *user* diminta untuk memasukkan pertanyaan yang akan diajukan pada sistem. Tipe pertanyaan yang dimasukkan untuk tipe pertanyaan APA, SIAPA, KAPAN, DIMANA, dan BERAPA. Tombol *search* adalah tombol yang digunakan untuk mengeksekusi pertanyaan yang telah diinputkan oleh *user* untuk mendapatkan jawaban dari sistem.

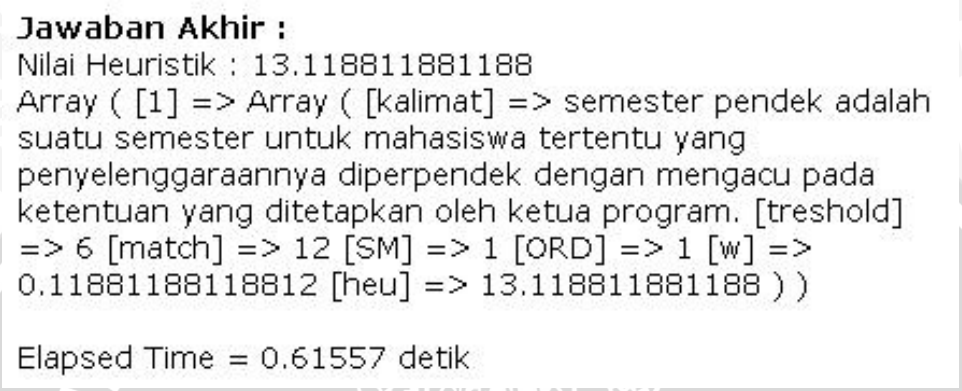


Gambar 4.1 Halaman utama

Sumber : [Implementasi]

Sistem akan menampilkan jawaban yang relevan dengan pertanyaan setelah dilakukan proses perhitungan. Adapun proses dalam melakukan pertanyaan yaitu normalisasi pertanyaan, *searching* dokumen, *preprocessing* (*case folding*, *tokenizing*, *filtering* dan *stemming*), dan perhitungan sesuai algoritma.

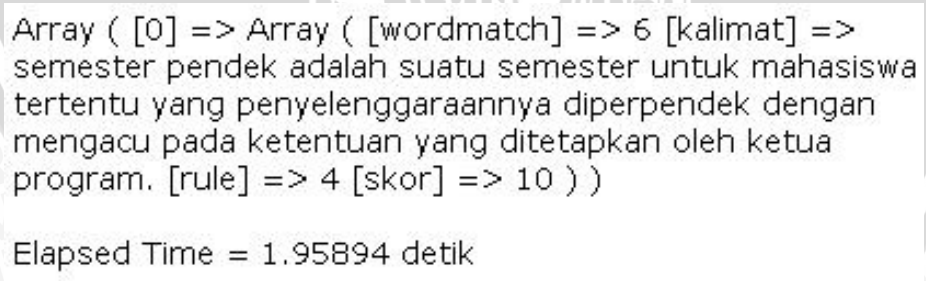
Pada algoritma *Rabin Karp* setelah *preprocessing* kemudian dihitung nilai *parsing* k-gram, *hashing*, *string matching*, dan pembobotan heuristiknya sampai ditemukan jawaban akhir dengan skor tertinggi.



Gambar 4.2 Halaman jawaban pembobotan Rabin Karp

Sumber : [Implementasi]

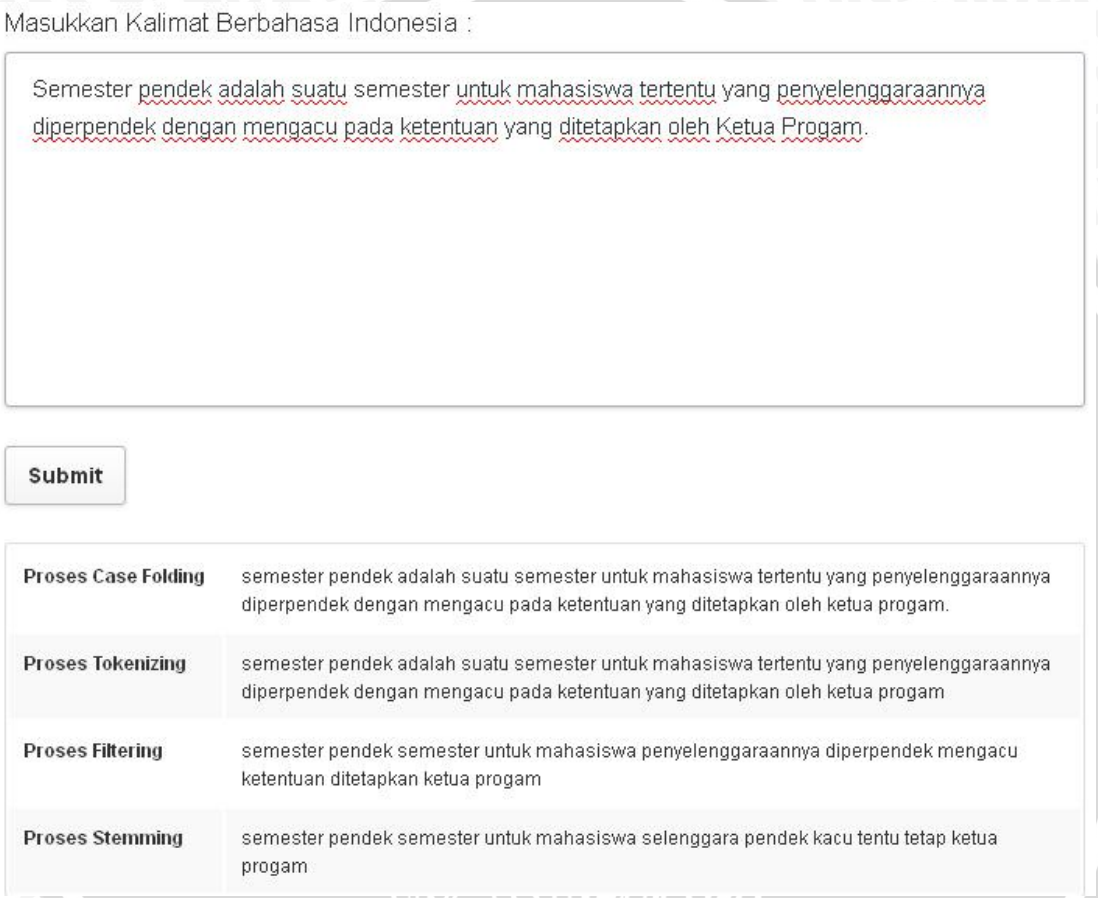
Pada algoritma *Rule Based* setelah melakukan *preprocessing* kemudian dihitung nilai *wordmatch* nya dan *scoring* jawaban di hitung sesuai aturan pada algoritma *Rule Based*. Jawaban akhir didapatkan dari kalimat dengan skor tertinggi.



Gambar 4.3 Halaman jawaban pembobotan Rule Based

Sumber : [Implementasi]

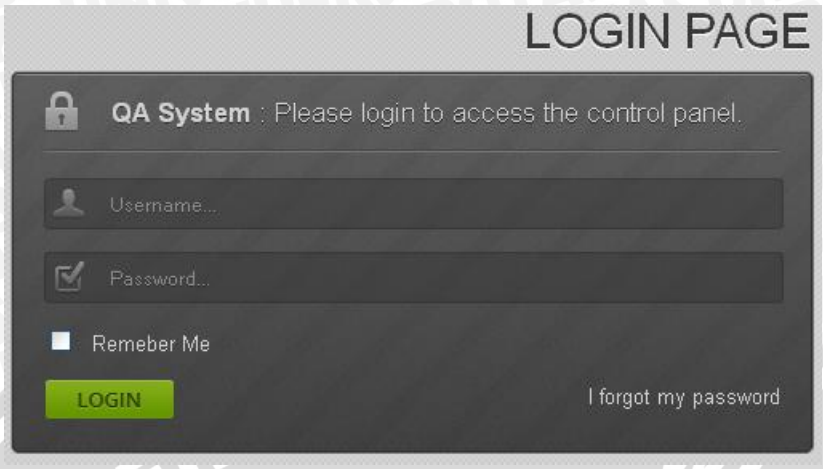
Juga terdapat rancangan interface untuk melakukan uji coba preprocessing secara terpisah diluar sistem *question answering*. *Preprocessing* kalimat terdiri dari *case folding*, *tokenizing*, *filtering* dan *stemming*.



Gambar 4.4 Halaman proses preprocessing

Sumber : [Implementasi]

Kalimat pertanyaan juga dianalisa berdasarkan tipe pertanyaan untuk mendapatkan pola pertanyaan, hal ini dimaksudkan untuk mendapatkan kandidat jawaban berdasarkan *named entity* pada pola pertanyaan. Oleh karena itu, diperlukan pengelolaan dokumen untuk melakukan *tagging* sesuai *named entity* nya, gambar implementasi pengelolaan dokumen ditunjukan pada gambar berikut :



Gambar 4.5 Halaman login pengelolaan dokumen

Sumber : [Implementasi]

Setelah user administrator login, maka akan muncul tampilan halaman pengelolaan dokumen untuk yang ditunjukan pada gambar berikut :

Gambar 4.6 Halaman pengelolaan *named entity tagging*

Sumber : [Implementasi]