**IMPLEMENTASI ALGORITMA *K-NEAREST NEIGHBOR* DALAM SISTEM *CASE BASED REASONING* UNTUK**

**PENCARI JAWABAN DARI SOAL ALGORITMA**

**TUGAS AKHIR**

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

pada jurusan Teknik Informatika Fakultas Sains dan Teknologi

Universitas Islam Negeri Sunan Gunung Djati Bandung

**Oleh**

**Nurida Ahsanti**

**1127050118**

****

**BANDUNG**

**2016M/1437H**

# BAB III

# ANALISIS DAN PERANCANGAN

## Analisis Sistem

Analisis sistem bertujuan untuk mengidentifikasi proses berjalannya suatu sistem yang akan dikembangkan. Analisis sistem dapat dijadikan sebagai acuan dasar dari pembuatan aplikasi baik itu dari segi kebutuhan fungsional, kebutuhan *non*-fungsional, pemodelan, deskripsi, analisis pengguna maupun desain yang akan digunakan dalam proses pengembangan perangkat lunak.

Penelitian ini dilakukan untuk membuat sebuah aplikasi yang dapat mencari kemiripan antara soal dan jawaban algoritma baru dengan seluruh pengetahuan yang aplikasi miliki. Aplikasi yang akan dibangun berbasis *java off-line,* hal ini disesuaikan dengan kebutuhan pengguna agar dapat mengakses sistem yang tidak bergantung pada koneksi internet. Aplikasi ini berguna untuk membantu para pelajar yang masih belum memahami kode program, sehingga dengan menjalankan aplikasi ini dapat membantu dalam mencari jawaban dari soal algoritma yang hasilnya berupa identitas jawaban dan gambaran potongan kode program berdasarkan tingkat kemiripan tertinggi dari hasil jawaban soal algoritma yang lalu.

Aplikasi ini dibangun dengan penerapan sistem *CBR* dan *KNN* yang melibatkan konsep *text mining* dalam proses pencarian jawaban. Tahapan yang dilakukan dalam pemprosesan terdiri dari *case folding,* *tokenizing, filtering, stemming,* pembobotan *similarity,* klasifikasi *KNN* dan algoritma *CBR* yang digunakan dalam pengetahuan kasus baru dan kasus lama. Adapun proses yang dilakukan ketika menginput pengetahuan adalah :

* 1. *Input*, admin harus login terlebih dahulu untuk dapat mengakses halaman. Hal ini dilakukan agar mencegah pengobrak-abrikan pengetahuan. Apabila halaman sudah dapat diakses, admin dapat memasukan data pengetahuan berupa soal, jawaban dan gambar potongan kode program.
  2. Proses, sistem akan mengolah data pengetahuan yang telah diinputkan, mulai dari membaca data yang berupa kalimat menjadi kata (*tokenzing*), selanjutnya melakukan proses penghapusan pada kata yang dianggap kata tidak penting (*filtering*), kemudian kata-kata yang tersisa tersebut dicari imbuhan baik awalan maupun akhiran sehingga dapat dijadikan kata dasar (*stemming*).
  3. Output, hasil proses disimpan ke dalam database.

Proses yang dilakukan ketika mencari jawaban adalah :

* 1. *Input*, pengguna memasukan data berupa soal.
  2. Proses, melakukan proses yang sama seperti cara penginputan oleh admin hanya saja setelah proses tersebut dilakukan akan dilanjutkan dengan proses pembobotan dengan *TF-IDF* dengan cara membandingkan dengan pengetahuan yang telah ada pada database, kemudian menggunakan metode perhitungan *similarity* dan pengklasifikasian *KNN.*
  3. Output, menampilkan hasil pencarian dan pengklasifikasian berupa identitas jawaban dan gambar potongan kode program. Setelah hasil proses didapatkan data akan disimpan sebagai pengetahuan baru ke dalam database ketika data tersebut memiliki hasil dibawah batasan nilai *tressholding*.

### Analisis Kebutuhan *Non­-*fungsional

Analisis kebutuhan *non-*fungsional dibutuhkan untuk membangun sistem yang terdiri dari:

1. Analisis Kebutuhan Perangkat Keras

Perangkat keras yang digunakan untuk membuat program menggunakan komputer dengan spesifikasi minimum sebagai berikut:

1. *Intel*® *Atom*™ InsideTM CPU @ 2.40GHz 2.40GHz;
2. RAM 2 GB (1.89 GB usable);
3. *Harddisk* 464.3 GB;
4. Sistem operasi *Windows* 7 Pro;
5. Analisis Kebutuhan Perangkat Lunak

Pembuatan pemodelan sistem dan pembuatan program dan simulasi membutuhkan perangkat lunak sebagai berikut.

1. *Java* *Development Kit* versi 1.7;
2. *Netbeans 7.2.1*
3. *Xampp*
4. *Mozilla Firefox*
5. Analisis Kebutuhan Pengguna

Karekteristik pengguna yang nantinya akan berperan dalam penggunaan aplikasi ini terbagi menjadi 2 kategori yaitu pengguna sebagai pemakai (*user*) dan pengguna sebagai pengelola data (*admin*).

### Analisis Kebutuhan *Fungsional*

Analisis kebutuhan *fungsional* merupakan penjabaran kebutuhan yang dibutuhkan pada aplikasi yang akan dibangun. Pada analisis kebutuhan *fungsional* mencangkupi deskripsi global dari aplikasi seperti yang terdapat pada tabel berikut:

Tabel 3.1 Kebutuhan *Fungsional*

| **No** | **Kebutuhan** | **Deskripsi** |
| --- | --- | --- |
| 1. | Aplikasi dapat mengelola data. | Aplikasi dapat mengelola data seperti membaca, menyimpan, mengubah dan menghapus data yang saling berhubungan dengan data yang ada pada database. |
| 2. | Aplikasi dapat mengolah soal. | Aplikasi dapat mengolah soal sebelum dimasukkan pada database. Pengolahan tersebut meliputi tahapan *Pre-processing (case folding*, *tokenisasi*, *filter* dengan menggunakan metode *stopword*, dan *stemming* dengan menggunakan algoritma porter). |
| 3. | Aplikasi dapat mencari identitas dan jawaban otomatis hanya dengan menginput soal. | Aplikasi dapat mencari identitas dan jawaban otomatis hanya dengan menginput soal. Proses pencarian tersebut pertama-tama menggunakan pengolahan pada soal yang diinputkan (*Pre-processing*), pemberian bobot menggunakan *TF-IDF*, pencarian kemiripan dengan metode *Similarity* dan algoritma *KNN*. |
| 4. | Aplikasi dapat mengelola soal dan jawaban baru. | Aplikasi dapat mengelola soal dan jawaban baru dengan menggunakan metode (*CBR*) yang mempunyai 4 tahapan yaitu: *Retrieve, Reuse, Revise* dan *Retain.* |

### Analisis Data

Untuk menjalankan sistem ini dibutuhkan data dari dari admin untuk dijadikan pengetahuan yang dianggap benar dan data dari pengguna berupa soal algoritma. Dari data-data tersebut dapat dijadikan perbandingan antara data baru dan data yang sudah ada di dalam *database.* Hal ini dilakukan agar dapat dijadikansebagai acuan terhadap perhitungan nilai kemiripan dan pengkategorian dalam menentukan apakah data tersebut sebagai pengetahuan baru atau bukan. Jika data tersebut merupakan pengetahuan baru, maka akan disimpan kedalam database.

### Analisis Proses *Text Maining*

Sistem yang dibangun menggunakan tahapan-tahapan *text maining* sehingga dibagi menjadi beberapa tahapan proses, setiap tahap proses mempunyai fungsi yang berbeda dalam memproses kata. Tahapan proses tersebut diantaranya sebagai berikut :

* + - 1. *Case Folding*

Proses *Case folding* berfungsi untuk mengubah semua huruf dalam teks menjadi huruf kecil dan menghilangkan karakter selain a-z. Sebagai pengimpementasian digunakan 3 data *training sample* sebagai berikut:

Tabel 3.2 Proses *Case Folding*

|  |  |  |
| --- | --- | --- |
| **Dok** | **Kalimat Asli** | **Proses *Case Folding*** |
| 1. | Tulis program untuk menentukan lama bekerja seorang pegawai, jika jam masuk dan jam pulang diinput. Catatan: jam berupa angka 1-12, dan seorang pegawai bekerja kurang dari 12 jam. | tulis program untuk menentukan lama bekerja seorang pegawai, jika jam masuk dan jam pulang diinput. catatan: jam berupa angka 1-12, dan seorang pegawai bekerja kurang dari 12 jam. |
| 2. | Tulis program untuk menentukan biaya parkir yang dihitung berdasarkan lama parkir. Lama parkir dihitung dari selisih jam masuk dan jam keluar diinput. Biaya parkir 2 jam pertama 2000, perjam berikutnya 500. | tulis program untuk menentukan biaya parkir yang dihitung berdasarkan lama parkir. lama parkir dihitung dari selisih jam masuk dan jam keluar diinput. biaya parkir 2 jam pertama 2000, perjam berikutnya 500. |
| 3. | Tulis program untuk menghitung resistor pengganti dari 3 buah resistor yang disusun secara seri atau paralel. Rumus untuk resistor pengganti sbb: Seri: Rgab = R1 + R2 + R3 Paralel : 1/Rgab =1/R1+ 1/R2+1/R3. Besar R1, R2 dan R3 serta jenis penyusunan (seri/paralel) diinput oleh user. | tulis program untuk menghitung resistor pengganti dari 3 buah resistor yang disusun secara seri atau paralel. rumus untuk resistor pengganti sbb: seri: rgab = r1 + r2 + r3 paralel : 1/rgab=1/r1+1/r2 +1/r3. besar r1, r2 dan r3 serta jenis penyusunan (seri/paralel) diinput oleh user. |
| 4. | Program melakukan pengurutan sekumpulan nilai ujian mahasiswa. Data nilai ujian mahasiswa dibaca dari keyboard, lalu diurutkan dari kecil ke besar dan akhirnya hasil pengurutan ditampilkan ke monitor. | program melakukan pengurutan sekumpulan nilai ujian mahasiswa. data nilai ujian mahasiswa dibaca dari keyboard, lalu diurutkan dari kecil ke besar dan akhirnya hasil pengurutan ditampilkan ke monitor. |
| 5. | Program untuk mencetak tabel Fahrenheit-Celcius dari x sampai y dengan kenaikan sebesar step. Masukan program ini adalah suhu awal, suhu akhir, step dan keluarannya adalah tabel konversi suhu dalam C dan F. | program untuk mencetak tabel fahrenheit-celcius dari x sampai y dengan kenaikan sebesar step. masukan program ini adalah suhu awal, suhu akhir, step dan keluarannya adalah tabel konversi suhu dalam c dan f. |

* + - 1. *Tokenizing*

Proses *Tokenizing* dilakukan setelah proses *case folding* selesai. Proses ini berfungsi untuk memecah kalimat menjadi kata-kata serta membuang tanda baca, operator dan angka.

Tabel 3.3 Proses *Tokenizing*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dok.** | **Kata** | **Kata** | **Kata** | **Kata** | **Kata** | **Kata** |
| 1. | tulis | program | untuk | menentukan | lama | bekerja |
| seorang | pegawai | jika | jam | masuk | dan |
| jam | pulang | diinput | catatan | jam | berupa |
| angka | dan | seorang | pegawai | bekerja | kurang |
| dari | jam |  | | | |
| 2. | tulis | program | untuk | menentukan | biaya | parkir |
| yang | dihitung | berdasarkan | lama | parkir | lama |
| parkir | dihitung | dari | selisih | jam | masuk |
| dan | jam | keluar | diinput | biaya | parkir |
| jam | pertama | perjam | berikutnya |  | |
| 3. | tulis | program | untuk | menghitung | resistor | pengganti |
| dari | buah | resistor | yang | disusun | secara |
| seri | atau | paralel | rumus | untuk | resistor |
| pengganti | sbb | seri | rgab | r | r |
| r | paralel | rgab | r | r | r |
| besar | r | r | dan | r | serta |
| jenis | penyusunan | seri | paralel | diinput | oleh |
| user |  | | | | |
| 4. | program | melakukan | pengurutan | sekumpulan | nilai | ujian |
| mahasiswa | data | nilai | ujian | mahasiswa | dibaca |
| dari | keyboard | lalu | diurutkan | dari | kecil |
| ke | besar | dan | akhirnya | hasil | pengurutan |
| ditampilkan | ke | monitor |  | | |
| 5. | program | untuk | mencetak | tabel | fahrenheit | celcius |
| dari | x | sampai | y | dengan | kenaikan |
| sebesar | step | masukan | program | ini | adalah |
| suhu | awal | suhu | akhir | step | dan |
| keluarannya | adalah | tabel | konversi | suhu | dalam |
| c | dan | f |  | | |

* + - 1. *Filtering*

Proses *Filtering* dilakukan setelah *tokenizing* dilakukan. Proses ini berfungsi untuk membuang kata yang dianggap tidak penting dengan menggunakan metode *stopword.* Berikut ini adalah kata-kata yang dibuang dari data *training sample.*

Tabel 3.4 Kata Yang Dibuang

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Kata Yang Dibuang** | | | | | |
| untuk | menentukan | seorang | jika | dan | berupa |
| angka | dari | yang | atau | r | serta |
| oleh | lalu | ke | ini | dalam | x |
| y | c | f | dengan | adalah |  |

* + - 1. *Stemming*

Proses *Stemming* adalah proses untuk mengubah bentuk kata menjadi kata dasar. Cara kerjanya adalah dengan membuang imbuhan, sisipan, dan akhiran. Pada proses ini menggunakan algoritma  *porter* dalam pembentukan kata dasar.

Tabel 3.5 Algoritma *Porter Stemmer*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dok** | **Kata** | **Kata** | **Kata** | **Kata** | **Kata** | **Kata** |
| 1. | tulis | program | lama | kerja | gawai | jam |
| masuk | jam | pulang | input | catat | jam |
| gawai | kerja | kurang | jam |  | |
| 2. | tulis | program | biaya | parkir | hitung | dasar |
| parkir | parkir | hitung | selisih | jam | masuk |
| jam | keluar | input | biaya | parkir | jam |
| pertama | jam | ikut |  | | |
| 3. | tulis | program | hitung | resistor | ganti | resistor |
| susun | cara | seri | paralel | rumus | resistor |
| ganti | sbb | seri | rgab | paralel | rgab |
| besar | jenis | susun | seri | paralel | input |
| user |  |  |  |  |  |
| 4. | program | laku | urut | kumpul | nilai | uji |
|  | mahasiswa | data | nilai | uji | mahasiswa | baca |
|  | keyboard | urut | kecil | besar | akhir | hasil |
|  | urut | tampil | monitor |  |  |  |
| 5. | program | cetak | tabel | fahrenheit | celcius | naik |
|  | besar | step | masuk | program | suhu | awal |
|  | suhu | step | luar | tabel | konversi | suhu |

### Analisis Proses Pembobotan *TF-IDF*

Proses pembobotan pada aplikasi yang dibangun menggunakan metode *TF-IDF. TF* (*Term Frequency*) digunakan untuk menghitung jumlah kata yang muncul pada tiap dokumen, sedangkan *IDF* (*Inverse Document Frequency*) digunakan untuk menghitung jumlah kata yang muncul pada seluruh dokumen.

Sebagai implementasi perhitungan pembobotan dengan mengambil 5 data *training sample* hasil dari pemprosesan *text maining*.

Tabel 3.6 Data *Training Sample*

|  |  |
| --- | --- |
| Dokumen 1 | tulis program lama kerja gawai jam masuk jam pulang input catat jam dan gawai kerja kurang jam |
| Dokumen 2 | tulis program biaya parkir hitung dasar parkir parkir hitung selisih jam masuk jam keluar input biaya parkir jam pertama jam ikut |
| Dokumen 3 | tulis program hitung resistor ganti resistor susun cara seri paralel rumus resistor ganti sbb seri rgab paralel rgab besar jenis susun seri paralel input user |
| Dokumen 4 | program laku urut kumpul nilai uji mahasiswa data nilai uji mahasiswa baca keyboard urut kecil besar akhir hasil urut tampil monitor |
| Dokumen 5 | program cetak tabel fahrenheit celcius naik besar step masuk program suhu awal suhu step luar tabel konversi suhu |
| Dokumen uji | hitung jumlah nilai data hasil uji naik suhu zat laku mahasiswa |

Menentukan banyaknya *TF (Term Frequency)*, *DF (Document Frequency)* dan *IDF* (*Inverse Document Frequency*) untuk menghitung jumlah kata yang muncul pada seluruh dokumen dari kata kunci(*query*) : “tulis program hitung biaya parkir kendaraan gawai cara susun paralel”*.* Pada nilai *idf* dihitung dengan menggunakan rumus persamaan : *Idf* =

*Idf* (hitung) = = 0,176 *Idf* (jumlah) = = 0,778

*Idf* (nilai) = = 0,301 *Idf* (data) = = 0,477

*Idf* (hasil) = = 0,477 *Idf* (uji) = = 0,301

*Idf* (naik) = = 0,477 *Idf* (suhu) = = 0,301

*Idf* (zat) = = 0,778 *Idf* (laku) = = 0,477

*Idf* (mahasiswa) = = 0,301

Tabel 3.7 Penentuan *TF*, *DF* dan *IDF*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Term** | **D.Uji** | **D1** | **D2** | **D3** | **D4** | **D5** | **DF** | **Idf** |
| akhir | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0,778 |
| awal | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0,778 |
| baca | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0,778 |
| besar | 0 | 0 | 0 | 1 | 1 | 1 | 3 | 0,301 |
| biaya | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0,477 |
| cara | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0,778 |
| catat | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0,778 |
| celcius | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0,778 |
| cetak | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0,778 |
| dasar | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0,778 |
| data | 1 | 0 | 0 | 0 | 1 | 0 | 2 | 0,477 |
| fahrenheit | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0,778 |
| ganti | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0,477 |
| gawai | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0,477 |
| hasil | 1 | 0 | 0 | 0 | 1 | 0 | 2 | 0,477 |
| hitung | 1 | 0 | 2 | 0 | 1 | 0 | 4 | 0,176 |
| ikut | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0,778 |
| input | 0 | 1 | 1 | 1 | 0 | 0 | 3 | 0,301 |
| jam | 0 | 4 | 4 | 0 | 0 | 0 | 8 | -0,125 |
| jenis | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0,778 |
| jumlah | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0,778 |
| kecil | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0,778 |
| keluar | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0,778 |
| kerja | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0,477 |
| keyboard | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0,778 |
| konversi | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0,778 |
| kumpul | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0,778 |
| kurang | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0,778 |
| laku | 1 | 0 | 0 | 0 | 1 | 0 | 2 | 0,477 |
| lama | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0,778 |
| luar | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0,778 |
| mahasiswa | 1 | 0 | 0 | 0 | 2 | 0 | 3 | 0,301 |
| masuk | 0 | 1 | 1 | 0 | 0 | 1 | 3 | 0,301 |
| monitor | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0,778 |
| naik | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 0,477 |
| nilai | 1 | 0 | 0 | 0 | 2 | 0 | 3 | 0,301 |
| paralel | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0,301 |
| parkir | 0 | 0 | 4 | 0 | 0 | 0 | 4 | 0,176 |
| pertama | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0,778 |
| program | 0 | 1 | 1 | 1 | 1 | 2 | 6 | 0 |
| pulang | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0,778 |
| resistor | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0,301 |
| rgab | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0,477 |
| rumus | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0,778 |
| sbb | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0,778 |
| selisih | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0,778 |
| seri | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 0,301 |
| step | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0,477 |
| suhu | 1 | 0 | 0 | 0 | 0 | 2 | 3 | 0,301 |
| susun | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0,477 |
| tabel | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0,778 |
| tampil | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0,778 |
| tulis | 0 | 1 | 1 | 1 | 0 | 0 | 3 | 0,301 |
| uji | 1 | 0 | 0 | 0 | 2 | 0 | 3 | 0,301 |
| urut | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0,301 |
| user | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0,778 |
| zat | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0,778 |

Sebagai hasil akhir dari proses pembobotan termasuk dengan menghitung kata kunci digunakan rumus adalah sebagai berikut:

Tabel 3.8 Perhitungan *Tf \* Idf*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Term** | **Wuji** | **W1** | **W2** | **W3** | **W4** | **W5** |
| akhir | 0 | 0 | 0 | 0 | 0,778 | 0 |
| awal | 0 | 0 | 0 | 0 | 0 | 0,778 |
| baca | 0 | 0 | 0 | 0 | 0,778 | 0 |
| besar | 0 | 0 | 0 | 0,301 | 0,301 | 0,301 |
| biaya | 0 | 0 | 0,9542 | 0 | 0 | 0 |
| cara | 0 | 0 | 0 | 0,778 | 0 | 0 |
| catat | 0 | 0,778 | 0 | 0 | 0 | 0 |
| celcius | 0 | 0 | 0 | 0 | 0 | 0,778 |
| cetak | 0 | 0 | 0 | 0 | 0 | 0,778 |
| dasar | 0 | 0 | 0,7782 | 0 | 0 | 0 |
| data | 0,477 | 0 | 0 | 0 | 0,477 | 0 |
| fahrenheit | 0 | 0 | 0 | 0 | 0 | 0,778 |
| ganti | 0 | 0 | 0 | 0,954 | 0 | 0 |
| gawai | 0 | 0,954 | 0 | 0 | 0 | 0 |
| hasil | 0,477 | 0 | 0 | 0 | 0,477 | 0 |
| hitung | 0,176 | 0 | 0,3522 | 0 | 0,176 | 0 |
| ikut | 0 | 0 | 0,7782 | 0 | 0 | 0 |
| input | 0 | 0,301 | 0,301 | 0,301 | 0 | 0 |
| jam | 0 | -0,5 | -0,5 | 0 | 0 | 0 |
| jenis | 0 | 0 | 0 | 0,778 | 0 | 0 |
| jumlah | 0,778 | 0 | 0 | 0 | 0 | 0 |
| kecil | 0 | 0 | 0 | 0 | 0,778 | 0 |
| keluar | 0 | 0 | 0,7782 | 0 | 0 | 0 |
| kerja | 0 | 0,954 | 0 | 0 | 0 | 0 |
| keyboard | 0 | 0 | 0 | 0 | 0,778 | 0 |
| konversi | 0 | 0 | 0 | 0 | 0 | 0,778 |
| kumpul | 0 | 0 | 0 | 0 | 0,778 | 0 |
| kurang | 0 | 0,778 | 0 | 0 | 0 | 0 |
| laku | 0,477 | 0 | 0 | 0 | 0,477 | 0 |
| lama | 0 | 0,778 | 0 | 0 | 0 | 0 |
| luar | 0 | 0 | 0 | 0 | 0 | 0,778 |
| mahasiswa | 0,301 | 0 | 0 | 0 | 0,602 | 0 |
| masuk | 0 | 0,301 | 0,301 | 0 | 0 | 0,301 |
| monitor | 0 | 0 | 0 | 0 | 0,778 | 0 |
| naik | 0,477 | 0 | 0 | 0 | 0 | 0,477 |
| nilai | 0,301 | 0 | 0 | 0 | 0,602 | 0 |
| paralel | 0 | 0 | 0 | 0,903 | 0 | 0 |
| parkir | 0 | 0 | 0,7044 | 0 | 0 | 0 |
| pertama | 0 | 0 | 0,7782 | 0 | 0 | 0 |
| program | 0 | 0 | 0 | 0 | 0 | 0 |
| pulang | 0 | 0,778 | 0 | 0 | 0 | 0 |
| resistor | 0 | 0 | 0 | 0,903 | 0 | 0 |
| rgab | 0 | 0 | 0 | 0,954 | 0 | 0 |
| rumus | 0 | 0 | 0 | 0,778 | 0 | 0 |
| sbb | 0 | 0 | 0 | 0,778 | 0 | 0 |
| selisih | 0 | 0 | 0,7782 | 0 | 0 | 0 |
| seri | 0 | 0 | 0 | 0,903 | 0 | 0 |
| step | 0 | 0 | 0 | 0 | 0 | 0,954 |
| suhu | 0,301 | 0 | 0 | 0 | 0 | 0,602 |
| susun | 0 | 0 | 0 | 0,954 | 0 | 0 |
| tabel | 0 | 0 | 0 | 0 | 0 | 0,778 |
| tampil | 0 | 0 | 0 | 0 | 0,778 | 0 |
| tulis | 0 | 0,301 | 0,301 | 0,301 | 0 | 0 |
| uji | 0,301 | 0 | 0 | 0 | 0,602 | 0 |
| urut | 0 | 0 | 0 | 0 | 0,903 | 0 |
| user | 0 | 0 | 0 | 0,778 | 0 | 0 |
| zat | 0,778 | 0 | 0 | 0 | 0 | 0 |

Penjumlahan bobot dokumen yang mengandung kata sama seperti query dengan memakai *TF-IDF* dinyatakan sebagai berikut:

Dok1 : 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

Dok2 : 0 + 0 + 0,3522 + 0 + 0 + 0 + 0 + 0+ 0 + 0 + 0 = 0,3522

Dok3 : 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 0

Dok4 : 0,477 + 0,477 + 0,176 + 0 + 0,477 + 0,602 + 0 + 0,602 + 0 + 0,602 + 0

= 2,209

Dok5 : 0 + 0 + 0 + 0 + 0 + 0 + 0,477 + 0 + 0,602 + 0 + 0 = 1,079

Tabel 3.9 Hasil Perbandingan Dokumen Dan *Query* dari *TFIDF*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dokumen 1** | **Dokumen 2** | **Dokumen 3** | **Dokumen 4** | **Dokumen 5** |
| 0 | 0,3522 | 0 | 2,209 | 1,079 |

### Analisis Pencari Kemiripan *Similarity*

Algoritma *Cosine similarity* digunakan untuk menghitung pendekatan relevansi *query* terhadap dokumen. Penentuan relevansi sebuah *query* terhadap suatu dokumen dipandang sebagai pengukuran kesamaan antara vektor *query* dengan vektor dokumen. Untuk menghitung vektor tersebut dapat dihitung dengan menghitung panjang skalar setelah itu menggunakan rumus Algoritma *Cosine similarity* sebagai berikut:

Perhitungan skalar didapatkan dari hasil perkalian antara bobot query dengan perkalian bobot tiap dokumen atau Sdi = WD \* Wdi, untuk pengimplementasian dibuat kedalam bentuk tabel berikut:

Tabel 3.10 Perhitungan Panjang Skalar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Term** | **S1** | **S2** | **S3** | **S4** | **S5** |
| akhir | 0 | 0 | 0 | 0 | 0 |
| awal | 0 | 0 | 0 | 0 | 0 |
| baca | 0 | 0 | 0 | 0 | 0 |
| besar | 0 | 0 | 0 | 0 | 0 |
| biaya | 0 | 0 | 0 | 0 | 0 |
| cara | 0 | 0 | 0 | 0 | 0 |
| catat | 0 | 0 | 0 | 0 | 0 |
| celcius | 0 | 0 | 0 | 0 | 0 |
| cetak | 0 | 0 | 0 | 0 | 0 |
| dasar | 0 | 0 | 0 | 0 | 0 |
| data | 0 | 0 | 0 | 0,228 | 0 |
| fahrenheit | 0 | 0 | 0 | 0 | 0 |
| ganti | 0 | 0 | 0 | 0 | 0 |
| gawai | 0 | 0 | 0 | 0 | 0 |
| hasil | 0 | 0 | 0 | 0,228 | 0 |
| hitung | 0 | 0,062 | 0 | 0,031 | 0 |
| ikut | 0 | 0 | 0 | 0 | 0 |
| input | 0 | 0 | 0 | 0 | 0 |
| jam | 0 | 0 | 0 | 0 | 0 |
| jenis | 0 | 0 | 0 | 0 | 0 |
| jumlah | 0 | 0 | 0 | 0 | 0 |
| kecil | 0 | 0 | 0 | 0 | 0 |
| keluar | 0 | 0 | 0 | 0 | 0 |
| kerja | 0 | 0 | 0 | 0 | 0 |
| keyboard | 0 | 0 | 0 | 0 | 0 |
| konversi | 0 | 0 | 0 | 0 | 0 |
| kumpul | 0 | 0 | 0 | 0 | 0 |
| kurang | 0 | 0 | 0 | 0 | 0 |
| laku | 0 | 0 | 0 | 0,228 | 0 |
| lama | 0 | 0 | 0 | 0 | 0 |
| luar | 0 | 0 | 0 | 0 | 0 |
| mahasiswa | 0 | 0 | 0 | 0,181 | 0 |
| masuk | 0 | 0 | 0 | 0 | 0 |
| monitor | 0 | 0 | 0 | 0 | 0 |
| naik | 0 | 0 | 0 | 0 | 0,228 |
| nilai | 0 | 0 | 0 | 0,181 | 0 |
| paralel | 0 | 0 | 0 | 0 | 0 |
| parkir | 0 | 0 | 0 | 0 | 0 |
| pertama | 0 | 0 | 0 | 0 | 0 |
| program | 0 | 0 | 0 | 0 | 0 |
| pulang | 0 | 0 | 0 | 0 | 0 |
| resistor | 0 | 0 | 0 | 0 | 0 |
| rgab | 0 | 0 | 0 | 0 | 0 |
| rumus | 0 | 0 | 0 | 0 | 0 |
| sbb | 0 | 0 | 0 | 0 | 0 |
| selisih | 0 | 0 | 0 | 0 | 0 |
| seri | 0 | 0 | 0 | 0 | 0 |
| step | 0 | 0 | 0 | 0 | 0 |
| suhu | 0 | 0 | 0 | 0 | 0,181 |
| susun | 0 | 0 | 0 | 0 | 0 |
| tabel | 0 | 0 | 0 | 0 | 0 |
| tampil | 0 | 0 | 0 | 0 | 0 |
| tulis | 0 | 0 | 0 | 0 | 0 |
| uji | 0 | 0 | 0 | 0,181 | 0 |
| urut | 0 | 0 | 0 | 0 | 0 |
| user | 0 | 0 | 0 | 0 | 0 |
| zat | 0 | 0 | 0 | 0 | 0 |
| **Total** | **0** | **0,062** | **0** | **1,258** | **0,409** |

Perhitungan skalar diatas hasilnya sama dengan potongan rumus *Cosine Similarity* yaitu , kemudian dilanjutkan dengan menghitung panjang vektor yang disajikan pada tabel berikut:

Tabel 3.11 Perhitungan Panjang Vektor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Term** | **Vuji** | **V1** | **V2** | **V3** | **V4** | **V5** |
| akhir | 0 | 0 | 0 | 0 | 0,606 | 0 |
| awal | 0 | 0 | 0 | 0 | 0 | 0,606 |
| baca | 0 | 0 | 0 | 0 | 0,606 | 0 |
| besar | 0 | 0 | 0 | 0,091 | 0,091 | 0,091 |
| biaya | 0 | 0 | 0,911 | 0 | 0 | 0 |
| cara | 0 | 0 | 0 | 0,606 | 0 | 0 |
| catat | 0 | 0,606 | 0 | 0 | 0 | 0 |
| celcius | 0 | 0 | 0 | 0 | 0 | 0,606 |
| cetak | 0 | 0 | 0 | 0 | 0 | 0,606 |
| dasar | 0 | 0 | 0,606 | 0 | 0 | 0 |
| data | 0,228 | 0 | 0 | 0 | 0,228 | 0 |
| fahrenheit | 0 | 0 | 0 | 0 | 0 | 0,606 |
| ganti | 0 | 0 | 0 | 0,911 | 0 | 0 |
| gawai | 0 | 0,911 | 0 | 0 | 0 | 0 |
| hasil | 0,228 | 0 | 0 | 0 | 0,228 | 0 |
| hitung | 0,031 | 0 | 0,124 | 0 | 0,031 | 0 |
| ikut | 0 | 0 | 0,606 | 0 | 0 | 0 |
| input | 0 | 0,091 | 0,091 | 0,091 | 0 | 0 |
| jam | 0 | 0,25 | 0,25 | 0 | 0 | 0 |
| jenis | 0 | 0 | 0 | 0,606 | 0 | 0 |
| jumlah | 0,606 | 0 | 0 | 0 | 0 | 0 |
| kecil | 0 | 0 | 0 | 0 | 0,606 | 0 |
| keluar | 0 | 0 | 0,606 | 0 | 0 | 0 |
| kerja | 0 | 0,911 | 0 | 0 | 0 | 0 |
| keyboard | 0 | 0 | 0 | 0 | 0,606 | 0 |
| konversi | 0 | 0 | 0 | 0 | 0 | 0,606 |
| kumpul | 0 | 0 | 0 | 0 | 0,606 | 0 |
| kurang | 0 | 0,606 | 0 | 0 | 0 | 0 |
| laku | 0,228 | 0 | 0 | 0 | 0,228 | 0 |
| lama | 0 | 0,606 | 0 | 0 | 0 | 0 |
| luar | 0 | 0 | 0 | 0 | 0 | 0,606 |
| mahasiswa | 0,091 | 0 | 0 | 0 | 0,362 | 0 |
| masuk | 0 | 0,091 | 0,091 | 0 | 0 | 0,091 |
| monitor | 0 | 0 | 0 | 0 | 0,606 | 0 |
| naik | 0,228 | 0 | 0 | 0 | 0 | 0,228 |
| nilai | 0,091 | 0 | 0 | 0 | 0,362 | 0 |
| paralel | 0 | 0 | 0 | 0,816 | 0 | 0 |
| parkir | 0 | 0 | 0,496 | 0 | 0 | 0 |
| pertama | 0 | 0 | 0,606 | 0 | 0 | 0 |
| program | 0 | 0 | 0 | 0 | 0 | 0 |
| pulang | 0 | 0,606 | 0 | 0 | 0 | 0 |
| resistor | 0 | 0 | 0 | 0,816 | 0 | 0 |
| rgab | 0 | 0 | 0 | 0,911 | 0 | 0 |
| rumus | 0 | 0 | 0 | 0,606 | 0 | 0 |
| sbb | 0 | 0 | 0 | 0,606 | 0 | 0 |
| selisih | 0 | 0 | 0,606 | 0 | 0 | 0 |
| seri | 0 | 0 | 0 | 0,816 | 0 | 0 |
| step | 0 | 0 | 0 | 0 | 0 | 0,911 |
| suhu | 0,091 | 0 | 0 | 0 | 0 | 0,362 |
| susun | 0 | 0 | 0 | 0,911 | 0 | 0 |
| tabel | 0 | 0 | 0 | 0 | 0 | 0,606 |
| tampil | 0 | 0 | 0 | 0 | 0,606 | 0 |
| tulis | 0 | 0,091 | 0,091 | 0,091 | 0 | 0 |
| uji | 0,091 | 0 | 0 | 0 | 0,362 | 0 |
| urut | 0 | 0 | 0 | 0 | 0,816 | 0 |
| user | 0 | 0 | 0 | 0,606 | 0 | 0 |
| zat | 0,606 | 0 | 0 | 0 | 0 | 0 |
| **Total** | 2,515 | 4,765 | 5,08 | 8,478 | 6,946 | 5,921 |
| **SQRT** | 1,586 | 2,183 | 2,254 | 2,912 | 2,636 | 2,433 |

Perhitungan vektor diatas hasilnya sama dengan potongan rumus *Cosine Similarity* yaitu dan sehingga dapat dihitung:

*Cosine Similarity* (Dok1) = 0

*Cosine Similarity* (Dok2) = 0.017

*Cosine Similarity* (Dok3) = 0

*Cosine Similarity* (Dok4) = 0,301

*Cosine Similarity* (Dok5) = 0,106

Tabel 3.12 Urutan Kemiripan *Cosine Similarity*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dokumen 1** | **Dokumen 2** | **Dokumen 3** | **Dokumen 4** | **Dokumen 5** |
| 0 | 0.017 | 0 | 0,301 | 0,106 |
| Rank 4 | Rank 3 | Rank 5 | Rank 1 | Rank 2 |

### Analisis Klasifikasi *KNN*

Penganalisis untuk klasifikasi data pada penelitian ini menggunakan algoritma klasifikasi *K-Nearest Neighbor*. Algoritma *KNN* bekerja berdasarkan jarak terpendek dari *training sample* dan *test sample* dalam penentuannya, kemudian menggambil mayoritas dari hasil ketentuan yang telah didapatkan untuk dijadikan prediksi dari *test sample*.

Sebagai pemgimplementasian algoritma dalam aplikasi, maka diambil dari 5 data *training* yang sebelumnya sudah dihitung hasil *similarity* dari data-data tersebut. Dari data *training* tersebut sebelumnya telah memiliki klasifikasi masing-masing, yaitu data dokumen *training* 1, 2, 3 mempunyai klasifikasi pemilihan sedangkan pada data dokumen *training* ke 4 dan 5 mempunyai klasifikasi pengulangan.

Langkah yang dilakukan untuk *KNN* adalah dengan mengambil sebanyak *k* (*k* = 3) sehingga yang paling tinggi tingkat kemiripannya dengan data test *sample* yaitu data *training* dokumen ke 2, 4, dan 5. Dengan data *training* dokumen ke 2 mempunyai klasifikasi pemilihan dan pada data *training* dokumen ke 4 dan 5 mempunyai klasifikasi pengurutan, sehingga data test *sample* tersebut diklasifikasi berdasarkan *voting* terbanyak dan akan masuk ke dalam klasifikasi pengurutan. Karena dari hasil *similarity* nilai terbesar adalah jarak kemiripan tertinggi, sedangkan *KNN* dengan nilai terkecil adalah jarak kemiripan tertinggi maka untuk perhitungan *KNN* tersebut adalah 1 - *similarity.* Dapat dilihat dari tabel berikut dengan *k* = 3:

Tabel 3.13 Klasifikasi *KNN*

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen 4** | **Dokumen 5** | **Dokumen 2** | **Dokumen *test*** |
| 1 - 0,301 = 0,699  (Pengulangan) | 1 - 0,106 = 0,894  (Pengulangan) | 1 - 0.017 = 0,983  (Pemilihan) | Pengulangan |
| Rank 1 | Rank 2 | Rank 3 |

### Analisis Berbasis Kasus *CBR*

Penelitian ini menggunakan Algoritma *Case Based Reasoning* (*CBR*) yaitu model penalaran yang menggabungkan pemecahan masalah, pemahaman, pembelajaran dan memadukan keseluruhannya dengan pemrosesan yang dilakukan dengan memanfaatkan kasus yang pernah dialami oleh sistem, sebagai dasar dari pengetahuan yang mewakili suatu pengalaman untuk dijadikan pembelajaran demi terencapainya tujuan sistem. Alur proses *CBR* dalam memecahkan kasus dari penelitian ini menggunakan 4 langkah RE, yaitu:

1. RETRIEVE, digunakan untuk mencari kemiripan kasus lama dan kasus baru dengan memakai algoritma *similarity.*
2. REUSE, digunakan untuk menentukan klasifikasi data *test sample* dengan menggunakan algoritma *KNN*, hal tersebut dilakukan karena agar dapat menggunakan kembali informasi dan pengetahuan berdasarkan klasifikasi dari data kasus *training sample* untuk memecahkan masalah kasus baru (proses ini disebut “tansfer solusi”).
3. REVISE, digunakan untuk kasus baru yang berupa soal input telah mendapat hasil jawaban yang paling mirip dapat direvisi atau diperbaiki jawaban dari soal baru yang diusulkan.
4. RETAIN, digunakan untuk menyimpan semua pengalaman kasus untuk memecahkan masalah kasus yang akan datang ke dalam basis kasus.

### Deskripsi Global Aplikasi

Sistem yang dibangun berupa aplikasipencarian kemiripan soal dan jawaban algoritma merupakan aplikasi berbasis *java off-line.* Aplikasiini bergunauntuk membantu para pelajar yang masih belum memahami kode program sehingga dengan menerapkan aplikasi ini dapat membantu dalam mencari jawaban dari soal algoritma yang hasilnya berupa identitas jawaban dan gambaran potongan kode program berdasarkan tingkat kemiripan tertinggi dari hasil jawaban soal algoritma yang lalu.

### Arsitektur Sistem

Data identitas jawaban

Hasil Akurasi

Pengujian Algoritma *KNN*

*Pra-proses* soal

Input soal

Pembobotan *TF-IDF*

*CBR*

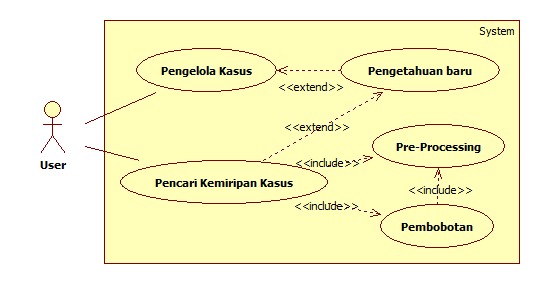
Gambar 3.1 Arsitektur Sistem

Program menerima inputan berupa soal algoritma, kemudian dilakukan tahapan pra-proses untuk mengekstrak soal menjadi identitas/label jawaban. Tahapan praproses dilakukan dengan *Text Miming* (pengolahan teks untuk menemukan pengetahuan baru) yang meliputi tahap: *Tokenizing* (pemotongan string input berdasarkan tiap kata penyusun), *Filtering* (mengambil kata-kata penting dari hasil token dengan menggunakan algoritma *stopword*), dan *Stemming* (membuang imbuhan dan mencari katadasar dari tiap kata hasil *filtering*). Kemudian pemberian bobot pada tiap *term* dalam dokumen menggunakan algoritma *TF-IDF,* sehingga hasilnya dapat diuji dengan Algoritma *KNN* untuk menemukan kemiripan antara soal baru dengan soal terdahulu yang ada pada *database* agar mendapatkan hasil berupa identitas jawaban. Tahapan-tahapan yang telah diproses tersebut merupakan bagian-bagian dari metode *CBR* (Sistem Pendukung Keputusan untuk pemecahan masalah) yang dilakukan untuk membandingkan akurasi nilai terbaik dari kemiripan jawaban dengan hasil dari pertimbangan pengetahuan baru.

## Perancangan Sistem

Perancangan aplikasi ini menggunakan pendekatan berorientasi objek dengan menggunakan metode *Unified Modeling Language* (*UML*) yakni sebagai berikut:

### *Use Case* Diagram



Gambar 3.2 *Use Case* Diagram

*Use case* diagram dibuat untuk menggambarkan rancangan sistem secara global. Pada *use case* diagram dibuat menjadi 5 use case yaitu:

1. *Use case* Pengelola Kasus, pada *use case* ini menggambarkan semua pengetahuan kasus yang telah ada pada database dan siap untuk digunakan untuk proses sistem.
2. *Use case* Pengetahuan Baru, pada *use case* ini menggambarkan data pengetahuan yang sebelumnya belum pernah ada dan belum ditemukan solusi penyelesaian yang benar sehingga pengetahuan baru tersebut perlu ditinjau ulang oleh Si pengelola aplikasi. Pengetahuan baru didapat dari hasil proses perhitungan yang sangat jauh kemiripannya dengan pengetahuan yang ada.
3. *Use case* Pencari Kemiripan Kasus, pada *use case* ini menggambarkan proses inti dari aplikasi karena disinilah sumber kajian penelitian. Use case ini juga terlibat dengan proses pada use case lain seperti *Use case pre-processing, Use case* Pembobotan dan proses dalam pembentukan pengetahuan baru.
4. *Use case pre-processing*, pada *use case* ini menggambarkan suatu proses yang wajib dilakukan sebelum perhitungan diproses. Pada pengimplementasiannya proses ini dilakukan ketika user ingin mencari kemiripan dari soal yang diinput dengan soal dan jawaban yang ada pada *database.*
5. *Use case* Pembobotan, pada *use case* ini menggambarkan proses lanjutan dari proses *pre-processing* karena disini akan dihitung berapa hasil kemiripan dari soal yang diinput dengan soal dan jawaban yang ada pada *database* sehingga akan menampilkan jawaban berdasarkan klasifikasi jawaban soal dengan *rating* tertinggi.

### Definisi *Actor*

Bagian ini diisi dengan daftar *actor* beserta deskripsinya. Deskripsi harus menjelaskan wewenang yang dapat dilakukan dalam perangkat lunak. Sehingga dapat dibuat kedalam bentuk tabel sebagai berikut:

Tabel 3.14 Definisi *Actor*

|  |  |  |
| --- | --- | --- |
| **No.** | ***Actor*** | **Deskripsi** |
| 1. | *User* | *User* mempunyai dua kriteria yaitu: admin dan pengguna. Apabila *user* sebagai admin, maka dapat mengakses kelola data dengan syarat berhasil *login*, pada pengelolaan dapat *insert, update, delete* pengetahuan dan dapat juga merevisi data pengetahuan baru untuk dipatenkan.  Apabila *user* sebagai Pengguna hanya dapat mengakses menu pilihan pencari jawaban dan dapat mengusulkan hasil jawaban dari pengolahan soal untuk dijadikan bahan revisi bagi pengelolaan data. |

### Skenario *Use Case*

Tabel 3.15 Skenario *Use Case* Pengelola Kasus

|  |  |  |
| --- | --- | --- |
| **Identifikasi** | | |
| **Nomor** | 1 | |
| **Nama** | Pengelola Kasus | |
| **Tujuan** | Menggambarkan semua pengetahuan kasus yang telah ada pada database dan siap untuk digunakan untuk proses sistem. | |
| **Aktor** | *User* sebagai admin | |
| **Skenario Utama** | | |
| **Kondisi awal** | Memilih menu kelola aplikasi dan berhasil login. | |
| **Aksi Aktor** | | **Reaksi Sistem** |
| 1. Menginput data | | 1. Sistem menyediakan beberapa kolom inputan. |
| 1. Memillih menu simpan | | 1. Sistem memproses soal dengan tahapan *pre-processing* (*tokenizing, filtering* dan *stemming*), kemudian menyimpan hasil ke *database.* |
| 1. Memilih menu lihat data | | 1. Sistem akan pindah ke halaman tabel data. |
| 1. Memilih salah satu data dan menekan tombol pilih data | | 1. Sistem akan mentransfer pilihan ke halaman kelola aplikasi didalam kolom-kolom data. |
| 1. Merubah data dan menekan tombol edit | | 1. Sistem akan meng-u*pdate* data pada *database* sesuai dengan id. |
| 1. Memilih data pada tabel lihat data dan menekan tombol hapus | | 1. Sistem akan men-*delete* data pada *database* sesuai dengan id. |
| 1. Memilih data pada tabel lihat data dan menekan tombol batal | | 1. Sistem akan mengosongkan kolom-kolom data. |
| **Kondisi akhir** | Admin dapat mengelola pengetahuan kasus. | |
| **Kondisi Pengecualian** | 1. Gagal menyimpan data maka akan menampilkan bahwa sistem gagal menyimpan dan admin harus mengisi atau mengubah kembali kolom data yang ingin disimpan. 2. Gagal mengedit data maka akan menampilkan bahwa sistem gagal mengedit dan admin harus mengubah kembali kolom data yang ingin diedit. 3. Gagal menghapus data maka akan menampilkan bahwa sistem gagal menghapus dan admin harus memilih kembali tabel data dan menghapus ulang. | |

Tabel 3.16 Skenario *Use Case* Pengetahuan Baru

|  |  |  |
| --- | --- | --- |
| **Identifikasi** | | |
| **Nomor** | 2 | |
| **Nama** | Pengetahuan Baru | |
| **Tujuan** | Menggambarkan pengetahuan kasus baru hasil proses sistem pencarian kemiripan soal dan pengklasifikasian jawaban. | |
| **Aktor** | *User* sebagai admin | |
| **Skenario Utama** | | |
| **Kondisi awal** | Sistem berada pada tabel usulan revisi | |
| **Aksi Aktor** | | **Reaksi Sistem** |
| 1. Melihat tabel data dan memilih salah satu usulan solusi kasus. | | 1. Sistem akan mentransfer pilihan ke halaman kelola aplikasi didalam kolom-kolom data. |
| 1. Merubah atau merevisi data dan menekan tombol edit | | 1. Sistem akan meng-u*pdate* data pada *database* sesuai dengan id. |
| 1. Menekan tombol hapus | | 1. Sistem akan men-*delete* data pada *database* sesuai dengan id. |
| **Kondisi akhir** | Admin dapat merevisi pengetahuan kasus baru. | |
| **Kondisi Pengecualian** | 1. Gagal mengedit data maka akan menampilkan bahwa sistem gagal mengedit dan admin harus mengubah kembali kolom data yang ingin diedit. 2. Gagal menghapus data maka akan menampilkan bahwa sistem gagal menghapus dan admin harus memilih kembali tabel data dan menghapus ulang. | |

Tabel 3.17 Skenario *Use Case* Pencari Kemiripan Kasus

|  |  |  |
| --- | --- | --- |
| **Identifikasi** | | |
| **Nomor** | 3 | |
| **Nama** | Pencari Kemiripan Kasus | |
| **Tujuan** | Menggambarkan proses inti dari aplikasi yang terlibat dengan proses pada *use case* lain seperti *use case pre-processing, use case* pembobotan dan proses dalam pembentukan pengetahuan baru. | |
| **Aktor** | *User* | |
| **Skenario Utama** | | |
| **Kondisi awal** | Sistem berada pada halaman pencari jawaban | |
| **Aksi Aktor** | | **Reaksi Sistem** |
| 1. Menginput soal kasus algoritma | | 1. Sistem menyediakan kolom untuk penginputan soal. |
| 1. Menekan tombol cari jawaban | | 1. Sistem memproses soal dengan tahapan pada *use case* *pre-processing* dan menyimpan hasil ke *database.* Selanjutnya sistem mengambil data kasus lama dan soal kasus baru untuk dilakukannya proses perhitungan pada *use case* pembobotan dan pengklasifikasian soal agar menampilkan hasil jawaban termirip. |
| 1. Melihat hasil jawaban pada kolom identitas jawaban, gambar potongan program dan hasil perhitungan *similarity* dan *KNN.* | | 1. Sistem akan menyimpan hasil sebagai solusi dari pengetahuan baru kedalam *database*. |
| **Kondisi akhir** | *User* dapat melihat hasil jawaban termirip dengan soal yang diinputkan. | |
| **Kondisi Pengecualian** | Gagal memproses maka *user* akan disarankan untuk meninjau kembali soal yang diinputkan dengan benar. | |

Tabel 3.18 Skenario *Use Case Pre-processing*

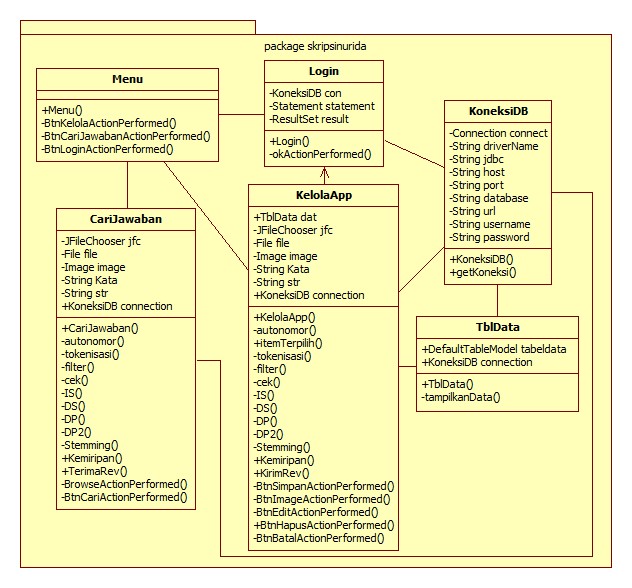
|  |  |  |
| --- | --- | --- |
| **Identifikasi** | | |
| **Nomor** | 4 | |
| **Nama** | *Pre-processing* | |
| **Tujuan** | Menggambarkan proses yang wajib dilakukan sebelum perhitungan diproses dan dilakukan ketika user ingin mencari kemiripan dari soal yang diinput dengan soal dan jawaban yang ada pada *database.* | |
| **Aktor** | *User* | |
| **Skenario Utama** | | |
| **Kondisi awal** | Sistem berada pada halaman pencari jawaban | |
| **Aksi Aktor** | | **Reaksi Sistem** |
| 1. Menginput soal kasus algoritma | | 1. Sistem menyediakan kolom untuk penginputan soal. |
| 1. Menekan tombol cari jawaban | | 1. Sistem memproses soal dengan *pre-processing* meliputi tahap: *tokenizing, filtering* dan *stemming* kemudian menyimpan hasil ke *database.* |
| **Kondisi akhir** | Sistem menyimpan hasil *pre-processing* kedalam *database.* | |
| **Kondisi Pengecualian** | Gagal memproses maka *user* akan disarankan untuk meninjau kembali soal yang diinputkan dengan benar. | |

Tabel 3.19 Skenario *Use Case* Pembobotan

|  |  |  |
| --- | --- | --- |
| **Identifikasi** | | |
| **Nomor** | 5 | |
| **Nama** | Pembobotan | |
| **Tujuan** | Menggambarkan proses lanjutan dari proses *pre-processing* karena disini akan dihitung hasil kemiripan dari soal yang diinput dengan soal dan jawaban yang ada pada *database* sehingga akan menampilkan jawaban berdasarkan klasifikasi jawaban soal dengan *rating* tertinggi. | |
| **Aktor** | *User* | |
| **Skenario Utama** | | |
| **Kondisi awal** | Sistem telah melakukan proses *pre-processing* | |
| **Aksi Aktor** | | **Reaksi Sistem** |
|  | | 1. Sistem menghitung bobot soal dan vektor dari semua data kasus dengan menggunakan algoritma *similarity* kemudian mengklasifikasikan soal dengan algoritma *KNN* dan hasil perhitungan ditampilkan di *UI.* |
| 1. Melihat hasil jawaban pada kolom identitas jawaban, gambar potongan program dan hasil perhitungan *similarity* dan *KNN.* | | 1. Sistem akan menyimpan hasil sebagai solusi dari pengetahuan baru kedalam *database*. |
| **Kondisi akhir** | Sistem menyimpan hasil pembobotan dan pengklasifikasian kedalam *database.* | |
| **Kondisi Pengecualian** | Gagal memproses maka *user* akan disarankan untuk meninjau kembali soal yang diinputkan dengan benar. | |

### *Sequence* Diagram

### *Class* Diagram



Gambar 3.3 *Class* Diagram

Tabel 3.20 *Class* Diagram

|  |  |  |
| --- | --- | --- |
| **Nama Kelas** | **Daftar Tanggung-Jawab** | **Daftar Atribut** |
| Menu.java | 1. Menyediakan pilihan *button* Kelola Data. 2. Menyediakan pilihan b*utton* Pencari Jawaban. | - |
| Login.java | 1. Menyediakan kolom input *User Name*  dan *Password.* 2. Menyediakan pilihan b*utton* *Login.* 3. Menyediakan pemberitahuan apabila login gagal. 4. Mengosongkan kolom apabila menekan *button* batal. | 1. con 2. statement 3. result |
| KoneksiDB.java | 1. Menghubungkan bahasa pemprograman dengan *database.* | 1. *connect* 2. *driverName* 3. *jdbc* 4. *host* 5. *port* 6. *database* 7. *url* 8. *username* 9. *password* |
| KelolaApp.java | 1. Menyediakan kolom input yang harus diisi. 2. Menyediakan fasilitas untuk melihat dan memilih tabel. 3. Melakukan proses simpan, edit, hapus dan batal 4. Memproses data sebelum dimasukkan kedalam *database.* 5. Menerima usulan revisi untuk dikonfirmasi. 6. Menyimpan perubahan pada *database.* | 1. data 2. jfc 3. file 4. image 5. Kata 6. Str 7. connection |
| CariJawaban.java | 1. Menyediakan kolom input untuk soal algoritma yang harus diisi. 2. Memproses soal dan memberi nilai bobot kemiripan berdasarkan nilai urutan tingkat kemiripan paling tinggi. 3. Menyediakan fasilitas untuk pengguna jika ingin mengirim usulan revisi soal dan jawaban dari hasil proses yang dilakukan sebelumnya. | 1. Jfc 2. File 3. Image 4. Kata 5. Str 6. connection |
| TblData.java | 1. Menampilkan seluruh data yang terdapat pada database. 2. Menyediakan fasilitas untuk memilih salah satu data untuk diberi aksi pada kelas KelolaApp.java. | 1. Tabeldata 2. connection |