IMPLEMENTASI ALGORITMA *K-NEAREST NEIGHBOR* DALAM SISTEM *CASE BASED REASONING* UNTUK PEMBENTUKAN IDENTITAS JAWABAN OTOMATIS DAN PENCARI KEMIRIPAN JAWABAN DARI SOAL-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

2016 M / 1438 H

LEMBAR PERSEMBAHAN

Alhamdulilahirobilalamin....

Puji Syukur kepada Allah SWT karena hanya atas izin dan karuniaNyalah maka skripsi ini dapat dibuat dan selesai pada waktunya. Skripsi ini dipersembahkan untuk orang-orang yang saya sayangi:

Bapak dan Mamah, yang telah memberikan dukungan moril maupun materi serta do'a yang tiada henti. Ucapan terimakasih saja takkan pernah cukup untuk membalas kebaikan orang tua, karena itu terimalah persembahan bakti dan cinta ku untuk bapakku (H.Jahidi) dan ibuku (Hj.Siti Rohaeda).

Adik-adikku (Nur Tursina, M.Manaf Hadi, M.Rafi'ul Hakim Maulana, Najma Azkiya Humaira), yang senantiasa memberikan semangat, senyum dan do'anya, terimakasih dan sayangku untuk kalian.....

Keluarga besar Alm.Kakek Salim dan Nenek Euis Fatimah, terimakasih banyak untuk segala dukungan, semangat dan do'anya.

Sahabatku Mia Amalia, S.T. dan Neneng Sapuroh S.T., terimakasih untuk canda tawa, tangis, dan perjuangan yang kita lewati bersama serta kenangan manis yang telah terukir selama ini. Dengan perjuangan dan kebersamaan kita pasti bisa! Semangat!! :D

Terimakasih yang sebesar-besarnya untuk kalian semua. Dan semoga skripsi ini dapat bermanfaat dan berguna untuk kemajuan ilmu pengetahuan di masa yang akan datang, Aamiinnn.

MOTTO

"Belajar dan bekerja dengan giat, serta tidak lupa bersyukur,

tentu akan memberikan hasil yang baik."

"Ikhtiar dan tawakal adalah kunci kesuksesan."

RIWAYAT HIDUP

Nurida Ahsanti, lahir di Bandung pada tanggal 08 Mei 1994 dari seorang ayah yang bernama Bapak H.Jahidi dan seorang ibu yang bernama Ibu Hj.Siti Rohaeda, adalah anak pertama dari lima bersaudara. Menyelesaikan pendidikan Sekolah Dasar di SD Negeri Rajamandala Kulon 1 pada tahun 2000, dan lulus pada tahun 2006. Kemudian melanjutkan pendidikan menengah pertama di MTs.Al-Mukhtariyah Rajamandala pada tahun 2006, dan lulus pada tahun 2009. Dan melanjutkan pendidikan menengah atas di MAN 1 Cililin pada tahun 2009 dan lulus pada tahun 2012. Setelah lulus pendidikan menengah atas, kemudian melanjutkan pendidikan Strata 1 di Universitas Islam Negeri Sunan Gunung Djati Bandung, Jurusan Teknik Informatika, Fakultas Sains dan Teknologi pada tahun 2012 dan lulus pada Desember 2016 sebagai Sarjana Teknik.

KATA PENGANTAR

Puji syukur senantiasa tertuju pada kehadirat Allah SWT. karena atas rahmat serta karunia-Nya penulisan skripsi yang berjudul "**Implementasi Algoritma *K-Nearest Neighbor* dalam *Sistem Case Based Reasoning* untuk Pembentukan Identitas Jawaban Otomatis dan Pencari Kemiripan Jawaban dari Soal-Soal Algoritma**" dapat diselesaikan dengan baik.

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Dr.Yana Aditia Gerhana, S.T., M.Kom. selaku dosen pembimbing I yang sekaligus sebagai pemberi ide skripsi dan Aldy Rialdy Atmadja, M.T. selaku dosen pembimbing II. Terimakasih atas kesabaran serta keikhlasan dalam memberikan bimbingan dan saran selama penyusunan skripsi.
2. Bapak Beki Subaeki, M.Kom. selaku penguji I dan Bapak Nur Lukman, S.T., M.Kom. selaku penguji II.
3. Sahabat, rekan-rekan seperjuangan dan semua pihak yang namanya tidak dapat disebutkan satu persatu yang telah membantu selama ini.

Semoga segala bantuan dan dukungan yang telah diberikan mendapatkan balasan yang berlipat ganda. Mohon maaf atas segala kealfaan dan semoga dengan penulisan skripsi ini dapat memberikan informasi serta manfaat bagi pembaca. Amin.

Bandung, Desember 2016

Penulis

“IMPLEMENTASI ALGORITMA *K-NEAREST NEIGHBOR* DALAM SISTEM *CASE BASED REASONING* UNTUK PEMBENTUKAN IDENTITAS JAWABAN OTOMATIS DAN PENCARI KEMIRIPAN JAWABAN DARI SOAL-SOAL ALGORITMA”

Nurida Ahsanti
NIM. 1127050118

ABSTRAK

Minat persaingan dalam mengembangkan teknologi baru terutama pada *Software* semakin ketat, sehingga pembelajaran dari berbagai studi kasus sangat dicari. Namun, apabila mengoleksi referensi studi kasus secara manual pada nyatanya tidak semudah yang diperkirakan apalagi jika harus membandingkan satu persatu dengan kasus sedang dicari solusinya. Salahsatu bentuk untuk membantu dalam pembelajaran tersebut yaitu dengan membangun aplikasi yang dapat menampung semua referensi atau sumber pengetahuan kasus dan dapat menghitung hasil kemiripan serta mengklasifikasikan soal kasus baru yang sedang dicari solusinya dengan metode pengembangan perangkat lunak *Prototype*. Dalam pengimplementasian, ada beberapa hal yang harus dilakukan oleh sistem diantaranya *Pre-Processing*, kemudian menentukan jumlah frekuensi dari tiap *term* dalam dokumen dengan *TF-IDF* dan menggunakan metode untuk mencari kemiripan dengan *Cosine Similarity*. Hal-hal tersebut dilakukan sebagai pendukung dalam mencari kemiripan dari studi kasus yang dicari solusinya dan pengetahuan kasus yang telah ada sebelumnya sehingga proses pencarian dapat dilakukan secara efektif. Algoritma *K-Nearest Neighbor* dipilih sebagai algoritma pengklasifikasi karena dari hasil penelitian yang telah dilakukan sebelumnya banyak yang menyatakan bahwa Algoritma *KNN* mempunyai kinerja yang baik apalagi jika diimplementasikan dengan algoritma *Case Based Reasoning*, sehingga hasil pengimplementasian pada sistem yang dibangun menunjukkan pernyataan yang sama dalam mengklasifikasikan struktur dasar algoritma dengan nilai k = 5 memperoleh akurasi sebesar 0,9 dari pengujian dengan menggunakan 10 soal algoritma dan data *training* sebanyak 90 dokumen yang terbagi kedalam 3 kategori yaitu runtunan, pemilihan dan pengulangan.

Kata Kunci: *Prototype, Pre-Processing, TF-IDF, Cosine Similarity, K-Nearest Neighbor, Case Based Reasoning.*

"IMPLEMENTATION OF K-NEAREST NEIGHBOR ALGORITHM IN CASE BASED REASONING SYSTEM FOR IDENTITY ANSWER TO AUTOMATED ESTABLISHMENT AND SEEKERS STRIKING REPLY FROM ALGORITHM PROBLEMS"

Nurida Ahsanti
NIM. 1127050118

ABSTRACT

Competitive interest in new technologies development, especially in the Software increasingly stringent, lessons learned from the case studies are highly sought after. However, if the reference case studies collected manually in fact is not easy as expected especially if you have to compare one by one with cases being solved. One of the main forms to help in learning to build applications that can accommodate any references or sources of knowledge case and can calculate results about similarities and classify new cases are being searched for a solution with software development method Prototype. In the implementation, there are some things that must be done by systems including the Pre-Processing, and then determine number of frequencies each term in the document with the TF-IDF and using methods to find similarities with Cosine Similarity. Those things are done as support in finding a semblance of a case study to find the solution and knowledge of cases that have been there before the search process can be done effectively. Algorithm K-Nearest Neighbor been voted algorithm classifier because of research has been done before many stating algorithm KNN has a good performance especially when implemented with algorithms Case Based Reasoning results of implementation system built shows same statement in classifying structure basic algorithm with a value of k = 5 gain accuracy 0.9 out 10 questions testing using algorithms and data training has 90 documents are divided into three categories: runs, selection and repetition.

Keywords: Prototype, Pre-Processing, TF-IDF, Cosine Similarity, K-Nearest Neighbor, Case Based Reasoning, Problem Algorithm.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Era digital saat ini cukup jelas membuktikan betapa pesatnya perkembangan teknologi dan informasi. Hal ini akan terus berlangsung karena semakin banyaknya minat dalam persaingan mengembangkan atau membuat berbagai teknologi baru. Pada nyatanya, bukan hanya orang yang khusus terlibat di bidang IT saja yang berpotensi dalam persaingan tersebut tetapi orang-orang di luar IT pun ikut tertarik. Oleh karena itu, pembelajaran tentang proses pembuatan perangkat lunak melalui contoh program/*coding* yang sudah ada sangat diperlukan.

Pembuatan sebuah program dapat diawali dengan mengetahui metodelogi untuk memecahkan masalah yang ada kemudian menuangkan algoritma kedalam suatu notasi tertentu. Namun pada nyatanya untuk pemahaman algoritma tidak dapat disepelekan walaupun hal tersebut tidak dikatakan sulit. Hal tersebut dibuktikan dari hasil kuesioner yang dapat dilihat pada lampiran, dimana kuesioner tersebut disebarkan pada 17 orang mahasiswa jurusan Teknik Informatika di UIN Sunan Gunung Djati Bandung. Sehingga dapat diambil kesimpulan bahwa meskipun pengkodean program bukan sesuatu yang baru, namun dalam memahami dan menuangkan algoritma ke dalam kode program masih terdapat kesulitan dan masih ada yang meragukan kesulitan ketika mencari atau mengoleksi referensi apalagi jika disertai dengan membandingkan antara soal yang ingin dipecahkan dengan refensi yang banyak. Sebagian besar dari mereka

merasa perlu untuk mengetahui soal yang dipecahkan termasuk ke dalam struktur dasar algoritma yang mana dan 16 orang diantaranya merasa perlu adanya aplikasi yang dapat membandingkan kemiripan antara refensi dengan soal algoritma dan dapat memberikan rekomendasi untuk pengerjaan jawaban soal-soal algoritma. Selain itu, dari hasil peninjauan nilai akhir mahasiswa angkatan 2012-2014 pada mata perkuliahan Algoritma dan Struktur Data bahwa 28.52% diantaranya masih mempunyai hasil nilai menengah kebawah. Oleh sebab itu, pembelajaran Algoritma sangat diperlukan dalam upaya peningkatan pembelajaran agar mencapai hasil yang lebih maksimal.

Sebelum dapat memahami Algoritma perlu diketahui bahwa algoritma dibangun dari tiga struktur dasar yaitu Runtunan (*Sequence*), Pemilihan (*Selection*), dan Pengulangan (*Repetition*). Sedangkan pada sebuah kode program terdiri dari tipe data, konstanta, variabel dan algoritma dalam bentuk notasi itu sendiri. Salah satu metode yang dapat dipakai dalam konsep pembelajaran yaitu dengan menggunakan metode *Case Based Reasoning* (*CBR*), karena metode ini mempunyai cara untuk memecahkan masalah yaitu dengan memanfaatkan pengalaman sebelumnya dalam domain pengetahuan tertentu, dan didukung dengan penelitian oleh Gerhana YA dan Djohar A, membuktikan bahwa metode *CBR* dapat meningkatkan kemampuan pemecahan masalah dan model pembelajaran tersebut lebih baik dari model konvensional [1]. Dalam penelitian lain dilakukan oleh Luthfi ET, mengungkapkan bahwa penggunaan *CBR* dapat dilakukan untuk mencari level kedekatan data kasus baru dengan data kasus lama yang menjadi acuan dalam pengambilan keputusan terhadap kasus baru [2].

Perhitungan tingkat kemiripan (jarak) didasarkan pada penggunaan beberapa atribut yang terdefinisi sebelumnya dan dapat menggunakan algoritma pengklasifikasi seperti *K-Nearest Neighbor*, *Naive bayes*, serta gabungan *LVQ* dan *K-Means.* Seperti pada hasil penelitian yang telah dilakukan sebelumnya oleh Santoso D, Ratnawati DE dan Indriati bahwa rata-rata akurasi *KNN* 96%, *Naive Bayes* 98%, serta gabungan *k-means* dan *LVQ* 92,2% [3]. Pada keempat algoritma tersebut dikatakan baik untuk digunakan terutama algoritma *K-Nearest Neighbor* (*KNN*) yang diperkirakan akan menghasilkan pengklasifikasian obyek baru yang lebih cocok untuk dijadikan bahan penelitian karena *KNN* berdasar pada atribut dan data *training* atau mengklasifikasikan dengan *voting* terbanyak dari data yang telah ada sebelumnya, dan apabila menggunakan tambahan metode lain seperti *Decision Rule* dalam penelitian oleh Samuel Y, Delima R dan Rachmat A mempunyai hasil kurang mampu memaksimalkan performa *KNN* [4], sedangkan dengan *Improved KNN* pada peneliitian oleh Rizki AS, Indriati, dan Muflikhah menghasilkan performa yang kurang bagus [5].

Pencarian kemiripan soal dan jawaban algoritma dengan menggunakan algoritma klasifikasi merupakan sebuah solusi dalam memecahkan permasalahan dari soal algoritma. Untuk itu, maka disusunlah penelitian "**Implementasi Algoritma *K-Nearest Neighbor* dalam *Sistem Case Based Reasoning* untuk Pembentukan Identitas Jawaban Otomatis dan Pencari Kemiripan Jawaban dari Soal-Soal Algoritma**".

1.2 Rumusan Masalah

Rumusan masalah dari latar belakang diatas yang akan dijadikan objek pada penelitian adalah sebagai berikut:

1. Bagaimana hasil kolaborasi antara sistem *Case Based Reasoning* dan algoritma *K-Nearest Neighbor* dalam membentuk identitas jawaban otomatis dari soal algoritma dan mencari kemiripan jawaban tersebut?
2. Bagaimana penerapan algoritma *K-Nearest Neighbor* dan metode *Case Based Reasoning* untuk mengetahui tingkat akurasi hasil pembentukan identitas jawaban otomatis dan pencari kemiripan jawaban?

1.3 Tujuan

Tujuan yang ingin dicapai dari permasalahan pada penelitian ini yakni sebagai berikut:

1. Mengetahui hasil kolaborasi dari sistem *Case Based Reasoning* dan algoritma *K-Nearest Neighbor* dalam membentuk identitas jawaban otomatis dari soal algoritma dan mencari kemiripan jawaban.
2. Menerapkan algoritma *K-Nearest Neighbor* dan metode *Case Based Reasoning* untuk mengetahui tingkat akurasi hasil pembentukan identitas jawaban otomatis dan pencari kemiripan jawaban.

1.4 Batasan Masalah

Batasan permasalahan dari penelitian berguna agar penulisan penelitian lebih teratur dan terarah. Adapun batasan permasalahan tersebut meliputi:

1. Implementasi dari sistem ini berbasis *desktop offline*.

2. Pembentukan identitas jawaban otomatis diperoleh dari perbandingan kemiripan antara soal inputan yang ingin dicarikan jawaban dengan soal dalam pengetahuan yang telah memiliki identitas jawaban sebelumnya.
3. Jawaban hasil pencarian berupa 5 rekomendasi yang diurutkan berdasarkan tingkat kemiripan tertinggi dan klasifikasi dari *vote* terbanyak sehingga dapat dijadikan sebagai gambaran untuk pengerjaaan soal-soal algoritma yang jawabannya harus menggunakan *coding.*
4. Potongan kode program berupa gambar dalam bahasa pemrograman C, Pascal dan C++.
5. Tingkat akurasi diperoleh dari hasil pengklsifikasian terhadap pengujian sistem menggunakan Algoritma *K-Nearest Neighbor* dalam sistem *Case Based Reasoning*.

1.5 Metodologi Penelitian

Metodologi yang digunakan pada penelitian ini terbagi menjadi 2 bagian yaitu teknik pengumpulan data dan model proses pengembangan perangkat lunak.

1.5.1 Teknik Pengumpulan Data

Metodologi Penelitian diperlukan sebagai alat bantu untuk memudahkan pekerjaan didalam melakukan perancangan aplikasi. Metodologi penelitian yang digunakan yaitu sebagai berikut:

a. Studi Kepustakaan

Studi pustaka merupakan langkah awal dalam metode pengumpulan data. Studi pustaka merupakan metode pengumpulan data yang diarahkan kepada pencarian data dan informasi melalui jurnal penelitian, internet, buku dan *e-*

book yang dapat mendukung dalam proses penulisan. ”Hasil penelitian juga akan semakin kredibel apabila didukung foto-foto atau karya tulis akademik dan seni yang telah ada [6].” Maka dapat dikatakan bahwa studi pustaka dapat memengaruhi kredibilitas hasil penelitian yang dilakukan.

b. Wawancara

Wawancara atau *interview* merupakan teknik pengumpulan data dengan cara bertatap muka secara langsung dengan informan dan dosen yang bersangkutan. Wawancara dilakukan jika data yang diperoleh kurang mendalam. Wawancara digunakan sebagai teknik pengumpulan data apabila peniliti ingin mengetahui hal-hal dari informan yang lebih mendalam [6].

1.5.2 Model Proses Pengembangan Perangkat Lunak

Adapun untuk pembuatan aplikasi ini, menggunakan metode pengembangan perangkat lunak *Prototype,* karena metode ini lebih memudahkan dalam proses membangun sebuah perancangan aplikasi. Alur dari metode *Prototype* terdiri dari langkah-langkah sebagai berikut:

a. *Communication,* yaitu *Developer* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diinginkan dan gambaran bagian-bagian yang akan dibutuhkan berikutnya.

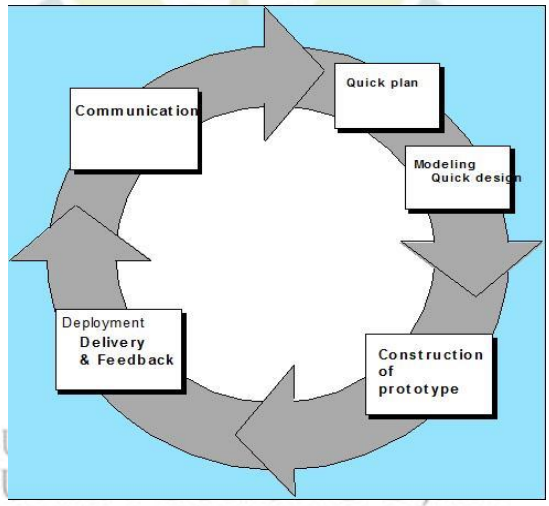
b. *Quick Plan* atau Perancangan yang dilakukan dengan cepat dan mewakili semua aspek *software* yang diketahui, dan rancangan ini menjadi dasar pembuatan *Prototype*.

c. *Modelling Quick Design*, langkah ini berfokus pada representasi aspek *software* yang bisa dilihat *customer/User* dan cenderung ke pembuatan *prototype*.

d. *Construction of Prototype*, yaitu membangun kerangka atau rancangan *prototype* dari *software* yang akan dibangun.

e. *Deployment Delivery & Feedback* merupakan *Prototype* yang telah dibuat oleh *developer* akan disebarkan kepada *User*/klien untuk dievaluasi, kemudian klien akan memberikan *feedback* yang akan digunakan untuk merevisi kebutuhan *software* yang akan dibangun. Pengulangan proses ini terus berlangsung sampai semua kebutuhan terpenuhi [7].

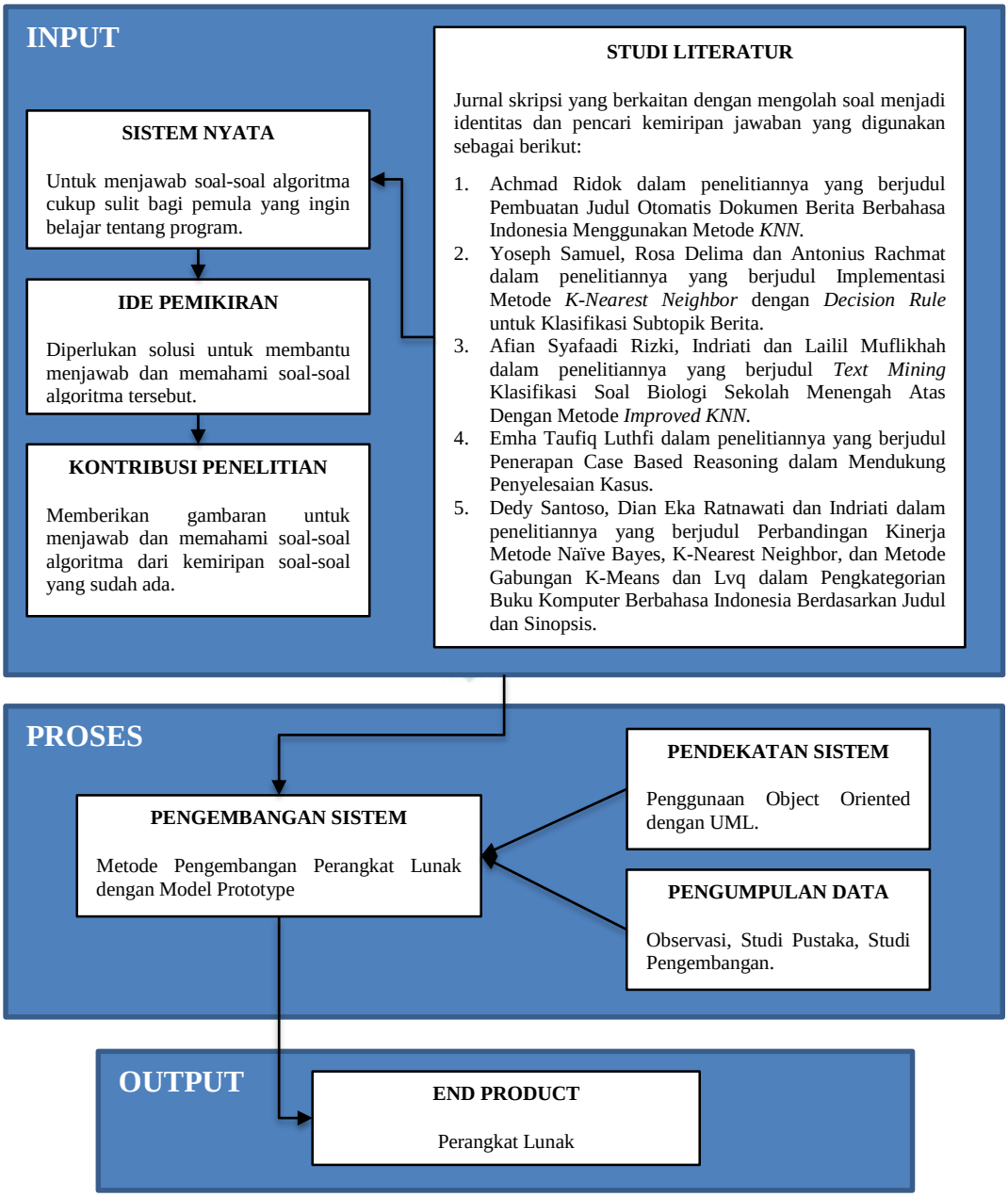
Alur tersebut dapat dijelaskan pada gambar ilustrasi proses model *prototype* sebagai berikut:



Gambar 1.1 Alur Model Pengembangan *Prototype* [7]

1.6 Kerangka Kerja Konseptual

Kerangka kerja konseptual atau alur logis dari penelitian yang akan dibuat adalah sebagai berikut:



Gambar 2.2 Skema Kerangka Kerja Konsep

Dilihat dari kerangka kerja konsep diatas terbagi menjadi 3 tahapan yang bertujuan untuk memudahkan dalam urutan perjabaran dan klasifikasi pencapaian dari skema tersebut. Tahapan tersebut terdiri dari:

1.6.1 *Input*

Tahapan awal yaitu input yang akan menjadi cangkupan pada beberapa kegiatan yang dilakukan seperti:

a. Studi Literatur

Studi Literatur adalah cara untuk menyelesaikan persoalan dengan menelusuri sumber-sumber tulisan yang pernah dibuat sebelumnya. Studi literatur yang dilakukan adalah dengan mencari data dan informasi yang dibutuhkan. Data dan informasi yang dibutuhkan dalam penelitian ini dicari melalui jurnal-jurnal ilmiah terakreditasi, dan hasil-hasil penelitian skripsi, tesis, disertasi, dan laporan praktikum yang sebelumnya pernah dijadikan bahan penelitian.

b. Sistem Nyata

Setelah studi literatur dilakukan, maka didapat permasalahan nyata yang saat ini masih dijadikan perbincangan dan belum ditemukan solusi yang lebih baik. Pada sistem nyata penelitian ini ditemukan permasalahan dalam menjawab soal-soal algoritma cukup sulit bagi pemula yang ingin belajar tentang program.

c. Ide Pemikiran

Setelah ditemukan permasalahan pada sistem nyata munculah ide pemikiran mengenai perlunya solusi untuk membantu menjawab dan memahami soal-soal algoritma.

d. Kontribusi Penelitian

Untuk mendukung solusi permasalahan dari ide yang didapatkan, diperlukan kontribusi penelitian yang berguna dalam memberikan gambaran untuk menjawab dan memahami soal-soal algoritma dari kemiripan soal-soal yang sudah ada.

1.6.2 *Proses*

Proses merupakan tahapan yang selanjutnya yang dilakukan setelah tahap input dan sebelum tahap output. Pengertian dari proses itu sendiri yaitu berupa serangkaian kegiatan yang saling terkait atau berinteraksi yang nantinya akan mengubah input menjadi output. Adapun tahapan proses mencangkup pada beberapa kegiatan yakni sebagai berikut:

a. Pendekatan Sistem

Pendekatan sistem dilakukan dengan membuat perancangan terhadap sistem dengan menggunakan metode pengembangan sistem *Unified Modelling Language* (UML).

b. Pengumpulan Data

Pengumpulan data dilakukan dengan melakukan Studi kepustakaan dan wawancara. Studi pustaka merupakan metode pengumpulan data yang diarahkan kepada pencarian data dan informasi melalui jurnal penelitian, internet, buku dan *e-book* yang dapat mendukung dalam proses penulisan. Sedangkan wawancara dilakukan dengan cara bertatap muka secara langsung dengan informan atau dosen yang bersangkutan. Wawancara digunakan sebagai teknik pengumpulan data apabila peniliti ingin mengetahui berbagai informasi dari informan dengan lebih mendalam.

c. Pengembangan Sistem

Setelah pendekatan sistem dan pengumpulan data selesai, barulah dapat dilakukan pengembangan terhadap sistem. Pengembangan sistem untuk penelitian ini akan menggunakan metode pengembangan perangkat lunak dengan model *prototype*, karena metode ini dianggap lebih memudahkan dalam proses membangun sebuah perancangan aplikasi.

1. *Communication*

   Developer dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diinginkan dan gambaran bagian-bagian yang akan dibutuhkan berikutnya.

2. *Quick Plan*

   Perancangan dilakukan cepat dan mewakili semua aspek *software* yang diketahui, dan rancangan ini menjadi dasar pembuatan *Prototype*.

3. *Modelling Quick Design*

   Berfokus pada representasi aspek *software* yang bisa dilihat *customer/User*. *Modelling QuickDesign* cenderung ke pembuatan *prototype*.

4. *Construction of Prototype*

   Membangun kerangka atau rancangan *Prototype* dari *software* yang akan dibangun.

5. *Deployment Delivery & Feedback*

   Prototype yang telah dibuat oleh *developer* akan disebarkan kepada *User*/klien untuk dievaluasi, kemudian klien akan memberikan

feedback yang akan digunakan untuk merevisi kebutuhan *software* yang akan dibangun. Pengulangan proses ini terus berlangsung sampai semua kebutuhan terpenuhi.

1.6.3 *Output*

Tahapan yang terakhir adalah tahap *output*, yang dimana merupakan hasil akhir keputusan yang dipertimbangkan dari pemikiran kerangka kerja konsep. Tahapan ini mencangkupi hasil yang berupa perangkat lunak.

1.7 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir disusun dalam beberapa bab yang masing-masing bab menguraikan beberapa pokok pembahasan. Adapun sistematika penulisan laporan ini adalah sebagai berikut:

BAB 1 PENDAHULUAN

Bab pendahuluan terdiri dari enam subbab yaitu latar belakang menggambarkan hal-hal yang perlu dijadikan penelitian dari realita yang ada, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, kerangka kerja konsep dan sistematika penulisan. Subbab Latar belakang masalah menggambarkan hal-hal yang perlu dijadikan penelitian dari realita yang ada. Subbab Rumusan masalah dituliskan kedalam bentuk poin yang menjadi sasaran utama pada objek yang akan diteliti. Subbab Tujuan menggambarkan hal-hal yang ingin dicapai. Subbab Batasan masalah berisi batasan yang ditentukan dalam perancangan sistem berupa hal-hal terkait dengan sistem. Subbab Kerangka Kerja Konsep menggambarkan keseluruhan konsep umum dan ide dari penelitian. Subbab Metodologi penelitian terbagi kedalam dua tahapan yaitu teknik

pengumpulan data dan menggambarkan model proses pengembangan perangkat lunak yang dibuat. Dan bagian terakhir dari bab ini yakni dengan subbab sistematika penulisan yang menguraikan urutan penyajian yang digunakan dalam penyusunan skripsi.

BAB II LANDASAN TEORI

Bab landasan teori menjelaskan tentang teori-teori yang digunakan dalam menganalisis permasalahan yang ada dan teori-teori yang akan digunakan untuk membangun perangkat lunak ini.

BAB III ANALISIS DAN PERANCANGAN

Bab analisis dan perancangan menguraikan hasil analisis dan perancangan perangkat lunak yang akan dibangun.

BAB IV IMPLEMENTASI

Bab implementasi menjelaskan tentang spesifikasi perangkat lunak, kebutuhan perangkat lunak, implementasi perangkat lunak, dan pengujian yang dilakukan terhadap perangkat lunak yang dibangun.

BAB V PENUTUP

Bab penutup berisi tentang pernyataan singkat berupa kesimpulan dari pembahasan perangkat lunak yang dibuat secara keseluruhan dan saran untuk mengembangkan perangkat lunak yang lebih baik.

BAB II

LANDASAN TEORI

2.1 Landasan Teori

2.1.1 Definisi Algoritma

Algoritma berasal dari kata *Algoritmi*, yaitu bentuk Latin dari al-Khwarizmi, yang merupakan seorang tokoh islam terkemuka dari Persia yang ahli dalam bidang matematika, astronomi, dan geografi. Untuk memecahkan masalah dengan instansiasi yang kecil, maka solusi dapat ditemukan dengan mudah dan cepat. Lain halnya dengan instansiasi masalah yang berukuran besar, jelas tidak mudah mengurutkan data sebanyak itu. Oleh karena itu, menuliskan prosedur yang berisi langkah-langkah pengurutan sangat diperlukan agar prosedur tersebut dapat dijalankan oleh sebuah pemroses (komputer, manusia, robot, dan sebagainya) untuk menghasilkan solusi masalah pengurutan dalam setiap instansiasi. Prosedur yang berisi langkah-langkah penyelesaian masalah tersebut disebut "algoritma". Algoritma adalah urutan langkah-langkah untuk memecahkan suatu masalah [8]. Dalam matematika dan ilmu komputer, algoritma adalah prosedur langkah-demi-langkah untuk penghitungan. Algoritma digunakan untuk penghitungan, pemrosesan data, dan penalaran otomatis.

Algoritma adalah metode efektif diekspresikan sebagai rangkaian terbatas dari instruksi-instruksi yang telah didefinisikan dengan baik untuk menghitung sebuah fungsi. Dimulai dari sebuah kondisi awal dan input awal, instruksi-instruksi tersebut menjelaskan sebuah komputasi yang bila dieksekusi prosesnya

melalui sejumlah urutan kondisi terbatas yang terdefinisi dengan baik, sehingga pada akhirnya menghasilkan "keluaran" dan berhenti di kondisi akhir [6].

Algoritma biasa disebut sebagai langkah pertama yang harus ditulis sebelum menuliskan program, karena kebanyakan dari masalah yang timbul dapat diselesaikan dengan pemrograman komputer dengan perhitungan matematik. Sehingga sebelum menguasai pemrograman, terlebih dahulu logika dalam berpikir harus lebih terbuka agar dapat memahami konsep dan cara untuk memecahkan masalah pemrograman yang akan dibuat.

Algoritma adalah jantung ilmu komputer atau informatika. Banyak cabang ilmu komputer yang diacu dalam terminologi algoritma. Dalam kehidupan sehari-haripun banyak terdapat proses yang digambarkan dalam suatu algoritma. Sebuah algoritma merupakan deskripsi langkah-langkah pelaksanaan suatu proses. Setiap langkah di dalam algoritma dinyatakan dalam sebuah pernyataan (*statement*) atau istilah lainnya instruksi. Sebuah pernyataan berisi aksi (*action*) yang dilakukan. Bila sebuah pernyataan dieksekusi oleh pemroses, maka aksi yang bersesuaian dengan pernyataan itu dikerjakan [8].

2.1.2 Struktur Dasar Algoritma

Algoritma mempunyai beberapa jenis pernyataan seperti ekspresi, pemilihan, pengulangan, prosedur, gabungan, dan sebagainya. Algoritma berisi langkah-langkah penyelesaian masalah yang dapat membentuk tiga buah konstruksi atau struktur dasar seperti runtunan (*sequence*), pemilihan (*selection*), dan pengulangan (*repetition*).

Konstruksi yang pertama adalah runtunan. Sebuah runtunan terdiri dari satu atau lebih pernyataan sehingga sering disebut dengan "pernyataan gabungan" (*compound statements*). Setiap pernyataan ditulis dalam satu baris atau dipisahkan dengan tanda titik koma. Tiap pernyataan dikerjakan secara berurutan (sekuensial) sesuai dengan urutan di dalam teks algoritma atau secara singkatnya instruksi dilaksanakan setelah instruksi sebelumnya selesai dilaksanakan. Urutan instruksi menentukan keadaan akhir algoritma. Oleh karena itu, apabila urutan instruksi tersebut diubah maka hasil akhirnya mungkin akan berubah pula.

Konstruksi selanjutnya adalah pemilihan. Struktur pemilihan dapat dikodekan dengan *if-then* yang artinya hanya memberikan satu pilihan aksi apabila kondisi (persyaratan) dipenuhi (bernilai benar), dan tidak memberi pilihan aksi lain apabila kondisi bernilai salah. Selain itu, pada konstruksi pemilihan dapat ditambahkan *else* "kalau tidak" yang diletakkan setelah kode *if-then*, kegunaan *else* ini mempunyai arti bahwa apabila kondisi terpenuhi maka aksi 1 akan dikerjakan dan apabila kondisi salah maka aksi 2 yang akan dikerjakan. Kelebihan struktur pemilihan terletak pada kemampuannya yang memungkinkan memproses mengikuti jalur aksi yang berbeda berdasarkan kondisi yang ada. Tanpa struktur pemilihan, maka memungkinkan tidak dapat menuliskan algoritma untuk memecahkan permasalahan praktis meskipun sangat kompleks.

Konstruksi yang terakhir adalah pengulangan, dalam algoritma terdapat banyak notasi pengulangan yang dapat digunakan, antara lain *repeat N times*, *for*, *repeat-until*, dan *while*. Notasi pertama, *repeat N times* yang artinya ulangi sebanyak *N* kali. Notasi kedua, struktur pengulangan yang mirip dengan *repeat N times* adalah *for* yang artinya aksi dilakukan sebanyak hitungan cacah

pengulangan, yaitu dari 1 sampai *N* (sebanyak *N* kali). Pencacah pengulangan dapat di atur tidak hanya mulai dari 1, tetapi juga dari sembarang nilai yang lain. Struktur pengulangan yang ketiga adalah *repeat-until* (*repeat* artinya "ulangi" dan *until* artinya "sampai" atau "hingga"), pengulangan dilakukan hingga kondisi (persyaratan) berhenti terpenuhi. Struktur pengulangan yang terakhir adalah *while* artinya "selagi" atau "selama". Yaitu selama kondisi (persyaratan) pengulangan masih benar, maka aksi dikerjakan. Perbedaannya dengan *repeat-until*, jika ada *repeat-until* kondisi pengulangan dievaluasi di akhir, sedangkan pada *while-do* kondisi pengulangan dievaluasi di awal pengulangan [8].

2.1.3 Notasi Algoritmik

Algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Langkah-langkah penyelesaian tersebut dapat dituliskan dalam notasi algoritmik sembarang agar mudah dibaca dan dipahami. Tidak ada notasi yang standar untuk menuliskan algoritma seperti notasi pada bahasa pemrograman. Setiap orang dapat mendefinisikan notasi algoritmiknya sendiri karena notasi algoritma tidak sama dengan kode program komputer. Program komputer adalah implementasi algoritma dalam notasi bahasa pemrograman tertentu.

Notasi algoritmik yang baik adalah notasi mudah dibaca dan ditranslasikan kedalam bahasa pemrograman. Notasi algoritmik berupa *pseudo-code* yang mempunyai korespondensi dengan notasi bahasa pemrograman sehingga proses penerjemah dari *pseudo-code* ke kode program menjadi lebih mudah. Tidak ada aturan baku membuat *pseudo-code*. Tidak seperti bahasa pemrograman yang direpotkan dengan tanda titik koma (*semicolon*), indeks, format keluaran, kata-

kata khusus, dan sebagainya. Sembarang versi *pseudo-code* dapat diterima asalkan notasinya bisa dipahami [8].

Penulisan *Pseudo-code* berbeda dengan penulisan struktur teks algoritma, karena pada struktur teks algoritma selalu disusun oleh tiga bagian yaitu:

1. Judul (*header*)

   Judul terdiri dari nama program dan penjelasan (spesifikasi) tentang program tersebut. Judul diawali dengan kata kunci program dan nama program *X*. Kata program ini bukan menyatakan program dalam bahasa komputer, tetapi menyatakan bahwa sedang menulis algoritma untuk program pemecahan masalah. Nama program sebaiknya singkat namun cukup menggambarkan apa yang dilakukan oleh program. Di bawah nama program sebaiknya disertai dengan penjelasan singkat mengenai apa yang sedang dilakukan oleh progam secara lengkap termasuk masukan dan keluaran program.

2. Deklarasi (*declaration*)

   Deklarasi digunakan untuk mengumumkan semua nama yang dipakai di dalam algoritma beserta propertinya. Nama tersebut dapat berupa nama konstanta, nama peubah, nama tipe, nama prosedur dan nama fungsi. Semua nama yang dipakai di dalam algoritma harus dikenali sebelum mereka digunakan. Deklarasi dapat dikosongkan apabila tidak ada penggunaan nama dalam program.

3. Algoritma

   Algoritma merupakan bagian inti dari sebuah program yang berisi instruksi-instruksi pemecahan masalah dalam notasi *pseeudo-code.*

2.1.4 *Case Based Reasoning* (*CBR*)

Penalaran berbasis kasus *(CBR)* adalah cara pemecahan masalah dengan memanfaatkan pengalaman sebelumnya pada domain pengetahuan tertentu. Maher et al (1995) mengungkapkan bahwa "*CBR* adalah sebuah pendekatan untuk masalah pemecahan yang menggunakan *database* atau kasus masalah sebelumnya yang selesai ketika pemecahan masalah baru pada *database* berupa kumpulan data yang disimpan dalam komputer" [1].

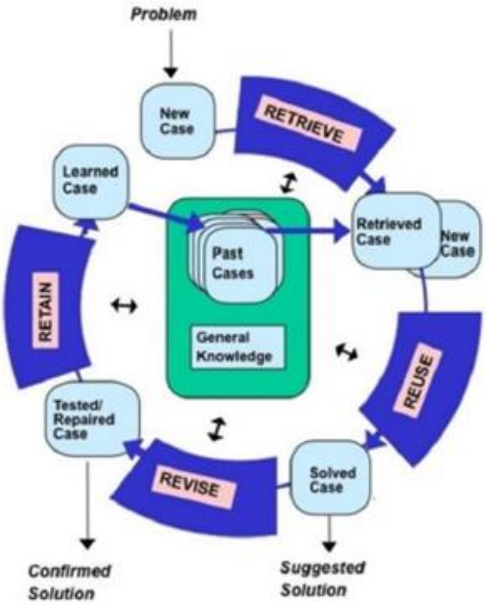
Case Based Reasoning (*CBR*) suatu model penalaran yang menggabungkan pemecahan masalah, pemahaman dan pembelajaran serta memadukan keseluruhannya dengan pemrosesan memori. Tugas tersebut dilakukan dengan memanfaatkan kasus yang pernah dialami oleh sistem, yang mana kasus merupakan pengetahuan dalam konteks tertentu yang mewakili suatu pengalaman yang menjadi dasar pembelajaran untuk mencapai tujuan sistem. Atau dalam definisi lain *CBR* merupakan metode pemecahan masalah/kasus baru dengan melakukan adaptasi terhadap metode yang digunakan untuk memecahkan masalah/kasus lama [2].

Secara umum siklus proses *CBR* adalah sebagai berikut:

1. *Retrieve,* yaitu mengambil kembali yang paling menyerupai/kasus yang relevan (mirip) dengan kasus baru. Fase pengambilan dimulai dengan menggambarkan/menguraikan beberapa masalah, dan berakhir ketika

menemukan kecocokan untuk masalah sebelumnya yang memiliki tingkat kompatibilitas tertinggi. Bagian ini mengacu pada istilah dari identifikasi, pencocokan awal, pencarian, penyeleksian dan pelaksanaan.

2. *Reuse*, yaitu menggunakan kembali pengetahuan dan informasi berdasarkan bobot kasus lama yang memiliki kesamaan paling relevan dengan kasus baru, menghasilkan pada sekumpulan solusi yang mungkin diperlukan untuk adaptasi dengan masalah baru.
3. *Revise*, yaitu merevisi solusi yang diusulkan dan mengujinya pada kasus nyata (simulasi) dan jika perlu meluruskan solusi ini untuk mencocokkan kasus baru.
4. *Retain*, yaitu mempertahankan atau menyimpan kasus baru yang telah mendapat solusi agar dapat digunakan oleh kasus berikutnya mirip dengan kasus ini. Tetapi jika solusi baru gagal, akan menjelaskan kegagalannya kemudian memperbaiki solusi yang digunakan dan menguji lagi.



Gambar 2.3 Siklus *CBR* [9]

Pada gambar diatas, terdapat plot metodologi *CBR* yang jelas dalam memecahkan suatu permasalahan. Ketika masalah baru datang, pertama-tama sistem akan melakukan proses *Retrieve*/ambil. Proses tersebut akan melakukan dua langkah pengolahan, yaitu pengenalan dari masalah dan mencari persamaan pokok masalah pada *Database*. Setelah proses *Retrieve* selesai, sistem akan melakukan proses *Reuse*. Dalam Proses *reuse*, sistem akan menggunakan informasi sebelumnya yang memiliki masalah yang sama untuk memecahkan masalah baru. Proses *Reuse* akan menyalin, memilih, dan melengkapi informasi yang akan digunakan. Kemudian pada proses *Revise*, informasi akan dihitung, dievaluasi dan diperbaiki untuk mengatasi kesalahan yang terjadi pada masalah baru. Dalam proses akhir, sistem akan melakukan proses *Retain* / Mempertahankan. Proses mempertahankan pada indeks, mengintegrasikan, dan mengekstrak solusi baru. Selain itu, solusi baru akan disetorkan ke pengetahuan dasar untuk memecahkan masalah yang akan datang. Tentu saja, masalah yang akan dipecahkan adalah masalah yang memiliki kesamaan dengan yang sebelumnya [1].

2.1.5 *Data Mining*

Data Mining merupakan serangkaian proses untuk menggali nilai tambah sekumpulan data berupa pengetahuan yang tidak diketahui secara manual. *Data Mining* dipakai dalam bermacam-macam bidang ilmu seperti kecerdasan buatan (*artificial intelligent*), *machine learning*, statistik dan *database*.

Proses penerapan metode *Data mining* mempunyai maksud untuk mengungkap pola-pola tersembunyi. Dengan kata lain *data mining* dapat disebut

sebagai proses untuk penggalian pola-pola dari data agar dapat menjadi alat untuk mengubah data menjadi informasi.

Penggunaan *data mining* adalah untuk membantu dalam analisis pengamatan perilaku. Beberapa teknik yang sering disebutkan pada literatur penerapan *Data Mining* antara lain: *clustering, classification, association rule mining, neural network, genetic algorithm* dan lain-lain.

2.1.6 *K-Nearest Neighbor* (*KNN*)

K-Nearest Neighbor (KNN) adalah suatu metode yang menggunakan algoritma *supervised* dimana hasil dari *query instance* yang baru diklasifikasi berdasarkan mayoritas dari kategori *KNN*. Tujuan dari algoritma ini adalah mengklasifikasikan obyek baru bedasarkan atribut dan data *training.* Pengklasifikasian tidak menggunakan model apapun untuk pencocokkannya dan hanya berdasarkan pada memori. Klasifikasi algoritma ini menggunakan *voting* terbanyak dari obyek *k* dan menggunakan ketetanggaan sebagai nilai prediksi dari *query instance* yang baru.

Algoritma metode *KNN* sangatlah sederhana, bekerja berdasarkan jarak terpendek dari *query instance* ke data *training* untuk menentukan *KNN*-nya. Data *training* diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data *training.* Sebuah titik pada ruang ini ditandai kelac *c* jika kelas *c* merupakan klasifikasi yang paling banyak ditemui pada *k* buah tetangga terdekat dari titik tersebut. Dekat atau jauhnya tetangga biasanya

dihitung berdasarkan jarak dan perhitungan jarak tersebut dapat menggunakan metode *cosine similarity.*

Ketepatan algoritma *KNN* sangat dipengaruhi oleh adanya fitur-fitur, sedangkan fitur tersebut tidak relevan terhadap pengklasifikasiannya atau dapat dikatakan tidak ditentukan berdasarkan apa yang diklasifikasikan. Menurut riset yang telah dilakukan, algoritma ini sebagian besar membahas tentang bagaimana memilih dari bobot terhadap fitur agar performa klasifikasi menjadi lebih baik.

KNN memiliki beberapa kelebihan yaitu ketangguhan terhadap *training* data yang memiliki banyak *noise* dan efektif apabila *training* data-nya besar. Sedangkan, kelemahan *KNN* adalah perlunya penentuan nilai pasti dari parameter *k* (jumlah dari tetangga terdekat), *training* pada *KNN* berdasarkan pada jarak yang tidak jelas tentang jenis jarak apa yang harus digunakan dan atribut mana yang harus digunakan untuk mendapatkan hasil terbaik, dan biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap *query instance* pada keseluruhan *training* [10].

Pengklasifikasian algoritma *KNN* mempunyai langkah-langkah sebagai berikut:

a. Tentukan parameter *K*.
b. Hitung jarak antara data yang akan dievaluasi dengan semua pelatihan.
c. Urutkan jarak yang terbentuk (urut naik).
d. Tentukan jarak terdekat sampai urutan *K*.
e. Pasangkan kelas yang bersesuaian.

f. Cari jumlah kelas dari tetangga yang terdekat dan tetapkan kelas tersebut sebagai kelas data yang akan dievaluasi.

2.1.7 *Text Mining*

Text mining adalah salah satu bidang khusus dari *data mining*. *Text mining* dapat didefinisikan sebagai suatu proses menggali informasi dimana seorang user berinteraksi dengan sekumpulan dokumen menggunakan *tools* analisis yang merupakan komponen-komponen dalam *data mining* yang salah satunya adalah kategorisasi. Adapun tugas khusus dari *text mining* antara lain yaitu pengkategorisasian teks *(text categorization)* dan pengelompokan teks *(text clustering)* [11].

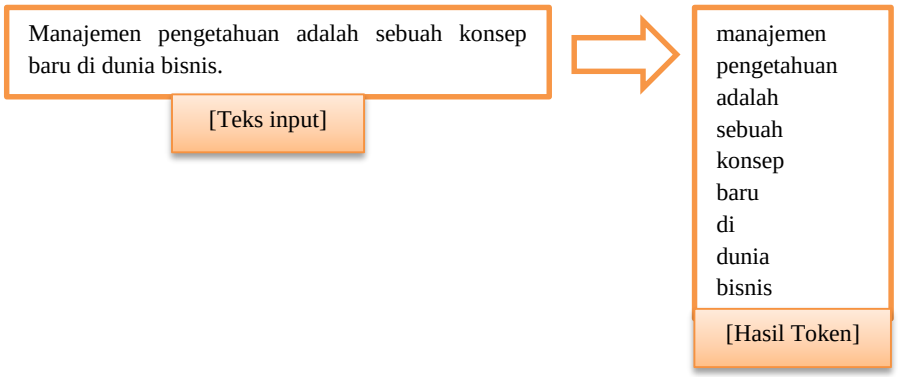
Secara umum dalam *text mining* pada dokumen atau suatu teks dilakukan tahap sebagai berikut:

a. *Case Folding*

Case folding merupakan tahapan yang biasa dilakukan pada awal proses. Tahapan ini berfungsi untuk mengubah semua huruf dalam teks menjadi huruf kecil dan menghilangkan karakter selain a-z.

b. *Tokenizing*

Proses *tokenizing* adalah tahap pemotongan string input berdasarkan tiap kata yang menyusunnya. Proses ini menghasilkan kata-kata yang berdiri sendiri [12].



Gambar 2.4 Contoh *Tokenizing*

c. *Filtering*

Proses *filtering* adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stop list* (membuang kata-kata yang kurang penting atau *word list*. Proses ini akan dihasilkan kata yang penting saja dan membuang kata kata yang kurang penting [12].

Gambar 2.5 Contoh *Filtering*

d. *Stemming*

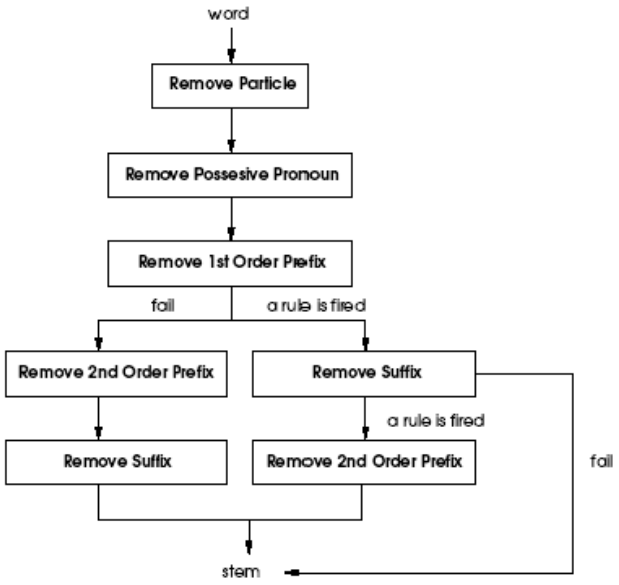
Proses *Stemming* adalah proses untuk mengubah bentuk kata menjadi kata dasar. Cara kerjanya adalah dengan membuang imbuhan, sisipan, dan akhiran. Tujuan proses *stemming* diantaranya adalah meningkatkan efisiensi sistem [5]. Proses pembentukan kata dasar dapat menggunakan algoritma *porter*. Algoritma *porter* adalah algoritma untuk pencarian kata dasar

khusus bahasa inggris yang ditemukan oleh Martin Porter 1980. Mekanisme pengerjaan algoritma untuk mencari kata dasar yaitu dengan membuang akhiran pada kata berbahasa Inggris yang mengandung imbuhan karena dalam bahasa Inggris tidak mengenal awalan. Untuk menyesuaikan Algoritma *Porter* dengan bahasa Indonesia, maka dilakukan beberapa modifikasi pada pembuangan kata karena bahasa Inggris memimiliki penyesuaian tata aturan bahasa yang sangat berbeda dengan bahasa Indonesia.

Selain algoritma porter stemmer, ada algoritma lain seperti algoritma Nazief Adriani. Dari kedua algoritma untuk *stemming* tersebut masing-masing mempunyai kelebihan dan kekurangan. Hal tersebut telah dibuktikan oleh penelitian sebelumnya dengan mengambil data pengujian pada 30 dokumen teks Bahasa Indonesia dengan ukuran dokumen yang bervariasi yang bertujuan untuk membandingkan hasil stem, waktu proses, dan presisi stemming pada dokumen dengan menggunakan kedua algoritma tersebut.

Berdasarkan perancangan dan implementasi dari riset tersebut, diperoleh kesimpulan untuk proses stemming dokumen yang menggunakan Algoritma Porter membutuhkan waktu yang lebih singkat dan memiliki presentase keakuratan (presisi) lebih kecil dibandingkan dengan stemming menggunakan Algoritma Nazief & Adriani. Selain itu, pada proses stemming Algoritma Nazief & Adriani, kamus yang digunakan sangat mempengaruhi hasil stemming sehingga semakin lengkap kamus yang digunakan maka semakin akurat pula hasil stemming. [13]

Dilihat dari hasil riset tersebut, maka penelitian ini akan menggunakan Algoritma *Porter* untuk proses *stemming* karena pada pemrosesan aplikasi dibutuhkan waktu yang relatif singkat sebagai pendukung proses *KNN* dalam menjalankan tugasnya. Untuk desain Algoritma *Porter Stemmer* untuk Bahasa Indonesia tersebut dapat dilihat pada gambar berikut:



Gambar 2.6 Porter Stemmer Untuk Bahasa Indonesia [14]

Langkah-langkah Pada Algoritma *Porter Stemmer* [15]

1. Menghapus partikel seperti: -kah, -lah, -tah.
2. Mengapus kata ganti (Possesive Pronoun), seperti –ku, -mu, -nya.
3. Mengapus awalan pertama.
4. Jika awalan pertama tidak ditemukan, maka menghapus awalan kedua dan dilanjutkan dengan menghapus akhiran sehingga kata akhir diasumsikan sebagai kata dasar (root word). Sedangkan jika ada maka menghapus akhiran, jika tidak ditemukan maka kata tersebut diasumsikan sebagai kata dasar (root word). Jika ditemukan maka

Menghapus awalan kedua dan kata akhir diasumsikan sebagai kata dasar (root word).

2.1.8 *TF-IDF*

Tahapan awal *Text Mining* sebelum dilakukan proses *TF-IDF* adalah tahap *tokenisasi* dan *stopword*. *TF-IDF* merupakan salah satu metode yang dapat dilakukan untuk pembobotan terhadap *term. TF* (*Term Frequency*) adalah pembobotan kata (*term*) yang didasarkan pada perhitungan jumlah kata yang muncul pada suatu dokumen. *IDF* (*Inverse Document Frequency*) adalah pembobotan kata (*term*) yang didasarkan pada perhitungan jumlah kata yang muncul pada seluruh dokumen [16]. Semakin banyak kata dalam dokumen, semakin besar bobot kata tersebut, begitu pula sebaliknya. *TF-IDF* (*Term Frequency - Inverse Document Frequency*) merupakan pembobotan sebuah kata dalam satu dokumen agar dapat diproses lebih lanjut oleh algoritma lain [4].

$W_{dt} = tf_{dt} * IDF_t = tf_{dt} \log \frac{D}{df_t}$ ..... [17]

Dimana:

d = dokumen ke-d

t = kata ke-t kata kunci

W = bobot dokumen ke-d terhadap kata ke-t

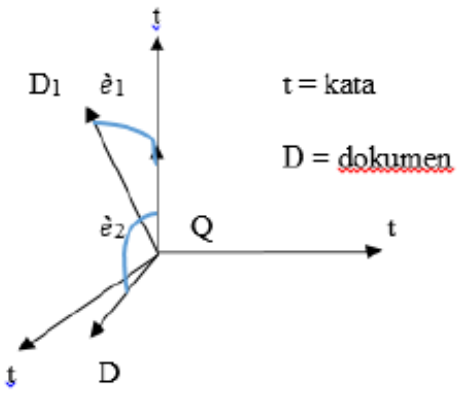
tf = banyaknya kata yang dicari pada sebuah dokumen

D = total dokumen

2.1.9 *Cosine Similarity*

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. Semakin besar nilai kesamaan vektor *query* dengan vektor dokumen maka *query* tersebut dipandang semakin relevan dengan dokumen. Saat mesin menerima *query*, mesin akan membangun sebuah vektor Q (Wq1 ,Wq2 ,…Wqt) berdasarkan istilah-istilah pada *query* dan sebuah vektor D (di1 ,di2 ,…dit) berukuran t untuk setiap dokumen. Pada umumnya *cosine similarity (CS)* dihitung dengan rumus *cosine measure*. [18]



Gambar 2.7 Vektor Skalar

Perhitungan *Cosine Similarity* dapat diimplementasikan dengan rumus:

$$CS(b_1, b_2) = \frac{\sum_{t=1}^{n} W_{t,b1}\, W_{t,b2}}{\sqrt{\sum_{t=1}^{n} W^2\, t,b1 \sum_{j=1}^{n} W^2 t,b2}} \quad \text{..... [18]}$$

Dimana:

T = *term* dalam kalimat

$W_{t,b1}$ = bobot *term* t dalam *block* b1

$W_{t,b2}$ = bobot *term* t dalam *block* b2

2.1.10 *Unified Modeling Languege*

Unified Modeling Language (UML) adalah sistem arsitektur yang bekerja dalam *OOAD* dengan bahasa yang konsisten untuk menentukan, *visualisasi*, mengkonstruksi, dan mendukomentasikan *artifact* yang terdapat dalam sistem

software untuk memodelkan bisnis dan sistem *non-software* lainnya. *Artifact* yaitu sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa perangkat lunak. *UML* merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. [19]

UML memiliki beberapa diagram dalam pembuatan suatu model, sebab diagram berfungsi untuk menjelaskan elemen-elemen dalam sistem secara grafis. Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisis atau rekayasa, yang diantaranya sebagai berikut:

a. *Use Case Diagram*

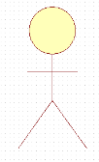
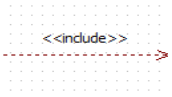
Use case diagram menjelaskan manfaat sistem jika dilihat menurut sudut pandang orang yang berada diluar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar. *Use case diagram* dapat digunakan selama proses analisis untuk menangkap *requirements* sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use case* diagram menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa *use case diagram*.

Use case diagram dapat sangat membantu dalam penyusunan *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari

proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Tabel 2.1 Komponen-komponen *Use Case*

Simbol	Nama	Keterangan
	*Actor*	Aktor adalah pengguna sistem. Aktor tidak terbatas hanya manusia saja, sehingga apabila sistem ingin berkomunikasi dengan sistem lain yang menerima *input* atau memberi *output*, maka sistem tersebut dapat dianggap pula sebagai aktor.
	Use Case	*Use case* digambarkan dengan lingkaran *elips* dengan tertulis nama *use case* didalamnya.
	Association	Asosiasi digambarkan dengan sebuah garis yang berguna untuk menghubungkan aktor dengan *use case*.
	*Include*	*Include* adalah Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya. [20]
	*Ekstends*	*Extend* adalah Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu. [20]

b. *Class Diagram*

Class digram membantu dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class diagram* memperlihatkan hubungan antar kelas dan penjelasan detail tiap-

tiap kelas didalam model desain dari suatu sistem. Selama proses analisis, *class diagram* memperhatikan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang yang membentuk arsitektur sistem yang dibuat. *Class diagram* juga merupakan pondasi untuk *component diagram* dan *deployment diagram*. Dalam sebuah model mungkin terdapat beberapa diagram kelas dengan spesifikasi tersendiri.

Pada sebuah *Class* memiliki tiga area pokok yaitu Nama (*stereotype*), Atribut dan Metoda. Sedangkan untuk menyatakan hubungan antar *class* dalam pembuatan *class diagram*, dapat menggunakan beberapa penghubung sebagai berikut:

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

c. *Sequence Diagram*

Sequence diagram menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosisasi dengan *use case*. *Sequence diagram* memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu didalam *use case*. Tipe diagram ini sebaiknya digunakan diawal tahap desain atau analisis karena kesederhanaanya dan mudah untuk dimengerti.

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Tujuan penggunaan sequence diagram yakni:

1. Mengkomunikasikan *requirement* kepada tim teknis karena diagram ini dapat lebih mudah untuk dielaborasi menjadi model *design*.
2. Merupakan *diagram* yang paling cocok untuk mengembangkan model deskripsi *use-case* menjadi spesifikasi *design*.

d. *Activity Diagram*

Activity diagram memodelkan alur kerja *(workflow)* sebuah proses bisnis dan urutan aktivitas dalam suatu proses. *Diagram* ini sangat mirip dengan sebuah *flowchart* karena kita dapat memodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas kedalam keadaan sesaat *(state)*. Seringkali bermanfaat bila membuat sebuah *activity diagram* terlebih dahulu dalam memodelkan sebuah proses secara keseluruhan. *Activity diagram* juga sangat berguna ketika menggambarkan perilaku

paralel atau menjelaskan bagaimana perilaku dalam berbagai *use case* berinteraksi.

2.2 Tinjauan Pustaka

Untuk menentukan posisi penelitian ini dengan penelitian yang terdahulu maka dibuatlah *State Of The Art*. Pada penelitian ini membandingkan dengan tiga jurnal yang berkaitan dengan "Implementasi Algoritma *K-Nearest Neighbor* Dalam *Sistem Case Based Reasoning* Untuk Pencari Jawaban dari Soal-soal Algoritma".

Achmad Ridok dalam penelitiannya yang berjudul "Pembuatan Judul Otomatis Dokumen Berita Berbahasa Indonesia Menggunakan Metode *KNN*", membuat judul otomatis dokumen berita. Analisis pembentukan judul secara otomatis menekankan ide pembelajaran korelasi antara kata-kata yang menyusun dokumen dari pelatihan dan menerapkan model tersebut untuk membuat judul pada data pengujian. Penelitian ini bertujuan untuk menghasilkan judul dari suatu artikel dokumen secara otomatis. Data yang digunakan adalah dokumen berita berbahasa Indonesia. Tahapan dari penelitian yaitu Preproses, persamaan dokumen, *KNN*, menemukan kesamaan terbesar, mendapatkan judul untuk dokumen. Penarikan kesimpulan implementasi algoritma disajikan dalam bentuk tabel yang berisi daftar judul asal dan judul hasil yang telah diuji cobakan [16].

Yoseph Samuel, Rosa Delima dan Antonius Rachmat dalam penelitiannya yang berjudul "Implementasi Metode *K-Nearest Neighbor* dengan *Decision Rule* untuk Klasifikasi Subtopik Berita", membuat sistem klasifikasi topik berita otomatis berdasarkan gabungan dari algoritma *K-Nearest Neighbor* dengan

algoritma *Decision Rule*. Analisis dari algoritma *K-Nearest Neighbor* yaitu sebuah algoritma yang sering digunakan untuk klasifikasi teks dan data, sedangkan *Decision Rule* digumakan untuk jumlah dokumen yang jauh lebih banyak. Penelitian ini bertujuan untuk memaksimalkan penggunaan algoritma *K-Nearest Neighbor*. Data yang digunakan adalah berita dari 3 *website* diantaranya bbc.com, cnn.com, dan foxnews.com. Tahapan dari penelitiannya yaitu *tokenizing*, *stopwords*, *stemming*, pembobotan pada tiap *term* dengan *TF-IDF*, perhitungan *cosine* dan *euclidean distance*. Penarikan kesimpulan implementasi algoritma disajikan dalam bentuk tabel persentase keakuratan [4].

Afian Syafaadi Rizki, Indriati dan Lailil Muflikhah dalam penelitiannya yang berjudul "*Text Mining* Klasifikasi Soal Biologi Sekolah Menengah Atas Dengan Metode *Improved KNN*", membuat pengkategorian soal-soal biologi SMA kedalam empat kategori yaitu hewan, tumbuhan, protista, ekosistem. Analisis dalam menghadapi tes banyak siswa yang merasa kesulitan yang salah satu penyebabnya adalah proses pembelajaran yang kurang optimal. Data yang digunakan adalah soal-soal biologi SMA. Penelitian ini bertujuan untuk dapat membantu pengajar dalam mengevaluasi siswanya, sekaligus dapat memberikan materi-materi pelajaran tertentu secara interaktif. Tahapan dari penelitiannya yaitu *Preprocessing*, Pembobotan, *cosine similiarity*, *Improved KNN*, Pengujian. Penarikan kesimpulan implementasi algoritma disajikan dalam bentuk grafik [5].

Emha Taufiq Luthfi dalam penelitiannya yang berjudul "Penerapan *Case Based Reasoning* dalam Mendukung Penyelesaian Kasus", membuat sistem yang lebih fleksibel dalam mendukung penyelesaian kasus/permasalahan yang bersifat

samar. Analisis pada *Rule Based System* dengan *database* konvensional memiliki kekurangan dalam mendukung penyelesaian kasus/permasalahan yang bersifat samar atau dengan level kemiripan yang tidak 100% terhadap informasi tersimpan. Penelitian ini bertujuan untuk mendukung penyelesaian kasus/permasalahan berdasar kasus/permasalahan terdahulu yang telah diketahui statusnya. Tahapan dari penelitiannya yaitu Pengambilan data lama, Pembobotan, Penginputan data kasus baru, Perhitungan level kemiripan kasus, Pencarian kemiripan kasus level terkecil. Penarikan kesimpulan implementasi metode dan algoritma disajikan dalam bentuk list/daftar [2].

Dedy Santoso, Dian Eka Ratnawati dan Indriati dalam penelitiannya yang berjudul "Perbandingan Kinerja *Metode Naïve Bayes, K-Nearest Neighbor*, dan Metode Gabungan *K-Means* dan *Lvq* dalam Pengkategorian Buku Komputer Berbahasa Indonesia Berdasarkan Judul dan Sinopsis", membuat perbandingan kinerja dari algoritma klasifikasi *Naïve Bayes, K-Nearest Neighbor*, dan Metode Gabungan *K-Means* dan *Lvq*. Analisis yang telah dialami oleh seorang pustakawan mendapatkan kendala dalam pengkategorian buku, karena biasanya pustakawan masih menggunakan cara yang kurang efisien. Penelitian ini bertujuan untuk mencari metode terbaik dalam mengkategorikan buku, khususnya buku komputer berbahasa Indonesia. Tahapan dari penelitiannya yaitu *Preprocessing*, Klasifikasi, Pengujian, Pembandingan hasil uji dan Penentuan metode terbaik. Penarikan kesimpulan pembandingan hasil uji algoritma disajikan dalam bentuk grafik [3].

Yana Aditia Gerhana dan As'ari Djohar dalam penelitiannya yang berjudul "Case-based Reasoning Learning Model to Develop Skill in Problem Solving of Student of Vocational Education", membuat penelitian tentang pengembangan model pembelajaran *CBR* dan metode *SDLC* untuk meningkatkan pemecahan masalah siswa SMK dalam keterampilan pemecahanan masalah komputer. Penelitian ini bertujuan untuk meningkatkan kemampuan pemecahan masalah siswa SMK dalam keterampilan pemecahan masalah komputer, membuktikan teori-teori mengenai keuntungan dari model pembelajaran yang memanfaatkan teknologi Informasi & Komunikasi, dan membuktikan keuntungan dari model pembelajaran *CBR*. Tahapan dari penelitiannya yaitu dan *CBR* (*Retrieve, Reuse, Revise, Retain*) [1].

Penarikan kesimpulan pembandingan dari penelitian disajikan dalam bentuk tabel dibawah ini:

Tabel 2.2 *State Of The Art*

No	Penelitian	Algoritma	Data	Hasil
1.	Achmad Ridok	*K-Nearest Neighbor*	Dokumen Berita Berbahasa Indonesia	Dengan metode *KNN* dapat melakukan pembentukan judul secara otomatis suatu dokumen dan menghasilkan kinerja terbaik dalam klasifikasi tetapi tidak dapat membuat judul baru dan sangat bergantung terhadap data latih.
2.	Yoseph Samuel, Rosa Delima dan Antonius Rachmat	*KNN* dan *Decision Rule*	Berita dari *website*	Penggunaan *Decision Rule* menambah keakuratan sekitar 2% dan kurang mampu memaksimalkan performa *KNN*. Algoritma *KNN* sendiri memberikan hasil keakuratan yang baik sekitar 88%.
3.	Afian Syafaadi Rizki, Indriati dan Lailil Muflikhah	*Improved KNN*	Soal-soal Biologi SMA	Performa metode *improved KNN* untuk klasifikasi soal biologi SMA masih kurang bagus karena adanya kesamaan term diantara kategori.
4.	Emha Taufiq Luthfi	*CBR* dan *Nearest*	Rule Based	Penggunaan *CBR* dengan algoritma *Nearest Neighbor* dapat dilakukan

		Neighbor	System	untuk mencari level kedekatan data kasus baru dengan data kasus lama yang menjadi acuan dalam pengambilan keputusan terhadap kasus baru.
5.	Dedy Santoso, Dian Eka Ratnawati dan Indriati	*KNN, Naive Bayes* serta gabungan *K-Means* dan *LVQ*	200 buku komputer dalam 5 kategori	Hasil pengujian menunjukkan bahwa rata-rata akurasi metode *KNN* adalah 96%, sedangkan metode *Naive Bayes* adalah 98%, dan metode gabungan *k-means* dan *LVQ* adalah 92,2%. Sehingga metode *Naive Bayes* adalah yang terbaik dalam mengkategorikan buku komputer bahasa Indonesia.
6.	Yana Aditia Gerhana dan As'ari Djohar	*Case Based Reasoning*	Sampel purposive dari 522 siswa dan 8 guru di SMK Garut	Hasil model pembelajaran CBR dapat meningkatkan kemampuan pemecahan masalah siswa SMK dalam keterampilan pemecahan masalah komputer. Hasil model pembelajaran ini lebih baik dari yang konvensional model.

BAB III

ANALISIS DAN PERANCANGAN

3.1 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 :

a. *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.

b. 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*).

c. *Output*, hasil proses disimpan ke dalam database.

Proses yang dilakukan ketika mencari jawaban adalah :

a. *Input*, pengguna memasukan data berupa soal.

b. 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*.

c. *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*.

3.1.1 Analisis Kebutuhan *Non-Fungsional*

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

a. Analisis Kebutuhan Perangkat Keras

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

1. *Intel® Atom™* Inside™ CPU @ 2.40GHz 2.40GHz
2. RAM 2 GB (1.89 GB usable)
3. *Harddisk* 464.3 GB
4. Sistem operasi *Windows* 7 Pro

b. 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*

c. 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*).

3.1.2 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 3.3 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.*

3.1.3 Analisis Data

Untuk menjalankan sistem ini dibutuhkan data yang *valid* dari admin untuk dijadikan pengetahuan dari soal dan jawaban dari studi kasus algoritma. Sebagai pendukung agar aplikasi dapat berjalan dengan baik, maka dibutuhkan ketelitian dalam penginputan pengetahuan tersebut. Hal tersebut harus dilakukan karena pemrosesan awal akan sangat berpengaruh terhadap pemrosesan lainnya yaitu hasil akhir dari kemiripan dan pengklasifikasian.

3.1.4 Analisis Proses *Text Mining*

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

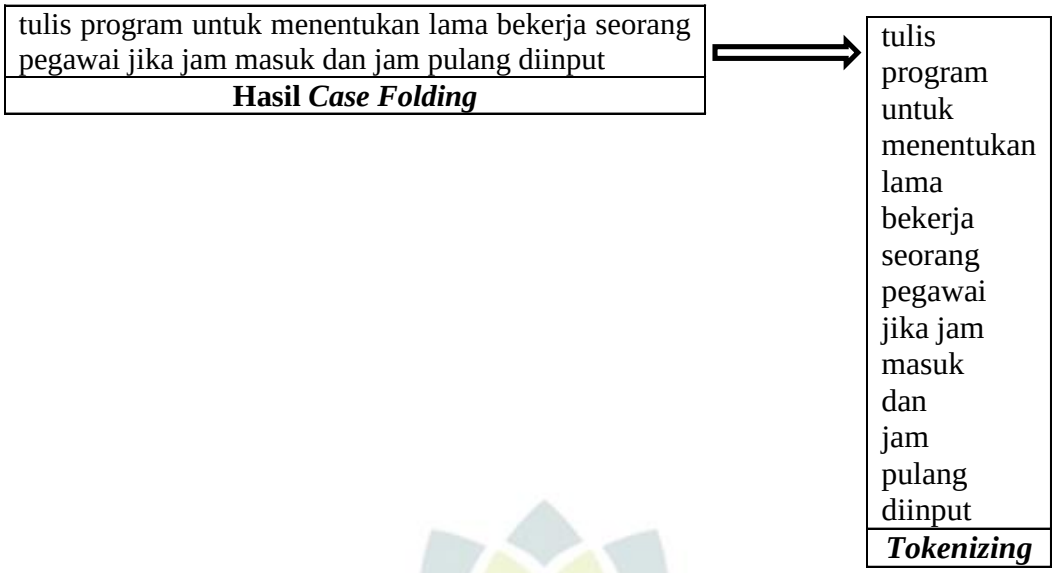
a. *Case Folding*

Proses *case folding* berfungsi untuk mengubah semua huruf dalam teks menjadi huruf kecil dan menghilangkan karakter selain a-z. Sebagai contoh pengimpementasian *case folding* dapat dilihat sebagai berikut:

Gambar 3.8 Proses *Case Folding*

b. *Tokenizing*

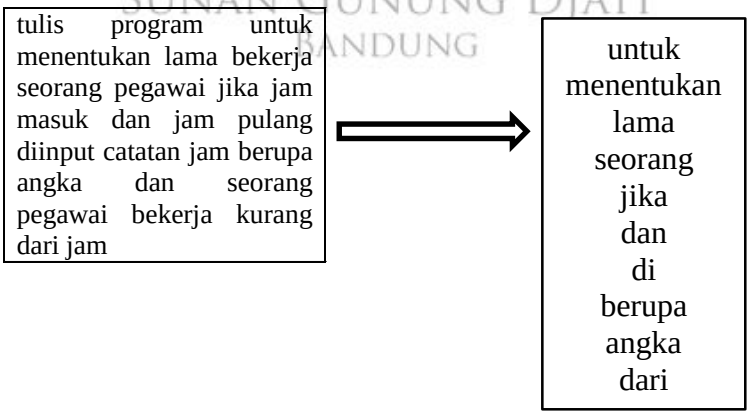
Proses *Tokenizing* dilakukan setelah proses *case folding* selesai. Proses ini berfungsi untuk memecah kalimat menjadi kata-kata.



Gambar 3.9 Proses *Tokenizing*

c. *Filtering*

Proses *Filtering* dilakukan setelah proses *tokenizing*, yang berfungsi untuk membuang kata yang dianggap tidak penting menggunakan *stopword removal*. Pembuangan kata tersebut dilakukan dengan cara menghapus kata yang termasuk pada kamus kata. Berikut ini adalah sebagian kata yang dibuang pada contoh pengimplementasian. Sedangkan untuk kamus kata *stopword removal* disertakan lebih lanjut pada halaman lampiran.



Gambar 3.10 Kata Yang Termasuk *Stopword Removal*

d. *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 yang dikembangkan oleh Fadillah Z Tala dalam penelitiannya yang berjudul "*A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*" [14].

Tabel 3.4 Algoritma *Porter Stemmer*

No.	Kata Asal	Kata Dasar	Keterangan
1.	bekerja	kerja	Kata asal memiliki imbuhan awalan "be-"
2.	diinput	input	Kata asal memiliki imbuhan awalan "di-"
3.	catatan	catat	Kata asal memiliki imbuhan akhiran "-an"
4.	kenaikkan	naik	Kata asal memiliki imbuhan awalan "ke-" dan akhiran "-kan"
5.	pendataan	data	Kata asal memiliki imbuhan awalan "pen-" dan akhiran "-an"
6.	penyusunan	susun	Kata asal memiliki imbuhan awalan "peny-" yang diganti dengan "s" dan akhiran "-an"

3.1.5 Analisis Pembobotan *TF-IDF*

Penganalisisan 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 bobot diambil dari 8 data *training* yang sebelumnya telah diproses pada *text mining*.

Analisis pembobotan dilakukan dengan menentukan banyaknya *TF (Term Frequency), DF (Document Frequency)* dan *IDF* (*Inverse Document Frequency*) untuk menghitung jumlah frekuensi kata yang muncul pada seluruh dokumen

karena semakin banyak kata yang muncul maka kata tersebut dianggap semakin penting. Pada nilai *idf* dihitung dengan menggunakan rumus persamaan :

$$Idf = \log \frac{N}{df} \quad \text{..... [18]}$$

Berikut ini contoh hasil perhitungan *tfidf* menggunakan pengujian dengan *Query*: "hitunglah kenaikkan suhu dari pengujian yang dilakukan oleh seorang mahasiswa", dengan 5 data *training* sebagai berikut:

Tabel 3.5 Contoh Data *Training*

No.	Soal Algoritma
D1	Tulis program untuk menentukan lama bekerja seorang pegawai, jika jam masuk dan jam pulang diinput.
D2	Tulis program untuk menentukan biaya parkir yang dihitung berdasarkan lama parkir.
D3	Program menenentukan urutan dari sekumpulan nilai ujian mahasiswa yang dibaca dari kecil ke besar.
D4	Program perhitungan suhu *Fahrenheit* ke suhu *Celcius* dengan kenaikkan sebesar *step*.
D5	Menghitung cepat rambat cahaya dalam ilmu fisika dengan satuan dalam kilometer.

Tabel 3.6 Perhitungan *TF*, *DF* dan *IDF*

Term	***TF***						***DF***	**Idf = log N/*df***
	Q	**D1**	**D2**	**D3**	**D4**	**D5**		
baca				1			1	0,778
besar				1	1		2	0,477
biaya			1				1	0,778
cahaya						1	1	0,778
celcius					1		1	0,778
cepat						1	1	0,778
dasar			1				1	0,778
fahrenheit					1		1	0,778
fisika						1	1	0,778
gawai		1					1	0,778
hitung	1				1	1	3	0,301
ilmu						1	1	0,778
input		1					1	0,778
jam		2					1	0,778
kecil				1			1	0,778

Term	*TF*						*DF*	Idf = log N/*df*
	Q	D1	D2	D3	D4	D5		
kerja		1					1	0,778
kilometer						1	1	0,778
kumpul				1			1	0,778
mahasiswa	1			1			2	0,477
masuk		1					1	0,778
naik	1				1		2	0,477
nilai				1			1	0,778
parkir			2				1	0,778
program		1	1	1	1		4	0,176
pulang		1					1	0,778
rambat						1	1	0,778
satuan						1	1	0,778
step					1		1	0,778
suhu	1				2		2	0,477
tulis		1	1				2	0,477
uji	1		1				2	0,477
urut			1				1	0,778

Dilihat dari segi *Query* yang diujicobakan pada tabel diatas disimpulkan oleh cara perhitungan sebagai berikut:

$Idf_{(hitung)} = \log \frac{6}{3} = 0{,}301$ $Idf_{(naik)} = \log \frac{6}{2} = 0{,}477$

$Idf_{(suhu)} = \log \frac{6}{2} = 0{,}477$ $Idf_{(uji)} = \log \frac{6}{2} = 0{,}477$

$Idf_{(mahasiswa)} = \log \frac{6}{2} = 0{,}477$

Sebagai tahapan akhir dari proses pembobotan yaitu dengan menghitung kata kunci serta digunakannya rumus adalah sebagai berikut:

$W_{d,t} = tf_{d,t} * IDF_t$ ..... [18]

Tabel 3.7 Perhitungan *Tf * Idf*

Term	$W_{d,t} = tf_{d,t} * IDF_t$					
	Q	**D1**	**D2**	**D3**	**D4**	**D5**
baca	0	0	0	0,778	0	0
besar	0	0	0	0,477	0,477	0
biaya	0	0	0,778	0	0	0
cahaya	0	0	0	0	0	0,778
celcius	0	0	0	0	0,778	0
cepat	0	0	0	0	0	0,778
dasar	0	0	0,778	0	0	0
fahrenheit	0	0	0	0	0,778	0
fisika	0	0	0	0	0	0,778
gawai	0	0,778	0	0	0	0
hitung	0,301	0	0	0	0,301	0,301
ilmu	0	0	0	0	0	0,778
input	0	0,778	0	0	0	0
jam	0	1,556	0	0	0	0
kecil	0	0	0	0,778	0	0
kerja	0	0,778	0	0	0	0
kilometer	0	0	0	0	0	0,778
kumpul	0	0	0	0,778	0	0
mahasiswa	0,477	0	0	0,477	0	0
masuk	0	0,778	0	0	0	0
naik	0,477	0	0	0	0,477	0
nilai	0	0	0	0,778	0	0
parkir	0	0	1,556	0	0	0
program	0	0,176	0,176	0,176	0,176	0
pulang	0	0,778	0	0	0	0
rambat	0	0	0	0	0	0,778
satuan	0	0	0	0	0	0,778
step	0	0	0	0	0,778	0
suhu	0,477	0	0	0	0,954	0
tulis	0	0,477	0,477	0	0	0
uji	0,477	0	0,477	0	0	0
urut	0	0	0,778	0	0	0

3.1.6 Analisis Pencari Kemiripan

Algoritma *Cosine similarity* digunakan untuk menghitung pendekatan relevansi *query* terhadap dokumen yang berprinsip semakin tinggi hasil maka *query* tersebut semakin mirip dengan dokumen yang telah tesedia. 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:

$$CS(b_1, b_2) = \frac{\sum_{t=1}^{n} W_{t,b1}\, W_{t,b2}}{\sqrt{\sum_{t=1}^{n} W^2\, t,b1 \sum_{j=1}^{n} W^2 t,b2}} \quad \text{..... [18]}$$

Perhitungan skalar didapatkan dari hasil perkalian antara bobot query dan perkalian bobot tiap dokumen dengan rumus *WD * Wdi* atau sama dengan potongan rumus *Cosine Similarity* yaitu $\sum_{t=1}^{n} W_{t,b1}\, W_{t,b2}$, untuk pengimplementasian dibuat kedalam bentuk tabel sebagai berikut:

Tabel 3.8 Perhitungan Panjang Skalar

Term	***WD * Wdi***				
	D1	**D2**	**D3**	**D4**	**D5**
baca	0	0	0	0	0
besar	0	0	0	0	0
biaya	0	0	0	0	0
cahaya	0	0	0	0	0
celcius	0	0	0	0	0
cepat	0	0	0	0	0
dasar	0	0	0	0	0
fahrenheit	0	0	0	0	0
fisika	0	0	0	0	0
gawai	0	0	0	0	0
hitung	0	0	0	0,091	0,091
ilmu	0	0	0	0	0
input	0	0	0	0	0
jam	0	0	0	0	0

Term	*WD * Wdi*				
	D1	**D2**	**D3**	**D4**	**D5**
kecil	0	0	0	0	0
kerja	0	0	0	0	0
kilometer	0	0	0	0	0
kumpul	0	0	0	0	0
mahasiswa	0	0	0,228	0	0
masuk	0	0	0	0	0
naik	0	0	0	0,228	0
nilai	0	0	0	0	0
parkir	0	0	0	0	0
program	0	0	0	0	0
pulang	0	0	0	0	0
rambat	0	0	0	0	0
satuan	0	0	0	0	0
step	0	0	0	0	0
suhu	0	0	0	0,455	0
tulis	0	0	0	0	0
uji	0	0,228	0	0	0
urut	0	0	0	0	0
	0	0,228	0,228	0,774	0,091

Perhitungan selanjutnya agar dapat menggunakan rumus *cosine similarity* terlebih dahulu harus menghitung panjang vektor, seperti pada tabel berikut:

Tabel 3.9 Perhitungan Panjang Vektor

Term	**Panjang Vektor**					
	Q	**D1**	**D2**	**D3**	**D4**	**D5**
baca	0	0	0	0,606	0	0
besar	0	0	0	0,228	0,228	0
biaya	0	0	0,606	0	0	0
cahaya	0	0	0	0	0	0,606
celcius	0	0	0	0	0,606	0
cepat	0	0	0	0	0	0,606
dasar	0	0	0,606	0	0	0
fahrenheit	0	0	0	0	0,606	0
fisika	0	0	0	0	0	0,606
gawai	0	0,606	0	0	0	0
hitung	0,091	0	0	0	0,091	0,091
ilmu	0	0	0	0	0	0,606

Term	Panjang Vektor					
	Q	**D1**	**D2**	**D3**	**D4**	**D5**
input	0	0,606	0	0	0	0
jam	0	2,422	0	0	0	0
kecil	0	0	0	0,606	0	0
kerja	0	0,606	0	0	0	0
kilometer	0	0	0	0	0	0,606
kumpul	0	0	0	0,606	0	0
mahasiswa	0,228	0	0	0,228	0	0
masuk	0	0,606	0	0	0	0
naik	0,228	0	0	0	0,228	0
nilai	0	0	0	0,606	0	0
parkir	0	0	2,422	0	0	0
program	0	0,031	0,031	0,031	0,031	0
pulang	0	0,606	0	0	0	0
rambat	0	0	0	0	0	0,606
satuan	0	0	0	0	0	0,606
step	0	0	0	0	0,606	0
suhu	0,228	0	0	0	0,911	0
tulis	0	0,228	0,228	0	0	0
uji	0,228	0	0,228	0	0	0
urut	0	0	0,606	0	0	0
	1,001	5,708	4,725	2,908	3,304	4,329
	1,001	2,389	2,174	1,705	1,818	2,081

Perhitungan vektor diatas hasilnya sama dengan potongan rumus *Cosine Similarity* yaitu $\sqrt{\sum_{t=1}^{n} W^2\, t, b1}$ dan $\sqrt{\sum_{j=1}^{n} W^2\, t, b2}$ sehingga dapat dihitung:

Cosine Similarity (Dok1) = $\frac{0}{1{,}001 * 2{,}389} = 0$

Cosine Similarity (Dok2) = $\frac{0{,}228}{1{,}001 * 2.174} = 0{,}105$

Cosine Similarity (Dok3) = $\frac{0{,}228}{1{,}001 * 1{,}705} = 0{,}133$

Cosine Similarity (Dok4) = $\frac{0{,}774}{1{,}001 * 1{,}818} = 0{,}425$

Cosine Similarity (Dok5) = $\frac{0{,}091}{1{,}001 * 2{,}081} = 0{,}044$

Tabel 3.10 Urutan Kemiripan *Cosine Similarity*

D1	D2	D3	D4	D5
0	0.105	0,133	0,425	0,044
Rank 5	Rank 3	Rank 2	Rank 1	Rank 4

3.1.7 Analisis Klasifikasi *KNN*

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

Sebagai pemgimplementasian algoritma dalam aplikasi, maka diambil dari 3 data *training* yang dijadikan sebagai nilai *k,* dimana dokumen tersebut sebelumnya sudah dihitung hasil dari *cosine similarity*. Pada data *training* tersebut telah memiliki klasifikasi masing-masing, yaitu data dokumen *training* 2 dan 4 mempunyai klasifikasi pengulangan, sedangkan pada data dokumen *training* ke 3 mempunyai klasifikasi pemilihan.

Tabel 3.11 Klasifikasi *KNN*

Dokumen 4	Dokumen 3	Dokumen 2	Dokumen *test*
0,425 (Pengulangan)	0,133 (Pengulangan)	0.105 (Pemilihan)	Pengulangan
Rank 1	Rank 2	Rank 3	

Hasil keputusan klasifikasi *KNN* ditentukan oleh klasifikasi terbanyak dari dokumen yang telah termasuk pada nilai *k.* Karena dokumen ke 2 dan 4 mempunyai klasifikasi pengurutan dan pada dokumen ke 3 mempunyai klasifikasi pemilihan, maka data *test* tersebut dimasukkan kedalam klasifikasi pengurutan.

3.1.8 Analisis Berbasis Kasus *CBR*

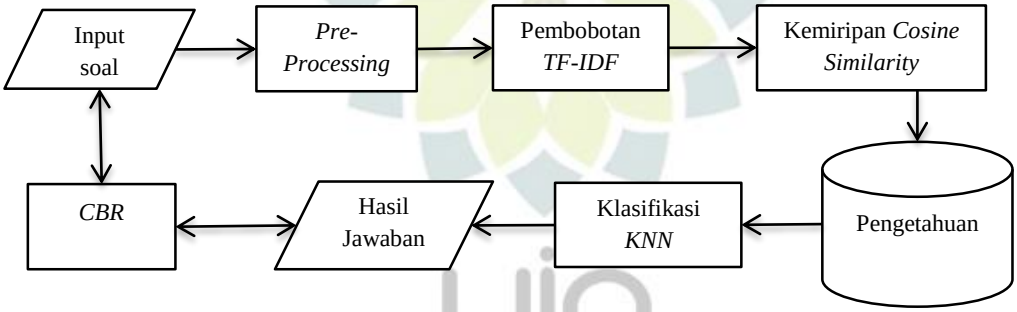
Penelitian ini menggunakan Algoritma *Case Based Reasoning* (*CBR*) yaitu model penalaran yang menggabungkan pemecahan masalah, pemahaman, pembelajaran dan memadukan keseluruhannya dengan pemrosesan yang dilakukan dengan memanfaatkan kasus yang pernah dialami oleh sistem, sebagai dasar dari pengetahuan 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:

a. *Retrieve,* digunakan untuk mencari kemiripan kasus lama dan kasus baru dengan memakai algoritma *similarity.*
b. *Reuse,* digunakan untuk menentukan klasifikasi data *test* dengan menggunakan algoritma *KNN,* hal tersebut dilakukan karena agar dapat menggunakan kembali informasi dan pengetahuan berdasarkan klasifikasi dari data kasus *training* untuk memecahkan masalah kasus baru (proses ini disebut “transfer solusi”).
c. *Revise,* digunakan untuk kasus baru hasil dari pencari jawaban yang disimpan dengan status “belum *valid*” dan apabila tidak di-*validasi* oleh admin maka pengetahuan baru tersebut tidak dapat dijadikan perbandingan dengan soal *input* baru pada pencarian jawaban.
d. *Retain,* digunakan untuk menyimpan semua pengalaman kasus termasuk kasus baru yang telah direvisi untuk memecahkan masalah kasus yang akan datang ke dalam *database.*

3.1.9 Deskripsi Global Aplikasi

Sistem yang dibangun berupa aplikasi pencarian kemiripan soal dan jawaban algoritma merupakan aplikasi berbasis *java off-line.* Aplikasi ini berguna untuk 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.

3.1.10 Arsitektur Sistem



Gambar 3.11 Arsitektur Sistem

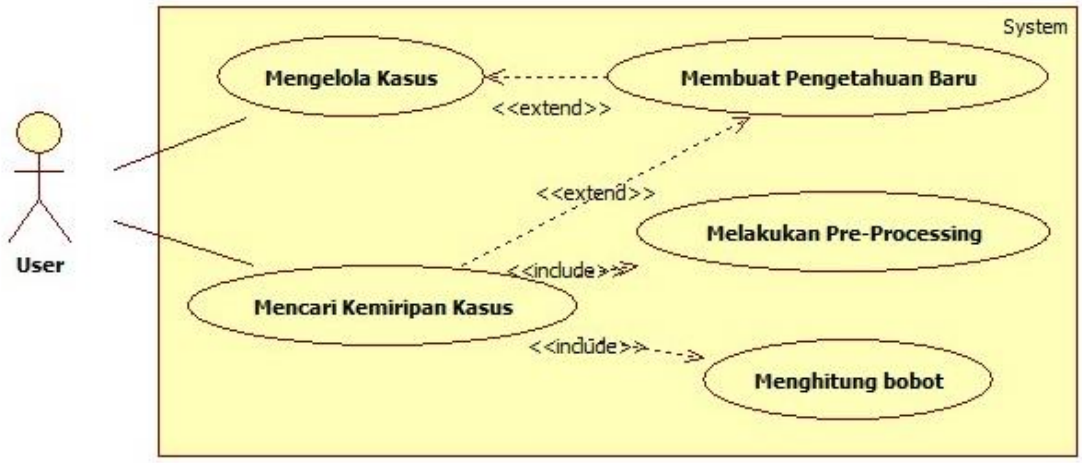
Program menerima inputan berupa soal algoritma, kemudian dilakukan tahapan pra-proses untuk mengekstrak soal menjadi identitas/label jawaban. Tahapan praproses dilakukan dengan *Text Mining* (pengolahan teks untuk menemukan pengetahuan baru) yang meliputi tahap: *Tokenizing* (pemotongan string input 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.

3.2 Perancangan Sistem

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

3.2.1 *Use Case* Diagram



Gambar 3.12 *Use Case* Diagram

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

a. *Use case* Mengelola Kasus, pada *use case* ini menggambarkan semua pengetahuan kasus yang telah ada pada *database* dan siap untuk digunakan untuk proses sistem.

b. *Use case* Membuat 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.

c. *Use case* Mencari 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.

d. *Use case* Melakukan *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.*

e. *Use case* Menghitung bobot, 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.

3.2.2 Definisi *Actor*

Pendeskripsian dari *actor* harus menjelaskan wewenang yang dapat dilakukan dalam perangkat lunak. Sehingga dapat dibuat kedalam bentuk tabel sebagai berikut:

Tabel 3.12 Definisi *Actor*

Actor	**Deskripsi**
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.

3.2.3 Skenario *Use Case*

Skenario *use case* berfungis untuk penggambaran alur tentang tata cara penggunaan sistem dimana setiap skenario digambarkan berdasarkan sudut pandang aktor yang berinteraksi dengan perangkat lunak. Berikut ini beberapa skenario *use case* berdasarkan semua *use case* yang ada yaitu:

Tabel 3.13 Skenario *Use Case* Mengelola Kasus

Identifikasi	
Nomor	1
Nama	Mengelola 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	2. Sistem menyediakan beberapa kolom inputan.
3. Memillih menu simpan	4. Sistem memproses soal dengan tahapan *pre-processing* (*tokenizing, filtering* dan *stemming*), kemudian menyimpan hasil ke *database.*

5. Memilih menu lihat data	6. Sistem akan pindah ke halaman tabel data.
7. Memilih salah satu data dan menekan tombol pilih data	8. Sistem akan mentransfer pilihan ke halaman kelola aplikasi didalam kolom-kolom data.
9. Merubah data dan menekan tombol edit	10. Sistem akan meng-u*pdate* data pada *database* sesuai dengan id.
11. Memilih data pada tabel lihat data dan menekan tombol hapus	12. Sistem akan men-*delete* data pada *database* sesuai dengan id.
13. Memilih data pada tabel lihat data dan menekan tombol batal	14. 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 3.14 Skenario *Use Case* Membuat Pengetahuan Baru

Identifikasi	
Nomor	2
Nama	Membuat 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.	2. Sistem akan mentransfer pilihan ke halaman kelola aplikasi didalam kolom-kolom data.
3. Merubah atau merevisi data dan menekan tombol edit	4. Sistem akan meng-u*pdate* data pada *database* sesuai dengan id.
5. Menekan tombol hapus	6. 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 3.15 Skenario *Use Case* Mencari Kemiripan Kasus

Identifikasi		
Nomor	3	
Nama	Mencari 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		2. Sistem menyediakan kolom untuk penginputan soal.
3. Menekan tombol cari jawaban		4. 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.
7. Melihat hasil jawaban pada kolom identitas jawaban, gambar potongan program dan hasil perhitungan *similarity* dan *KNN*.		8. 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 3.16 Skenario *Use Case* Membuat *Pre-processing*

Identifikasi	
Nomor	4
Nama	Membuat *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	2. Sistem menyediakan kolom untuk penginputan soal.
3. Menekan tombol cari jawaban	4. 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.

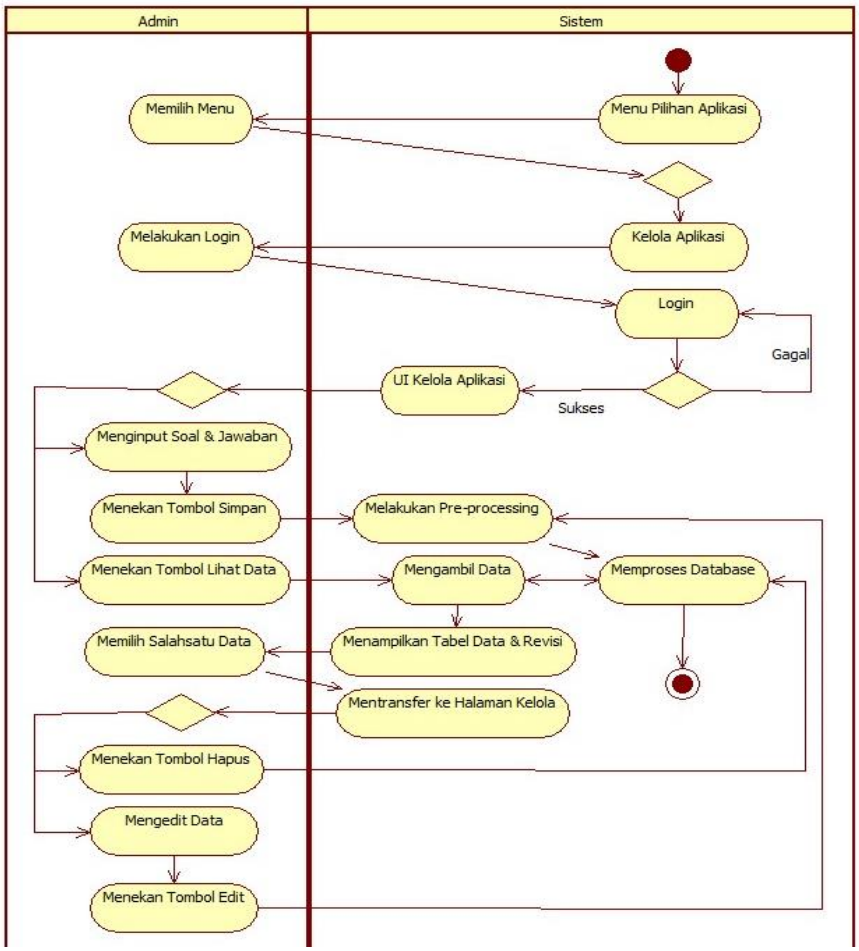
Tabel 3.17 Skenario *Use Case* Menghitung bobot

Identifikasi	
Nomor	5
Nama	Menghitung bobot
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.*
2. Melihat hasil jawaban pada kolom identitas jawaban, gambar potongan program dan hasil perhitungan *similarity* dan *KNN.*	3. 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.

3.2.4 *Activity* Diagram

Activity diagram berfungsi untuk memodelkan alur kerja dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah *flowchart* karena pemodelannya dapat dibentuk menjadi sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas kedalam keadaan sesaat *(state)*.

a. *Activity* diagram *User* Sebagai Admin

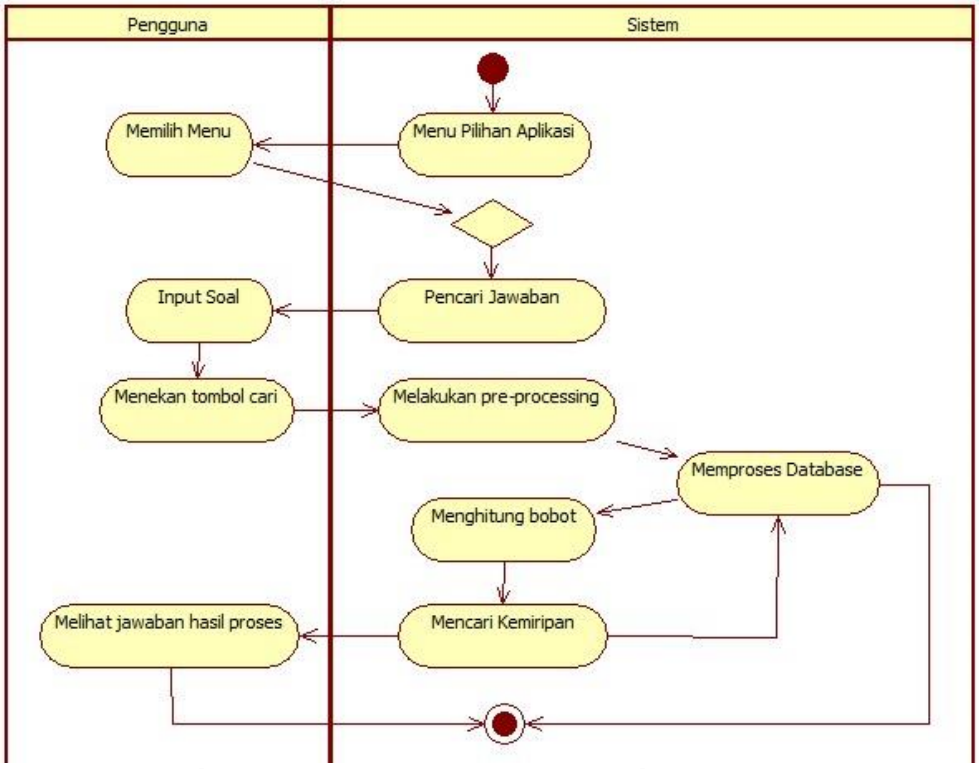


Gambar 3.13 *Activity* Diagram *User* Sebagai Admin

Activity diagram diatas menggambarkan alur dari interaksi antara Admin dan sistem, dimana ketika sistem dimulai maka akan menampilkan halaman menu dan Admin memilih kelola aplikasi. Ketika Admin memilih kelola aplikasi, maka akan langsung ke halaman login yang apabila login sukses,

maka akan ke halaman kelola aplikasi dan jika login gagal, maka halaman login akan memberi pernyataan bahwa login gagal sehingga Admin harus melakukan login ulang. Pada pengelolaan aplikasi ini Admin mempunyai hak akses untuk menginput data soal dan jawaban baru, mengedit dan menghapus dari data yang sudah ada maupun revisi yang diusulkan dari hasil pengolahan pencarian jawaban yang telah dilakukan oleh pengguna dengan pemrosesan yang hanya sampai pada tahap stemming dan langsung dimasukkan kedalam *database.*

b. *Activity* diagram *User* Sebagai Pengguna



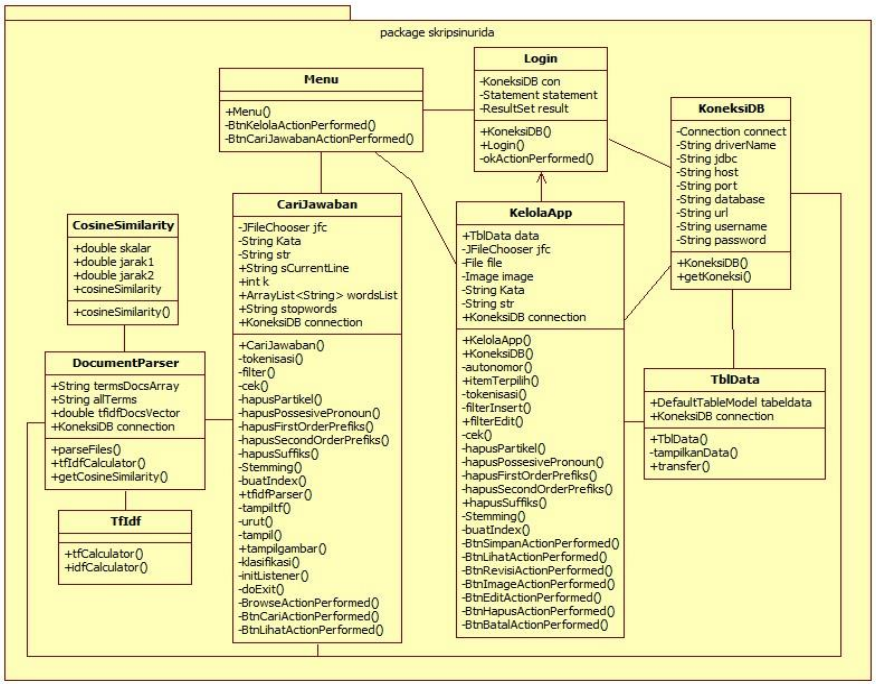
Gambar 3.14 *Activity* Diagram *User* Sebagai Pengguna

Activity diagram diatas menggambarkan alur dari interaksi antara pengguna dan sistem, dimana ketika sistem dimulai maka akan menampilkan halaman menu dan pengguna memilih pencari jawaban. Pada halaman pencari

jawaban ini pengguna dapat menginput soal dan memulai pencarian jawaban dengan menekan tombol cari saja. Pada pengolahan pencarian jawaban mempunyai beberapa tahapan, yaitu: 1) *pre-processing* yang terdiri dari proses *case folding, tokenizing* dan *stemming* ; 2) proses pembobotan tiap *term* dokumen dengan *TF-IDF*, perhitungan tingkat kemiripan oleh algoritma *cosine similarity* dan pengklasifikasian oleh algoritma *KNN*. Hasil dari pengolahan berupa identitas jawaban, klasifikasi dari soal, gambar potongan kode program dan hasil tersebut akan disimpan kedalam *database* untuk direvisi oleh Admin.

3.2.5 *Class* Diagram

Class diagram menggambarkan atribut sistem yang dapat memanipulasi keadaan tersebut (metoda/fungsi). Untuk itu maka dibuatlah *class* diagram untuk aplikasi dari penelitian, yakni sebagai berikut:



Gambar 3.15 *Class* Diagram

Fase analisis proses *class* diagram memperhatikan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem seperti pada tabel berikut:

Tabel 3.18 *Class* Diagram

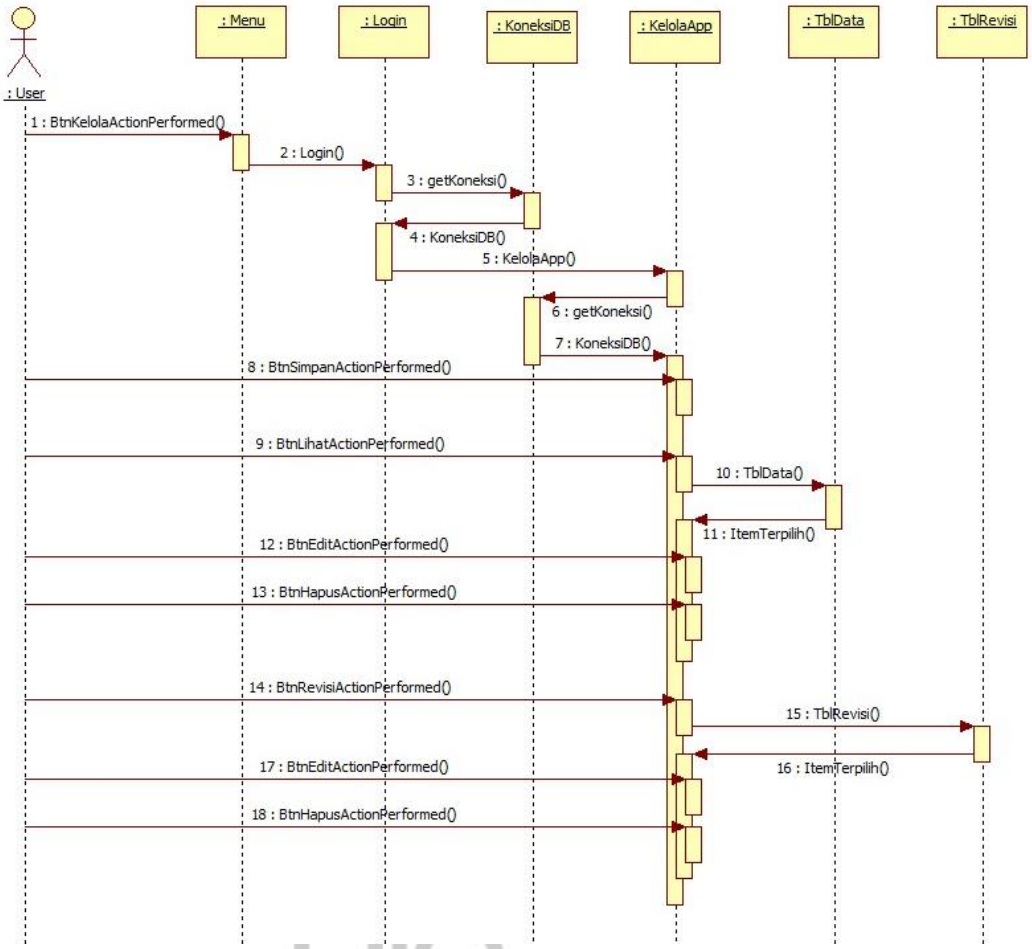
Nama Kelas	Daftar Tanggung-Jawab	Daftar Atribut
Menu.java	1. Menyediakan pilihan *button* Kelola Data. 2. Menyediakan pilihan b*utton* Pencari Jawaban.	-
Login.java	1. Menyediakan kolom input *User Name* dan *Password.* 2. Menyediakan pilihan b*utton Login.* 3. Menyediakan pemberitahuan apabila login 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

Nama Kelas	Daftar Tanggung-Jawab	Daftar Atribut
DocumentParser.java	1. Menyediakan fasilitas untuk membaca file yang akan diproses 2. Menyediakan proses untuk perhitungan bobot *TF-IDF* 3. Menyediakan proses untuk perhitungan bobot Similarity	1. termsDocsArray 2. allTerms 3. tfidfDocsVector 4. connection
CosineSimilarity.java	1. Menghitung bobot seluruh *term* dan dokumen menurut algoritma *Cosine Similarity.*	1. Skalar 2. jarak1 3. jarak2 4. cosineSimilarity
TfIdf.java	1. Menghitung nilai *TF* dari *term.* 2. Menghitung nilai *IDF* dari *term*	-
TblData.java	1. Menampilkan seluruh data yang terdapat pada database baik data valid maupun data yang belum valid tergantung dari status data. 2. Menyediakan fasilitas untuk memilih salah satu data untuk diberi aksi pada kelas KelolaApp.java.	1. Tabeldata 2. connection

3.2.6 *Sequence* Diagram

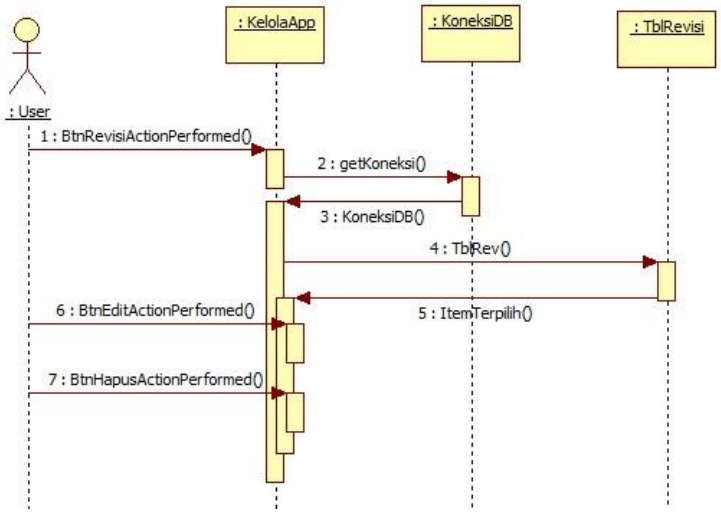
Sequence diagram digunakan untuk menjelaskan interaksi antar objek yang disusun dalam suatu urutan waktu. Diagram dengan khusus dapat berasosisasi dengan *use case* untuk memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu didalam *use case*. Oleh karena itu, dibuatlah lima *sequence* diagram pada penganalisisan sistem ini agar dapat menginterpretasikan masing-masing *use case.*

a. *Sequence* Diagram Mengelola Kasus



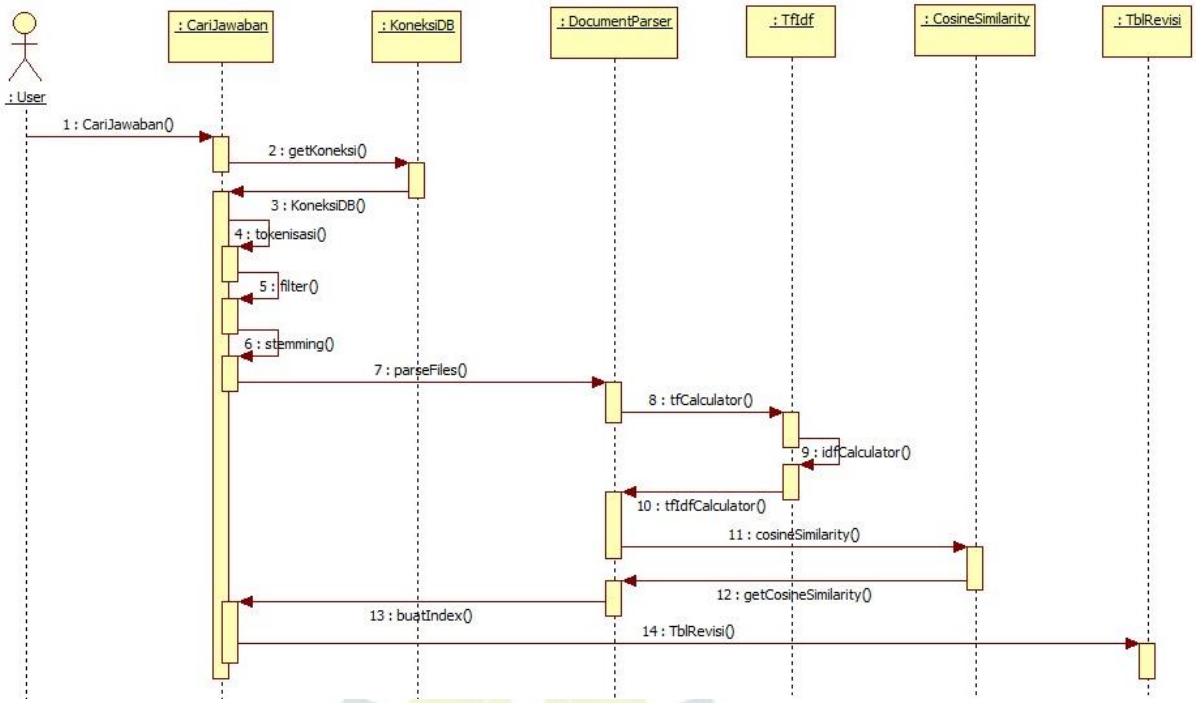
Gambar 3.16 *Sequence* Diagram Pengelolaan Kasus

b. *Sequence* Diagram Pengetahuan Baru



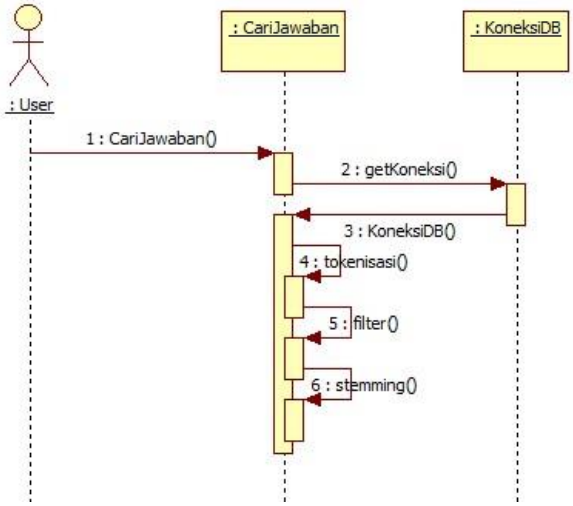
Gambar 3.17 *Sequence* Diagram Pengetahuan Baru

c. *Sequence* Diagram Pencari Kemiripan Kasus



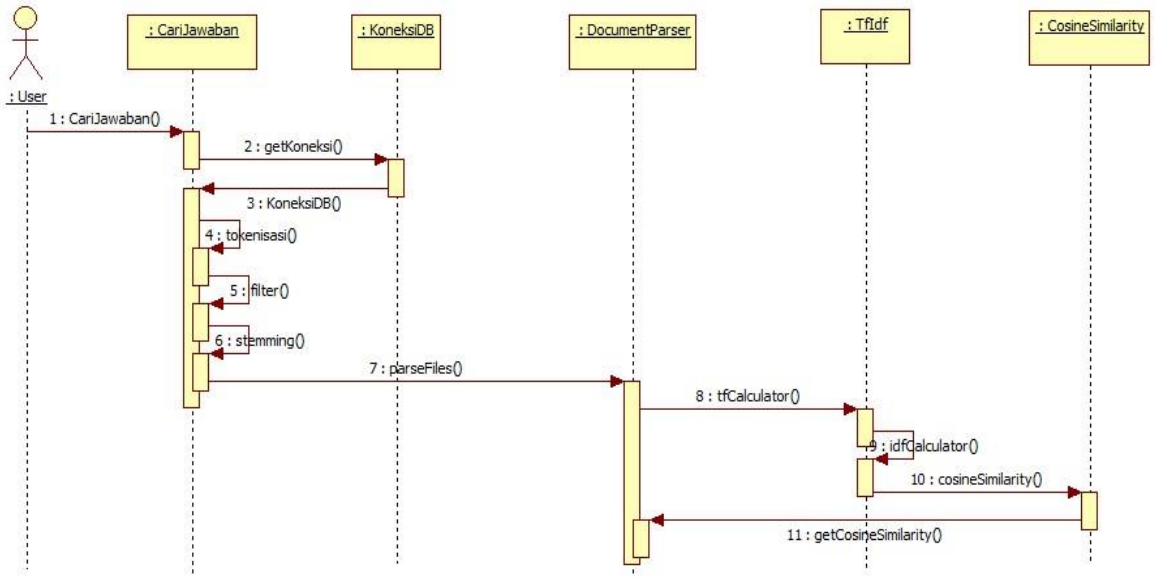
Gambar 3.18 *Sequence* Diagram Pencari Kemiripan Kasus

d. *Sequence* Diagram *Pre-processing*



Gambar 3.19 *Sequence* Diagram *Pre-processing*

e. *Sequence* Diagram Pembobotan



Gambar 3.20 *Sequence* Diagram Pembobotan

3.2.7 Perancangan *Database*

Berikut ini merupakan perancangan *database* dari aplikasi pencari kemiripan jawaban dari soal algoritma yang akan dibangun:

a. Tabel *tblogin*

Tabel *tblogin* berisi data pengelola aplikasi yang terdiri dari *username* dan *password*. Tabel ini digunakan untuk mengelola data admin. Dengan adanya fasilitas *login*, maka akan mengurangi pengelolaan pengetahuan yang tidak *valid*. Adapun struktur dari tabel *tblogin* adalah sebagai berikut:

Primary key : username

Foreign key : -

Tabel 3.19 Perancangan Tabel *tblogin*

Kolom	**Jenis**	**Atribut**	**Kosong**	***Default***
username	*varchar*(20)	-	Tidak	*None*
password	*varchar*(20)	-	Tidak	*None*

b. Tabel *appdb*

Tabel *appdb* berisi data pengetahuan yang sebelumnya dimasukkan oleh admin atau dari *appbaru* yang telah di-*validasi* oleh admin. Adapun struktur dari tabel *appdb* adalah sebagai berikut:

Primary key : autoid

Foreign key : -

Tabel 3.20 Perancangan Tabel *appdb*

Kolom	**Jenis**	**Atribut**	**Kosong**	***Default***
autoid	*int*(11)	-	Tidak	*None*
nama	*varchar*(200)	-	Ya	*NULL*
kelas	*varchar*(30)	-	Ya	*NULL*
konst	*varchar*(200)	-	Ya	*NULL*
tipe	*varchar*(200)	-	Ya	*NULL*
var	*varchar*(200)	-	Ya	*NULL*
kata	*varchar*(15000)	-	Ya	*NULL*
soal	*varchar*(15000)	-	Ya	*NULL*
foto	*longblob*	*binary*	Ya	*NULL*

c. Tabel *tb_katadasar*

Tabel *tb_katadasar* berisi data kata-kata dasar atau bisa dibilang kamus kata. Adapun struktur dari tabel *tb_katadasar* adalah sebagai berikut:

Primary key : Id_ktdasar

Foreign key : -

Tabel 3.21 Perancangan Tabel *tb_katadasar*

Kolom	**Jenis**	**Atribut**	**Kosong**	***Default***
Id_ktdasar	int(10)	-	Tidak	*None*
katadasar	*varchar*(20)	-	Tidak	*None*
Tipe_katadasar	*varchar*(20)	-	tidak	*None*

d. Tabel *appindex*

Tabel *appindex* berisi data pengetahuan yang disediakan untuk pemrosesan. Tabel ini akan selalu ter-*update* apabila pencari jawaban digunakan. Adapun struktur dari tabel *appindex* adalah sebagai berikut:

Primary key : autoid

Foreign key : appdb

Tabel 3.22 Perancangan Tabel *appindex*

Kolom	**Jenis**	**Atribut**	**Kosong**	***Default***
autoid	int(11)	-	Tidak	*None*
Kata	*varchar*(15000)	-	*Tidak*	*None*
SimDocAkhir	*double*	-	Ya	*NULL*

e. Tabel *tfidf*

Tabel *tfidf* berisi data hasil pemrosesan soal yang berisi jumlah *frekuensi* kata dari tiap dokumen. Adapun struktur dari tabel *tfidf* adalah sebagai berikut:

Primary key : autoid

Foreign key : appindex

Tabel 3.23 Perancangan Tabel *tfidf*

Kolom	**Jenis**	**Atribut**	**Kosong**	***Default***
autoid	*int*(11)	-	Tidak	*None*
Term	*varchar*(50)	-	Tidak	*None*
docId	*int*(11)	-	Tidak	*None*
jumlah	*int*(11)	-	Tidak	*None*
tfidf	*double*	-	Ya	*NULL*

f. Tabel *apptf*

Tabel *apptf* berisi data hasil pemrosesan soal yang berisi jumlah *frekuensi* kata dari tiap dokumen. Adapun struktur dari tabel *tfidf* adalah sebagai berikut:

Primary key : autoid

Foreign key : apptf

Tabel 3.24 Perancangan Tabel *apptf*

Kolom	**Jenis**	**Atribut**	**Kosong**	***Default***
autoid	*int*(11)	-	Tidak	*None*
Term	*varchar*(50)	-	Tidak	*None*
count	*int*(11)	-	Tidak	*None*

g. Tabel *appbaru*

Tabel *appbaru* berisi urutan data pengetahuan hasil pengurutan pada *appdb* dan *appindex* untuk menampilkan hasil kemiripan dan klasifikasi soal *input*-an. Adapun struktur dari tabel *appbaru* adalah sebagai berikut:

Primary key : autoid

Foreign key : appdb, appindex

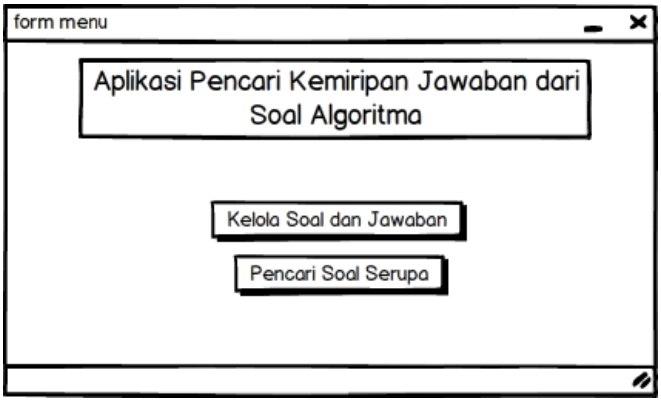
Tabel 3.25 Perancangan Tabel *appbaru*

Kolom	**Jenis**	**Atribut**	**Kosong**	***Default***
autoid	*int*(11)	-	Tidak	*None*
idlama	*int*(11)	-	Tidak	*None*
nama	*varchar*(200)	-	Ya	*NULL*
kelas	*varchar*(30)	-	Ya	*NULL*
konst	*varchar*(200)	-	Ya	*NULL*
tipe	*varchar*(200)	-	Ya	*NULL*
var	*varchar*(200)	-	Ya	*NULL*
kata	*varchar*(15000)	-	Ya	*NULL*
soal	*varchar*(15000)	-	Ya	*NULL*
foto	*longblob*	*binary*	Ya	*NULL*

3.2.8 Perancangan *Mock-Up*

a. *Mock-Up* Halaman Menu

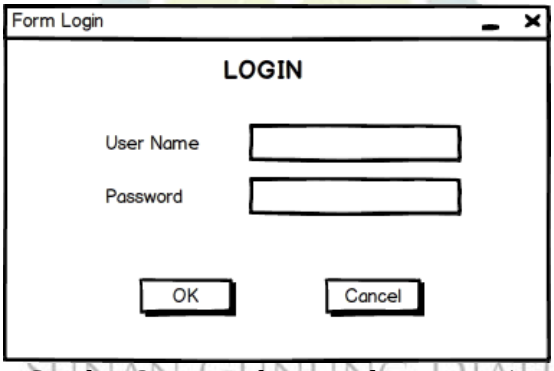
Halaman menu terdiri dari judul aplikasi dan dua tombol yang dapat diakses oleh *user.* Halaman ini merupakan halaman yang paling awal muncul ketika aplikasi dijalankan.



Gambar 3.21 *Mock-Up* Halaman Menu

b. *Mock-Up* Halaman *Login*

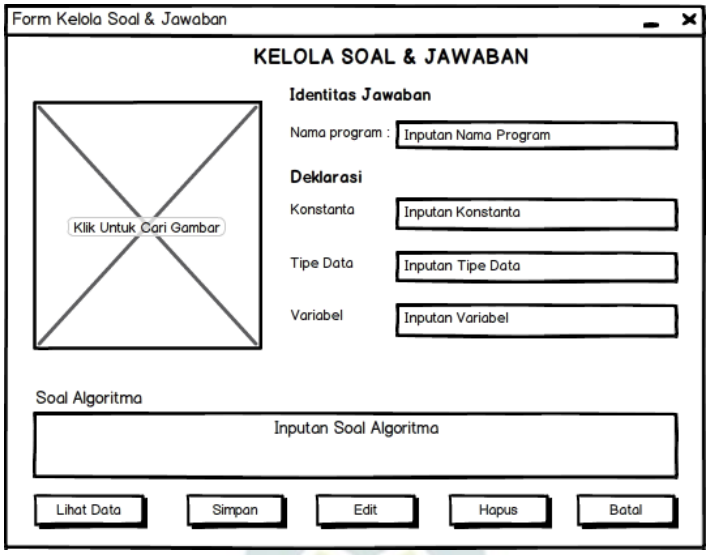
Halaman login ditampilkan ketika *user* memilih menu kelola soal dan jawaban pada halaman menu. Pada halaman ini *user* harus melakukan *login* agar dapat masuk ke halaman kelola soal dan jawaban.



Gambar 3.22 Mock-Up Halaman *Login*

c. *Mock-Up* Halaman Kelola

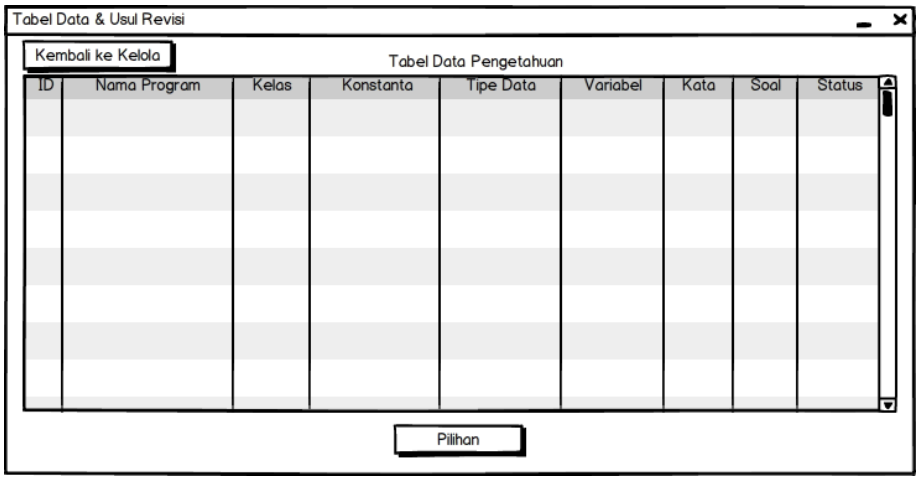
Halaman kelola dapat diakses apabila login berhasil. Pada halaman ini Admin dapat melakukan aktifitas pengelolaan terhadap data pengetahuan.



Gambar 3.23 *Mock-Up* Halaman Kelola

d. *Mock-Up* Tabel Data

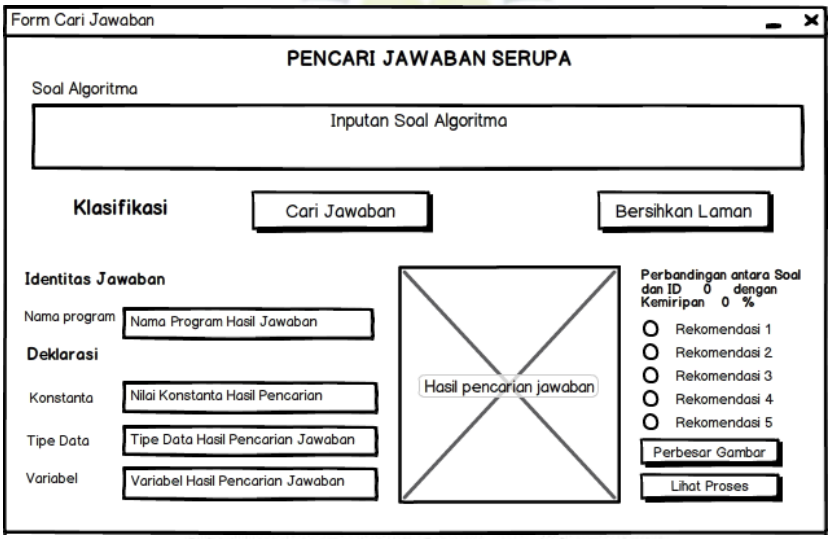
Halaman ini akan tampil apabila Admin menekan tombol "lihat data". Halaman ini menyediakan semua data pengetahuan lama dan pengetahuan baru yang akan direvisi oleh Admin. Pada halaman ini Admin dapat memilih salah satu data pengetahuan untuk diberi tindakan dan menekan tombol "pilihan" agar data yang dipilih ditransfer ke halaman kelola.



Gambar 3.24 *Mock-Up* Tabel Data

e. *Mock-Up* Halaman Pencari Jawaban Serupa

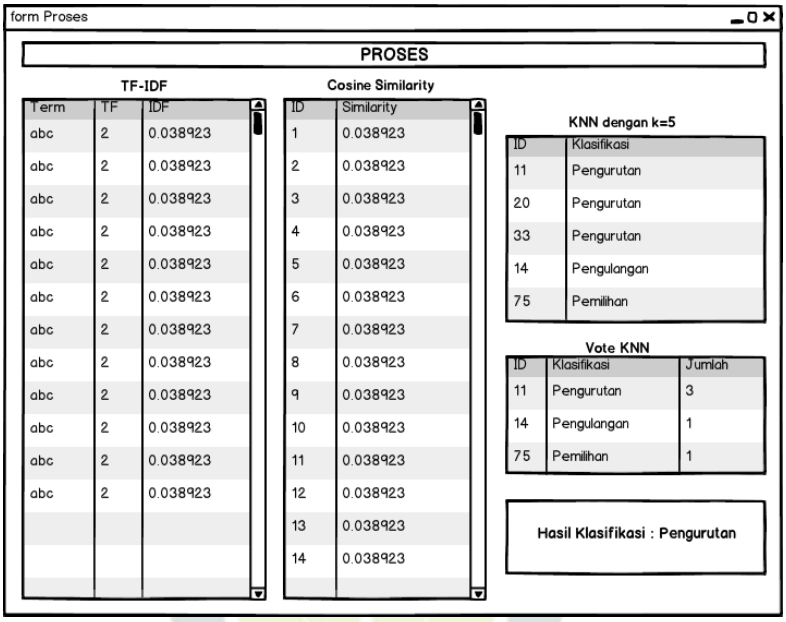
Halaman ini tampil ketika *user* menekan tombol pencari jawaban serupa. Disini pengguna hanya tinggal mengisi kolom dan menekan tombol cari jawaban saja ketika ingin memulai proses pencarian. Hasil pencarian yang berupa identitas jawaban akan disimpan pada kolom-kolom kosong yang tersedia, sedangkan potongan kode program akan disimpan pada kolom gambar. Pada bagian kanan bawah juga tersedia hasil perhitungan pencarian dan 5 rekomendasi jawaban berbeda sesuai urutan *rating* kemiripannya.



Gambar 3.25 *Mock-Up* Halaman Pencari Jawaban Serupa

f. *Mock-Up* Halaman Proses

Halaman ini tampil ketika *user* menekan tombol lihat proses pada halaman pencari jawaban serupa. Pada halaman ini terdapat beberapa tabel hasil dari pemrosesan *TF-IDF*, *Cosine Similarity* dan cara pengklasifikasian *KNN*. Berikut ini adalah perancangan tampilan untuk halaman proses:



Gambar 3.26 Perancangan Halaman Proses

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Implementasi bertujuan untuk mengetahui bagaimana penerapan dari aplikasi pencari kemiripan jawaban dari soal-soal algoritma. Pada pengimplementasian terbagi menjadi beberapa tahapan diantaranya implementasi perangkat lunak, implementasi perangkat keras, implementasi *database*, implementasi algoritma, dan implementasi antarmuka.

4.2 Implementasi Perangkat Keras

Implementasi perangkat keras yang digunakan untuk membuat aplikasi menggunakan komputer dengan spesifikasi minimum sebagai berikut:

a. *Intel® Atom™* Inside™ CPU @ 2.40GHz 2.40GHz

b. RAM 2 GB (1.89 GB *usable*)

c. *Harddisk* 464.3 GB

d. Sistem operasi *Windows* 7 Pro

4.3 Implementasi Perangkat Lunak

Implementasi perangkat lunak yang digunakan untuk pemrograman aplikasi menggunakan beberapa *software* lain sebagai berikut:

a. *Java Development Kit* versi 1.7

b. *Netbeans 7.2.1*

c. *Xampp*

d. *Mozilla Firefox*

4.4 Implementasi *Database*

Berikut ini merupakan implementasi *database* dari aplikasi pencari kemiripan jawaban dari soal algoritma yang dibangun:

4.4.1 Tabel *tblogin*

Implementasi dari tabel *tblogin* berisi data pengelola aplikasi yang terdiri dari *username* dan *password*. Tabel ini digunakan untuk mengelola data admin. Dengan adanya fasilitas *tblogin*, maka akan mengurangi pengelolaan pengetahuan yang tidak *valid*. Adapun implementasi dari tabel *tblogin* adalah sebagai berikut:

#	Kolom	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra
1	**username**	varchar(20)	latin1_swedish_ci		Tidak	*None*	
2	**password**	varchar(20)	latin1_swedish_ci		Tidak	*None*	

Gambar 4.27 Implementasi Tabel *tblogin*

4.4.2 Tabel *appdb*

Implementasi dari tabel *appdb* berisi data pengetahuan yang sebelumnya dimasukkan oleh admin atau dari *appbaru* yang telah di-*validasi* oleh admin. Adapun implementasi dari tabel *appdb* adalah sebagai berikut:

#	Kolom	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra
1	**autoid**	int(11)			Tidak	*None*	AUTO_INCREMENT
2	**nama**	varchar(200)	latin1_swedish_ci		Ya	*NULL*	
3	**kelas**	varchar(30)	latin1_swedish_ci		Ya	*NULL*	
4	**konst**	varchar(200)	latin1_swedish_ci		Ya	*NULL*	
5	**tipe**	varchar(200)	latin1_swedish_ci		Ya	*NULL*	
6	**var**	varchar(200)	latin1_swedish_ci		Ya	*NULL*	
7	**kata**	varchar(15000)	latin1_swedish_ci		Ya	*NULL*	
8	**soal**	varchar(15000)	latin1_swedish_ci		Ya	*NULL*	
9	**foto**	blob		BINARY	Ya	*NULL*	

Gambar 4.28 Implementasi Tabel *appdb*

4.4.3 Tabel *tb_katadasar*

Implementasi dari tabel *tb_katadasar* berisi data kata-kata dasar atau bisa dibilang kamus kata. Adapun implementasi dari tabel *tb_katadasar* adalah sebagai berikut:

#	Kolom	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra
1	id_ktdasar	int(10)			Tidak	None	AUTO_INCREMENT
2	katadasar	varchar(20)	latin1_swedish_ci		Tidak	None	
3	tipe_katadasar	varchar(20)	latin1_swedish_ci		Tidak	None	

Gambar 4.29 Implementasi Tabel *tb_katadasar*

4.4.4 Tabel appindex

Implementasi dari tabel *appindex* berisi data pengetahuan yang disediakan untuk pemrosesan. Tabel ini akan selalu ter-*update* apabila pencari jawaban digunakan. Adapun implementasi dari tabel *appindex* adalah sebagai berikut:

#	Kolom	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra
1	autoid	int(11)			Tidak	None	AUTO_INCREMENT
2	Kata	varchar(15000)	latin1_swedish_ci		Tidak	None	
3	SimDocAkhir	double			Ya	NULL	

Gambar 4.30 Implementasi Tabel *appindex*

4.4.5 Tabel *apptfidf*

Implementasi dari tabel *apptfidf* berisi data hasil pemrosesan soal yang berisi jumlah *frekuensi* kata dari tiap dokumen. Adapun implementasi dari tabel *apptfidf* adalah sebagai berikut:

#	Kolom	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra
1	autoid	int(11)			Tidak	None	AUTO_INCREMENT
2	Term	varchar(50)	latin1_swedish_ci		Tidak	None	
3	docId	int(11)			Tidak	None	

Gambar 4.31 Implementasi Tabel *apptfidf*

4.4.6 Tabel *apptf*

Implementasi dari tabel *apptf* berisi data hasil pemrosesan soal yang berisi jumlah *frekuensi* kata dari seluruh dokumen. Adapun implementasi dari tabel *apptf* adalah sebagai berikut:

#	Kolom	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra
1	autoid	int(11)			Tidak	None	AUTO_INCREMENT
2	Term	varchar(50)	latin1_swedish_ci		Tidak	None	
3	docId	int(11)			Tidak	None	

Gambar 4.32 Implementasi Tabel *tfidf*

4.4.7 Tabel *appbaru*

Implementasi dari tabel *appbaru* berisi urutan data pengetahuan hasil pengurutan pada *appdb* dan *appindex* untuk menampilkan hasil kemiripan dan klasifikasi soal *input*-an.

#	Kolom	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra
1	autoid	int(11)			Tidak	None	AUTO_INCREMENT
2	idlama	int(11)			Ya	NULL	
3	nama	varchar(200)	latin1_swedish_ci		Ya	NULL	
4	kelas	varchar(30)	latin1_swedish_ci		Tidak	None	
5	konst	varchar(200)	latin1_swedish_ci		Ya	NULL	
6	tipe	varchar(200)	latin1_swedish_ci		Ya	NULL	
7	var	varchar(200)	latin1_swedish_ci		Ya	NULL	
8	foto	longblob		BINARY	Ya	NULL	
9	SimDocAkhir	double			Ya	NULL	

Gambar 4.33 Implementasi Tabel *appbaru*

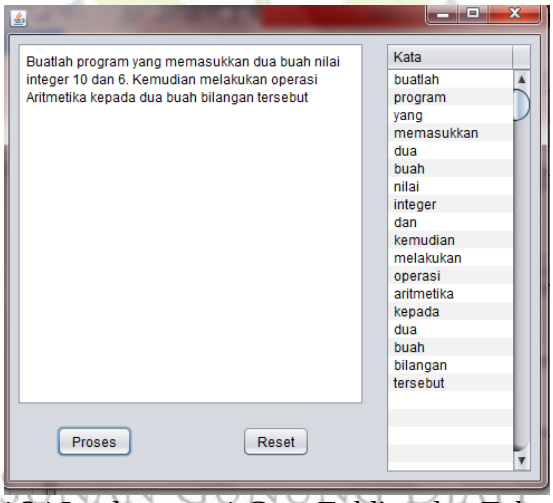
4.5 Implementasi Metode dan Algoritma

Algoritma yang dipakai pada pengimplementasiaan aplikasi menggunakan beberapa algoritma diantaranya *Case Folding*, *Tokenizing*, *Filtering* dengan *Stopword*, *Stemming* dengan algoritma *Porter*, pembobotan dengan *Tf-Idf*, pencari kemiripan soal dengan algoritma *Cosine Similarity*, pengklasifikasian dengan algoritma *KNN* dan algoritma *CBR*. Pada algoritma *CBR* tidak ada potongan program karena algoritma *CBR* diterapkan sebagai alur dari jalannya program.

Berikut ini merupakan implementasi dari algoritma yang diterapkan ketika dijalankan, sedangkan untuk potongan kode program algoritma disertakan pada lampiran.

4.5.1 *Case Folding* dan *Tokenizing*

Implementasi *case folding* dan *tokenizing* sebenarnya tidak ditampilkan pada aplikasi dan hanya diproses didalam pengolahan saja. Sebagai pembuktian pengimplementasi *case folding* dan *tokenizing* berjalan dengan baik, maka dibuatlah contoh program lain diluar aplikasi dengan menggunakan kode program yang sama sehingga menampilkan hasil sebagai berikut:



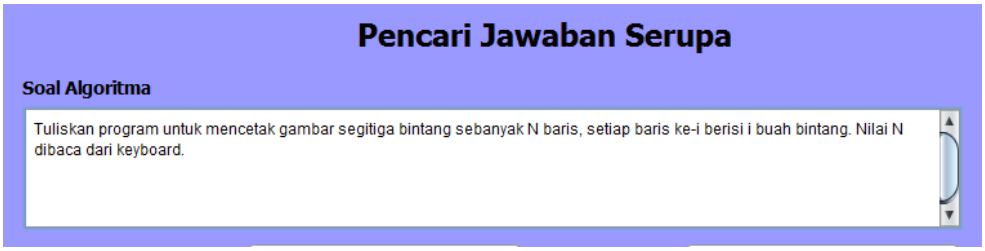
Gambar 4.34 Implementasi *Case Folding* dan *Tokenizing*

4.5.2 Algoritma *Stopword Removal* dan Algoritma *Porter Stemmer*

Implementasi dari algoritma *Stopword Removal* dan *Porter Stemmer* tidak ditampilkan pada aplikasi secara langsung, namun hasilnya disimpan langsung kedalam *database.* Berikut ini hasil pemrosesan algoritma *Stopword Removal* dan *Porter Stemmer* pada *database* yang diambil dari tabel *appindex* baris terakhir atau merupakan hasil *pre-processing* soal untuk mencari jawaban.

Gambar 4.35 Implementasi *Stopword* dan *Porter Stemmer* pada *Database*

Proses dari soal input pada program:



Gambar 4.36 Implementasi *Stopword* dan *Porter Stemmer* Soal Input

4.5.3 Metode *Tf-Idf*

Term	TF	IDF
abc	2	3.806662489770...
absolute	1	4.499809670330...
ada	2	3.806662489770...
agustus	2	3.806662489770...
air	2	4.499809670330...
akhir	6	2.890371757896...
algoritma	53	0.567984037605...
aman	1	4.499809670330...
ambah	3	3.806662489770...
ambil	1	4.499809670330...
anak	2	4.499809670330...
analisis	1	4.499809670330...
andai	3	3.806662489770...
anggap	1	4.499809670330...
aritmetika	2	3.806662489770...
arsip	1	4.499809670330...
arti	1	4.499809670330...
asosiasi	1	4.499809670330...
asumsi	3	3.401197381662...
atas	1	4.499809670330...
atm	1	4.499809670330...
awal	4	3.401197381662...
awar	1	4.499809670330...
ayam	2	4.499809670330...
baca	66	0.607989372219...
bagi	4	3.113515309210...

Gambar 4.37 Implementasi *TF-IDF* Pada Halaman Proses

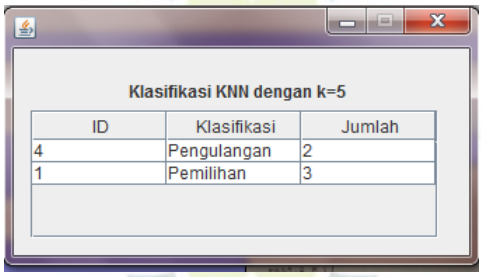
4.5.4 *Cosine Similarity*

ID	Similarity
1	0.003713526969 10...
2	0.002422296764 92...
3	0.012176955368 89...
4	0.261229921002 53...
5	0.032526705519 30...
6	0.003479393712 87...
7	0.001477282851 61...
8	0.108029951040 60...
9	0.015813236420 07...
10	0
11	0.183267026680 37...
12	0.147169735781559
13	0.666406717977 37...
14	0.033679995802 19...
15	0.009813200184 39...
16	0.116412921915 14...
17	0.018536079768 16...
18	0.029506430142 22...
19	0.012810831216 79...
20	0.007446871918 09...
21	0.002588141219 23...
22	0.005799796683 50...
23	0.006366362257 92...
24	0.002984951254 93...
25	0.124406718761 79...
26	0.007374365954 25...

Gambar 4.38 Implementasi *Cosine Similarity* Pada Halaman Proses

4.5.5 Algoritma *KNN*

Implementasi *KNN* didapatkan dari hasil pengurutan nilai kemiripan *cosine similarity* dengan nilai k=5 dan mengklasifikasikan berdasarkan *vote* kategori terbayak, apabila ternyata jumlah *vote* kategori berimbang 2:2:1 maka yang diambil adalah kategori yang memiliki kemiripan *cosine similarity* tertinggi dari jumlah dokumen yang berimbang tersebut. Contoh pengimplementasian dari *running* program dibawah ini akan menghasilkan klasifikasi runtunan karena *vote* dari kelima kategori seperti pada gambar berikut:



Gambar 4.39 Implementasi *KNN* Pada Halaman Proses

4.6 Implementasi Antar Muka

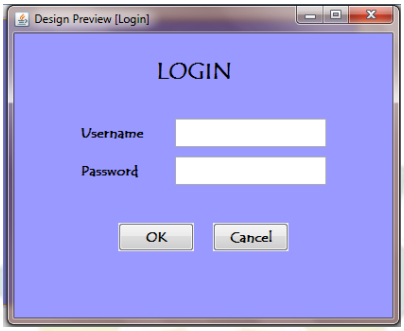
4.6.1 Halaman Menu

Halaman menu adalah halaman yang pertama kali muncul ketika aplikasi dijalankan. Halaman menu menyediakan 2 menu pilihan seperti pada gambar berikut:

Gambar 4.40 Implementasi Halaman Menu

4.6.2 Halaman *Login*

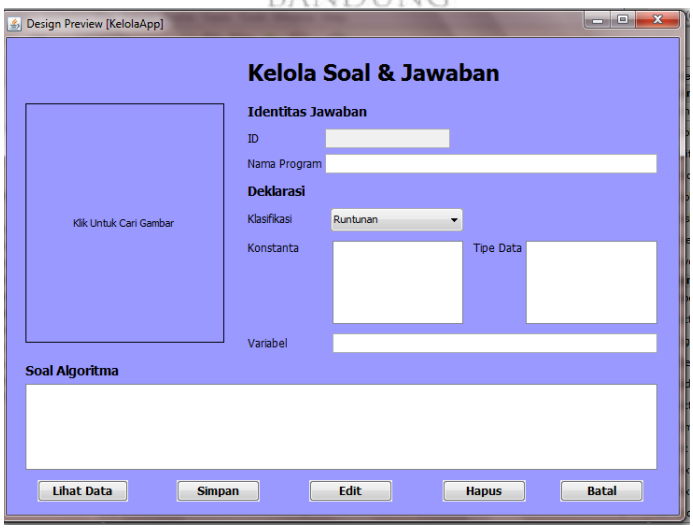
Halaman login ditampilkan ketika *user* memilih menu kelola soal dan jawaban pada halaman menu. Pada halaman ini *user* harus melakukan login agar dapat masuk ke halaman kelola soal dan jawaban. Halaman login menyediakan kolom pengisian untuk login dan 2 tombol pilihan untuk diberikan aksi seperti pada gambar berikut:



Gambar 4.41 Implementasi Halaman Login Admin

4.6.3 Halaman Kelola Aplikasi

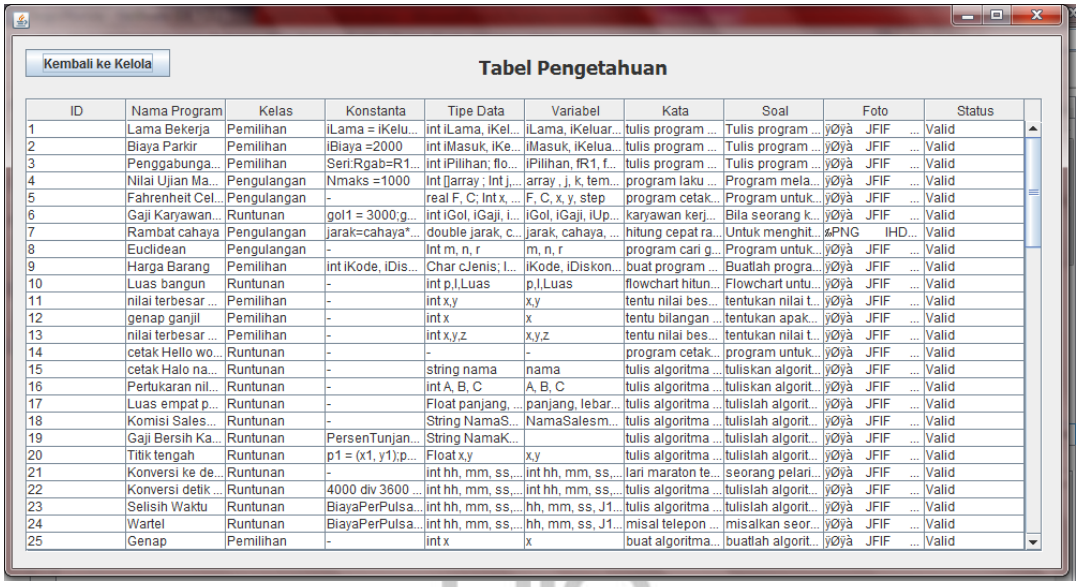
Halaman kelola dapat diakses apabila *login* berhasil. Pada halaman ini Admin dapat melakukan aktifitas pengelolaan terhadap data pengetahuan. Halaman kelola aplikasi menyediakan kolom pengisian data dan 5 tombol pilihan untuk diberikan aksi seperti pada gambar berikut:



Gambar 4.42 Implementasi Halaman Kelola Aplikasi

4.6.4 Halaman Tabel Data

Halaman ini akan tampil apabila Admin menekan *double* klik *row*. Halaman ini menyediakan semua data pengetahuan lama dan pengetahuan baru yang akan direvisi oleh Admin. Pada halaman ini Admin dapat memilih salah satu data pengetahuan untuk diberi tindakan agar data yang dipilih ditransfer ke halaman kelola. Berikut ini merupakan untuk halaman tabel data:



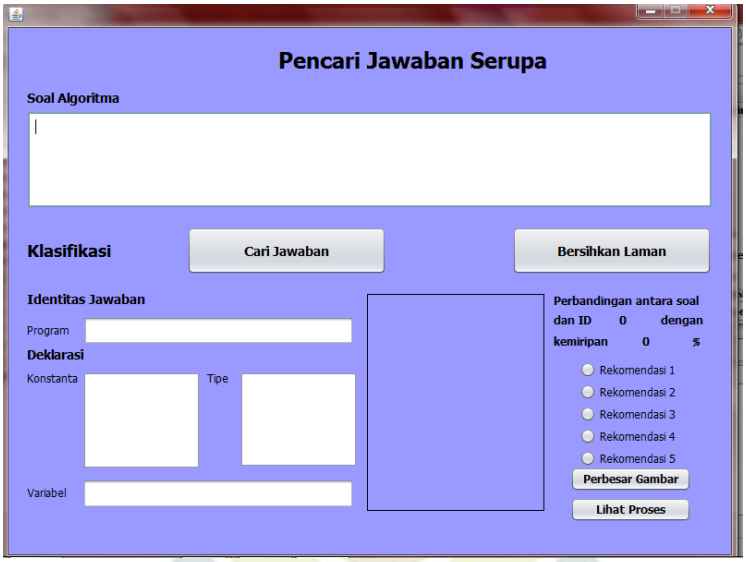
ID	Nama Program	Kelas	Konstanta	Tipe Data	Variabel	Kata	Soal	Foto	Status
1	Lama Bekerja	Pemilihan	iLama = iKelu...	int iLama, iKel...	iLama, iKeluar...	tulis program ...	Tulis program ...	ÿØÿà JFIF ...	Valid
2	Biaya Parkir	Pemilihan	iBiaya =2000	int iMasuk, iKe...	iMasuk, iKelua...	tulis program ...	Tulis program ...	ÿØÿà JFIF ...	Valid
3	Penggabunga...	Pemilihan	Seri:Rgab=R1...	int iPilihan; flo...	iPilihan, fR1, f...	tulis program ...	Tulis program ...	ÿØÿà JFIF ...	Valid
4	Nilai Ujian Ma...	Pengulangan	Nmaks =1000	Int []array ; Int j,...	array , j, k, tem...	program laku ...	Program mela...	ÿØÿà JFIF ...	Valid
5	Fahrenheit Cel...	Pengulangan	-	real F, C; Int x, ...	F, C, x, y, step	program cetak...	Program untuk...	ÿØÿà JFIF ...	Valid
6	Gaji Karyawan...	Runtunan	gol1 = 3000;g...	int iGol, iGaji, i...	iGol, iGaji, iUp...	karyawan kerj...	Bila seorang k...	ÿØÿà JFIF ...	Valid
7	Rambat cahaya	Pengulangan	jarak=cahaya*...	double jarak, c...	jarak, cahaya, ...	hitung cepat ra...	Untuk menghit...	‰PNG IHD...	Valid
8	Euclidean	Pengulangan	-	Int m, n, r	m, n, r	program cari g...	Program untuk...	ÿØÿà JFIF ...	Valid
9	Harga Barang	Pemilihan	int iKode, iDis...	Char cJenis; I...	iKode, iDiskon...	buat program ...	Buatlah progra...	ÿØÿà JFIF ...	Valid
10	Luas bangun	Runtunan	-	int p,l,Luas	p,l,Luas	flowchart hitun...	Flowchart untu...	ÿØÿà JFIF ...	Valid
11	nilai terbesar ...	Pemilihan	-	int x,y	x,y	tentu nilai bes...	tentukan nilai t...	ÿØÿà JFIF ...	Valid
12	genap ganjil	Pemilihan	-	int x	x	tentu bilangan ...	tentukan apak...	ÿØÿà JFIF ...	Valid
13	nilai terbesar ...	Pemilihan	-	int x,y,z	x,y,z	tentu nilai bes...	tentukan nilai t...	ÿØÿà JFIF ...	Valid
14	cetak Hello wo...	Runtunan	-	-	-	program cetak...	program untuk...	ÿØÿà JFIF ...	Valid
15	cetak Halo na...	Runtunan	-	string nama	nama	tulis algoritma ...	tuliskan algorit...	ÿØÿà JFIF ...	Valid
16	Pertukaran nil...	Runtunan	-	int A, B, C	A, B, C	tulis algoritma ...	tuliskan algorit...	ÿØÿà JFIF ...	Valid
17	Luas empat p...	Runtunan	-	Float panjang, ...	panjang, lebar...	tulis algoritma ...	tulislah algorit...	ÿØÿà JFIF ...	Valid
18	Komisi Sales...	Runtunan	-	String NamaS...	NamaSalesm...	tulis algoritma ...	tulislah algorit...	ÿØÿà JFIF ...	Valid
19	Gaji Bersih Ka...	Runtunan	PersenTunjan...	String NamaK...		tulis algoritma ...	tulislah algorit...	ÿØÿà JFIF ...	Valid
20	Titik tengah	Runtunan	p1 = (x1, y1);p...	Float x,y	x,y	tulis algoritma ...	tulislah algorit...	ÿØÿà JFIF ...	Valid
21	Konversi ke de...	Runtunan	-	int hh, mm, ss,...	int hh, mm, ss,...	lari maraton te...	seorang pelari...	ÿØÿà JFIF ...	Valid
22	Konversi detik ...	Runtunan	4000 div 3600 ...	int hh, mm, ss,...	int hh, mm, ss,...	tulis algoritma ...	tulislah algorit...	ÿØÿà JFIF ...	Valid
23	Selisih Waktu	Runtunan	BiayaPerPulsa...	int hh, mm, ss,...	hh, mm, ss, J1...	tulis algoritma ...	tulislah algorit...	ÿØÿà JFIF ...	Valid
24	Wartel	Runtunan	BiayaPerPulsa...	int hh, mm, ss,...	hh, mm, ss, J1...	misal telepon ...	misalkan seor...	ÿØÿà JFIF ...	Valid
25	Genap	Pemilihan	-	int x	x	buat algoritma...	buatlah algorit...	ÿØÿà JFIF ...	Valid

Gambar 4.43 Implementasi Halaman Tabel Data

4.6.5 Halaman Pencari Jawaban

Halaman ini tampil ketika *user* menekan tombol pencari jawaban serupa. Disini pengguna hanya tinggal mengisi kolom dan menekan tombol cari jawaban saja ketika ingin memulai proses pencarian. Hasil pencarian yang berupa identitas jawaban akan disimpan pada kolom-kolom kosong yang tersedia, sedangkan potongan kode program akan disimpan pada kolom gambar. Pada bagian kanan bawah juga tersedia hasil perhitungan pencarian dan 5 rekomendasi jawaban berbeda sesuai tingkat *rating* kemiripannya yang dapat dilihat oleh pengguna. Halaman pencari jawaban serupa menyediakan kolom untuk inputan soal, 3

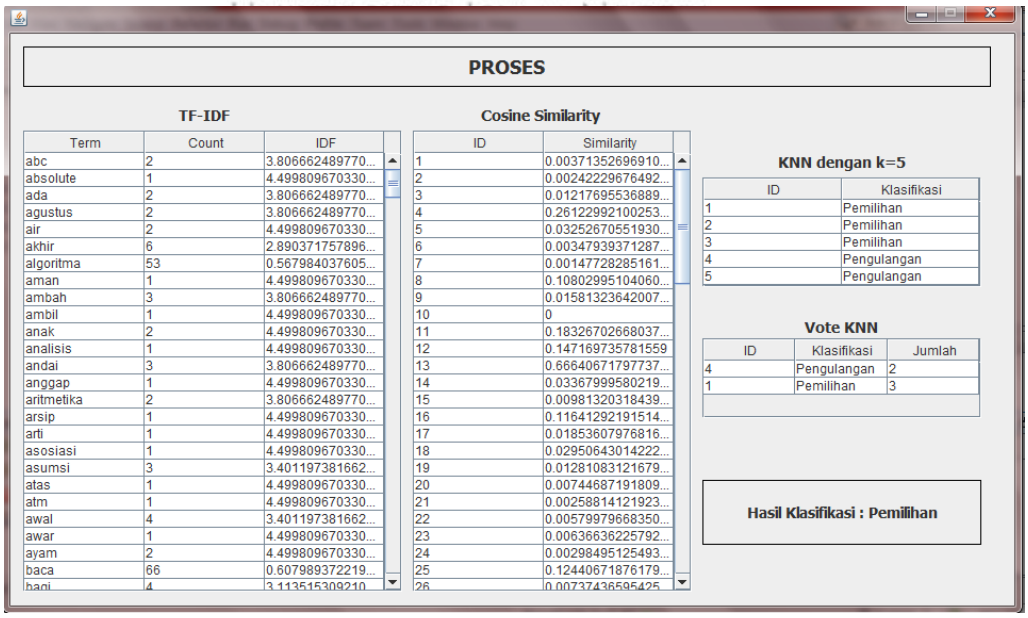
tombol pilihan untuk diberikan aksi dan beberapa kolom untuk hasil pencarian, seperti pada gambar berikut:



Gambar 4.44 Implementasi Halaman Pencari Jawaban

4.6.6 Halaman Proses

Halaman ini menampilkan beberapa proses yang dilakukan aplikasi dalam mencari kemiripan dan pengklasifikasian soal input pengguna. Berikut ini adalah tampilan pada halaman proses:



Gambar 4.45 Implementasi Halaman Proses

4.7 Pengujian Sistem

Pengujian sistem dilakukan dengan pengujian *Black box* dan berdasarkan *use case* yang dirancang sebelumnya. Hal tersebut dapat dilihat pada tabel berikut:

Tabel 4.26 Pengujian Sistem

No.	*Use Case*	Relasi yang diharapkan	Hasil Pengujian	Kesimpulan	
				Diterima	Ditolak
1.	Mengelola kasus	Mengelola kasus dengan menerapkan aksi *CRUD* dan melakukan *pre-processing* pada tiap dokumen.	Sistem dapat mengelola kasus dengan menerapkan aksi *CRUD* dan melakukan *pre-processing* pada tiap dokumen.	√	
2.	Membuat Pengetahuan Baru	Membuat pengetahuan baru sesuai hasil pencarian jawaban yang siap di-*validasi* oleh admin.	Sistem dapat membuat pengetahuan baru sesuai dengan hasil pencarian jawaban yang siap di-*validasi* oleh admin.	√	
3.	Mencari Kemiripan Kasus	Mencari kemiripan dengan mengurutkan berdasarkan tingkat yang paling mirip serta mengklasifikasikan kedalam struktur kategori dari algoritma.	Sistem dapat mencari kemiripan soal dengan mengurutkan berdasarkan tinggkat yang paling mirip serta melakukan klasifikasi untuk struktur kategori dari algoritma.	√	
4.	Membuat *Pre-processing*	Mengolah dari pencarian atau inputan pengelolaan menjadi kata dasar dalam bentuk array.	Sistem dapat mengolah dari pencarian atau inputan pengelolaan menjadi kata dasar dalam bentuk array.	√	
5.	Menghitung bobot	Menghitung bobot dokumen untuk diurutkan dan diklasifikasikan.	Sistem dapat menghitung bobot dokumen untuk diurutkan dan diklasifikasikan.	√	

4.8 Pengujian Dokumen

Pengujian yang dilakukan merupakan pengklasifikasian kategori dengan nilai $k = 5$, $k = 7$ dan $k = 9$. Untuk mengetahui hasil pengelohan pencarian kemiripan jawaban menggunakan 10 soal uji yang dibandingkan dengan 90 data pengetahuan kasus lama yang berstatus *valid* pada *database* dan terdiri dari 3 kategori yaitu runtunan, pemilihan serta pengulangan yang memiliki 30 dokumen disetiap kategori. Untuk pengujian tersebut dapat dilihat pada tabel berikut:

Tabel 4.27 Pengujian *k*=5

No	Nama Program	Uji Dok Input	Sistem	Hasil
1.	Uji Data Mahasiswa	Pengulangan	Pengulangan	Sesuai
2.	Konversi Waktu	Runtunan	Runtunan	Sesuai
3.	Luas Bangun Geometri	Runtunan	Runtunan	Sesuai
4.	Jarak Tanggal	Runtunan	Pemilihan	Tidak Sesuai
5.	Bilangan Bulat Positif	Pemilihan	Pemilihan	Sesuai
6.	Tiga Buah Bilangan Bulat	Pemilihan	Pemilihan	Sesuai
7.	Jumlah N ganjil Pertama	Pemilihan	Pemilihan	Sesuai
8.	Segitiga Bintang	Pengulangan	Pengulangan	Sesuai
9.	Membaca Sembarang Karakter	Pengulangan	Pengulangan	Sesuai
10.	Konversi Bilangan Bulat Positif	Pengulangan	Pengulangan	Sesuai

Tabel 4.28 Pengujian k=7

No	Nama Program	Uji Dok Input	Sistem	Hasil
1.	Uji Data Mahasiswa	Pengulangan	Pengulangan	Sesuai
2.	Konversi Waktu	Runtunan	Runtunan	Sesuai
3.	Luas Bangun Geometri	Runtunan	Runtunan	Sesuai
4.	Jarak Tanggal	Runtunan	Pemilihan	Tidak Sesuai
5.	Bilangan Bulat Positif	Pemilihan	Pengulangan	Tidak Sesuai
6.	Tiga Buah Bilangan Bulat	Pemilihan	Pemilihan	Sesuai
7.	Jumlah N ganjil Pertama	Pemilihan	Pengulangan	Tidak Sesuai
8.	Segitiga Bintang	Pengulangan	Pengulangan	Sesuai
9.	Membaca Sembarang Karakter	Pengulangan	Pengulangan	Sesuai
10.	Konversi Bilangan Bulat Positif	Pengulangan	Pengulangan	Sesuai

Tabel 4.29 Pengujian k=9

No	Nama Program	Uji Dok Input	Sistem	Hasil
1.	Uji Data Mahasiswa	Pengulangan	Pengulangan	Sesuai
2.	Konversi Waktu	Runtunan	Runtunan	Sesuai
3.	Luas Bangun Geometri	Runtunan	Runtunan	Sesuai
4.	Jarak Tanggal	Runtunan	Pemilihan	Tidak Sesuai
5.	Bilangan Bulat Positif	Pemilihan	Pemilihan	Sesuai
6.	Tiga Buah Bilangan Bulat	Pemilihan	Pemilihan	Sesuai
7.	Jumlah N ganjil Pertama	Pemilihan	Pengulangan	Tidak Sesuai
8.	Segitiga Bintang	Pengulangan	Pengulangan	Sesuai
9.	Membaca Sembarang Karakter	Pengulangan	Pengulangan	Sesuai
10.	Konversi Bilangan Bulat Positif	Pengulangan	Pengulangan	Sesuai

Pengujian dilakukan untuk mengetahui ketepatan dan keakuratan pengklasifikasian dokumen, yaitu dengan menggunakan pengujian nilai akurasi. Berdasarkan perbandingan dengan nilai *k* diatas, maka implementasi dari pengklasifikasian struktur dasar algoritma dapat bekerja dengan baik ketika menggunakan metode *KNN* dengan nilai *k* = 5 yang memiliki nilai akurasi sebagai berikut:

$$Akurasi = \frac{\sum dokumen\ relevan}{\sum dokumen\ total} = \frac{9}{10} = 0{,}9$$

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian yang dilakukan pada aplikasi pencari kemiripan soal dari jawaban algoritma dengan menggunakan algoritma *K-Nearest Neighbor* (*KNN*), disimpulkan bahwa:

1. Metode *CBR* dapat dianggap cocok ketika dikolaborasikan dengan algoritma *KNN*, karena dari hasil jawaban yang telah dicarikan kemiripannya yang dihitung dengan metode *TF-IDF* dan *Cosine Similarity* kemudian akan dipertimbangkan kembali oleh manusia sehingga jawaban tersebut dapat dikatakan sesuai. Selain itu kajian metode *CBR* pada umumnya hampir sama dengan algoritma *KNN* yaitu mengambil kasus berdasarkan pengetahuan. Semakin banyak pengetahuan kasus, maka semakin baik dalam pengklasifikasian.
2. Algoritma *KNN* dapat bekerja dengan baik ketika menggunakan nilai $k = 5$ untuk mengklasifikasikan struktur dasar algoritma dari pencari kemiripan soal. Karena, dari hasil nilai akurasi dengan *KNN* menggunakan $k = 5$ adalah 0,9 dari pengujian dengan menggunakan 10 soal algoritma dan data *training* sebanyak 90 dokumen dengan 3 kategori yaitu runtunan, pemilihan dan pengulangan yang masing-masing memiliki 30 data disetiap kategori, didapat hasil pengklasifikasian dengan benar. Namun dalam pengklasifikasi sangat bergantung pada token dan pengetahuan sehingga mempengaruhi hasil perhitungan dari *Cosine Similarity.*

5.2 Saran

Beberapa saran yang dipertimbangkan untuk pengembangan selanjutnya pada aplikasi pencari jawaban dari soal algoritma yaitu:

1. Aplikasi dapat diakses selain pada *desktop.*
2. Gunakan algoritma lain seperti *SVM, Decision Tree*, dan *Naive bayes* dan sebagainya.
3. Tambahkan kata dasar yang sekiranya dianggap perlu untuk kamus kata.
4. Temukan cara agar waktu pemrosesan lebih cepat ketika mencari kemiripan jawaban.

DAFTAR PUSTAKA

[1] Gerhana YA, Djohar A. Case-based Reasoning Learning Model to Develop Skill in Problem Solving of Student of Vocational Education. International Journal of Basic and Applied Science. 2016 April; 04(11).

[2] Luthfi ET. Penerapan Case Based Reasoning dalam Mendukung Penyelesaian Kasus. JURNA DASI. 2010;: p. 10.

[3] Santoso D, Ratnawati DE, Indriati. Perbandingan Kinerja Metode Naïve Bayes, K-Nearest Neighbor, Dan Metode Gabungan K-Means Dan Lvq Dalam Pengkategorian Buku Komputer Berbahasa Indonesia Berdasarkan Judul Dan Sinopsis. Jurnal Universitas Brawijata. 2014;: p. 14.

[4] Samuel Y, Delima R, Rachmat A. Implementasi Metode K-Nearest Neighbor dengan Decision Rule untuk Klasifikasi Subtopik Berita. Jurnal Informatika, Vol. 10 No. 1. 2014;: p. 15.

[5] Rizki AS, Indriati , Muflikhah L. Text Mining Klasifikasi Soal Biologi Sekolah Menengah Atas Dengan Metode Improved KNN. Repositori Jurnal Mahasiswa PTIIK UB. 2014;: p. 8.

[6] Sugiyono. Metode Penelitian Kualitatif Bandung: Alfabeta; 2005.

[7] Pressman RS. Rekayasa Perangkat Lunak Yogyakarta: Andi; 2002.

[8] Munir R. Algoritma dan Pemrograman dalam Bahasa Pascal dan C, Edisi ke-3, Buku 1 Bandung: Informatika Bandung; 2011.

[9] Aamodt A, Plaza E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. 1994 IOS Press; Vol. 7: 1, pp. 39-59.

[10] Yovianto E. Buku TA : K-Nearest Neighbor (KNN). [Online].; 2010 [cited 2016 Januari 9. Available from: https://kuliahinformatika.wordpress.com/2010/02/13/buku-ta-k-nearest-neighbor-knn/.

[11] Diaz R. Pengertian Data Mining,Teks Mining,dan Web Mining. [Online].; 2013 [cited 2016 Januari 10. Available from: http://yosephoriolryandiaz.blogspot.co.id/2013/03/pengertian-data-miningteks-miningdan.html.

[12] Hasanah U. Label: analyzing, dokumen, filtering, kata, proses filtering, stemming, tagging. [Online].; 2012 [cited 2016 Januari 27. Available from: http://sistemtemukembaliinformasi.blogspot.co.id/2012/07/tokenisasi.html.

[13] Agusta L. Konferensi Nasional Sistem dan Informatika. Perbandingan Algoritma Stemming Porter Dengan Algoritma Nazief & Adriani Untuk Stemming Dokumen Teks Bahasa Indonesia. 2009 November; 036(KNS&I09).

[14] Tala FZ. A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. The Netherlands: Institute for Logic, Language and Computation Universiteit van Amsterdam, Master of Logic Project; 2003.

[15] Baskoro DO, Malik H, Anshari MH. Porter Stemmer Information Retrieval. In Computer Science Gadjah Mada University; 2012: COMPUTER SCIENCE. p. 6.

[16] Ridok A. Pembuatan Judul Otomatis Dokumen Berita Berbahasa Indonesia Menggunakan Metode KNN. ISSN: 1907-5022. 2012;: p. 5.

[17] Herwansyah A. APLIKASI PENGKATEGORIAN DOKUMEN DAN PENGUKURAN TINGKAT SIMILARITAS DOKUMEN MENGGUNAKAN KATA KUNCI PADA DOKUMEN PENULISAN ILMIAH UNIVERSITAS GUNADARMA. Jakarta: Universitas Gunadarma, Sistem Informasi; 2009.

[18] Salsabilla SM. Sistem Peringkasan Jurnal Ilmiah Menggunakan Metode Maximal Marginal Relevance Bandung: Universitas Islam Negeri Sunan Gunung Djati Bandung; 2016.

[19] A. Suhendar SS, Hariman Gunadi SS. Visual Modeling Menggunakan UML dan Rational Rose Bandung: Informatika Bandung; 2002.

[20] Rosa , Shalahuddin. Rekayasa Perangkat Lunak Bandung: Informatika; 2013.

LAMPIRAN

1. Data Empiris

1.1 Hasil Penilaian Ujian Akhir Mahasiswa Angkatan 2012-2014

Tahun Angkatan	Kelas	Nilai Ujian Akhir				
		A	B	C	D	E
2012	A	2	16	4	1	2
	B	5	32	1	1	2
	C	2	29	1	0	3
	D	2	27	1	0	3
	E	26	9	1	0	2
2013	A	15	16	5	1	3
	B	4	13	6	2	2
	C	6	13	15	0	2
	D	1	6	7	0	2
	E	3	16	19	1	0
2014	A	13	21	5	0	0
	B	9	29	1	0	2
	C	3	31	2	0	4
	D	1	5	29	3	3
	E	1	10	4	1	5

1.2 Hasil Quesioner Dari 17 Mahasiswa

No.	Pertanyaan	Ya	Ragu	Tidak
1.	Apakah pengkodean program merupakan sesuatu yang baru bagi anda?	5	1	11
2.	Apakah memahami langkah pengerjaan dari soal algoritma adalah sesuatu yang mudah?	4	9	4
3.	Adakah kesulitan dalam menuangkan algoritma kedalam kode program?	10	3	4
4.	Apakah mencari jawaban dari soal agoritma membutuhkan referensi yang banyak?	9	4	4
5.	Adakah kesulitan dalam mencari atau mengoleksi referensi?	3	7	7
6.	Adakah kesulitan untuk mencari kemiripan antara soal algoritma yang ingin dicari jawabannya dengan referensi?	3	5	9
7.	Perlukah mengetahui termasuk kedalam struktur dasar mana pada soal yang ingin dicari jawabannya?	14	3	0
8.	Perlukah adanya aplikasi yang dapat mencari kemiripan antara referensi dan soal algoritma sehingga dapat memberikan rekomendasi jawaban dari soal algoritma?	16	1	0

2. Pembuangan Kata Pada *Stopword Removal*

Kata Yang Dibuang						
ada	beginilah	depan	kalau	makanya	rupanya	semua
adanya	sebegini	di	kalaulah	makin	serupa	semuanya
adalah	begitu	diberikan	kalaupun	malah	saat	sendiri
adapun	begitukah	diketahui	kalian	malahan	saatnya	sendirinya
agak	begitulah	dia	kami	mampu	sesaat	seolah
agaknya	begitupun	dialah	kamilah	mampukah	saja	seperti
agar	sebegitu	dini	kamu	mana	sajalah	sepertinya
angka	belum	diri	kamulah	manakala	saling	serta
akan	belumlah	dirinya	kan	manalagi	bersama	siapa
akankah	sebelum	terdiri	kapan	masih	sama	siapakah
aku	sebelumnya	dong	kapankah	masihkah	sesama	siapapun
akulah	sebenarnya	dulu	kapanpun	semasih	sambil	disini
amat	berapa	enggak	dikarenakan	masing	sampai	disinilah
amatlah	berapakah	enggaknya	karena	mau	sana	sini
anda	berapalah	entah	karenanya	maupun	sangat	sinilah
andalah	berapapun	entahlah	ke	semaunya	sangatlah	sesuatu
antar	berikut	terhadap	kemudian	memang	saya	sesuatunya
diantaranya	berikutnya	terhadapnya	kenapa	mereka	sayalah	suatu
antara	berupa	hal	kepada	merekalah	sebab	sesudah
antaranya	betulkah	hampir	kepadanya	meski	sebabnya	sesudahnya
diantara	sebetulnya	hanya	ketika	meskipun	sebuah	sudah
apa	biasa	hanyalah	seketika	semula	tersebut	sudahkah
apaan	biasanya	harus	khususnya	mungkin	tersebutlah	sudahlah
mengapa	bila	haruslah	kini	mungkinkah	untuk	supaya
apabila	bilakah	harusnya	kinilah	menentukan	sedang	tadi
apakah	bisa	seharusnya	kiranya	nah	sedangkan	tadinya
apalagi	bisakah	hendak	sekiranya	namun	sedikit	tanpa
apatah	sebisanya	hendaklah	kita	nanti	sedikitnya	setelah
atau	boleh	hendaknya	kitalah	nantinya	segala	telah
ataukah	bolehkah	hingga	kok	nyaris	segalanya	tentang
ataupun	bolehlah	sehingga	lagi	oleh	segera	seterusnya
bagai	buat	ia	lagian	olehnya	sesegera	tapi
bagaikan	buah	ialah	selagi	seorang	sejak	tetapi
sebagai	bukan	ibarat	lah	seseorang	sejenak	setiap
sebagainya	bukankah	ingin	lain	pada	sekali	tiap
bagaimana	bukanlah	inginkah	lainnya	padanya	sekalian	setidaknya
bagaimanapun	bukannya	inginkan	melainkan	padahal	sekalipun	tidak
sebagaimana	cuma	ini	selaku	paling	sesekali	tidakkah
bagaimanakah	percuma	inikah	lalu	sbb	sekaligus	tidaklah

Kata Yang Dibuang						
bagi	dahulu	inilah	melalui	sepanjang	sekarang	toh
bahkan	dalam	itu	terlalu	pantas	sekarang	tak
bahwa	dan	itukah	lama	sepantasnya	sekitar	waduh
bahwasanya	dapat	itulah	lamanya	sepantasnyalah	sekitarnya	wah
sebaliknya	dari	jangan	selama	para	sela	wahai
banyak	daripada	jangankan	selama	pasti	selain	sewaktu
sebanyak	dekat	janganlah	selamanya	pastilah	selalu	walau
beberapa	demi	jika	terlebih	per	seluruh	walaupun
seberapa	demikian	jikalau	bermacam	pernah	seluruhnya	wong
begini	demikianlah	juga	macam	pula	semakin	yaitu
beginian	sedemikian	justru	semacam	pun	sementara	yakni
beginikah	dengan	kala	maka	merupakan	sempat	yang

3. Kode Program Metode & Algoritma

3.1 Metode *Case Folding* & *Tokenizing*

```
public void tokenisasi() {
    String Kalimat = TACari.getText().toLowerCase()
        .replace("!", " ").replace("@", " ").replace("#", " ").replace("$", " ")
        .replace("%", " ").replace("^", " ").replace("&", " ").replace("*", " ")
        .replace("(", " ").replace(")", " ").replace("-", " ").replace("_", " ")
        .replace("=", " ").replace("+", " ").replace("{", " ").replace("[", " ")
        .replace("}", " ").replace("]", " ").replace(";", " ").replace(":", " ")
        .replace("'", " ").replace(",", " ").replace("<", " ").replace(".", " ")
        .replace(">", " ").replace("/", " ").replace("?", " ").replace("~", " ")
        .replace("`", " ").replace("\"", " ").replace("|", " ")..replace("\'", " ")
        .replaceAll("[0-9]", " ").replaceAll("\\n+", " ").replaceAll("\\s+", " ")
        .replaceAll("\\t+", " ");
    Kata = Kalimat.split(" ");
}
```

3.2 Metode *Stopword Removal*

```
public String filter(String str) throws SQLException {
      tokenisasi();
      FileReader fr = null;
      try {
         fr = new FileReader("D:\\stopwords_id.txt");
      } catch (FileNotFoundException ex) {
         Logger.getLogger(CariJawaban.class.getName()).log(Level.SEVERE, null, ex);
          JOptionPane.showConfirmDialog(null, "File tidak ditemukan!",
         "Confirmation", JOptionPane.OK_OPTION);
      }
      BufferedReader br = new BufferedReader(fr);
```

```
        try {
            while ((sCurrentLine = br.readLine()) != null) {
                stopwords[k] = sCurrentLine;
                k++;
            }
        } catch (IOException ex) {
            Logger.getLogger(CariJawaban.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showConfirmDialog(null, "Tidak dapat membuang kata!",
            "Confirmation", JOptionPane.OK_OPTION);
        }

        wordsList.addAll(Arrays.asList(Kata));
        for (int i = 0; i < wordsList.size(); i++) {
            for (int j = 0; j < k; j++) {
                if (wordsList.contains(stopwords[j])) {
                    wordsList.remove(stopwords[j]);
                }
            }
        }

        StringBuilder buffer = new StringBuilder();
        boolean processedFirst = false;
        for (Iterator<String> it = wordsList.iterator(); it.hasNext();) {
            str = it.next();
            if (processedFirst) {
                buffer.append(" ");
            }
            buffer.append(Stemming(str));
            processedFirst = true;
        }

        String query01 = "ALTER TABLE appindex AUTO_INCREMENT = 1";
        PreparedStatement preparedStmt01 = connection.getKoneksi()
        .prepareStatement(query01);
        preparedStmt01.executeUpdate();
        String firstParam = buffer.toString();
        String query = "INSERT INTO appindex(Kata) VALUES (?)";
        PreparedStatement preparedStmt = connection.getKoneksi()
        .prepareStatement(query);
        preparedStmt.setString(1, firstParam);
        preparedStmt.executeUpdate();
        return str;
    }
```

3.3 Algoritma *Porter Stemmer*

```
private String Stemming(String str) throws SQLException {
      if (cek(str) == true) {
         return str;
      } else {
         if (cek(str) == true) {
            return str;
         } else {
            if (cek(hapusPartikel(str)) == true) {
               str = hapusPartikel(str);
            }

            if (cek(hapusPossesivePronoun(hapusPartikel(str))) == true) {
               str = hapusPossesivePronoun(hapusPartikel(str));
            }

            if ((str.startsWith("se") || str.startsWith("ber") && str.endsWith("lah"))
                  || (str.startsWith("ber") && str.endsWith("ku"))) {
               str = hapusFirstOrderPrefiks(str);
               if (cek(hapusSecondOrderPrefiks(hapusSuffiks(str))) == true) {
                  str = hapusSuffiks(str);
                  if (cek(hapusSecondOrderPrefiks(str))) {
                     str = hapusSecondOrderPrefiks(str);
                  }
               } else if (cek(hapusSuffiks(hapusSecondOrderPrefiks(str))) == true) {
                  str = hapusSecondOrderPrefiks(str);
                  if (cek(hapusSuffiks(str))) {
                     str = hapusSuffiks(str);
                  }
               }
         } else {
         str=hapusFirstOrderPrefiks(hapusPossesivePronoun(hapusPartikel(str)));
             if (cek(hapusSecondOrderPrefiks(hapusSuffiks(str))) == true) {
                  str = hapusSuffiks(str);
                  if (cek(hapusSecondOrderPrefiks(str))) {
                     str = hapusSecondOrderPrefiks(str);
                  }
               } else if (cek(hapusSuffiks(hapusSecondOrderPrefiks(str))) == true) {
                  str = hapusSecondOrderPrefiks(str);
                  if (cek(hapusSuffiks(str))) { str = hapusSuffiks(str); }
               }
            }
         }
      }
      return str;
}
```

3.4 Metode *TF-IDF*

```
public class TfIdf {
    public double tfCalculator(String[] totalterms, String termToCheck) {
        double count = 0;
        for (String s : totalterms) {
            if (s.equalsIgnoreCase(termToCheck)) {
                count++;
            }
        }
        return count / totalterms.length;
    }

    public double idfCalculator(List<String[]> allTerms, String termToCheck) {
        double count = 0;
        for (String[] ss : allTerms) {
            for (String s : ss) {
                if (s.equalsIgnoreCase(termToCheck)) {
                    count++;
                    break;
                }
            }
        }
        return Math.log(allTerms.size() / count);
}
```

3.5 Metode *Cosine Similarity*

```
public double cosineSimilarity(double[] docVector1, double[] docVector2) {
        double skalar = 0.0;
        double jarak1 = 0.0;
        double jarak2 = 0.0;
        double cosineSimilarity = 0.0;
        for (int i = 0; i < docVector1.length; i++) {
            skalar += docVector1[i] * docVector2[i];  //a.b
            jarak1 += Math.pow(docVector1[i], 2);  //(a^2)
            jarak2 += Math.pow(docVector2[i], 2); //(b^2)
        }
        jarak1 = Math.sqrt(jarak1);//sqrt(a^2)
        jarak2 = Math.sqrt(jarak2);//sqrt(b^2)
        if (jarak1 != 0.0 | jarak2 != 0.0) {
            cosineSimilarity = skalar / (jarak1 * jarak2);
        } else { return 0.0; }
        return cosineSimilarity;
}
```

3.6 Algoritma *KNN*

```
private void klasifikasi() {
        try {
            int K = 5;
            String query2 = "select autoid, kelas, count(kelas) as jum from"
            + "(select autoid, kelas from appbaru order by SimDocAkhir DESC limit "
            + K + ") as a group by kelas order by jum";
            Statement st2 = connection.getKoneksi().createStatement();
            ResultSet rs2 = st2.executeQuery(query2);
            while (rs2.next()) {
                String kelas = rs2.getString("kelas");
                TFKelas.setText(kelas);
            }
        } catch (SQLException ex) {
            Logger.getLogger(CariJawaban.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```