# **PROPOSAL**

**IMPLEMENTASI *LATENT SEMANTIC ANALIYSIS METHOD* UNTUK MENDETEKSI KEMIRIPAN KALIMAT**

**(Studi Kasus: Jurusan Teknik Informatika Universitas Halu Oleo)**

**Diajukan untuk memenuhi salah satu syarat memperoleh derajat Sarjana Teknik**

****

**MUHAMMAD BUDI DHARMAWAN P**

**E1E1 15 034**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS HALU OLEO**

**KENDARI**

**2019**

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Dalam perkembangannya, internet telah banyak membawa dampak perubahan sebagian besar aspek kehidupan, hal ini dibuktikan dengan tingginya angka penggunaan internet di Indonesia. Perkembangan dan jumlah internet di Indonesia pada tahun 2014, Kementrian Teknologi dan Informatika (KOMINFO) mengeluarkan hasil riset sebuah lembaga *e-Marketer* yang mengatakan populasi netter tanah air mencapai 83,7 juta orang (Hidayat, 2014). Pada tahun 2017 pengguna internet di Indonesia meningkat 8% menjadi 143, 26 juta pengguna. Jika dibandingkan dengan data populasi penduduk Indonesia yang sebanyak 262 juta jiwa, maka pengguna internet di Indonesia sudah mencapai 54,68% dari total populasi di Indonesia (Tim APJII, 2018).

Pendidikan di tanah air juga tidak luput dari pengaruh internet, hal ini dilihat dari banyaknya keikutsertaan internet dalam aspek-aspek pendidikan, mulai dari administrasi-administrasi pendidikan, sampai dengan model pembelajaran yang telah *ter-mordenisasi* menggunakan internet. Adanya mata pelajaran Teknologi Informasi dan Komunikasi (TIK), juga membuktikan betapa pentingnya mempelajari kemajuan teknologi khususnya internet itu sendiri. Salah satu bentuk pemanfaatan internet dalam dunia pendidikan adalah denga hadirnya model pembelajaran *e-learning*. Pada awal mulanya penerapan *e-learning* yaitu pada saat memanfaatkan *disk* untuk menyampaikan materi melalui video, banyak orang berpendapat bahwa *e-learning* mirip dengan televisi (Suranto, 2009).

Saat ini mulai banyak yang tertarik untuk mengimplementasikan sebuah kelas konvensional kedalam bentuk *virtual classroom* (VC). Ujian online merupakan bentuk virtual classroom, dalam pemanfaatanya ujian online sering digunakan untuk menghilangkan keterbatasan berupa ruang kelas untuk ujian dan dengan memberikan kemudahan akses oleh peserta ujian. Namun disisilain, pemanfaatan internet dalam *virluar classroom* memberikan dampak kerentanan .

terhadap kecurangan saat ujiaan online dilaksanakan, tindakan curang ini dilakukan oleh peserta ujian seperti dalam ujian *essay*. Penggunaan *virtual calssroom* saat unjian online akan memberikan kesempatan kepada peserta ujian untuk menjiplak/mencontek pekerjaan *essay* peserta lainya kerena ketiadaan pengawasan pelaksanaan ujian secara fisik. Tentu sebagai sebuah lembaga pendidikan yang menjunjung tinggi kejujuran, kerentanan dari ujian online ini akan membawa dampak kemerosotan akhlak dan kemampuan intelektual peserta ujian.

Dalam pengembangun sebuah *virtual classroom* dapat dilakukan pendeteksian untuk mengindikasikan tindak kecurangan yang terjadi saat ujian online, tentu hal ini berkaitan dengan implementasi metode yang sesuai terhadap sistem *e-learning*. Salah satu metode yang dapat digunakan dalam mendeteksi kemiripan jawaban *essay* adalah *Latent Semantik Analiysis Method (LSA).* Metode ini merupakan sebuah teori dan metode untuk menggali dan mempresentasikan konteks yang digunakan sebagai sebuah arti kata dengan memanfaatkan komputasi statick untuk sejumlah *corpus*  yang besar. (Nova, Dadang, & Ardiyanto , 2012).

Berdasarkan pemaparan diatas, maka peneliti mengembangkan sebuah sistem yang mampu mendeteksi kemiripan jawaban *essay test* peserta ujian, agar dapat melakukan pertimbangan terhadap tingkat originalitas jawaban dari masing masing peserta ujian. Atas dasar inilah peneliti mengangkat judul ***“*Implementasi *Latent Semantic Analiysis Method* Untuk Mendeteksi Kemiripan Kata*”*.** Implementasi ini akan dilakukan pada sistem *e-learning* *essay test* di jurusan Teknik Informatika Fakultas Teknik Universitas Halu Oleo.

* 1. **Rumusan Masalah**

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah dalam penilitian ini adalah bagaimana merancang dan membangun sistem *e-learning* yang dapat digunakan untuk melakukan ujian online yang dapat mendeteksi kemiripan kalimat jawaban *essay* *test* dengan menggunakan metode *Latent Semantic Analiysis.*

* 1. **Batasan Masalah**

Adapun hal-hal yang menjadi batasan masalah pada penelitian ini adalah sebagai berikut :

1. Peneliti hanya berfokus pada pendeteksian kemiripan jawaban *essay* dan bukan untuk jawaban pilihan ganda atau *optional* lainya.
2. Pendeteksian tidak secara *real time*, artinya pendeteksian hanya dapat dilakukan setelah ujian *online* berlangsung.
3. Pendekteksian kemiripan jawaban *essay* dilakukan pada tiap nomor dengan jumlah peserta maksimal 15 orang. soal dalam akumulasi persen.
4. Sistem hanya mendeteksi kemiripan kata pada jawaban *essay* bahsa Indonesia dan bukan mendeteksi makna kalimat.
   1. **Tujuan Penelitian**

Tujuan dari penelitian ini adalah mengetahui hasil dari ujicoba implementasi metode *Latent Semantic Analiysis* pada sistem *e-learning* untuk mendeteksi kemiripan kalimat pada jawaban *essay* peserta ujian.

* 1. **Manfaat Penelitan**

Manfaat yang diharapkan dari penelitian ini adalah dapat memberikan kemudahan bagi tenaga penndidik (dosen) selaku pemberi ujian dalam menentuka kemiripan kalimat jawaban *essay* antara peserta ujian (mahasiswa).

* 1. **Sistematika Penulisan**

Sistematika penulisan tugas akhir digunakan untuk menjelaskan penulisan perbab sebagai berikut :

**BAB I PENDAHULUAN**

Merupakan bab pendahuluan yang menguraikan latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

**BAB II LANDASAN TEORI**

Membahas mengenai dasar-dasar teori pendukung yang berhubungan dengan masalah yang diambil dan program aplikasi yang akan digunakan dalam pembangunan sistem.

**BAB III METODE PENELITIAN**

Membahas mengenai waktu dan tempat penelitian, metode pengumpulan data, metode pembangunan sistem yang digunakan dalam penelitian tugas akhir ini.

**BAB IV ANALISIS DAN PERANCANGAN**

Bab ini akan membahas tentang analisis dan perancangan dari aplikasi yang akan dibuat, dengan menggunakan desain UML (*Unified Modelling Language*) serta desain *user interface*.

**BAB V IMPLEMENTASI DAN PEMBAHASAN**

Dalam bab ini akan dikaji mengenai implementasi hasil perancangan aplikasi yang dibuat serta melakukan pengujian terhadap sistem.

**BAB VI PENUTUP**

Bab ini berisi kesimpulan dari program yang telah dibuat serta saran yang diperlukan untuk pengembangan program berikutnya.

* 1. **Tinjauan Pustaka**

Penelitian ini didasarkan pada permintaan seorang dosen akan kebutuhan sistem *e-learning* yang dapat membantu aktifitas akademik di Fakultas Teknik Universitas Haluoleo.

Penelitian yang dilakukan oleh Diah Yustiana (2015) dalam membuat sebuah sistem penilaian otomatis terhadap jawaban *essay* menggunakan metode *Latent Semantic Analiysis* (LSA) menyimpulkan bahwa adanya beberapa proses yang mempengaruhi secara langsung performa metode *Latent Semantic Analiysis* (LSA), yaitu: *stopword removal*, pembobotan kata, pengurangan dimensi, metode pengukuran kemiripan, dan banyaknya data (korpus) yang akan diolah. *Stopword removal* dapat mengurangi banyaknya data yang akan diolah, karena proses ini berfungsi untuk menghilangkan kata-kata yang dianggap tidak penting. Cara pembobotan kata yang digunakan akan dapat mempengaruhi kecepatan sistem, hal ini disebabkan karena pengaruh banyaknya perhitungan yang harus dilakukan dengan banyaknya data yang akan dihitung (Yustiana, 2015).

Diah Yustina juga menyebutkan bahwa, sistem yang dirancang telah berhasil mencapai tingkat korelasi hingga 71% dengan kategori cukup. Maka untuk meningkatkan performa dari sistem penilaian otomatis ini, disarankan untuk melakukan pengembangan seperti berinovasi dengan beberapa teknik yang ada pada proses *prepocessing*-nya, khususnya pada metode pembobotan kata (bisa dilakukan dengan tehnik GfIDF atau Probabilistic IDF), dan metode pengukuran kemiripannya (bisa dilakukan dengan *Cosin Similarity* atau *Language Based Matching*, dan juga bisa dilakukan pengembangan untuk menggunakan beberapa macam alternatif kunci jawaban, sehingga perhitungan pembobotan kata untuk setiap soal pada kunci jawaban bisa dilakukan dengan menggunakan tema yang sama. Penulis juga menyarankan untuk menambahkan sebuah proses integrasi terhadap *software-software e-learning* yang sudah ada, sehingga kinerja sistem ini akan jauh lebih baik dan benar-benar dapat sepenuhnya meringankan tugas para pengajar dalam melakukan proses *evaluasi* yang cepat dan akurat. (Yustiana, 2015)

Penelitian yang dilakukan oleh Asih Sekarsiwi dan Hendri (2015) dalam perancangan aplikasi test online dengan memanfaatkan algoritma *Latent Semantic Analiysis (LSA)*, menyimpulkan bahwa *Latent Semantic Analiysis (LSA)* dapat digunakan untuk menilai soal berbentuk *essay*, dengan cara membuat basis data untuk menyimpulkan data kunci jawaban sebagai patokan, jawaban mahasiswa, list kata dasar, *stopword list,* dan skor nilai. Kemudian melakukan proses *streaming*, pembobotan kata, memecahkan matriks dan mencari nilai similaritas. Hasil dari penelitian ini adalah standar deviasi hasil nilai *human rates* sebesar 3.48, standar deviasi untuk hasil nilai menggunakan LSA sebesar 3.08, *Imean Sequare Error* (MSE) sebesar 10,12. (Asih & Hendry, 2014)

Peneliti juga menyarankan untuk pengembangan aplikasi ini dengan menambah jumlah kosa kata yang lebih luas baik untuk *stopword list,* dan kata dasar. Juga agar dalam pengembangannya lebih memerhatikan pemilihan kunci jawaban, menyusun daftar persamaan kata kunci jawabn dan pembobotan terhadap posisi kata kunci dalam kalimat.

Penelitian yang dilakukan oleh Muhammad Saiful (2019) dalam implementasi metode *Roccio* untuk klasifikasi jenis pelaporan di laman PROPAM POLDA SULTRA myimpulkan bahwa lgoritma *Rocchio* memiliki kinerja yang baik untuk klasifikasi laporan pengaduan berdasarkan isi laporan. Hal ini dibuktikan pada pengujianmenggunakan data laporan yang diambil dari Propam POLDA SULTRA*,* kemudian laporan diklasifikasikan pada delapan kategori yaitu asusila, laka lantas,narkotika, pelayanan yang kurang baik, penganiayaan, pengancaman, penghinaan dan penyimpangan dalam melaksanakan tugas (penangananperkara). Hasil klasifikasi menggunakan 79 laporan didapatkan akurasi 89,87 %. Algoritma *Rocchio* mampu melakukan proses klasifikasi laporan yang masuksecara otomatis dan proses klasifikasi semakin akurat jika sampel kata dasar semakin banyak yang digunakan dalam pembelajaran. (Saiful, 2019)

Berdasarkan penelitian-penelitian yang telah dipaparkan di atas, maka akan dilakukan penelitian lebih lanjut dengan judul ***“*Implementasi *Latent Semantic Analiysis Method* Untuk Mendeteksi Kemiripan Kalimat.** Penelitian ini nantinya akan diimplementasikan untuk sistem *e-learning* jurusan Teknik Informatika Universitas Halu Oleo.

**BAB II**

**LANDASAN TEORI**

**2.1** ***E-Learning***

*E-learning* (electronik learning) atau dalam bahasa indonesia yang berarti “pembelajaran elektronik”, merupakan sebuah fasilitas elektronik yang dewasa ini sudah menjadi bagian dari gaya pembelajaran hampir disemua lembaga atau instansi pendidikan. *E-learning* saat ini sering dikaitkan dengan internet dan *website*, padahal sebenarnya *e-learning* tidak hanya sebatas penggunaan internet dan *website* semata, namun sejatinya *e-learning* adalah pengggunaan alat elektronik dalam proses belajar mengajar.

Para ahli memberikan pernyataan yang berbeda-beda mengenai pengertian *e-learning*, namun mengarah pada pokok yang sama. Menurut Rusman (2012) *e-learning* merupakan segala aktivitas belajar yang menggunakan bantuan teknologi elektronik. Melalui *e-learning*, pemahaman siswa tentang sebuah materi tidak tergantung pada guru/instruktur tetapi dapat diperoleh dari media elektronik. Teknologi elektronik yang banyak digunakan misalnya internet, tape video atau audio, penyiaran melalui satelit, televisi interaktif, *virtual reality* serta CD-ROM. Menurut Daryanto (2010) *e-learning* merupakan sistem pembelajaran yang dapat membantu kegiatan pembelajaran dengan memanfaatkan media elektronik. Dari beberapa pengertian dari para ahli tersebut memusatkan pengertian *e-learning* pada sistem pembelajaran yang memanfaatkan penggunaan media elektronik. (Jhon, 2019)

Triluqman (2007) menjelaskan dengan terperinci mengenai kelemahan dan kelebihan dari sistem pembelajaran *e-learning*. Berikut penjabaran mengenai kekurangan *e-learning*:

1. Kurangnya interaksi antara pendidik dan peserta didik bahkan antar-peserta didik itu sendiri. Kurangnya interaksi ini bisa memperlambat terbentuknya *values* dalam proses belajar-mengajar.
2. Kecenderungan mengabaikan aspek akademik atau aspek sosial dan sebaliknya mendorong tumbuhnya aspek bisnis. Juga proses belajar dan mengajarnya cenderung ke arah pelatihan daripada pendidikan.
3. Peserta didik yang tidak mempunyai motivasi belajar yang tinggi cenderung gagal. Serta berubahnya peran pendidik dari yang semula menguasai teknik pembelajaran konvensional.

Sedangakan untuk penjabaran mengenai kelebihan *e-learning* menurut Triluqman (2007) adalah sebagai berikut:

1. Tersedianya fasilitas *e-moderating* dimana pendidik dan peserta didik dapat berkomunikasi dengan mudah melalui fasilitas internet secara regular atau kapan saja kegiatan berkomunikasi itu dilakukan dengan tanpa dibatasi oleh jarak, tempat, dan waktu.
2. Pendidik dan peserta didik dapat menggunakan bahan ajar atau petunjuk belajar yang tersruktur dan terjadwal melalui internet, sehingga keduanya bisa saling menilai sampai berapa jauh bahan ajar dipelajari. Serta peserta didik dapat belajar atau mengulas kembali bahan ajar setiap saat dan dimana saja jika bahan ajar tersimpan di komputer.
3. Apabila peserta didik memerlukan tambahan informasi yang berkaitan dengan bahan yang dipelajarinya, ia dapat melakukan akses di internet. Serta baik pendidik maupun peserta didik dapat melaksanakan diskusi melalui internet yang dapat diikuti dengan jumlah peserta yang banyak, sehingga menambah ilmu pengetahuan dan wawasan yang lebih luas. Juga berubahnya peran peserta didik dari yang biasanya pasif menjadi aktif. (Triluqman, 2017)

**2.2 Python**

Python adalah bahasa pemrograman *interpretatif* multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python merupakan salah satu bahasa pemrograman yang populer di dunia kerja Indonesia. Python juga merupakan bahasa pemrograman yang relatif mudah untuk pemula. Selain itu di akademik pun banyak akademisi yang menggunakan python untuk menyelesaikan penelitiannya di bidang robotika, data science, ekonomi, antariksa dan berbagai macam bidang lainnya. Python secara default telah terpasang di beberapa sistem operasi berbasis Linux seperti Ubuntu, Linux Mint, dan Fedora. Untuk sistem operasi lain, sudah tersedia installer yang disediakan untuk sistem operasi tersebut.

Python memiliki kelebihan tersendiri dibandingkan dengan bahasa pemrograman lainnya terutama dalam hal penanganan modul, ini yang membuat beberapa programmer menyukai Python. Selain itu Python merupakan salah satu produk yang bersifat open source, free, dan multiplatform. Beberapa feature yang dimiliki Python antara lain.

1. Memiliki library yang luas, dalam distribusi Python telah disediakan modul-modul siap pakai untuk berbagai keperluan
2. Memiliki tata bahasa yang jernih dan mudah dipelajari
3. Memiliki aturan layout source code yang memudahkan pengecekan, pembacaan kembali, dan penulisan ulang source code tersebut
4. Bersifat OOP
5. Memiliki sistem pengelolaan memory yang auto, garbage collection, seperti Java
6. Bersifat modular, mudah dikembangkan dengan menciptakan modul-modul baru, dimana modul-modul tersebut dapat dibangun dengan bahasa Python maupun C/C++

**2.3** ***Hyper Text Markup Language (HTML)***

*HTML* yang merupakan singkatan dari *Hyper Text Markup Language* adalah serangkaian kode program yang merupakan dasar dari representasi visual sebuah halaman *Web*. Didalamnya berisi kumpulan informasi yang disimpan dalam *tag-tag* tertentu, dimana *tag*-*tag* tersebut digunakan untuk melakukan format terhadapinformasi yang dimaksud.

Berbagai pengembangan telah dilakukan terhadap kode *HTML* dan telah melahirkan teknologi-teknologi baru di dalam dunia pemrograman *web*. Kendati demikian, sampai sekarang *HTML* tetap berdiri kokoh sebagai dasar dari bahasa *web* seperti *PHP*, *ASP*, *JSP* dan lainnya. Bahkan secara umum, mayoritas situs *web* yang ada di Internet pun masih tetap menggunakan *HTML* sebagai teknologi utama (Constantianus & Suteja, 2010).

**2.4** **MySQL**

Menurut Abdul Kadir (2008), *MySQL* (dibaca: mi-se-kyu-el) merupakan software yang tergolong sebagai *DBMS* (*Database Management System*) yang bersifat *open source. Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (code yang dipakai untuk membuat *MySQL*). Selain tentu saja bentuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi. Dan bisa diperoleh secara gratis dengan mendownload di internet.

*SQL* *(Structured Query Language)* merupakan sebuah bahasa relational yang berisi pernyataan yang digunakan untuk memasukkan, mengubah, menghapus, memilih dan melindungi data (Prihatna, 2005). *SQL* bukan *database* aplikasi, tetapi lebih berarti dengan suatu bahasa yang digunakan untuk mengajukan pertanyaan ke dalam *database* berupa pengguna *SQL*. *Database* sistem yang memiliki konsep sama dengan *SQL*, adalah Postgres dan *MySQL*, dimana *database* tersebut bisa didapatkan gratis atau dengan harga yang murah. *MySQL* adalah *server multithreaded*, sehingga sangat memungkinkan daemon untuk menghandle permintaan layanan secara stimultan. Model koneksi dengan protocol TCP-IP membuat akses ke server *database* lebih cepat jika dibandingkan dengan menggunakan *mapping drive*.

**2.5** ***Database* (Basis Data)**

Basis data atau *Database* merupakan kumpulan data yang saling berhubungan satu dengan yang lainnya yang diaorganisasikan sesuai struktur tertentu dan disimpan dengan baik.  Untuk mendapatkan informasi yang berguna dari kumpulan data maka diperlukan suatu perangkat lunak (*software*) untuk memanipulasi data sehingga mendapatkan informasi yang berguna. (Sukamto dan Shalahuddin, 2013). *Database* terbentuk dari beberapa komponen, yaitu (Bunafit Nugroho, 2008):

1. *Table*

*Table* atau Tabel adalah sekumpulan data dengan struktur yang sedemikian rupa, terbentuk dari *record* dan *field.* Istilah Tabel disini berbeda dengan istilah Tabel pada HTML, walaupun secara visual hampir sama.

1. *Record*

*Record* adalah sekumpulan *field* yang membentuk suatu objek tertentu.

1. *Field*

*Field* adalah atribut dari objek yang memiliki tipe data tertentu.

**2.6 Walfrom Alpha**

Wolfram Alpha (WolframAlpha atau Wolfram Alpha ) adalah mesin pengetahuan komputasi atau [mesin jawaban yang](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Answer_engine&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhgSsi3feG60AVzF_xZUgrigYWYcmw) dikembangkan oleh Wolfram Alpha LLC. anak perusahaan dari [Wolfram Research](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Wolfram_Research&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhiGG4QylHA1DCvi9ZqX3ebdUwYupw). Situs ini adalah layanan online yang menjawab pertanyaan faktual secara langsung dengan menghitung jawaban dari data terkuratori yang bersumber dari luar, daripada memberikan daftar dokumen atau halaman web yang mungkin berisi jawaban seperti yang mungkin dilakukan oleh [mesin pencari](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Search_engine&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhg592AQgFzL-lPHiyvLrr_c_8zr0Q)

Wolfram Alpha hanya dapat memberikan hasil kueri yang kuat berdasarkan fakta komputasi, bukan pertanyaan tentang ilmu sosial, studi budaya atau bahkan banyak pertanyaan tentang sejarah di mana respons membutuhkan lebih banyak parameter dan kompleksitas. Ia mampu menjawab pertanyaan berbasis fakta [data angka yang](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Natural_language_understanding&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhjA-BFgv2KPXbkwmQGyp__aekKAHA) diutarakan secara khusus seperti "Di mana [Mary Robinson](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Mary_Robinson&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhiuVSrE_ISdqUDoCJqpqBmCbPP6cw) dilahirkan?" atau pertanyaan yang lebih rumit seperti "Berapa umur [Ratu Elizabeth II](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Queen_Elizabeth_II&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhgnGiXZDTfHCxI9iAttiYjRhDfLDA) pada tahun 1974?". Wolfram Alpha tidak menjawab pertanyaan yang memerlukan respons naratif seperti "Apa perbedaan antara kalender [Julian](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Julian_calendar&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhhpmpvY0juJLRr5w_2Z4DceVnPzsg) dan kalender [Gregorian](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.co.id&sl=en&sp=nmt4&tl=id&u=https://en.m.wikipedia.org/wiki/Gregorian_calendar&xid=17259,15700023,15700186,15700190,15700256,15700259&usg=ALkJrhi6V--P_Np0-4xUQOSMc1IQ4j5qDg) ?" tetapi akan menjawab pertanyaan faktual atau komputasi seperti "1 Juni di kalender Julian". Salah satu yang populer di Indonesia (jalantikus.com) menyebutkan Wolfram Alpha menjadi situs denga nomine 7 situs tempat belajar online terbaik dan berada diperingkat pertama. (Rifai, 2015).

**2.7** **Algoritma *Latent Symantik Analisis***

*Latent Semantic Analysis* (LSA) adalah suatu teori atau algoritma untuk mengekstrak dan merepresentasikan kesamaan makna dari kata-kata menggunakan perhitungan statistika yang diterapkan pada keseluruhan teks. LSA menggunakan metode aljabar linear yaitu *singular* *value* *decomposition* (SVD). LSA akan membentuk matriks yang merepresentasikan asosiasi antara term-dokumen yang merupakan *semantic space*, yakni kata–kata dan dokumen-dokumen yang berasosiasi dekat akan ditempatkan dekat satu sama lain (Asih Sekarsiwi, 2014).

Metode *Latent Semantic Analysis* (LSA) memiliki beberapa langkah langkah dalam mendeteksi kemiripan teks pada beberapa dokumen string antara lain :

1. *Parsing Text*

*Parsing* adalah sebuah proses yang dilakukan seseorang untuk menjadikan sebuah kalimat menjadi lebih bermakna atau berarti dengan cara memecah kalimat tersebut menjadi kata-kata atau frase-frase. *Parsing* di dalam pembuatan aplikasi dokumen yang semula berupa kalimat-kalimat berisi kata-kata dan tanda pemisah antar kata seperti titik (.), koma(,), spasi dan tanda pemisah lainnya menjadi kata-kata saja baik itu berupa kata-kata penting maupun tidak penting. *Parsing text* dibagi menjadi tiga bagian seperti:

1. *Tokenizing*

*Tokenizing* merupakan proses mengidentifikasi unit terkecil (*token*) dari suatu struktur kalimat. Tujuan dilakukannya *tokenizing* ini adalah untuk mendapatkan *term-term* yang nantinya akan diindeks. Pengklasifikasian *token* dilakukan untuk teks yang dipisahkan dengan spasi atau enter adalah suatu dokumen.

1. *Filtering*

*Filtering* merupakan proses dimana *token-token* yang didapat dari proses *tokenizing* akan diseleksi dari *token-token* yang dianggap tidak penting (*stoplist*). *Stoplist* merupakan kata yang sering muncul dan bisa diabaikan pada proses *filtering*.

1. *Stemming*

*Stemming* adalah suatu proses yang bertujuan untuk mengambil kata dasar dari kata yang berimbuhan atau kata tunggal dari kata bentukan. Hal itu mengurangi jumlah *term* yang berbeda dalam koleksi.

1. *Singular Value Decomposition* (SVD)

*Singular Value Decomposition* (SVD) adalah satu metode untuk memecahkan masalah-masalah matematik linier. Pada SVD, matriks akan didekomposisi menjadi tiga komponen matriks. Komponen matriks pertama mendeskripsikan entitas baris sebagai vektor orthogonal matriks. Komponen matriks kedua mendeskripsikan matriks diagonal yang berisi nilai skalar dan yang ketiga adalah matriks entitas kolom sebagai vektor orthogonal matriks.

Metode SVD berdasarkan pada teori aljabar linier yang menyatakan bahwa suatu matriks A yang berukuran mxn, mempunyai nilai singular yang merupakan akar pangkat dua *eigenvalue* AᵀA. Adapun rumus yang digunakan sebagai berikut:



Dimana:

A : matriks yang didekomposisi

U : matriks ortogonal U (matriks vektor singular kiri)

S : matriks diagonal S (matriks nilai singular)

V : transpose matriks ortogonal V(matriks vektor singular kanan)

m : jumlah baris matriks

n : jumlah kolom matriks

Setelah memperoleh 3 matriks dari proses SVD, proses berikutnya yang perlu dilakukan adalah mereduksi dimensi dari matriks dengan cara mengurangi dimensi dari matriks kedua yang berupa matriks diagonal. Pengurangan dimensi matriks diagonal ini dilakukan dengan cara mengeset semua nilai diagonal matriks ke dua menjadi nol kecuali diagonal dimensi yang dipilih. Dan jika ketiga komponen matriks tersebut dikalikan maka menghasilkan matriks rekonstruksi yang lain dengan tujuan untuk nilai korelasi yang diinginkan.

Jika nilai-nilai singulir dari matriks S diurutkan berdasarkan nilainya, maka *k* nilai terbesar dapat tetap disimpan dalam matriks tersebut, dan nilai-nilai lain yang lebih kecil dapat diset menjadi nol. Jika baris dan kolom yang berkaitan pada matriks U dan V juga diset menjadi nol, maka hasil kali dari ketiga matrik ini akan membentuk matriks baru yang merupakan matriks *least square approximation*.

Dengan menghapus elemen yang tidak menunjukkan arti, berarti menghapus *noise* yang berada pada vektor pada matriks. Sehingga vektor menjadi lebih pendek dan mengandung elemen yang sangat signifikan dengan data awal saja. Karena tujuan dari LSA bukanlah mendeskripsikan vektor secara verbal tetapi mampu untuk merepresentasikan *term*, dokumen dan menghasilkan *query* yang tidak ambigu dan beredundansi dengan sesama *term*, dimensi nilai k untuk mereduksi matrik bisa menggunakan dua atau tiga vektor saja.

1. Pengukuran kemiripan (*similarity*)

Salah satu pengukuran kemiripan (*similarity*) adalah dengan menghitung sudut antara dua vector yang bersangkutan, disini antara vector dari permintaan (*query*) dengan vector dokumen yang akan dinilai (*document*). Adapun rumus yang digunakan yaitu rumus *cosine similarity* :



Dimana:

A : vektor A

B : vektor B

||A|| : panjang vektor A

||B|| : panjang vektor B

Antara vektor A dan B membentuk sudut a dimana cosinus dari sudut tersebut menunjukkan kemiripan (*similarity*) dari kedua sudut tersebut, bukan menunjukkan jarak (*distance*) antara vektor yang satu dengan yang lainnya.

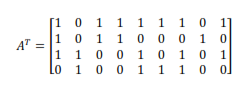
**2.8** **Contoh Perhitungan Metode *Latent Symantik Analisis***

Diasumsikan bahwa sistem telah menerima *text* dari 3 sumber. Dan telah melewati prose *parsing text*, sehingga ditemukan *matrix* sebagai berikut :

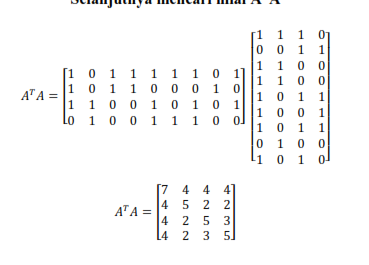
**Tabel 2.1 Perhitungan Bobot Kemunculan Kata**

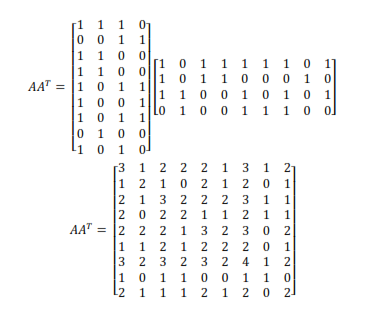
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Q | D1 | D2 | D3 |
| Mampu | 1 | 1 | 1 | 1 |
| Mata | 2 | 1 | 2 | 1 |
| Cembung | 1 | 1 | 0 | 0 |
| Pipih | 1 | 1 | 1 | 1 |
| Lensa | 1 | 1 | 1 | 1 |
| Lihat | 1 | 0 | 1 | 0 |
| Objek | 1 | 0 | 1 | 0 |
| Daya | 0 | 0 | 1 | 0 |
| Tebal | 0 | 0 | 1 | 1 |

Dibutuhkan nilai eigen dan nilai vektor terlebih dahulu. Dimana untuk mencari nilai eigen dan nilai vektor, dimulai dengan mentranspose matriks awal.

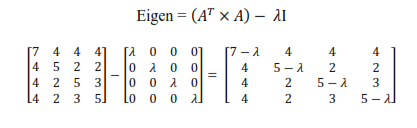


Selanjutnya mencari nilai matriks awal yang dikalikan dengann matriks transpose kemudian sebaliknya.

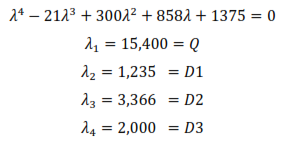




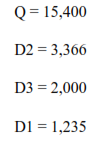
Cari nilai *eigen* dengan cara hasil perkalian matriks AtA dikurangi dengan λ



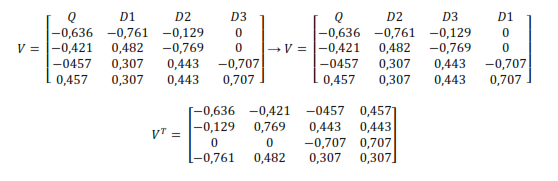
Buatlah persamaan *polynomial* dari matriks diatas dengan mengalikan setiap elemen dan dikurangi elemen dari sisi berlawanan. Dari persamaan dibawah akan didapatkan nilai *eigen* masing-masing dokumen.



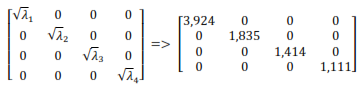
Urutkan nilai *eigen* dari urutan besar ke urutan kecil



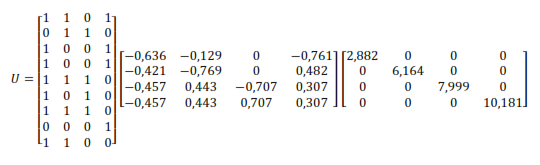
Selanjutnya pembentukan matriks V dengan menggunakan hasil dari kalkulasi normalisasi *eigen* vektor kemudian transpose matriks V.

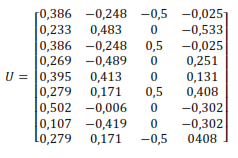


Matriks S didapatkan dari diagonal akar nilai *eigen* yang sudah diurutkan dari yang terbesar.

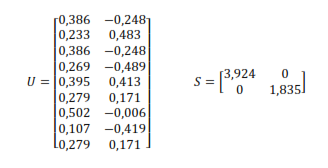
  


Selanjutnya pembentukan matriks U dengan mengkalikan A.V.S-1. Matriks A dan V diurutkan terlebih dahulu dari urutan yang paling besar ke urutan paling kecil. Dengan persamaan

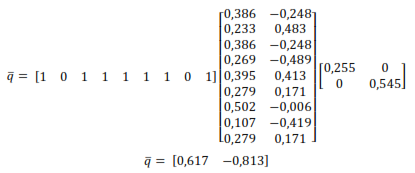




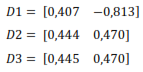
Penyederhanaan sesuai dengan nilai dimensi(k). Penyederhanaan pada matriks U dengan mengurangi jumlah kolom. Pada matriks S akan dilakukan pengurangan pada jumlah baris dan kolom. Jika k bernilai 2 maka matriks U hanya memiliki 2 kolom, sedangkan matriks S memiliki 2 kolom dan 2 baris.



Melakukan pencarian matriks vektor Q pada vektor D sebelum masuk ke proses perhitungan *cosine similarity* pada tahap selanjutnya. Dengan membentuk vektor *query* qt yang dibentuk dengan cara yang sama dengan membentuk matriks A.



Dengan melakukan perhitungan yang sama pada tiap D, maka didapatkan nilai vektor sebagai berikut.



Sekarang dapat dihitung hasil kali titik antara vektor *query* yang sudah dipetakan dengan masing-masing vektor dokumen untuk mendapatkan nilai similaritas dari matriks vektor masing-masing D terhadap Q. Menghitung nilai *cosine similarity* untuk membandingkan kemiripan antara dua vektor, yaitu kunci jawaban(Q) dengan jawaban siswa (D)







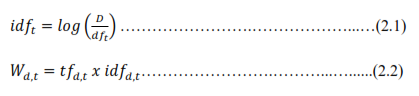


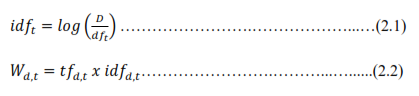
Dari perhitungan terakhir, didapatkan nilai masing-masing siswa yang terdapat pada D1, D2, dan D3.

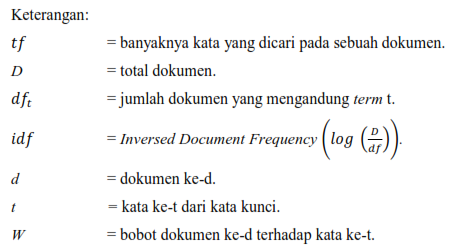
**2.9 Metode *Term Frequency-Inverse Document Frequency (TF-IDF)***

Metode *TF-IDF* (Robertson, 2004) merupakan suatu cara untuk memberikan bobot hubungan suatu kata (*term*) terhadap dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu frekuensi kemunculan sebuah kata di dalam sebuah dokumen tertentu dan *inverse* frekuensi dokumen yang mengandung kata tersebut. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi kata tersebut tinggi di dalam dokumen dan frekuensi keseluruhan dokumen yang mengandung kata tersebut yang rendah pada kumpulan dokumen.

Metode *TF- IDF* merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat. Metode ini akan menghitung nilai *Term Frequency (TF)* dan *Inverse Document Frequency (IDF)* pada setiap *token* (kata) di setiap dokumen dalam korpus. Rumus untuk *TF-IDF*:







**2.10** ***Flowchart***

*Flowchart* adalah simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang saling terhubung sehingga setiap simbol *flowchart* melambangkan pekerjaan dan instruksinya (Ewolf, 2011). Simbol–simbol yang digunakan dalam *flowchart* adalah sebagai berikut:

**Tabel 2.2 Simbol-simbol *flowchart***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO.** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | Proses | Mempresentasikan operasi. |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. |  | *Input / Output* | Mempresentasikan *Input* atau *Output* data yang diproses atau informasi. |

**Tabel 2.2 Simbol-simbol *flowchart* (Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| 3. |  | Keputusan | Keputusan dalam program. |
| 4. |  | Dokumen | Dokument I / O dalam format cetak. |
| 5. |  | *Terminal points* | Awal / akhir *flowchart*. |
| 6. |  | *Preparation* | Pemberian harga awal. |
| 7. |  | Manual *input* | *Input* yang dimasukkan secara manual dari *keyboard*. |
| 8. |  | Penghubung | Keluar atau masuk dari bagian lain *flowchart* khususnya. |
| 9. |  | Penghubung | Keluar atau masuknya dari bagian lain *flowchart* khususnya halaman lain. |
| 10. |  | *Display* | *Output* yang ditampilkan pada terminal. |
| 11. |  | Anak panah | Mempresentasikan alur kerja. |

Sumber: Ewolf, 2011.

**2.11** ***Usnified Modeling Language* (UML)**

*Unified Modeling Language* (UML) merupakan bahasa dalam mendesain perangkat lunak secara visual. Dengan UML, desainer dapat melihat konsep global suatu desain. Desain kemudian dapat dijadikan panduan dalam proses pengembangan dan rekayasa perangkat lunak. Selain itu, UML dapat menjadi media komunikasi gagasan antara pengembang perangkat lunak dengan pengguna (Shalahudin, 2011). Ada beberapa jenis diagram dalam UML yaitu:

1. ***Use Case Diagram***

*Use case diagram* menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat *user*) dan bukan bagaimana fungsi yang ada di dalam sistem. (Shalahudin, 2011).

**Tabel 2.3 Simbol-simbol *Use Case Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case.* |
| 2. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*dependent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). |
| 3. |  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*). |
| 4. |  | *Include* | Menspesifikasikan bahwa *use case* sumber secara eksplisit. |
| 5. |  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 6. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7. |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8. |  | *Use case* | Deskripsi dari uraian aksi-aksi yang ditampilkan system yang menghasilkan suatu hasil yang terukur bagi suatu *actor.* |
| 9. |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |

Sumber: Shalahuddin, M., 2011.

1. ***Activity Diagram***

*Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana alir berakhir. *Activity* *diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

**Tabel 2.4 Simbol *Activity Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | *Activity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. |
| 2. |  | *Action* | State dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3. |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| 4. |  | *Activity Final Node* | Bagaimana objek dibentuk dan diakhiri. |

Sumber: Shalahuddin, M., 2011.

1. ***Sequence Diagram***

*Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu objek dan pesan yang dikirimkan dan diterima antar objek. Jumlah *Sequence diagram* yang akan digambar adalah minimal sama dengan jumlah *use case* yang didefinisikan (Sukamto, 2013).

**Tabel 2.5 Simbol *Sequence Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1 |  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |

**Tabel 2.5 Simbol *Sequence Diagram* (Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| 2 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. |
| 3 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. |
| 4 |  | *Message* | Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy.* |

Sumber: Sukamto dan Shalahuddin, 2013.

1. ***Class Diagram***

*Class diagram* adalah kumpulan objek-objek yang mempunyai struktur umum, *behavior* umum, relasi umum, dan *semantic* atau kata yang umum. *Class-class* ditentukan atau ditemukan dengan cara memeriksa objek-objek dalam *sequence diagram* dan *collaboration diagram*. Sebuah *class* digambarkan seperti sebuah bujur sangkar dengan tiga bagian ruangan.

**Tabel 2.6 Simbol *Class Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2. |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. | |
| 3. |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor. |
| 4. |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |

**Tabel 2.6 Simbol *Class Diagram* (Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| 5. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri. |

Sumber: Sukamto dan Shalahuddin, 2013.

**2.12** ***Entity Relationship Diagram* (ERD)**

*ERD* merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. *ERD* untuk memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan simbol.

*Entity Relationship diagram* *(ERD)* merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *System Analyst* dalam tahap analisis persyaratan proyek pengembangan *system*. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain database relasional yang mendasari sistem informasi yang dikembangkan. *ERD* bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk database. (Brady dan Loonam, 2010).

Elemen-elemen ERD adalah sebagai berikut (Rosa, 2013):

* + - 1. Entitas (*Entity*)

*Entity* adalah obyek yang dapat dibedakan dalam dunia nyata. *Entity set* adalah kumpulan dari *entity* yang sejenis. *Entityset* dapat berupa:

1. Obyek secara fisik seperti Rumah, Kendaraan, Peralatan.
2. Obyek secara konsep seperti Pekerjaan, Perusahaan, Rencana.
   * + 1. Hubungan / Relasi (*Relationship*)

*Relationship* adalah hubungan yang terjadi antara satu atau lebih *entity*. *Relationship set* adalah kumpulan *relationship* yang sejenis.

* + - 1. Atribut

Atribut adalah karakteristik dari *entity* atau *relationship*, yang menyediakan penjelasan detail tentang *entity* atau *relationship* tersebut. Nilai Atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu *entity* atau *relationship*.

* + - 1. Derajat Relasi (*Relationship Degree*)

*Relationship Degree* atau Derajat Relasi adalah jumlah entitas yang berpartisipasi dalam satu *relationship*.

* + - 1. Kardinalitas (*Cardinality*)

Kardinalitas menyatakan jumlah himpunan relasi antar entitias. Terdapat tiga macam kardinalitas relasi yaitu:

* 1. Satu ke satu (*One to One*)

Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B.

* 1. Satu ke banyak (*One to Many*)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B dapat berhubungan dengan satu entitas pada himpunan entitas A.

* 1. Banyak ke banyak (*Many to Many*)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.

**Tabel 2.7** **Simbol *Entity Relationship Diagram***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1 |  | Entitas *(Entity)* | Kumpulan dari objek yang dapat diidentifikasikan secara unik |
| 2 |  | Entitas Lemah  ( *Weak Entity)* | Entitas yang keberadaannya sangat bergantung dengan entitas lain |

**Tabel 2.7 Simbol *Entity Relationship Diagram* (Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| 3 |  | Relasi *(Relationship)* | Hubungan yang terjadi antara satu atau lebih entitas. |
| 4 |  | *Identifying Realtionship* | Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy* |
| 5 |  | Atribut | Unsur penyusun suatu entitas = *field* |
| 6 |  | Atribut *Primary Key* | Atribut yang digunakan untuk menentukan suatu *entity* secara unik dan berbeda |
| 7 |  | Atribut *Multivalue* | Nilai dari suatu attribute yang mempunyai lebih dari satu (multivalue) nilai dari atribut yang bersangkutan |
| 8 |  | Atribut Komposif | Atribut yang terdiri dari beberapa atribut yang lebih mendasar/lebih kecil lagi |
| 9 |  | Atribut Derivatif | Atribut yang dihasilkan dari atribut lain atau dari suatu *relationship* |
| 10 |  | Garis | Penghubung antara relasi dan entitas. |

Sumber: Sukamto dan Shalahuddin, 2013.

**2.13** **Metode Pengembangan Sistem**

Metode Pengembangan sistem yang digunakan pada tugas ini adalah metode *Rational Unified Process* *(RUP). RUP* merupakan salah satu proses rekayasa perangkat lunak yang menyediakan pendekatan untuk menentukan tugas dan tanggung jawab dalam pengembangan suatu organisasi, tujuannya adalah untuk memastikan produksi kualitas tinggi, software memenuhi dengan kebutuhan user sesuai dengan jadwal dan biaya yang telah dirancang. Dalam metode RUP ini, terdiri dari 4 tahap, yaitu:

* 1. *Inception*

Tahap ini membangun *business case* untuk sistem dan membatasi ruang lingkupnya, untuk melakukan hal ini diharuskan untuk mengidentifikasi semua entitas eksternal yang akan berinteraksi dengan sistem, dan mendefinisikan interaksi pada level tertentu. Ini juga termasuk mengidentifikasi semua *use cases* dan menjelaskan beberapa yang signifikan. *Business case* termasuk kriteria keberhasilan, perkiraan resiko, dan mengestimasikan sumber daya yang dibutuhkan.

* 1. *Elaboration*

Tujuan dari tahap *elaboration* adalah menganalisis domain masalah, membuat sebuah dasar arsitektur, membangun rencana proyek, dan mengeliminasi resiko terbesar dari proyek. Untuk menjalankan objek-objek tersebut diperlukan melihat lebih luas dan lebih dalam terhadap sistem. Pada tahap ini merupakan tahap paling sulit karena pada tahap ini memastikan bahwa arsitektur, kebutuhan, dan perencanaan cukup stabil sehingga waktu dan biaya tidak berubah.

* 1. *Construction*

Dalam tahap ini semua komponen dan fitur aplikasi yang dibuat dan di integrasikan kedalam *software*. Dalam tahapan ini juga dituntut untuk mengoptimalkan sumber daya, biaya, jadwal dan kualitas. Pada tahapan ini meliputi bagaimana suatu aplikasi biasa diimplementasikan dan diuji coba.

1. Implementasi

Penjelasan mengenai perangkat keras dan perangkat lunak apa saja yang dibutuhkan untuk mengimplementasikan sistem.

1. *Coding*

Proses pengkodean dilakukan dengan menggunakan bahasa pemrograman. Pengkodean sendiri berisi tahapan-tahapan perhitungan metode

1. *Testing*

Pada tahap ini dilakukan pengujian terhadap aplikasi yang telah dibangun untuk mengetahui tingkat akurasi dan kualitas dari aplikasi tersebut, apakah sudah sesuai dengan yang diharapkan atau tidak. *Testing* dilakukan dengan menguji semua tombol-tombol yang terdapat pada aplikasi apakah sudah berjalan sesuai dengan fungsi nya atau tidak

* 1. *Transition*

Pada tahap ini dilakukan *testing* akhir pada sistem yang telah jadi, kemudian dilakukan sosialisasi penggunaan perangkat lunak yang telah dibangun ke *administrator*.

**BAB III**

**METODOLOGI PENELITIAN**

**3.1 Waktu dan Tempat Penelitian**

**3.1.1 Waktu**

Waktu pelaksanaan penelitian tugas akhir dilaksanakan mulai dari bulan April 2019 sampai dengan Juli 2019. Rincian kegiatan dapat dilihat pada Tabel 3.1 berikut:

**Tabel 3.1** ***Gannt Chart* Waktu Penelitian**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Uraian | Waktu (2019) | | | | | | | | | | | | | | | |
| April | | | | Mei | | | | Juni | | | | Juli | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 |
| 1 | *Inception* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | *Elaboration* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | C*onstruction* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | *Transition* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Studi literatur dilakukan selama 5 minggu yaitu pada minggu ke 3 dan ke 4 bulan Februari, dan dilanjutkan pada bulan Maret smpai minggu ke 3. Studi literatur dilakukan dengan cara membaca artikel, buku, situs dan referensi lainya yang berkaitan dengan pembuatan sistem.
2. Analisis dilakukan selama 4 minggu yaitu pada minggu ke 4 nulan Maret dan dilanjutkan sampai minggu ke 3 bulan April. Analisis dilakkan dengan menganalisis aplikasi yang akan dibuat dengan melihat permasalahan yang terjadi.
3. Desain dilakukan selama 4 minggu yaitu pada minggu 3 bulan April sampai dengan minggu ke 2 bulan Mei. Desain dilakukan dengan memperhatikan hasil perancangan sebelumnya.
4. Pembuatan Program dilakukan pada minggu selanjutnya setelah desain dilakukan yaitu selama 4 minggu pula, dimulai dari minggu ke 3 bulan Mei sampai dengan minggu ke 2 bulan Juni.
5. Pengujian dimulai pada minggu terakhir pembuatan program selama 3 minggu sampai dengan minggu pertama bulan Juli. Tahap ini dilakukan untuk memastikan kesesuaian sistem yang telah dibangun terhadap rancangan awal
6. Pemeliharaan dilakukan dalam 3 minggu terahir dimulai dari minggu pertama bulan Juli sampai dengan minggu ke 3 bulan Juli. Pemeliharaan dilakukan untuk memastikan sistem tidak menemui masalah saat digunakan.

**3.1.2 Tempat Penelitian**

Adapun tempat penelitian tugas akhir yang akan dilakukan tidak mengacu pada tempat tertentu.

**3.2 Metode Pengumpulan Data**

Dalam penelitian ini metode pengumpulan data yang digunakan pada perancangan aplikasi tersebut adalah studi *literatur*, mulai dari buku-buku, jurnal maupun artikel dan sumber-sumber lain. Metode ini dilaksanakan dengan melakukan studi kepustakaan yang relevan. Studi kepustakaan ini dilakukan untuk mencari sumber pelengkap yang berhubungan dengan sistem yang akan dibangun, yaitu dengan mencari referensi yang membahas tentang penggunaan metode *Latent Semantic Analysis* (*LSA*), sehingga metode *LSA* dapat diimplementasikan dalam *website e-learning* Jurusan Teknik Informatika.

**3.3 Metode Pengembangan Sistem**

Metode pengembangan sistem yang digunakan dalam sistem ini adalah metode *Rational Unified Process* (*RUP*). Dalam metode ini, terdapat empat tahap pengembangan perangkat lunak, yaitu:

1. Permulaan (*Inception*)

Pada *fase* ini dilakukan proses pengidentifikasian sistem, dilakukan dengan analisis kebutuhan akan aplikasi, melakukan kajian terhadap penelitian yang terkait dengan metode *Latent Semantic Analysis* (*LSA*).

1. Perluasaan/Perencanaan (*Elaboration*)

Setelah menentukan ruang lingkup penelitian, tahap ini akan dilakukan perancangan dan analisis sistem menggunakan *flowchart* meliputi *flowchart* *flowchart* Sistem *e-learning,* dan metode *flowchart Latent Semantic Analysis* (*LSA*). Pada perancangan ini, digunakan juga UML (*Unified Modelling Language*) yang meliputi *use case diagram,* *activity diagram, class diagram* dan *sequence diagram*.

1. Konstruksi (*Construction*)

Proses yang dilakukan pada tahap ini yaitu membangun aplikasi dengan perancangan yang telah dilakukan sebelumnya, mulai dari tampilan *interface* sampai implementasi rancangan UML. Proses yang juga dilakukan pada tahap ini yaitu penerapan *coding* metode *Latent Semantic Analysis* (*LSA*) pada sistem.

1. Transisi (*Transition*)

Pada tahap *Transition* difokuskan untuk melakukan proses pengujian terhadap aplikasi. Dalam penelitian ini, dilakukan pengujian menggunakan *black box* terhadap aplikasi yang meliputi pengujian pendekteksian *text* terdapat kemiripan dan pengujian pendekteksian *text* tidak terdapat kemiripan.