**Perbandingan Algoritma *String Matching Latent Semantic Analisis* dengan**

**Algortima String Matching *Constraint Based Genetic Algorithm* dalam Pembuatan**

**Essay Auto Grading**

Proposal Tugas Akhir

Disampaikan Sebagai Bagian Dari Persyaratan Kelulusan Sarjana

Program Studi Teknik Informatika



Oleh :

|  |  |
| --- | --- |
| 11S14035 | Michael Arianto |
| 11S15045 | Reikard Napitupulu |
|  |  |

Institut Teknologi Del

2018/2019

# DAFTAR ISI

[DAFTAR ISI i](#_Toc32764222)

[DAFTAR GAMBAR iv](#_Toc32764223)

[DAFTAR TABLE v](#_Toc32764224)

[BAB 1 PENDAHULUAN 1](#_Toc32764225)

[1.1 Latar Belakang 1](#_Toc32764226)

[1.2 Perumusan Masalah 3](#_Toc32764227)

[1.3 Tujuan Penelitian 4](#_Toc32764228)

[1.4 Ruang Lingkup Penelitian 4](#_Toc32764229)

[Adapun lingkup pembuatan sistem auto grading ini adalah : 4](#_Toc32764230)

[1.5 Metode Penelitian 4](#_Toc32764231)

[1.6 Sistematika Penyajian 5](#_Toc32764232)

[BAB 2 Tinjauan Pustaka 7](#_Toc32764233)

[2.1 Penilaian (Grading) 7](#_Toc32764234)

[2.2 Auto Grading 8](#_Toc32764235)

[2.3 Algoritma 9](#_Toc32764236)

[2.3.1 Constraint-based Genetic Algorithm 9](#_Toc32764237)

[2.3.2 String matching Algorithm 13](#_Toc32764238)

[2.3.1 Latent Semantic Analysis 14](#_Toc32764239)

[2.4 WordNet 14](#_Toc32764240)

[2.4.1 Sejarah Wordnet 15](#_Toc32764241)

[2.4.2 Struktur Database Wordnet 15](#_Toc32764242)

[2.5 Tokenizing 18](#_Toc32764243)

[2.6 Case Folding 18](#_Toc32764244)

[2.7 Stemming 19](#_Toc32764245)

[2.8 Fitness Score 21](#_Toc32764246)

[2.9 Black Box Testing 21](#_Toc32764247)

[BAB 3 ANALISIS MASALAH DAN ANALISIS ALGORITMA 23](#_Toc32764248)

[3.1 Analisis Masalah 23](#_Toc32764249)

[3.1.1 Penilaian Otomatis 23](#_Toc32764250)

[3.2 Analisis Algoritma dan Pengambilan Data 24](#_Toc32764251)

[3.2.1 Pengambilan Data 24](#_Toc32764252)

[3.2.2 Constraint-Based Genetic 24](#_Toc32764253)

[3.2.2.1 Mutasi 25](#_Toc32764254)

[3.2.2.2 Delesi 27](#_Toc32764255)

[3.2.2.3 Persilangan 27](#_Toc32764256)

[3.2.2.4 Insersi 28](#_Toc32764257)

[3.2.3 Sistem Penilaian 29](#_Toc32764258)

[3.2.4 Rancangan Eksperimen 31](#_Toc32764259)

[3.2.5 Permasalahan yang Mungkin Muncul 32](#_Toc32764260)

[BAB 4 Rancangan 32](#_Toc32764261)

[4.1 Rancangan Prototype 32](#_Toc32764262)

[4.1.1 Proses kerja Prototype 33](#_Toc32764263)

[4.1.1.1 Preprocessing 33](#_Toc32764264)

[4.1.1.2 Mutation 34](#_Toc32764265)

[4.1.1.3 Linked List 34](#_Toc32764266)

[4.1.1.4 Crossing 34](#_Toc32764267)

[4.1.2 Class Diagram 36](#_Toc32764268)

[4.1.3 Desain 37](#_Toc32764269)

[4.1.4 Sequence Diagram 38](#_Toc32764270)

[4.1.5 Desain mock up layout 39](#_Toc32764271)

[4.2 Rancangan Pengujian 43](#_Toc32764272)

[4.3 Implementasi Prototype 43](#_Toc32764273)

[4.3.1 Hasil Implementasi Desain 43](#_Toc32764274)

[4.3.2 Lingkungan Implementasi 46](#_Toc32764275)

[4.3.3 Batasan Prototype 46](#_Toc32764276)

[BAB 5 Eksperimen dan Pembahasan 47](#_Toc32764277)

[5.1 Pengujian 47](#_Toc32764278)

[5.1.1 Tujuan penelitian 47](#_Toc32764279)

[5.1.2 Data pengujian 47](#_Toc32764280)

[Lampiran 49](#_Toc32764281)

[DAFTAR REFERENSI 58](#_Toc32764282)

# DAFTAR GAMBAR

[Gambar 1 .*Genetic Flow* 10](#_Toc13502721)

[Gambar 2 .*String Matching Flow* 13](#_Toc13502722)

[Gambar 3. Skema mutasi 27](#_Toc13502723)

[Gambar 4. Skema proses Algoritma genetic pada *Auto grading* yang akan diimplentasikan 29](#_Toc13502724)

[Gambar 5.Usecase 33](#_Toc13502725)

[Gambar 6. Sequence Diagram Dosen 34](#_Toc13502726)

[Gambar 7.Sequence Diagram Mahasiswa 34](#_Toc13502727)

[Gambar 8.Class Diagram 41](#_Toc13502728)

[Gambar 9. Tampilan halaman Login 42](#_Toc13502729)

[Gambar 10.Tampilan halaman menu Dosen 43](#_Toc13502730)

[Gambar 11. Tampilan halama Upload soal dan jawaban oleh Dosen 43](#_Toc13502731)

[Gambar 12.Tampilan Halaman nilai Dosen 44](#_Toc13502732)

[Gambar 13. Tampilan halaman jawaban mahasiswa 44](#_Toc13502733)

[Gambar 14. Tampilan halaman nilai yang diakses mahasiswa 45](#_Toc13502734)

# DAFTAR TABLE

[Table 1.Database Wordnet 16](#_Toc13502840)

[Table 2.Contoh tokenizing 18](#_Toc13502841)

[Table 3. Contoh Case Folding 19](#_Toc13502842)

[Table 4. Contoh Stemming 19](#_Toc13502843)

[Table 5. Contoh Name Entity Recognition 22](#_Toc13502844)

[Table 6. Penilaian Essay Auto Grading 30](#_Toc13502845)

# PENDAHULUAN

Pada bab ini dapat ditemukan hal yang terkait dengan latar belakang Tugas Akhir, tujuan pemilihan topik untuk Tugas Akhir, lingkup penelitian yang dilaksanakan dalam pelaksanaan Tugas Akhir, pendekatan yang dilakukan pada Tugas Akhir, dan sistematika penyajian hasil penelitian Tugas Akhir

## Latar Belakang

Pada proses pembelajaran, evaluasi terhadap pencapaian hasil belajar pelajar adalah satu faktor yang sangat penting. Evaluasi ini biasanya dilakukan dalan pengujian, atau biasa disebut ujian, yang biasanya memiliki 2 (dua) jenis soal yaitu esai dan pilihan berganda (Isti Nafah, 2010). Pengujian dengan soal berformat pilihan berganda dikritik secara luas, walaupun tetap diadopsi, karena pelajar bisa saja menebak secara beruntung jawaban. Pada format ini juga pelajar lebih mudah untuk mencontek jawaban dari pelajar lain. Format ini juga mungkin membuat kemampuan menulis dari pelajar berkurang. Soal esai, di sisi lain, dapat lebih menggambarkan pemahaman dari pelajar secara lebih luas dan dalam daripada pilihan berganda, tetapi lebih sulit untuk diperiksa menggunakan komputer (Steele 1997). ( Khalid Nahar and Izzat Alsmadi)

Penilaian pada pengujian yang telah dilakukan, biasanya dilakukan secara manual oleh pengajar. Tetapi hal ini dapat memakan waktu yang sangat lama dikarenakan beberapa faktor, salah satu yang paling memepengaruhi adalah rasio jumlah pelajar terhadap pengajar. Pada saat ini, rasio jumlah pelajar terhadap jumlah pengajar sangat berat sebelah ke arah jumlah pelajar. Menurut Dinas Pendidikan dan Kebudayaan, rasio ideal untuk jumlah pelajar dan pengajar seharusnya tidak melebihi 20:1. Tetapi menurut data *Analytical and Capacity Development Partnership (ACDP)*, rasio pengajar dan pelajar telah mencapai 26:1 untuk daerah Asia. Hal ini menimbulkan masalah dalam proses belajar-mengajar terutama pada saat pemeriksaan lembar jawaban ujian. Dengan menggunakan rasio yang ada, yaitu 26:1, ini berati satu orang pengajar harus memeriksa lembar jawaban dari sekitar 26 orang pelajar. Hal ini menyebabkan ketidak efisiensian dalam aspek waktu dan tenaga yang dikeluarkan, dimana masalah ini menyebabkan gangguan pada proses belajar-mengajar dalam aspek waktu, yaitu laporan nilai hasil pembelajaran akan terlambat pengedarannya. Hal ini berdampak pada pengajar karena pengajar harus mengejar *deadline* publikasi nilai pada saat proses pemeriksaan lembar jawaban pelajar. Menurut Dr. Peng Li dan Mr. Lee Toderick dari East Carolina University , bagi pelajar, kesempatan untuk memperbaiki kesalahannya yang dilakukan oleh pelajar tersebut dalam sebuah pengujian sangat penting karena belajar dari kesalahan adalah salah satu metode pembelajaran yang sangat efektif. Akan tetapi, pelajar tidak memeiliki kesempatan tersebut karena penilaian manual oleh pengajar memakan waktu yang banyak. Untuk menyelesaikan masalah ini, dikembangkanlah sistem *autograder* atau sistem *automated grading* yang dinamakan BroadReachExtended. (Dr. Peng Li, Mr. Lee Toderick, East Carolina University, 2015)

Untuk mengatasi permasalahan ini, dilakukan pengujian secara *online*, dimana pengujian secara *online* menggunakan program *automatic grading* yang diharapkan dapat mempercepat proses penilaian dari evaluasi atau pengujian yang telah dilakukan. Pada pengujian *online* pelajar juga diuji untuk memahami penggunaan komputer dan jaringan komputer sesuai dengan evolusi pembelajaran yang sudah masuk pada era *e-learning.* Pengujian ini lebih praktis dan lebih hemat dalam segi biaya, penyimpanan yang dibutuhkan dan juga menghemat penggunaan kertas karena pengujian ini dapat dilakukan secara lokal dan tidak membutuhkan biaya untuk kertas dan alat tulis. (Ikdam Alhami, Izzat Alsmadi, 2011)

Pengujian secara onlineberformat pilihan berganda lebihmudah untuk dinilai menggunakan *auto grading* daripada penilaian terhadap pengujian secara esai menggunakan *auto grading*. (Steele 1997). (Khalid Nahar and Izzat Alsmadi). *Auto grading* yang digunakan pada pengujian secara online saat ini pada umumnya menggunakan *string-based similiarity* atau yang disebut juga *string matching* yang mencocokkan kunci jawaban yang ditentukan oleh pihak pengajar pada esai (Abdulaziz Shehab, Mahmoud Faroun, Magdi Rashad). Metode *string matching* ini memang efektif, tetapi membatasi penggunaan kosa kata oleh pelajar. Metode ini hanya menerima jawaban yang sama seperti kunci jawaban yang diberikan oleh pengajar sebagai jawaban yang benar. Jawaban lain akan dikategorikan salah, terlepas dari seberapa tinggi *similiarity* jawaban dengan kunci jawaban. Tetapi pada bidang genetika, ada metode yang mencari tingkat kesamaan antar 2 (dua) kromosom menggunakan algoritma *constraint based genetic* atau disebut juga algoritma genetik*.*Kami memprediksi bahwa menggunakan algoritma dan metode ini, dapat dibuat sebuah program *essay auto grading* yang dapat menilai kebenaran suatu jawaban berdasarkan kesamaan jawaban tersebut dengan kunci jawaban yang diberikan. Sampai sekarang, belum ada penelitian yang membuat *essay auto grading* menggunakan algoritma *constraint-based* serta melakukan perbandingan antara *essay auto grading* menggunakan algoritma tersebut dengan *essay auto grading* menggunakan algoritma *exact string matching* atau biasa disebut algoritma *string matching.* Dan oleh karena itu kami ingin membandingkan performa *auto grading* yang menggunakan kedua algoritma tersebut dalam segi keefektivitasan performa dari kedua *auto grading.*

## Perumusan Masalah

1. Apakah dapat dibuat sebuah program *essay auto grading* yang menilai jawaban berdasarkan *similiarity* jawaban dengan kunci jawabanmenggunakan algoritma *constraint based genetic* atau algoritma genetik?

2. Apakah performa *essay auto grading* menggunakan algoritma *constraint-based* lebih efektif daripada performa *essay auto grading* yang menggunakan algoritma *string matching?*

## Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah:

1. Pembuatan sebuah program *auto grading* yang menggunakan algoritma *constraint-based genetic*.
2. Membandingkan keefektifan dan efisiensi performa *essay auto grading* menggunakan algoritma *constraint-based* dan *essay auto grading* yang menggunakan algoritma *string matching.*

.

## Ruang Lingkup Penelitian

## Adapun lingkup pembuatan sistem auto grading ini adalah :

1. Penggunaan algoritma cb *(constrain based)*, dimana algoritma ini menggunakan pendekatan *search and scoring* dan *dependency analysis* dalam pembuatan program *short answer essay auto grading.*

2. Penelitian ini menghasilkan perbandingan keefektifan performa antara *autograding* menggunakan algoritma *string matching* dan *auto grading* menggunakan algoritma *contraint-based*

## Metode Penelitian

Metode Penelitian yang akan digunakan adalah dengan menggunakan metode kuantitatif dalam hal mengembangkan dengan pendekatan baru

1. Studi Literatur

Studi literatur diperoleh melalui *website*, jurnal, *paper*, buku, dan artikel yang mana data yang diperoleh berupa hal-hal yang mendukung untuk penelitian Tugas Akhir ini seperti informasi yang berhubungan dengan program.

1. Implementasi

Mengimplementasikan essay *auto grading* dengan menggunakan metode *constraint based*.

1. Analisis Pembangunan *Auto grading*

Analisis pada sistem *essay autograding* dilakukan untuk memastikan bahwa model yang dibangun dapat digunakan.

1. Perbandingan

Dari data yang diperoleh dari pengujian langsung maupun dari sumber lain, data tersebut akan digunakan untuk membandingkan metode *constraint based* dengan metode *string matching*.

1. Analisis Penelitian

Data hasil perbandingan yang dilakukan kemudian dianalisis kembali agar data yang diperoleh semakin akurat dan mencapai tujuan dari penelitian ini dilakukan.

1. Pembuatan Dokumen

Hasil dari perbandingan dan data – data terkait didokumentasikan sebagai bukti dari penelitian tersebut pernah dilakukan.

## Sistematika Penyajian

Pada dokumen ini dapat ditemukan mulai dari latar belakang hingga hasil akhir penelitian yang disajikan dalam bentuk beberapa Bab. Adapun Bab tersebut adalah sebagai berikut:

1. Pada Bab 1 dapat ditemukan informasi yang berhubung dengan alasan kenapa penelitian ini dilakukan hingga bagaimana data penelitian tersebut disajikan.
2. Pada Bab 2 dapat ditemukan informasi yang berhubungan dengan tinjauan pustaka yang mana menguraikan beberapa deskripsi yang berhubungan dengan penelitian yang dilakukan.
3. Pada Bab 3 dapat ditemukan informasi yang menyangkut analisis dan desain dari penelitian yang dilakukan penulis. Disini juga dapat ditemukan bagaimana cara penulis memperoleh hasil analisis dan implementasi dari penelitian ini.
4. Pada Bab 4 dapat ditemukan informasi yang menyangkut dengan implementasi dan pengujian dari dua *framework* tersebut.
5. Pada Bab 5 dapat ditemukan informasi yang menyangkut dengan hasil dan pembahasan dari analisis dan implementasi dua *framework* tersebut.
6. Pada Bab 6 dapat ditemukan informasi berupa kesimpulan dan saran dari penulis terkait dari penelitian ini.

# Tinjauan Pustaka

Pada bab ini dapat ditemukan tinjauan pustaka yang berhubungan dengan Tugas Akhir ini. Tinjauan pustaka yang disajikan berdasarkan referensi yang dicantumkan pada dokumen ini.

## Penilaian (Grading)

Penilaian adalah komponen yang sangat penting dalam sebuah proses pembelajaran karena melalui penilaian, peserta didik dapat mengetahui pencapaian hasil dari proses pembelajaran yang sudah dilakukannya terhadap hasil yang sudah ditetapkan (Mustofa Abi Hamid, 2016).

Penilaian otentik merupakan pernilaian yang mampu memfasilitasi peserta didik untuk menggunakan kombinasi dari kompetensi pengetahuan, keterampilan dan sikap yang diambil dalam mengaplikasikan sesuatu yang dibutuhkan dalam kehidupannya(Rina Melly Suciyati, Nurhaida, dan Linda Vitoria, 2017), di mana karakteristik yang harus dimiliki oleh penilaian otentik, sebagaimana yang dikemukakan oleh Nurhadi (dalam Daryanto dan Sudjendro, 2014 : 90-91), itu sendiri adalah sebagai berikut :

a. Melibatkan pengalaman nyata (*involves real-word experience*)

b. Dilaksanakan selama dan sesudah proses pembelajaran berlangsung.

c. Mencakup penilaian pribadi (*self assessment*) dan refleksi.

d. Yang diukur keterampilan dan performansi, bukan mengingat fakta.

e. Berkesinambungan.

f. Terintegrasi.

g. Dapat digunakan sebagai umpan balik.

h. Kriteria keberhasilan dan kegagalan diketahui siswa dengan jelas.

Kriteria-kriteria tersebut didukung oleh (Anderson,2003), yang mengungkapkan bahwa hasil belajar dari peserta didik harus bersifat bermakna, transparant dan adil.

Namun, metode penilaian secara manual memiliki beberapa kendala dalam pelaksanaannya. Penilaian untuk pertanyaan yang bersifat subjektif biasanya menggunakan persepsi, pengertian dan interpretasi dari pengajar sehingga adanya beberapa kata yang dipilih sebagai kata kunci untuk jawaban yang benar. Pada saat ujian, penilaian dapat menjadi proses yang cukup berat karena pengajar harus memeriksa lembar jawaban dalam jumlah yang sangat banyak, dimana setiap lembar jawaban memiliki gaya tulisan yang beragam mulai dari yang baik sampai yang hampir tidak terbaca. Hal ini tidak dapat dkategorikan sebagai proses yang efektif karena memakan waktu yang lama.( Khalid Nahar and Izzat Alsmadi)

## Auto Grading

*Auto grading system* adalah sistem yang dibuat untuk melakukan penilaian(*grading)* secara otomatis yang digunakan untuk mengecek kebenaran dari pekerjaannya secara interaktif, dimana nilai untuk solusi atau jawaban yang benar sudah disimpan secara otomatis. (Urs von Matt, 1994)

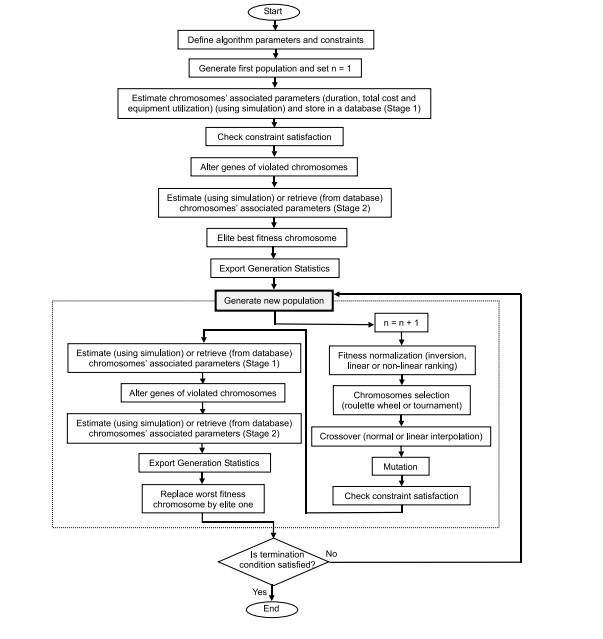
Pengujian dengan soal berformat pilihan berganda mudah untuk dinilai menggunakan komputer. Format pertanyaan ini dikritik secara luas, walaupun tetap diadopsi, karena pelajar bisa saja menebak secara beruntung jawaban. Pada format ini juga pelajar lebih mudah untuk mencontek jawaban dari pelajar lain . format ini juga mungkin membuat kemampuan menulis dari pelajar berkurang. Soal esai, di sisi lain, dapat lebih menggamparkan pemahaman dari pelajar secara lebih luas dan dalam daripada pilihan berganda, tetapi lebih sulit untuk diperiksa menggunakan komputer(Steele 1997). ( Khalid Nahar and Izzat Alsmadi)

Automated Essay Scoring (AES) didefinisikan sebagai computer technology yang dapat mengevaluasi dan menuliskan nilai (Shermis & Barrera2002; Shermis & Burstein, 2003; Shermis, Raymat, & Barrera, 2003). Tujuan untuk menggunakan AES untuk memangkas beberapa isu yang terkandung pada saat melakukan ujian tertulis dimana isu tersebut adalah: waktu,biaya,kemampuan secara umum dan keterpercayaan (Burstein, 2003;Chung & O’Neil,1997; Hamp-Lyons,2001;Ellis Batten Page,2003;).  Keuntungan menggunakan AES telah menarik perhatian publik, sekolah, universitas, dan peneliti (Burstein et al., 1998; Shermis &Burstein, 2003; Sireci & Rizavi, 2000). Beberapa keuntungan ini adalah: meringankan beban penilaian dari para pendidik dan menambahkan tingkat yang konsisten yang terkadang tidak dapat dicapai oleh pendidik (Shermis & Barrera, 2002).

## Algoritma

### Constraint-based Genetic Algorithm

Algoritma genetika adalah algoritma komputasi yang diinspirasi teori evolusi Darwin yaitu “yang kuat adalah yang menang” yang kemudian diadopsi menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan cara yang lebih “alamiah”. Salah satu aplikasi algoritma genetika adalah pada permasalahan optimasi kombinasi, yaitu mendapatkan suatu nilai solusi optimal terhadap suatu permasalahan yang mempunyai banyak kemungkinan solusi. (Hermawanto, 2003). Algoritma genetik adalah sebuah fungsi optimisasi yang digunakan untuk mencari solusi optimum dari nilai maksimum dan minimum sebuah fungsi (Jenna Carr, 2014). Berikut adalah alur umum dari algoritma genetik :



Gambar 1 .*Genetic Flow*

Dimana menurut (Rahul Malhotra, Narinder Singh dan Yaduvir Singh), alur tersebut dapat didefinisikan dalam beberapa langkah sebagai berikut:

Step I [Start] *Generate* sebuah populasi kromosom yang cocok dijadikan solusi permasalahan.

Step II [Fitness] Evaluasi nilai *fitness* dari setiap kromosom

Step III [New population] Membuat generasi populasi yang baru dengan mengikuti langkah-langkah berikut dan mengulangnya sampai populasi baru terbentuk dengan menggunakan kromosom yang dimodifikasi.

1. [Selection] Pilih 2 (dua) kromosom sebagai “orang tua”. Semakin tinggi nilai fitness suatu kromosom, semakin tinggi kemungkinan kromosom tersebut terpilih.
2. [Crossover] Lakukan persilangan antar “orang tua” untuk membuat kromosom baru atau kromosom anak. Jika tidak ada persilangan yang terjadi, maka dapat disimpulkan kromosom anak sama persis seperti kromosom”orang tua”.
3. [Mutation] Mutasi setiap kromosom anakan dengan mengubah secara acak nilai pada kromosom anak seperti mutasi buatan pada genetika.
4. [Accepting] Masukkan kromosom yang baru ke dalam populasi. Step IV [Replace] Gunakan populasi yang baru untuk menggantikan populasi yang lama

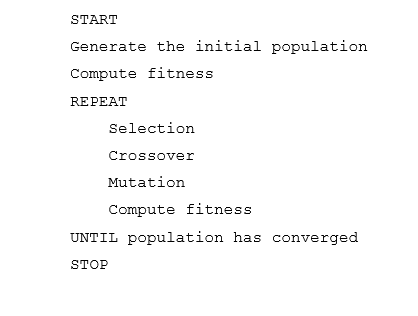
Step V [Test] Jika seluruh *constraint* sudah terpenuhi, maka proses dinyatakan selesai. Jika tidak, maka ulangi proses sampai *constraint* terpenuhi.

Berikut adalah penjelasan secara umum proses pada algoritma genetika :

Mutasi adalah perubahan yang terjadi pada DNA atau kromosom dalam sebuah rangkaian beberapa DNA aatau kromosom. Mutasi menyebabkan munculnya variasi baru pada spesies. Mutasi ini disebakan oleh perubahan yang sangat besar pada lingkungan tempat tinggal dari individu yang bermutasi. Mutasi biasanya terjadi pada basa bernitrogen(ATGC atau AUGC pada RNA) dari susunan gen pada makhluk hidup. Mutasi pada basa bernitrogen terbagi menjadi tiga jenis yaitu Substitusi, insersi dan delesi.

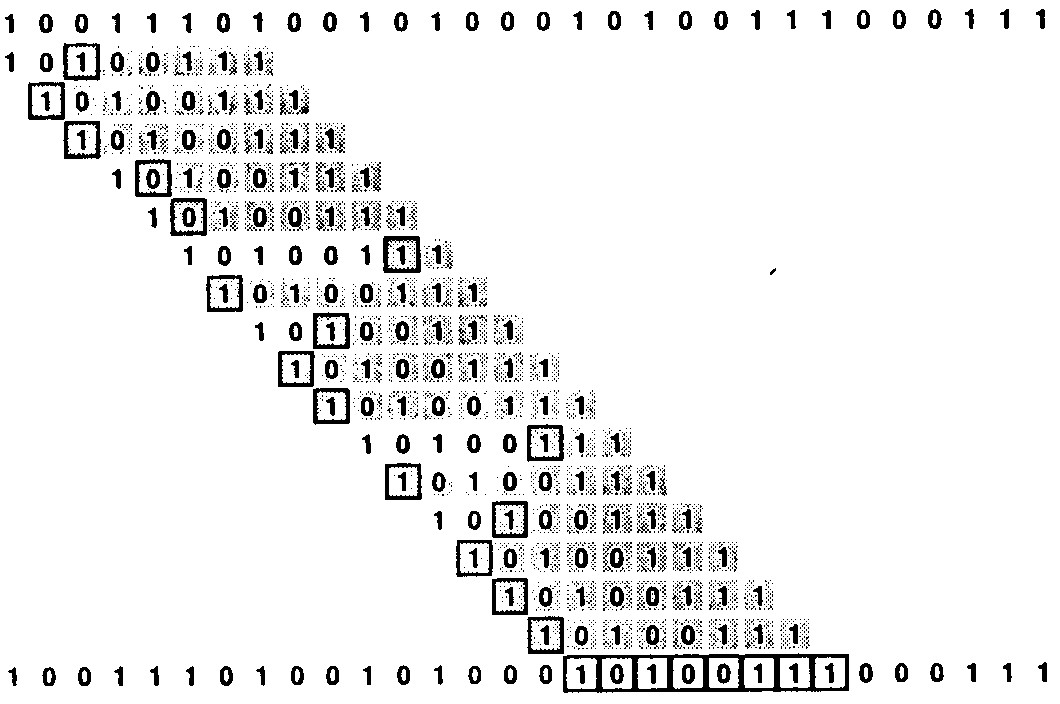
Substitusi terbagi menjadi dua bagian yaitu mutasi transisi, dimana beberapa basa bernitrogennya bertukar tempat, dan mutasi transversi, dimana basa bernitrogen berubah secara individual menjadi basa bernitrogen tetapi memiliki jenis lain.

Pada umumnya, pseudocode untuk logika dari jalannya algoritma ini adalah :



### String matching Algorithm

*String matching algorithm* adalah algoritma yang digunakan untuk mencari suatu pola kumpulan string secara persis dalam sebuah grup besar *text*. ( Koloud Al-Khamaiseh, Shadi ALShagarin, 2014). Algoritma ini biasanya digunakan dalam *spell checkers system,* sistem *filter* dan deteksi spam, sistem pendeteksi gangguan,mesin pencari (*sarch engine)*, *DNA squencing system, digital forensic* dan data mining. *Exact string matching* akan mencari pola dari string yang diberikan secara identik pada text tujuan dan jika tidak identik, maka algoritma ini akan menyatakan tidak ada pola string tersebut pada text. (Kapil Kumar Soni, Rohit Vyas, Amit Sinhal, 2014). *String matching* pada umumnya berjalan menggunakan metode *brute force,* dimana akan dilakukan pengecekan untuk mencari *string* yang dituju pada teks yang diberikan dengan mencarinya di setiap kemungkinan yang ada seperti pada gambar



Gambar 2 .*String Matching Flow*

Pada gambar dapat dilihat bahwa *string* yang ditentukan sebagai kunci akan dicari pada setiap kemungkinan yang ada dengan mengecek setiap kombinasi yang mungkin pada teks yang diberikan.

### Latent Semantic Analysis

*Latent Semantic Analysis* (LSA) adalah sebuah teori dan metode untuk mengekstrak dan merepresentasikan penggunaan kontekstual arti kata-kata pada korpus teks yang besar dengan menggunakan perhitungan statistik (Landauer, et al., 1998). LSA dapat juga didefensikan sebagai sebuah perkiraan hubungan matriks dengan peringkat yang lebih rendah menggunakan teknik SVD. Merupakan salah satu metode yang dipakai pada himpunan dokumen yang banyak dan terstruktur unutk mengekstraksi dan mewakili penggunaan arti kata dengan penggunaan statistic dan aljabar linier. Pada tauhun 1990 Scoot Deerwester ,dkk memperkenalkan algoritma yang digunakan untuk dapat melakukan indexing kata pada dokumen dan memplot ke basis vector. Vektor digunakan untuk menangkap inti sari(semantic) dari suatu kalimat. Salah satu fungsi utama Latent Semantic Analysis ini adalah untuk menghitung kemiripan (similarity) kalimat dengan membandingkan representasi setiap vector dalam kalimat, LSA akan membentuk sebuah matriks yang memppresentasika hubungan antara term dan kalimat yang dinamakan semantic space.

## WordNet

WordNet merupakan sebuah sumber daya leksikal online yang berguna untuk memetakan kata kerja, kata benda dan kata sifat kedalam sebuah himpunan sinonim (*Milller, et al, 1993*). Kata-kata ini dikumpulkan secara bersama berdasarkan makna mereka kedalam sinonim set, atau yang sering disebut sebagai synsets. Setiap contoh dari sebuah kaya yang terdapat pada synset ditandai dengan terpisah dengan makna kata itu. Contoh synset atau kumpulan sinonim adalah {car, auto, automobile, machine, motorcar}. Wordnet telah digunakan dalam sebuah proyek pengembangan analisis teks bermakna ambigu dengan menggunakan Wordnet pada tahun 1998.

### Sejarah Wordnet

Wordnet dikembangkan sejak tahun 1985 di laboratorium Cognitive Science Princeton University dan saat ini pengembangannya diteruskan oleh Christiane Fellbaum. Christiane Fellbaum dianugerahi penghargaan Antonio Zampolli Prize atas kontribusinya terhadap Wordnet.

Versu terbaru Wordnet dalam Bahasa inggris adalah versi 3.0(dirilis pada bulan desember 2016). Pada saat ini, Wordnet dapat digunakan dengan bebas baik untuk tujuan pengembagan maupun untuk tujuan komersil sesuai dengan lisensi yang telah diterbitkan (*Fellbaum,2005*).

Wordnet Bahasa telah dikembangkan oleh Lab Information Retrieval Universitas Indonesia pada tahun 2012 dan oleh Massachusetts Institute of technology pada tahun 2008.

### Struktur Database Wordnet

Wordnet bahasa Inggris membedakan antara kata benda, kata kerja, dan kata keterangan. Setiap Bahasa memiliki tata bahasai yang berbeda sehingga struktur *database* untuk setiap bahasa berbeda.Wordnet terdiri dari kata-kata yang memiliki hubungan semantik. Hubungan semantic dalam database Wordnet dibagi berdasarakan jenis kata, meliputi:

1. Kata Benda

* Hypernym : X adalah hypernym dari Y jika setiap Y adalah (*kind of*) X (wangi adalah hypernym dari aroma)
* Hyponym : X adalah hyponym dari Y jika setiap X adalah (*kind of*)Y(aroma adalah hyponym dari wangi)
* *Coordinate Terms* : X adalah coordinate terms dari Y jika Y dan X memiliki hubungan hypernym (orangutan coordinate term dari monyet, dan monyet adalah coordinate term dari orangutan )
* Meronym :X adalah meronym dari X dan Y adalah bagian dari X(Pintu adalah meronym dari bangunan)
* Holonym :X adalah holonym dari Y jika dalam X ada Y, Y adalah bagian dari Y(bangunan adalah holonym pintu)

1. Kata Kerja

* Hypernym :kata kerja X adalah hypernym dari kata kerja Y jika aktivitas Y adalah melakukan X (memandang hyoernym dari melihat)
* Troponym : kata kerja X adalah troponym dari kata kerja Y jika aktivitas X melakukan Y dalam beberapa cara(berbisik adalah troponym dari berbicara)
* Entailment : kata kerja X yang terkangung oleh Y jika dengan melakukan Y anda melakukan X (tidur adalah entailment dari mendengkur)
* Coordinate Terms : kata kerja yang memiliki hubungan hypernum (memandang dan melihat)

Wordnet bahasa sudah dikembangkan oleh Lab Information Retrieval Fakultas Ilmu Komputer Universitas Indonesia dan MIT. Pengembangan yang dilakukan menggunakan pendekatan expand approach sehingga struktur dari wordnet bahasa menyerupai struktur dari wordnet yang dikembangkan oleh Princeton University.

|  |  |  |  |
| --- | --- | --- | --- |
| Synset | Tlang | tgoodness | Tlemma |
| 00001740-a | B | L | Boleh |
| 00001740-n | B | O | Hakikat |
| 00001740-r | I | O | Akapela |
| 00001740-v | B | O | Menarik |
| 00002098-a | B | L | Tidak berdaya |

Table 1.Database Wordnet

1. Synset

Synset merupakan indentifier yang menandakan keterhubungan satu kata dengan kata lainnya synset memiliki struktur penulisan sebagai berikut:

* Synsey\_oggset adalah 8 digit decimal integer
* Lex\_filenum adalah 2 digit terakur dari synset offset yang menandakan hubungan dengan kata lain apabila lex\_filenum sama
* Ss\_type satu karakter yang mengindikasikan tipe synset yaitu: n:NOUN v:VERB a:ADJECTIVE s:ADJECTIVE SATELLITE r:ADVERB

1. tlang

Merupakan identifier dari bahasa pada kata dalam Wordnet, yaitu:

* B :Bahasa(Generalisasi dari bahasa Indonesia dan Malaysia)
* I :Indonesia
* M :Malaysia

1. tgoodness

Merupakan indetifier dari kualitas kata berdasarkan hasil pengecekan pada saat pendaftaran kata ke dalam Wordnet.

Kualitas kata pada Wordnet didefinisikan sebagai berikut:

* Y : Diverifikasi secara manual dan kualitasnya baik
* O : Diverifikasi secara otomatis dan kualitasnya baik
* M : Diverifikasi secara otomatis dan kualitasnya sedang
* L : Diverifikasi secara otomatis dan kualitasnya rendah
* X : Diverifikasi secara manual dan kualitanya rendah

1. tlemma

Merupakan kata yang disimpan pada Wordnet

## Tokenizing

Tokenizing merupakan tahap pemotongan *string input* berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan melakukan *scan* kalimat dengan *white space* (space, tab dan *newline*) (Mustaqfiri, 2016) sehingga setiap huruf yang diikuti oleh white space akan secara otomatis terbentuk kata tunggal baru.  
Contoh :

|  |  |
| --- | --- |
| Data Input | Hasil Tokenizing |
| kapan kamu masuk kuliah | Kapan  kamu  masuk  kuliah |

Table 2.Contoh tokenizing

Dari contoh tersebut dapat telihat bahwa data input merupakan kalimat input hasil dari proses case folding. Sedangkan hasil tokenizing kalimat input telah dipotong-potong berdasarkan tiap kata yang menyusunnya. Sehingga setiap kata telah terbebas dari *white space*

## Case Folding

Case folding merupakan tahapan untuk mengubah huruf dalam dokumen menjadi huruf kecil atau huruf capital semua. Case folding digunakan karena tidak semua dokumen teks konsisten dalam menggunakan huruf kapital. Pada tahapan ini juga menghilangkan tanda baca dan angka sebagai delimiter (pembatas), pada proses case folding hanya menggunakan huruf a sampai z (mustaqhfiri,2016)

Contoh :

|  |  |
| --- | --- |
| Data Input | Hasil Case Folding |
| Kapan kamu masuk kuliah? | kapan kamu masuk kuliah |

Table 3. Contoh Case Folding

Dari contoh di atas, data input merupakan kalimat input dari pengguna. Sedangkan hasil case folding kalimat input yang telah diubah menjadi huruf kecil dan tanpa tanda baca atau angka selain karakter huruf a sampai z.

## Hash Map

Hash Map adalah sebuah *class* yang memiliki nilai dan kunci yang saling berpasangan, pada java hash map lebih sering disebut sebagai array asosiatif, nilai pada hash map biasanya berupa string, integer, Boolean, float, dan objek. Sedangkan untuk key berupa string ataupun integer. Hash map juga merupakan sebuah struktur data yang mengimplementasikan data yang bertipe abstrak array yang akan memetakan sebuah sebuah kunci terhadap nilai.

## Stemming

*Stemming* merupakan suatu proses pengambilan kata dasar atau akar kata (stem) berbagai bentuk suatu kata (Tala, 2003). Tujuan dari stemming ini untuk menghilangkan imbuhan-imbuhan yang terdapat pada suatu kata. Teknik stemming juga diperlukan untuk memperkecil jumlah indeks yang berbeda dari suatu dokumen dan mekalukan pengelompokan kata-kata lain yang memiliki kata dasar dan arti yang serupa memiliki bentuk berbeda mendapakan imbuhan yang berbeda.

|  |  |
| --- | --- |
| Data input | Hasil stemming |
| Terpaksa  Kebakaran | Paksa  Bakar |

Table 4. Contoh Stemming

hasil stemming merupakan sebuah kata dasar dari suatu kata yang diberi imbuhan.

Algoritma untuk proses stemming untuk teks able Indonesia yaitu algoritma Nazief dan andriani. Tahapan proses stemming menggunakan algoritma nazief & andriani :

1. Cari kata yang akan di stemming dalam database. Jika ditemukan maka diasumsikan bahwa kata tersebut adalah root word. Maka algoritma berhenti.
2. Inflection suffixes (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa particles (“-lah”, “-kah”, “-tah”, atau “-pun”) maka langkah ini diulangi lagi untuk menghapus ablesive pronouns (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus derivation suffixes (“-i”, “-an”, atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka lakukan langkah 3a.
4. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan, maka lakukan langkah 3b.
5. Akhiran yang dihapus (“-i”, “-an”, atau “-kan”) dikembalikan, lanjut ke langkah 4.
6. Hapus derivation prefix. Jika pada langkah 3 ada sufiks yang dihapus, maka lakukan ke langkah 4a. Jika tidak, lakukan langkah 4b.
7. Periksa able kombinasi awalan-akhiran yang tidak diizinkan. Jika ditemukan maka algoritma berhenti, jika tidak lakukan langkah 4b.
8. For = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika root word belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
9. Melakukan re-coding

Jika semua langkah telah selesai tetapi tidak juga berhasil, maka kata awal diasumsikan sebagai kata dasar dan proses *stemming* selesai (Agusta, 2009).

## Fitness Score

*Fitness score* pada algoritma genetika adalah tingkat kesamaan dari suatu kromosom pada populasi terhadap kromosom yang dianggap ideal oleh sistem yang dapat dilihat dari kesamaan basa bernitrogen yang ada kedua kromosom (Patel, 2018). *Fitness score* ini yang akan menjadi alat ukur untuk nilai yang didapatkan oleh pelajar pada setiap soal. Pada dasarnya, *fitness score* merupakan banyaknya perbedaan yang ada pada kromosom pada populasi dan kromosom yang ideal menurut sistem.

*Fitness Score* pada umumnya memiliki konsep hitungan :

***)***

dimana *number of char correct* adalah jumlah basa bernitrogen pada kromosom yang sesuai dengan basa bernitrogen yang ada pada kromosom ideal dan *total number of char* adalah total dari basa bernitrogen yang ada pada kromosom.

## Black Box Testing

*Black box testing* merupakan sebuah pengujian yang digunakan hanya untuk melihat dan output melalui data uji dan memeriksa proses fungsional dari perangkat lunak (Mustaqbal,2016). *Black box testing* tidak memperhatikan dari setiap proses detail yang dilakukan. Sehingga *black box testing* dapat memilih subset test secara efektif dan efisien dalam medeteksi *bug. Black box testing* juga dapat memebantu untuk memaksimalkan proses *testing investment*. Pada penelitian ini *black box testing* digunakan untuk membandingkan hasil penilaian yang dilakukan oleh prototype dengan hasil penilaian yang dilakukan oleh para ahli. Metode *black box testing* memungkinkan perekayasa perangkat lunak untuk mendapatkan serangkaian kondisi input yang seluruhnya menggunakan seluruh persyaratan fungsional dari suatu program.

Black box dapat menemukan kesalahan dalam beberapa kategori berikut:

* + - 1. Fungsi-fungsi yang tidak benar atau hilang.
      2. Kesalahan *interface*.
      3. Kesalahan dalam struktur data atau akses basisdata eksternal.
      4. Inisalisasi dan kesalahan terminasi.
      5. Validitas fungsional.
      6. Kesensitifan sistem terhadap nilai *input* tertentu.
      7. Batasan dari suatu data (Abdul, 2012)

# ANALISIS MASALAH DAN ANALISIS ALGORITMA

Pada bab ini dijelaskan analisis masalah dalam *auto grading* yang sudah ada pada saat ini dan algoritma yang akan digunakan dalam pembuatan program yang akan dibangun ini.

## Analisis Masalah

### Penilaian Otomatis

Pada saat ini, penilaian yang ada pada umumnya menggunakan metode *exact string mathcing,* dimana metode ini melakukan pengecekan terhadap kecocokan kunci jawaban dan jawaban input secara *exact*, dengan kata lain jawaban harus sama secara seratus persen dengan kunci jawaban. Adapun *auto grading* lain yang menggunakan *Latent Semantic Analysis* (LSA) yang menganalisis persen kesamaan dari jawaban dan kunci jawaban. *Auto grading* seperti ini juga biasanya hanya melakukan pengecekan berdasarkan kata yang ada pada kunci jawaban secara *exact*. Sedangkan , banyak kata yang memiliki sinonim sehingga jawaban yang benar hanya terpaku pada satu jenis kata dan tidak pada sinonimnya. Kesulitan juga terjadi pada saat bentuk kalimat jawaban berbeda, sebagai contoh : kalimat aktif, kalimat pasif, dan imbuhan. Padahal seharusnya jawaban dalam kalimat aktif dan pasif adalah benar selama format dan logika jawaban sesuai dengan kunci jawaban.Jika imbuhan sekedar dihilangkan menggunakan *stemming,* maka jawaban masih belum tentu akurat karena kalimat “ Budi menendang bola” akan bernilai sama dengan “Bola menendang Budi”. Oleh karena itu, pada *auto grading* yang menggunakan algoritma genetik ini dilakukan penyisipan makna imbuhan pada jawaban dan kunci jawaban untuk mengecek kesamaan dari arti dari kata-kata berimbuhan.

## Analisis Algoritma dan Pengambilan Data

### Pengambilan Data

Pada program,data yang diperlukan adalah seluruh sinonim dari kata-kata yang ada dalam bahasa Indonesia.Pengambilan data kata sinonim dari kata-kata yang ada pada input akan dilakukan menggunakan *database* wordnet yang sudah memiliki *database* kata dalam bahasa Indonesia dalam bentuk sql. Data tersebut akan diimport kedalam program dan digunakan pada proses mutasi pada program.

### Constraint-Based Genetic

Pada program ini, akan di*-import ­*database berupa kamus bahasa Indonesia untuk mendukung proses dari Algoritma *genetic****.*** Algoritma ini digunakan untuk mengecek jawaban pelajar berdasarkan seluruh kombinasi dari susunan dan sinonim kata dari kunci jawaban yang merupakan *constraint* dari program

Step I [Start] *Generate*  *input* jawaban dari pelajar dan kunci jawaban dari pengajar. Jawaban pelajar sebagai populasi yang akan dikaji dan kunci jawaban dari pengajar sebagai *constraint.*

Step II [Fitness] Mencari kunci jawaban yang sudah ada pada jawaban dari pelajar yang sesuai degan kunci jawaban. Jika seluruh kunci jawaban sudah terpenuhi, maka pross akan langsung selesai.

Step III [New population] Membuat set jawaban baru dari seluruh kemungkinan dari susunan dan sinonim dari jawaban pengajar dan mencarinya sebagai *constraint* pada jawaban pelajar dengan cara sebagai berikut:.

1. [Selection] Pilih seluruh kalimat dari kunci jawaban lalu kategorikan setiap kata berdasarkan jenisnya yaitu kata kerja, subjek, predikat, objek, keterangan, kata penghubung dan kata sifat..
2. [Crossover] Lakukan persilangan dari susunan kalimat kunci jawaban. Persilangan ini mencari kemungkinan untuk susunan kata yang berbeda tetapi masih memeiliki makna yang sama, seperti kalimat aktif dan kalimat pasif, yaitu dengan menukar posisi subjek dan objek serta menambahkan kata di- pada kata kerja.
3. [Mutation] Mutasi setiap kata yang memungkinkan menjadi sinonim dari kata tersebut menggunakan kamus yang sudah di-*import* kedalam program.
4. [Accepting] Simpan seluruh kemungkinan yang didapat

Step IV [Replace] Gunakan kata kunci baru untuk menggantikan kata kunci awal.

Step V [Test] Jika seluruh *constraint* sudah terpenuhi, maka proses dinyatakan selesai. Jika tidak, maka ulangi proses sampai *constraint* terpenuhi. Jika *constraint* masih belum terpenuhi sampai seluruh kemungkinan sudah dicek, maka sistem akan mencatat berapa dari berapa kunci jawaban yang terpenuhi sebagai nilai kemiripan (*fitness)* dari jawaban pelajar.

Berikut akan dijelaskan konsep dari mutasi dan persilangan secara genetika dan pemanfaatannya pada program yang akan dibangun :

#### Mutasi

Mutasi pada program akan menggunakan *wordnet* sebagai media untuk mengenali sinonim dari suatu kata. Mutasi akan mencari seluruh sinonim dari suatu kata yang belum sesuai dengan kunci jawaban menggunakan *wordnet*. Mutasi pada program akan menggunakan metode *adaptive random*, dimana metode tersebut menghapus pilihan yang sudah terpilih. Kemudian mutasi akan memilih satu sinonim dari kata tersebut secara acak untuk menggantikan posisi kata tersebut. Lalu akan dilakukan pengecekan apakah kata tersebut ada pada kalimat atau tidak. Pada proses ini susunan kalimat belum berubah.

**Contoh :**

Diketahui :

Soal adalah “Apakah faktor yang menyebabkan perubahan bentuk tepi pantai?”.

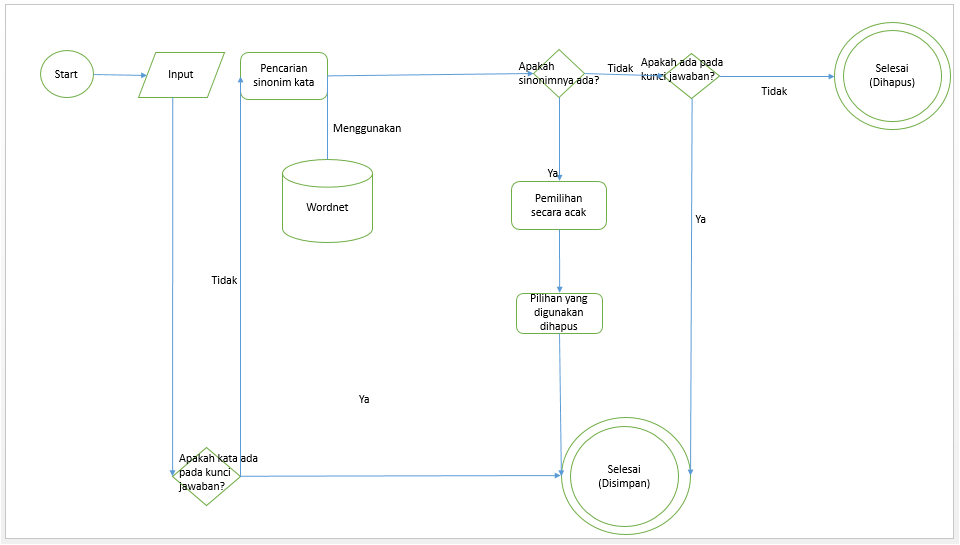
Kunci jawaban adalah “ Penyebab perubahan adalah abrasi”

Jawaban dari pelajar adalah “ Pengikisan adalah penyebab dari perubahan”.

Maka proses mutasi yang akan terjadi adalah :

* + - 1. Kata pertama yaitu “pengikisan” akan dicari pada kunci jawaban. Karena tidak ditemukan, maka kata tersebut akan dicek apakah memiliki sinonim.
      2. Sinonim dari kata tersebut ditemukan pada database *wordnet,* yaitu “erosi” dan “abrasi”. Kedua kata tersebut akan disimpan sebagai kemungkinan jawaban.
      3. Sistem memilih satu sinonim secara acak lalu menggantikan kata “pengikisan”. Misalkan yang terpilih adalah kata “Erosi”
      4. Kemudian sistem akan menghapus kata “pengikisan” dari kemungkinan jawaban.

1. Sistem mengecek kembali apakah kata “erosi”ada pada jawaban. Ternyata kata tersebut tidak ditemukan, maka sistem akan mengulangi proses 1 sampai 4 terhadap kata “Erosi” dengan kata “abrasi” sebagai pengganti kata tersebut.
2. Sistem mendeteksi adanya kata “abrasi” pada kunci jawaban. Maka kata “Abrasi” akan disimpan sesuai dengan posisi dari kata “pengikisan” yang ada pada jawaban pelajar.
3. Jika ternyata kata tersebut tidak ditemukan sampai daftar kemungkinan jawaban habis, maka kata tersebut akan dihapus.

Alur mutasi akan digambarkan pada skema berikut : 

Gambar 3. Skema mutasi

#### Delesi

Delesi menghapus kata yang dianggap tidak akan mempengaruhi pokok pikiran dari jawaban.

**Contoh :**

Diketahui :

Soal “ Kapan dia lahir?”

Kunci jawaban adalah “ Dia lahir pada tahun 1990”

Jawaban pelajar adalah “Di tahun 1990”

maka proses delesi yang akan terjadi adalah :

1. Sistem mengecek kata pertama pada soal yaitu “dia”
2. Berdasarkan NER, maka sistem mengetahui bahwa kata “dia tidak terlalu berpengaruh sehingga dapat dihilangkan.
3. Sistem menghilangkan kata “dia”
4. Sistem melanjutkan untuk memproses kata selanjutnya sampai seluruh kata sudah dicek .
5. Pada akhirnya, jawaban akan menjadi “tahun 1990”
6. Sistem melakukan proses yang sama pada jawaban pelajar danjawaban pelajar menjadi “tahun 1990”

#### Persilangan

Persilangan akan melakukan pertukaran posisi kata benda dan subjek sehingga jawaban dapat berupa kalimat aktif maupun kalimat pasif.

**Contoh** :

Diketahui :

Soal :“Apa yang dilakukan Budi dengan bola tersebut?”

Kunci jawaban adalah “ budi menendang bola”

Jawaban mahasiswa adalah “bola ditendang budi” .

Maka akan dilakukan persilangan dengan proses:

* + - 1. Sistem akan mengecek nomor urutan pada setiap kata yang didapat pada proses *preprocessing.*
      2. Sistem akan malakukan pengubahan urutan kata. Dimana jika kata kedua pada jawaban mahasiswa sudah sama dengan kata pertama dari kunci jawaban maka urutan kata tersebut diubah menjadi urutan pertama pada jawaban. Pada contoh ini, kata pertama pada jawaban mahasiswa yaitu “bola” akan dipindahan ke urutan ketiga.
      3. Proses akan dilakukan kepada seluruh kata pada jawaban mahasiswa sehingga pada akhirnya jawaban juga akan menjadi “budi tendang bola”

#### Insersi

Insersi menyisipkan makna dari imbuhan yang sudah dipisahkan dari kata dasar kedalam kalimat. Insersi akan dilakukan setelah persilangan dan akan menyesuaikan dengan terjadinya persilangan subjek dan objek.

**Contoh** :

Soal : “dimana gitar Andi?”

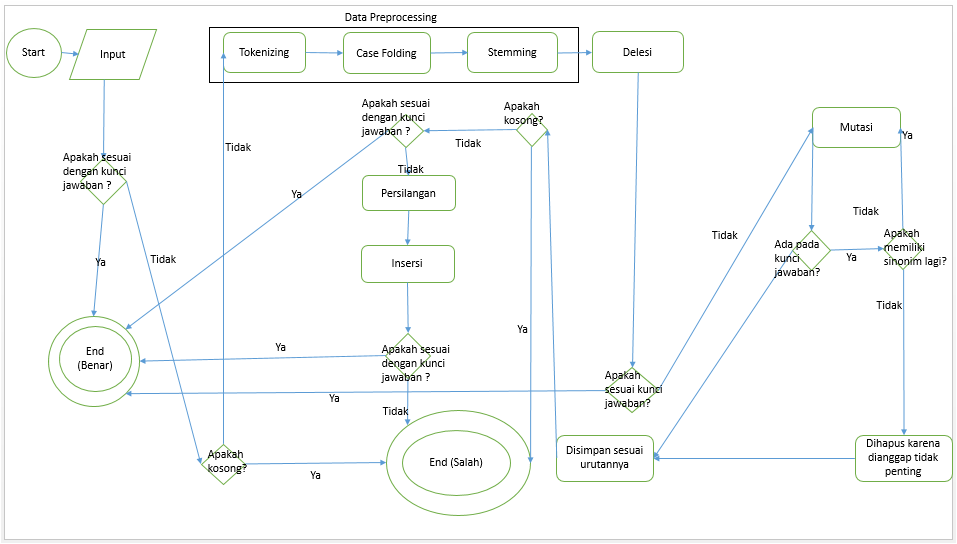
Kunci jawaban : “Nina membawa gitar Andi”

Jawaban pelajar: “Gitar Andi dibawa oleh Nina”

Maka proses insersi yang terjadi :

1. Sistem akan mencari kata yang memiliki imbuhan. Pada soal maka sistem mendapatkan kata “bawa” pada jawaban pelajar dengan imbuhan me-, pe, ke- dan ber- , karena pengubahan yang terjadi dari kalimat pasif ke kalimat aktif pada, dan dengan imbuhan me- pada kunci jawaban.
2. Sistem akan mencari seluruh arti dari imbuhan yang ada pada kunci jawaban dan memilih satu secara acak (Proses pemilihan sama seperti proses pemilihan sinonim pada mutasi).
3. Sistem akan mecari seluruh arti dari imbuhan pada jawaban pelajar dan melakukan pengecekan kesamaan dengan arti imbuhan pada kunci jawaban. Pada soal ini karena sama-sama menggunakan imbuhan me- maka pasti jawaban akan benar.

Berikut merupakan alur dari *auto grading* yang akan diimplementasikan menggunakan algoritma ini :



Gambar 4. Skema proses Algoritma genetic pada *Auto grading* yang akan diimplentasikan

### Sistem Penilaian

Pada *autograding* ini, sistem penilaian akan menggunakan *fitness score* yang memiliki sistem perhitungan yang jauh lebih sederhana dibandingkan konsep *fitness score* yang digunakan dalam perhitungan genetika. Pada sistem ini, rubik penilaian adalah :

|  |  |
| --- | --- |
| Mutasi | Setiap kata yang berhasil tedeteksi berada dalam kunci jawaban akan menambahkan *fitness score* sejumlah 1. |
| Insersi | Jika makna dari imbuhan yang terdapat pada jawaban pelajar sama dengan yang terdapat pada kunci jawaban, maka *fitness score* akan ditambahkan sebesar 2. |
| Total *fitness score* | Total *fitness score* pada satu soal dihitung dengan cara nomor urutan terbesar dari kata pada kunci jawaban setelah delesi ditambah 2. |
| Perhitungan skor |  |

Table 6. Penilaian Essay Auto Grading

**Contoh :**  
Diketahui :   
Soal : “ Apa yang sedang dilakukan oleh A?”. Soal memiliki bobot 5 nilai

Kunci Jawaban : “ A memetik senar gitar”  
Jawaban pelajar : “ tali gitar dipetik oleh A”  
maka proses penilaian yang terjadi adalah :

1. Sistem akan mengecek apakah jawaban langsung sama dengan kunci jawaban. Ternyata tidak, maka sistem melanjukan ke tahap *preprocessing.*
2. Sistem melakukan delesi pada kunci jawaban dan jawaban pelajar. Maka didapatkan “ A petik sinar gitar” sebagai kunci jawaban dan “tali gitar petik A” sebagai jawaban pelajar.
3. Kata pertama dari jawaban adalah “tali” , dimana karena kata tidak ditemukan pada kunci jawaban, maka akan dilakukan proses mutasi pada kata. Sinonim kata “tali” yaitu “senar” ada pada kunci jawaban maka jumlam *fitness score* bertambah 1. Kemudian dilakukan proses yang sama pada seluruh kata pada jawaban pelajar. Ketiga kata selanjutnya juga ditemukan pada kunci jawaban, maka jumlah *fitness score* bertambah 3 lagi menjadi 4.
4. Kemudian dilakukan pengecekan apakah jawaban setelah mutasi sama dengan kunci jawaban. Tenyata tidak, maka akan dilakukan proses persilangan.
5. Pada persilangan, kata pertama pada jawaban pelajar yaitu “senar” merupakan sama dengan kata ketiga pada kunci jawaban. Maka posisi kata “senar” pada jawaban pelajar bertukar tempat dengan kata pada posisi ketiga pada jawaban pelajar. Proses yang sama akan dilakukan pada seluruh kata pada jawaban pelajar.
6. Kemudian sistem mendeteksi bahwa adanya pertukaran posisi pada kata subjek dan kata objek, maka imbuhan di- pada kata “petik” akan menjadi disuaikan dengan kebutuhan yaitu imbuhan pe-, me-, ke- dan ber-. Yang diambil dari data imbuhan adalah arti dari imbuhan tersebut. Kemudian arti imbuhan tersebut akan disisipkan sebelum kata “petik”.
7. Sistem melakukan pengecekan apakah salah satu arti dari imbuhan yang terdapat pada kata di kunci jawaban sama dengan arti dari imbuhan kata di jawaban pelajar. Sistem mendeteksi bahwa kedua arti imbuhan memiliki kesamaan,maka jumlah *fitness score* ditambah 2. Total akhit jumlah *fitness score* adalah 6.
8. Sistem menghitung total *fitness score* yang ada pada kunci jawaban dengan cara mengambil besar dari nomor urutan yang adapada kunci jawaban , yaitu 4, kemudian ditambahkan dengan 2. Total *fitness score* yang terdeteksi oleh sistem adalah 6.

Kemudian sistem menilai *fitness function* dengan fungsi pada tabel. Dimana hasil akhir akan menjadi satu dikali lima, yaitu lima. Lima adalah nilai yang didapatkan oleh pelajar pada soal tersebut.  
Perhitungan dalam fungsi adalah :

### Rancangan Eksperimen

Eksprimen yang akan dilakukan untuk memvalidasi akurasi daripada program *autograding* ini akan dilakukan pada civitas Del, dimana akan diambil sampel sebesar 10% dari jumlah mahasiswa yaitu sekitar 25 orang, dan sekitar 5 orang sampai dengan 10 orang dosen atau asisten dosen, serta melibatkan enam pertanyaan sederhana yang akan dijawab oleh pelajar, kemudian diperiksa oleh dosen/asisten dosen yang bersedia membantu kemudian kunci jawaban dan jawaban pelajar akan dinilai oleh sistem. Hasil dari penilaian manual oleh dosen/asisten dosen akan digunakan sebagai alat ukur akurasi dari program *autograding* ini.

### Permasalahan yang Mungkin Muncul

Pada proses mutasi, saat proses pencarian sinonim yang tepat, proses terus berulang sampai sinonim yang tepat dari kata tersebut ditemukan. Hal ini berkemungkinan menimbulkan *error* berupa *looping* yang tidak berujung dikarenakan jika kata tersebut tidak ditemukan dan satu kata sinonim dapat terpakai lebih dari satu kali karena tidak ada penghapusan yang dilakukan terhadap sinonim yang sudah digunakan, dimana hal ini mendukung *error looping* tanpa henti tersebut, maka proses akan terus berulang karena sistem akan terus menerus mencari sinonim dari kata tersebut.

Untuk menghindari *looping* tanpa henti tersebut pada program, maka pada mutasi, penggantian kata menggunakan sistem *adaptive random*, dimana sistem pada awalnya sistem akan mencari seluruh kemungkinan sinonim dari sebuah kata, lalu sistem akan memilih secara acak sinonim dari sebuah kata untuk digunakan. kemudian sistem akan menghapus kata tersebut dari daftar sehingga kata sinonim tersebut tidak akan digunakan dua kali

# Rancangan

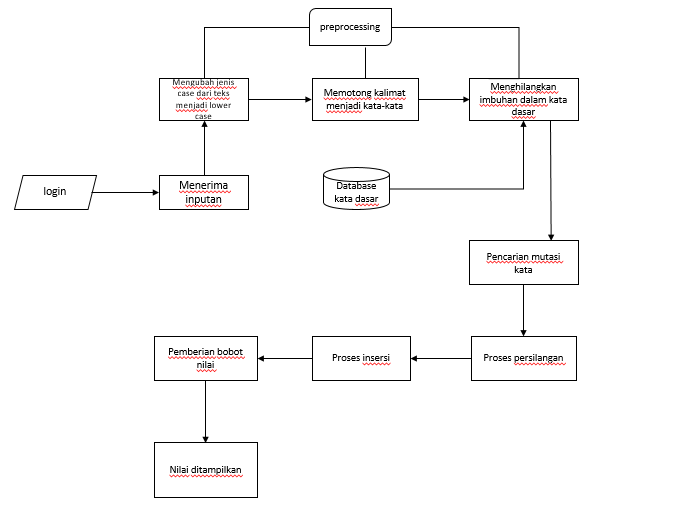
Pada bab ini dijelaskan rancangan *prototype* yang digunakan dalam penelitian

## Rancangan Prototype

Pada sub bab ini dijelaskan desain yang akan diimplementasikan pada prototype perangkat lunak yang akan dibangun. *Prototype* yang akan dibangun adalah sebuah aplikasi menggunakan java berikut adalah rancangan *prototype* yang akan dibangun.

### Proses kerja Prototype

Prototype melakukan proses yang telah dirincikan pada tahap analisis. Berikut adalah gambaran proses kerja *prototype* pada gambar berikut



#### Preprocessing

Proses ini dilakukan dengan tahapan pertama yaitu tahapan case folding untuk mengubah kalimat yang terdapat huruf besar akan diubah menjadi huruf kecil, kemudian akan masuk ketahapan proses tokenizing pada tahap ini kalimat yang telah mengalami case folding akan masuk ke tahap tokenizing, pada tahap tokenizing akan memenggal kalimat menjadi potongan kata kemudian akan dilanjutkan ke proses stemming untuk mencari kata dasar dari imbuhan suatu kata kedalam database kata dasar, apabila kata yang dicek tidak memiliki kata dasar akan dikembalikan kembali kedalam list kata

|  |
| --- |
| for(int i=0;i<jawaban\_mhs.size();i++){  sentence1 = jawaban\_mhs.get(i);  String sentence2 = sentence1.toLowerCase();  for (String word : sentence2.split(" ")) {  String jawaban = sentence2;  String sql = "INSERT INTO stemming (kalimat,kataDasar, hasilStem) VALUE('%s','%s', '%s')";  sql = String.format(sql,jawaban, word, stemmer.getRootWord(word,word));  stmt.execute(sql);  } |

#### Mutation

Proses pengecekan untuk list kata yang akan mengalami pengetesan synonym yang akan disimpan didalam list. Mutasi akan melakukan proses pengecekan kata pada kunci jawaban dengan jawaban mahasiswa dimana kata yang tidak sama akan dicek kedalam database wordnet untuk melihat synonym kata tersebut apakah terdapat pada database atau tidak, jikalau tidak terdapat maka jawaban akan dinilai

#### Linked List

Proses untuk mengatur posisi dari string kata yang akan mengalami proses ditahap crossing.Linkedlist akan dilakukan memasukkan kata kedalam linked list dimana kata tersebut akan dilakukan proses selanjutnya yaitu crossing dimana kata yang berada didalam linked list akan diproses menjadi kata yang sesuai dengan urutan yang diharapkan.

#### Crossing

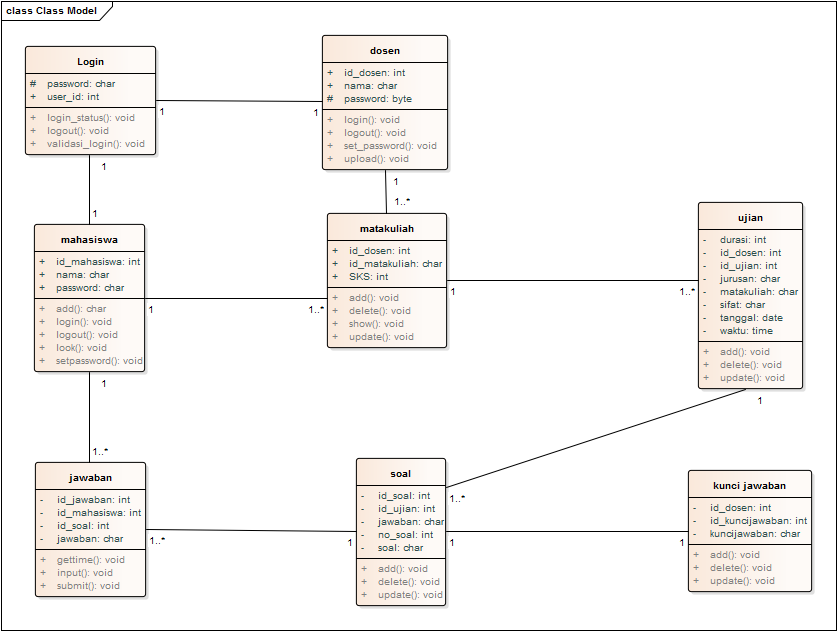
Proses persilangan untuk melihat susunan yang berada didalam linked list. Dimana susunan yang berada didalam linked list akan disamakan dengan susunan yang berada didalam kunci jawaban kemudian apabila tidak ditemukan akan langsung dilakukan pemberian nilai kepada jawaban tersebut

#### Hash map

Proses yang digunakan pada bagian synset untuk melakukan sebuah proses pencarian synonym menggunakan key dimana pada key tersebut terdapat sebuah kata synonym yang lain dari kata tersebut dan kemudian akan mereplace kata yang dicari diawal pencarian dengan kata yang dicek kemiripannya dari sebuah kata, dimana pada data csv telah dibentuk sebuah kata untuk nilai berupa kata synonym dan key sebagai string yang dimiliki nilai dimana pada saat proses pencarian akan diambil kata dari kunci jawaban dan kata dari jawaban siswa apabila tidak ditemukan kecocokan string akan dicek kata pada jawaban siswa kedalam data base synset untuk mencari synonym pada kata.

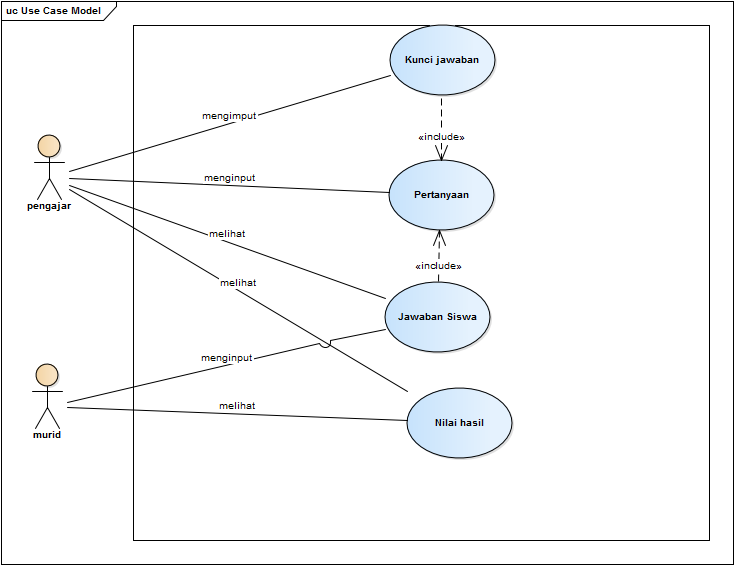
### Class Diagram

Sub bab ini menjelaskan tentang rancangan atau desai aplikasi dalam bentuk class diagram yang dapat dilihat pada gambar 6.

**Gambar 5.Class Diagram

Pada gambar class diagram dijelaskan sistem yang akan dibangun terdiri dari, login, dosen, mahasiswa, ujian, kuncijawaban, jawaban, soal, matakuliah, untuk dapat mengakses halaman pertama harus melakukan login dimana class login dipanggil untuk melakukan login yang dilakukan oleh mahasiswa dengan dosen, dimana akan dilanjutkan ke dosen dimana dosen akan melakukan upload soal pada matakuliah yang diampunya serta kunci jawaban kedalam ujian dimana ujian akan diakses oleh mahasiswa,dan soal akan terhubung dengan jawaban mahasiswa dan di dalam ujian akan terjadi proses preprocessing dan permeriksaan synsetnya

### Desain

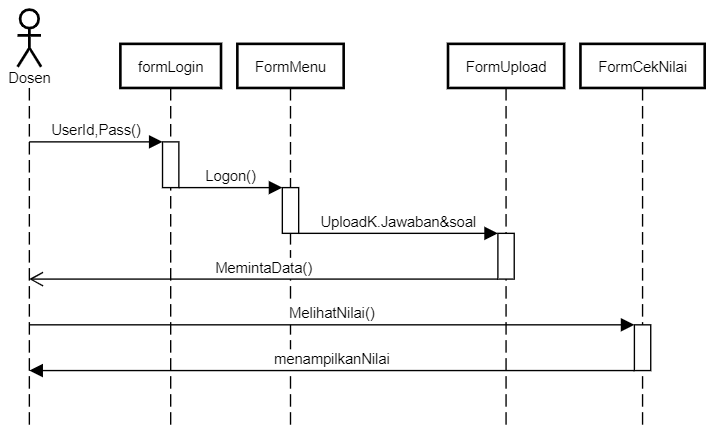


Gambar 6.Usecase

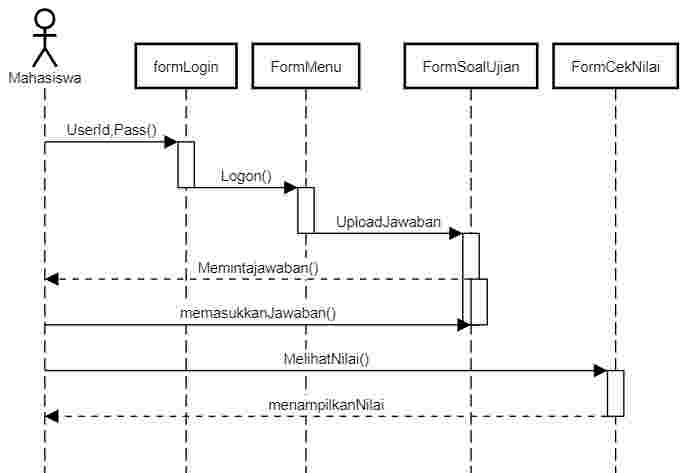
Dari gambar . dapat diketahui bahwa aktor yang terdapat ada 2 yaitu murid dengan pengajar dimana pengajar memiliki tugas untuk menginput kunci jawaban dan pertanyaan sedangkan murid melakukan tugas dengan menginput jawaban dari pertanyaan yang dibuat oleh pengajar, pengajar dan murid dapat melihat nilai hasil

Dari gambar dapat dilihat class-class yang terdapat dalam sistem essay autograding.

### Sequence Diagram



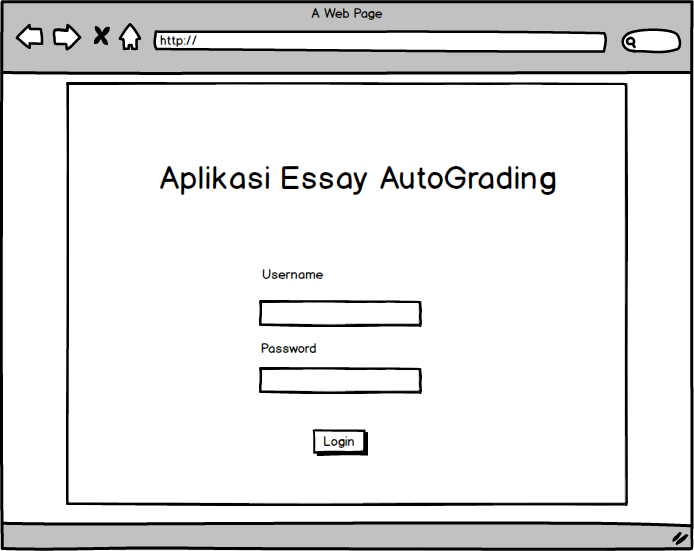
Gambar 7. Sequence Diagram Dosen



Gambar 8.Sequence Diagram Mahasiswa

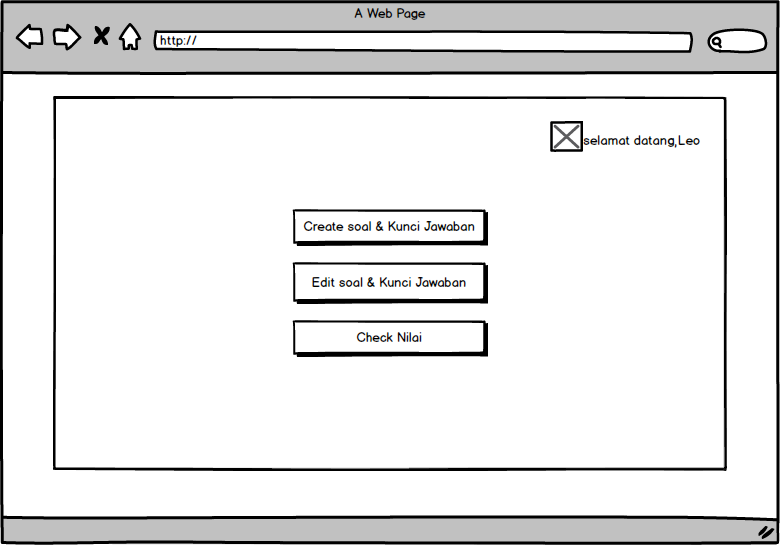
### Desain mock up layout

Pada bagian ini, akan dijelaskan mockup aplikasi yang akan digunakan sebagai contoh tampilan yang akan diimplementasikan

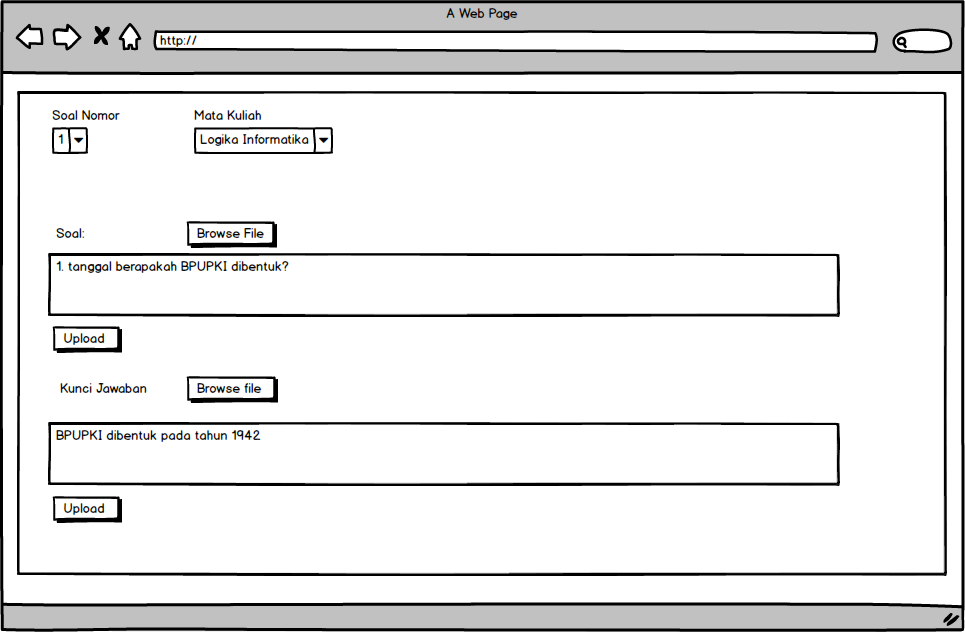


Gambar 9. Tampilan halaman Login

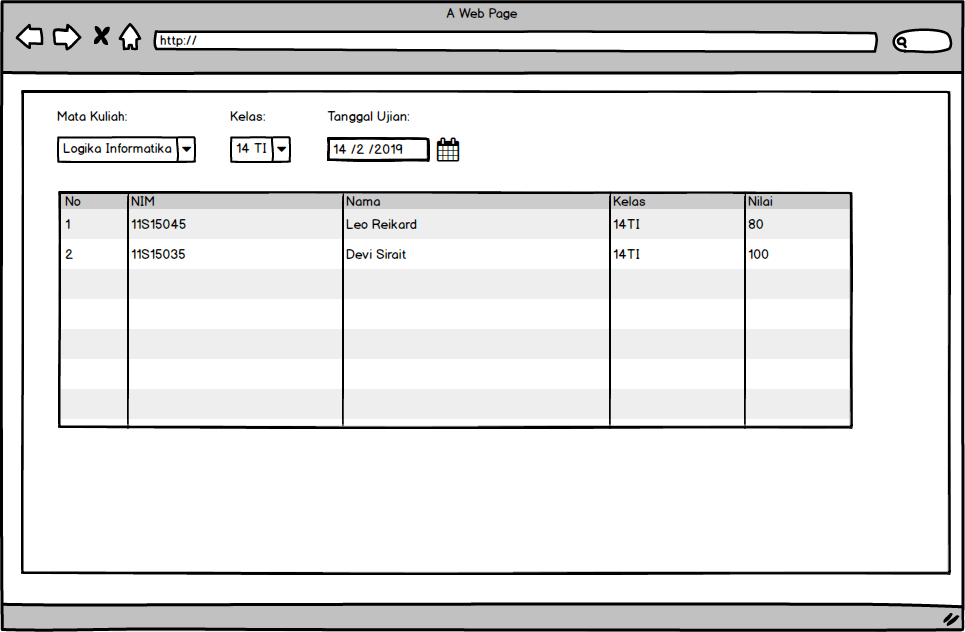
Pada gambar diatas halaman login akan diakses oleh mahasiswa dan dosen dimana setelah login dosen akan di lanjutkan kehalaman menu untuk melakukan upload soal dan jawaban dan mahasiswa akan dikirim menuju halaman ujian untuk memilih matakuliah



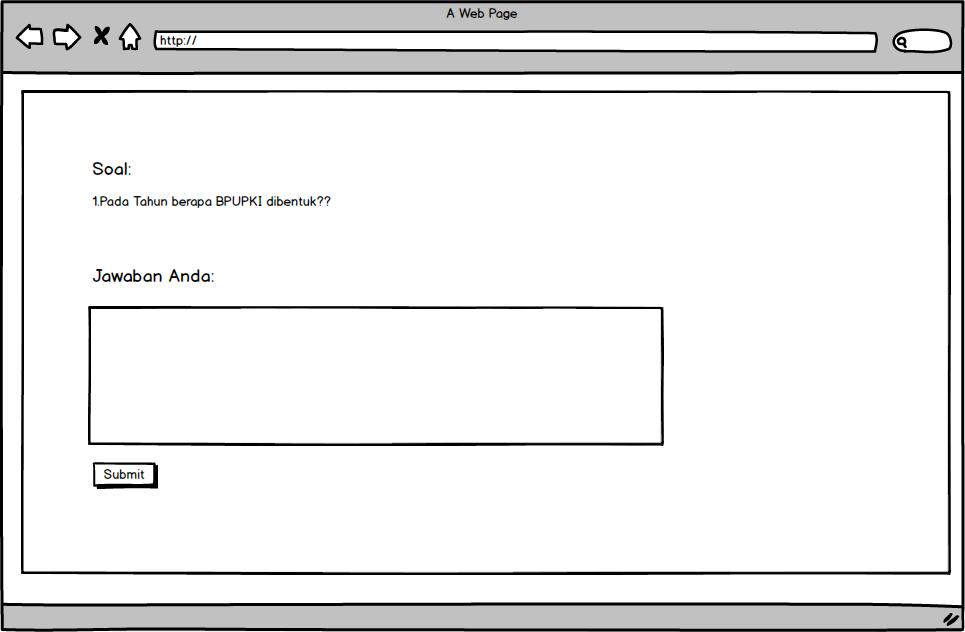
Gambar 10.Tampilan halaman menu Dosen



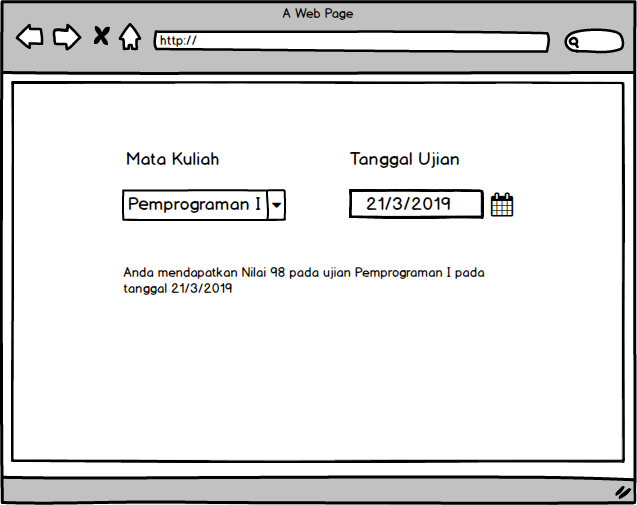
Gambar 11. Tampilan halama Upload soal dan jawaban oleh Dosen



Gambar 12.Tampilan Halaman nilai Dosen



Gambar 13. Tampilan halaman jawaban mahasiswa



Gambar 14. Tampilan halaman nilai yang diakses mahasiswa

## Rancangan Pengujian

Pengujian yang akan dilakukan dalam penelitian ini akan dilakukan dengan memanfaatkan data hasil percobaan dari beberapa mata pelajaran atau mata kuliah. Pada penelitian sebelumnya, jumlah data yang digunakan dalam pengujian yaitu sebanyak 40 data.jumlah data yang akan digunakan dalam pengujian kali ini memiliki targer sekitar 400 data yang terdiri dari 20 soal yang dijawab oleh 20 siswa dengan harapan memperoleh hasil yang lebih akurat. Sebelum melakukan pengujian data percobaan akan dikumpulkan, dan prototype akan diuji dengan metode Black-box testing untuk memastikan prototype telahdibangun sesuai dengan fungsi dan algoritma yang telah dirancang sebelumnya. *Prototype* dianggap dapat digunakan setelah berhasil melewati pengujian berikut:

* Kunci jawaban digunakan sebagai jawaban siswa maka skor yang diberikan oleh *prototype* adalah 4.
* Diberikan jawaban siswa yang sangat jauh dari kunci jawaban maka skor yang diberikan oleh *prototype* adalah 0-1.
* Setiap kata pada kunci jawaban dirubah menjadi sinonimnya dan digunakan sebagai jawaban siswa maka skor yang diberikan oleh *prototype* adalah 3-4

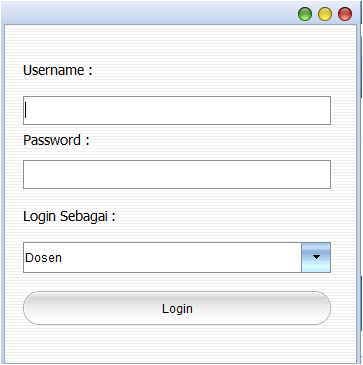
Pengukuran akurasi terhadap hasil penilian yang diperoleh melalui *prototype* dilakukan menggunakan pengukuran korelasi *Pearson.*

## Implementasi Prototype

Pada sub bab ini dijelaskan mengenai implementasi prototype penilian secara otomatis yang dibangun dengan menggunakan Bahasa pemrograman JAVA

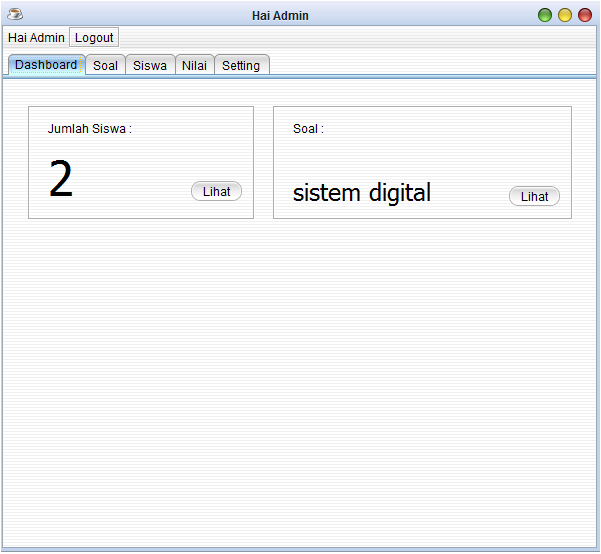
### Hasil Implementasi Desain

Pada bagian ini dijelaskan mengenai hasil implementasi dari desain *layout prototype.* Hasil implementasi dari desain dapat dilihat pada gambar .



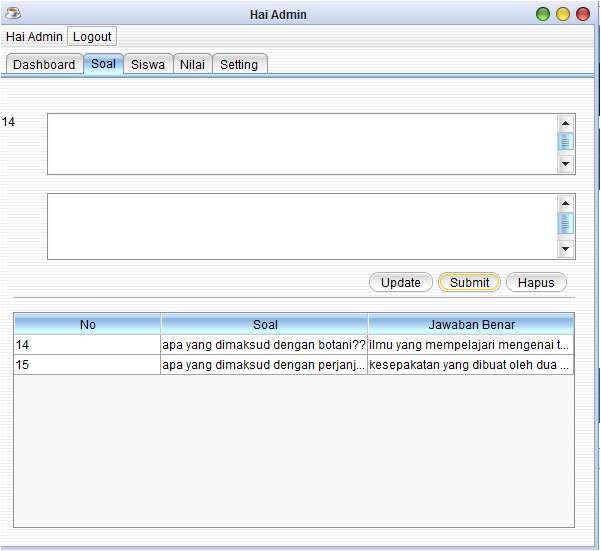
Gambar 15. Login form

Implementasi desain pada Gambar 15 merupakan tampilan dari prototype yang melakukan penilaian jawaban otomatis yang telah dikembangkan. Pada tampilan ini dosen dapat melakukan login kedalam aplikasi untuk mengakses dashboard yang akan ditampilkan pada gambar.16



Gambar 16. Dashboard

Pada gambar 16 dapat dilihat bahwa matakuliah dan jumlah siswa yang akan mengikuti ujian matakuliah yang dibuat dosen. Dan untuk membuat soal dosen dapat melakukannya pada gambar 17



Gambar 17. Form soal

Pada gambar 17 dosen dapat memasukkan soal dan kunci jawaban di form tersebut dan akan ditampilkan pada table dan dosen dapat menghapus soal yang telah diupload dengan cara menekan soal yang akan dihapus pada table dan menekan tombol hapus data akan dihapus dari database. Dan pada tombol update dapat melakukan update data yang dari database dengan mengambil id\_soal dari database.

### Lingkungan Implementasi

Lingkungan implemtasi mencakup spesifikasi perangkat lunak dan perangkat keras yang akan digunakan dalam pengembangan prototype penelian essai otomatis. Spesifikasi perangkat keras yang digunakan dalam pengembangan prototype adalah sebagai berikut:

1. Laptop : Lenovo G410
2. *Processor* : Intel Core i5 @2.7 Ghz
3. RAM : 4GB
4. *Hard Disk* : 500GB

Spesifikasi perangkat lunak yang dugunakan dalam pengembangan *prototype* adalah sebagai berikut:

1. Sistem Operasi: Windows 10 Pro
2. *Framework* : Netbeans
3. DBMS : Xampp
4. *Database* : MySQL
5. *Web Server* : Apache

### Batasan Prototype

Prototype di implementasikan dengan menggunakan Bahasa pemprograman java dengan framework yang digunakan adalah netbeans.

Adapun batasan yang didefinisikan dalam implementasi prototype adalah sebagai berikut:

* + - 1. Angka pada jawaban yang merupakan *input*  harus dituliskan dengan huruf
      2. Tidak dapat memahami maksud implisit
      3. Tidak dapat menilai siswa dengan kombinasi kunci jawaban
      4. Tidak dapat melakukan penilaian benar atau salah terhadap jawaban siswa

# Eksperimen dan Pembahasan

Pada bab ini dijelaskan mengenai tujuan pengujian, data pengujian dan hasil analisis dari hasil pengujian yang diperoleh melalui eksperimen

## Pengujian

Pada sub bab ini akan dijelaskan pengujian yang dilakukan terhadap *prototype* penilaian esai otomatis dan perbandingan nilai yang diperoleh dengan menggunakan prototype dengan nilai yang diberikan pengajar

### Tujuan penelitian

Adapun tujuan dilakukannya pengujian adalah sebagai berikut:

1. Untuk membuktikan bahwa metode yang digunakan dalam *prototype* telah dibangun sesuai dengan metode yang dianalisis dalam tahap tinjauan pustaka.
2. Untuk membandingkan hasil penilaian yang diperoleh menggunakan *prototype* dengan penilaian yang diberikan pengajar

### Data pengujian

Data pengujian yang digunakan untuk menguji aplikasi berupa hasil percobaan dari beberapa materi yang diperoleh melalui pengajar baik dosen maupun guru. Sebelum dilakukan pengujian, nilai yang diperoleh dari hasil percobaan dipetakan terlebih dahulu ke dalam skala 0-100 sesuai dengan algoritma yang digunakan pada prototype. Berikut adalah persamaan yang digunakan penulis dalam pemetaan nilai yang telah disetujui oleh ahli dalam hal ini guru/dosen terkait.

Hasil dari pemetaan untuk masing-masing data pada keseluruhan soal terlampir pada

### Pengujian *Prototype*

*Prototype* yang dikembangkan terlebih dahulu diuji fungsionalitasnya untuk memastikan *prototype* tersebut telah bekerja sesuai dengan algoritma yang telah dirancang sebelumnya. Hasil pengujian *prototype* yang telah dikembangkan dideskripsikan dalam bentuk tabel-tabel berikut:

1. Pengujian untuk jawaban yang serupa dengan kunci jawaban

Tabel 12. Pengujian Jawaban Serupa dengan Kunci Jawaban

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama Kasus Uji** | Pengujian jawaban serupa dengan kunci jawaban | | |
| **Tujuan** | Untuk melihat hasil penilaian yang diperoleh apabila jawaban siswa sama dengan kunci jawaban yang disediakan | | |
| **Deskripsi** | Pengujian dilakukan terhadap jawaban siswa yang sama dengan kunci jawaban yang disediakan | | |
| **Kondisi Awal** | - | | |
| **Skenario Uji** | | | |
| 1. Mengakses halaman *prototype* 2. Melakukan masukkan jawaban 3. System akan mecocokkan kunci jawaban dan jawaban yang terinput 4. Tekan tombol Nilai Jawaban | | | |
| **Kriteria Evaluasi Hasil** | | | |
| *Protoype* menghasilkan nilai 4 pada tabel hasil penilaian | | | |
| **Kasus dan Hasil Uji** | | | |
| **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Kesimpulan** |
|  | *Prototype* menghasilkan nilai 4 | *Prototype* menghasilkan nilai 4 | Nilai yang dihasilkan *prototype* pada jawaban yang serupa dengan kunci jawaban sudah baik. |
| **Catatan** | | | |
|  | | | |

1. Pengujian untuk jawaban yang jauh berbeda dengan kunci jawaban

Tabel 13. Pengujian Jawaban Jauh Berbeda dengan Kunci Jawaban

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama Kasus Uji** | Pengujian jawaban jauh berbeda dengan kunci jawaban | | |
| **Tujuan** | Untuk melihat hasil penilaian yang diperoleh apabila jawaban siswa jauh berbeda dengan kunci jawaban yang disediakan | | |
| **Deskripsi** | Pengujian dilakukan terhadap jawaban siswa yang jauh berbeda dengan kunci jawaban yang disediakan | | |
| **Kondisi Awal** | - | | |
| **Skenario Uji** | | | |
| 1. Mengakses halaman *prototype* 2. berisi jawaban dan kunci jawaban yang jauh berbeda 3. Tekan tombol Nilai Jawaban | | | |
| **Kriteria Evaluasi Hasil** | | | |
| *Prototype* menghasilkan nilai 0 atau 1 pada tabel hasil penilaian | | | |
| **Kasus dan Hasil Uji** | | | |
| **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Kesimpulan** |
| [  {  "idSoal" : "1",  "kunciJawaban" : "Mengemukakan pendapat secara bebas",  "jawabanSiswa" : [  {  "idSiswa" : "1",  "jawaban" : "suku, agama, warna kulit, golongan"  }  }  ] | *Prototype* menghasilkan nilai 0 atau 1 | *Prototype* menghasilkan nilai 0 | Hasil penilaian *prototype* pada kunci jawaban yang jauh berbeda belum presisi |
| **Catatan** | | | |
|  | | | |

1. Pengujian untuk jawaban yang mendekati kunci jawaban

Tabel 14. Pengujian Jawaban Mendekati Kunci Jawaban

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama Kasus Uji** | Pengujian jawaban mendekati kunci jawaban | | |
| **Tujuan** | Untuk melihat hasil penilaian yang diperoleh apabila jawaban siswa mendekati kunci jawaban yang disediakan | | |
| **Deskripsi** | Pengujian dilakukan terhadap jawaban siswa yang mendekati kunci jawaban yang disediakan | | |
| **Kondisi Awal** | - | | |
| **Skenario Uji** | | | |
| 1. Mengakses halaman *prototype* 2. berisi jawaban yang mendekati kunci jawaban 3. Tekan tombol Nilai Jawaban | | | |
| **Kriteria Evaluasi Hasil** | | | |
| *Prototype* menghasilkan nilai 3 atau 4 pada tabel hasil penilaian | | | |
| **Kasus dan Hasil Uji** | | | |
|  |  |  |  |
| **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Kesimpulan** |
|  | *Prototype* menghasilkan nilai 3 atau 4 | *Prototype* menghasilkan nilai 3 | Hasil penilaian *prototype* pada jawaban yang mendekati kunci jawaban sudah baik |
| **Catatan** | | | |
|  | | | |

1. Pengujian untuk jawaban menggunakan sinonim dari kata pada kunci jawaban

Tabel 15. Pengujian Jawaban menggunakan sinonim dari kata pada kunci jawaban

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama Kasus Uji** | Pengujian jawaban menggunakan sinonim dari kata pada kunci jawaban | | |
| **Tujuan** | Untuk melihat hasil penilaian yang diperoleh apabila jawaban menggunakan sinonim dari kata yang ada pada kunci jawaban | | |
| **Deskripsi** | Pengujian dilakukan terhadap jawaban siswa yang menggunakan sinonim dari kata pada kunci jawaban | | |
| **Kondisi Awal** | - | | |
| **Skenario Uji** | | | |
| 1. Mengakses halaman *prototype* 2. Memilih file JSON yang berisi jawaban yang mendekati kunci jawaban 3. Tekan tombol Nilai Jawaban | | | |
| **Kriteria Evaluasi Hasil** | | | |
| *Prototype* menghasilkan nilai 4 pada tabel hasil penilaian | | | |
| **Kasus dan Hasil Uji** | | | |
| **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Kesimpulan** |
|  | *Prototype* menghasilkan nilai 4 | *Prototype* menghasilkan nilai 4 | Hasil penilaian *prototype* pada jawaban menggunakan sinonim yang terdapat pada kunci jawaban sudah baik |
| **Catatan** | | | |
|  | | | |

1. Pengujian untuk jawaban kosong

Tabel 16. Pengujian Jawaban Untuk Jawaban Kosong

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama Kasus Uji** | Pengujian jawaban kosong | | |
| **Tujuan** | Untuk melihat hasil penilaian yang diperoleh apabila jawaban siswa kosong | | |
| **Deskripsi** | Pengujian dilakukan terhadap jawaban siswa yang kosong | | |
| **Kondisi Awal** | - | | |
| **Skenario Uji** | | | |
| 1. Mengakses halaman *prototype* 2. Memilih file JSON yang berisi jawaban dan kunci jawaban yang sama 3. Tekan tombol Nilai Jawaban | | | |
| **Kriteria Evaluasi Hasil** | | | |
| *Prototype* menghasilkan nilai 0 pada tabel hasil penilaian | | | |
| **Kasus dan Hasil Uji** | | | |
| **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Kesimpulan** |
|  | *Prototype* menghasilkan nilai 0 | *Prototype* menghasilkan nilai 0 | Hasil penilaian *prototype* pada jawaban yang kosong sudah baik |
| **Catatan** | | | |
|  | | | |

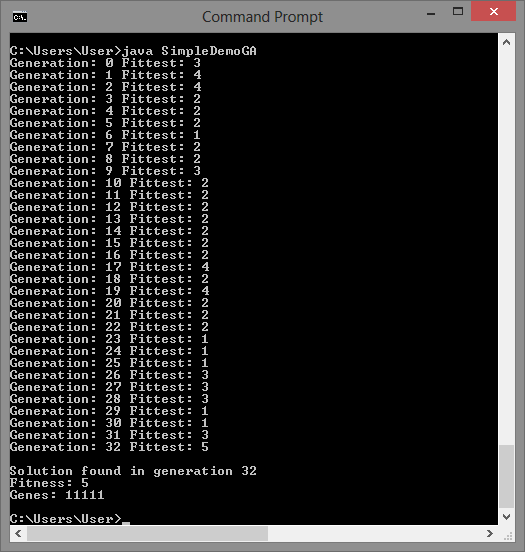
1. Pengujian untuk jawaban dengan nilai kebenaran kecil

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama Kasus Uji** | Pengujian jawaban dengan nilai kebenaran kecil | | |
| **Tujuan** | Untuk melihat hasil penilaian yang diperoleh apabila jawaban siswa memiliki nilai kebenaran yang kecil | | |
| **Deskripsi** | Pengujian dilakukan terhadap jawaban siswa yang memiliki nilai kebenaran kecil | | |
| **Kondisi Awal** | - | | |
| **Skenario Uji** | | | |
| 1. Mengakses halaman *prototype* 2. Tekan tombol Nilai Jawaban | | | |
| **Kriteria Evaluasi Hasil** | | | |
| *Prototype* menghasilkan nilai 1 atau 2 pada tabel hasil penilaian | | | |
| **Kasus dan Hasil Uji** | | | |
| **Data Masukan** | **Yang Diharapkan** | **Pengamatan** | **Kesimpulan** |
|  | *Prototype* menghasilkan nilai 1 atau 2 | *Prototype* menghasilkan nilai 2 | Hasil penilaian *prototype* pada jawaban yang memiliki nilai kebenaran kecil sudah baik |
| **Catatan** | | | |
|  | | | |

# Lampiran

|  |
| --- |
| import java.util.Random; |
|  |  |
|  | /\*\* |
|  | \* |
|  | \* @author Vijini |
|  | \*/ |
|  |  |
|  | //Main class |
|  | public class SimpleDemoGA { |
|  |  |
|  | Population population = new Population(); |
|  | Individual fittest; |
|  | Individual secondFittest; |
|  | int generationCount = 0; |
|  |  |
|  | public static void main(String[] args) { |
|  |  |
|  | Random rn = new Random(); |
|  |  |
|  | SimpleDemoGA demo = new SimpleDemoGA(); |
|  |  |
|  | //Initialize population |
|  | demo.population.initializePopulation(10); |
|  |  |
|  | //Calculate fitness of each individual |
|  | demo.population.calculateFitness(); |
|  |  |
|  | System.out.println("Generation: " + demo.generationCount + " Fittest: " + demo.population.fittest); |
|  |  |
|  | //While population gets an individual with maximum fitness |
|  | while (demo.population.fittest < 5) { |
|  | ++demo.generationCount; |
|  |  |
|  | //Do selection |
|  | demo.selection(); |
|  |  |
|  | //Do crossover |
|  | demo.crossover(); |
|  |  |
|  | //Do mutation under a random probability |
|  | if (rn.nextInt()%7 < 5) { |
|  | demo.mutation(); |
|  | } |
|  |  |
|  | //Add fittest offspring to population |
|  | demo.addFittestOffspring(); |
|  |  |
|  | //Calculate new fitness value |
|  | demo.population.calculateFitness(); |
|  |  |
|  | System.out.println("Generation: " + demo.generationCount + " Fittest: " + demo.population.fittest); |
|  | } |
|  |  |
|  | System.out.println("\nSolution found in generation " + demo.generationCount); |
|  | System.out.println("Fitness: "+demo.population.getFittest().fitness); |
|  | System.out.print("Genes: "); |
|  | for (int i = 0; i < 5; i++) { |
|  | System.out.print(demo.population.getFittest().genes[i]); |
|  | } |
|  |  |
|  | System.out.println(""); |
|  |  |
|  | } |
|  |  |
|  | //Selection |
|  | void selection() { |
|  |  |
|  | //Select the most fittest individual |
|  | fittest = population.getFittest(); |
|  |  |
|  | //Select the second most fittest individual |
|  | secondFittest = population.getSecondFittest(); |
|  | } |
|  |  |
|  | //Crossover |
|  | void crossover() { |
|  | Random rn = new Random(); |
|  |  |
|  | //Select a random crossover point |
|  | int crossOverPoint = rn.nextInt(population.individuals[0].geneLength); |
|  |  |
|  | //Swap values among parents |
|  | for (int i = 0; i < crossOverPoint; i++) { |
|  | int temp = fittest.genes[i]; |
|  | fittest.genes[i] = secondFittest.genes[i]; |
|  | secondFittest.genes[i] = temp; |
|  |  |
|  | } |
|  |  |
|  | } |
|  |  |
|  | //Mutation |
|  | void mutation() { |
|  | Random rn = new Random(); |
|  |  |
|  | //Select a random mutation point |
|  | int mutationPoint = rn.nextInt(population.individuals[0].geneLength); |
|  |  |
|  | //Flip values at the mutation point |
|  | if (fittest.genes[mutationPoint] == 0) { |
|  | fittest.genes[mutationPoint] = 1; |
|  | } else { |
|  | fittest.genes[mutationPoint] = 0; |
|  | } |
|  |  |
|  | mutationPoint = rn.nextInt(population.individuals[0].geneLength); |
|  |  |
|  | if (secondFittest.genes[mutationPoint] == 0) { |
|  | secondFittest.genes[mutationPoint] = 1; |
|  | } else { |
|  | secondFittest.genes[mutationPoint] = 0; |
|  | } |
|  | } |
|  |  |
|  | //Get fittest offspring |
|  | Individual getFittestOffspring() { |
|  | if (fittest.fitness > secondFittest.fitness) { |
|  | return fittest; |
|  | } |
|  | return secondFittest; |
|  | } |
|  |  |
|  |  |
|  | //Replace least fittest individual from most fittest offspring |
|  | void addFittestOffspring() { |
|  |  |
|  | //Update fitness values of offspring |
|  | fittest.calcFitness(); |
|  | secondFittest.calcFitness(); |
|  |  |
|  | //Get index of least fit individual |
|  | int leastFittestIndex = population.getLeastFittestIndex(); |
|  |  |
|  | //Replace least fittest individual from most fittest offspring |
|  | population.individuals[leastFittestIndex] = getFittestOffspring(); |
|  | } |
|  |  |
|  | } |
|  |  |
|  |  |
|  | //Individual class |
|  | class Individual { |
|  |  |
|  | int fitness = 0; |
|  | int[] genes = new int[5]; |
|  | int geneLength = 5; |
|  |  |
|  | public Individual() { |
|  | Random rn = new Random(); |
|  |  |
|  | //Set genes randomly for each individual |
|  | for (int i = 0; i < genes.length; i++) { |
|  | genes[i] = Math.abs(rn.nextInt() % 2); |
|  | } |
|  |  |
|  | fitness = 0; |
|  | } |
|  |  |
|  | //Calculate fitness |
|  | public void calcFitness() { |
|  |  |
|  | fitness = 0; |
|  | for (int i = 0; i < 5; i++) { |
|  | if (genes[i] == 1) { |
|  | ++fitness; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | } |
|  |  |
|  | //Population class |
|  | class Population { |
|  |  |
|  | int popSize = 10; |
|  | Individual[] individuals = new Individual[10]; |
|  | int fittest = 0; |
|  |  |
|  | //Initialize population |
|  | public void initializePopulation(int size) { |
|  | for (int i = 0; i < individuals.length; i++) { |
|  | individuals[i] = new Individual(); |
|  | } |
|  | } |
|  |  |
|  | //Get the fittest individual |
|  | public Individual getFittest() { |
|  | int maxFit = Integer.MIN\_VALUE; |
|  | int maxFitIndex = 0; |
|  | for (int i = 0; i < individuals.length; i++) { |
|  | if (maxFit <= individuals[i].fitness) { |
|  | maxFit = individuals[i].fitness; |
|  | maxFitIndex = i; |
|  | } |
|  | } |
|  | fittest = individuals[maxFitIndex].fitness; |
|  | return individuals[maxFitIndex]; |
|  | } |
|  |  |
|  | //Get the second most fittest individual |
|  | public Individual getSecondFittest() { |
|  | int maxFit1 = 0; |
|  | int maxFit2 = 0; |
|  | for (int i = 0; i < individuals.length; i++) { |
|  | if (individuals[i].fitness > individuals[maxFit1].fitness) { |
|  | maxFit2 = maxFit1; |
|  | maxFit1 = i; |
|  | } else if (individuals[i].fitness > individuals[maxFit2].fitness) { |
|  | maxFit2 = i; |
|  | } |
|  | } |
|  | return individuals[maxFit2]; |
|  | } |
|  |  |
|  | //Get index of least fittest individual |
|  | public int getLeastFittestIndex() { |
|  | int minFitVal = Integer.MAX\_VALUE; |
|  | int minFitIndex = 0; |
|  | for (int i = 0; i < individuals.length; i++) { |
|  | if (minFitVal >= individuals[i].fitness) { |
|  | minFitVal = individuals[i].fitness; |
|  | minFitIndex = i; |
|  | } |
|  | } |
|  | return minFitIndex; |
|  | } |
|  |  |
|  | //Calculate fitness of each individual |
|  | public void calculateFitness() { |
|  |  |
|  | for (int i = 0; i < individuals.length; i++) { |
|  | individuals[i].calcFitness(); |
|  | } |
|  | getFittest(); |
|  | } |
|  |  |
|  | } |

Output dari implementasi pada java tersebut adalah :



Gambar 4. Hasil dari process fittest genetic algorithm

# DAFTAR REFERENSI

Agusta, L., 2009. PERBANDINGAN ALGORITMA STEMMING PORTER DENGAN ALGORITMA NAZIEF & ADRIANI UNTUK STEMMING DOKUMEN TEKS BAHASA INDONESIA. *Konferensi Nasional Sistem dan Informatika.*

Al-Khamaiseh ,Koloud; ALShagarin ,Shadi (2014). A Survey of String Matching Algorithms

Attali, Y., & Burstein, J. (2006). Automated essay scoring with e-rater® V. 2. The Journal of Technology, Learning and Assessment, 4(3)

Burstein, J. (2003). The E-rater® scoring engine: Automated essay scoring with natural language processing

Dewi,Yayuk Nur Rohmani. (2017). Problematika Guru dalam Menerapkan Penilaian Autentik pada Kurikulum 2013 di SD Negeri Bayan No. 216 Surakarta.

E. Ukkonen, ‘Algorithms for approximate string matching’, Information Control, 64, 100–118 (1985).

Fellbaum, C., 2005. *Wordnet and wordnets.* 2nd ed. Oxford: Elsevier.

Hamid,Mustofa Abi. (2016). Pengembangan Instrumen Penilaian Hasil Belajar Siswa Berbasis TIK pada Pembelajaran Dasar Listrik Elektronika.

Indurkhya, N. dan Damerau, F.J., 2010. Handbook of Natural Language Prosesing Second Edition.

Matt, Urs von. (1994). Kassandra : The Automatic Grading System.

M. Crochemore, ‘String matching with constraints’, 13th Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 324, Berlin, 1988, pp. 44–58. Springer-Verlag

Nahar, Khalid Mohamed Oqlah& Alsmadi,Izzat Mahmoud . (n.d). The Automatic Grading for Online exams in Arabic with Essay Questions Using Statistical and computational linguistics Techniques.

Nurindra, Y., Baizak, Z. A. & A.w., Y. F., 2011. IMPLEMENTASI AUTOMATIC ESSAY GRADING SYS MENGGUNAKAN METODE PROBABILISTIC LATENT SEMANTIC ANALYSIS. *Tugas AKhir Teknik Informatika, Universitas Telkom.*

Shermis, M. D., & Barrera, F. D. (2002). Exit Assessments: Evaluating Writing Ability through Automated Essay Scoring.

Shermis, M. D., & Burstein, J. C. (2003). Automated essay scoring: A cross-disciplinary perspective. Routledge.

Shermis, M. D., Raymat, M. V., & Barrera, F. (2003). Assessing Writing through the Curriculum with Automated Essay Scoring

Soni, Kapil Kumar ;Vyas, Rohit;Sinhal, Amit (2014). Importance of String Matching in Real World Problems.

Suciyati,Rina Melly; Nurhaida dan Victoria,Linda. (2017). Pelaksanaan Hasil Belajar Siswa pada Sub Tema Hidup rukun dengan Teman Bermain di Kelas II SDN 14 Banda Aceh.

Tala, F. Z., 2003. A Study of Stremming Effects on Information Retrieval in Bahasa Indonesia.

.