# **PROPOSAL**

**IMPLEMENTASI ALGORITMA *LEVENSHTEIN DISTANCE* PADA SISTEM INFORMASI PENELUSURAN TUGAS AKHIR BERBASIS *WEB***

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

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

****

**FITNAWATI**

**E1E117030**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS HALU OLEO**

**KENDARI**

**2021**

# 

# **DAFTAR ISI**

[HALAMAN SAMPUL i](#_Toc65585309)

[HALAMAN PENGESAHAN ii](#_Toc65585309)

[DAFTAR ISI iii](#_Toc65585310)

[DAFTAR GAMBAR v](#_Toc65585311)

[DAFTAR TABEL vi](#_Toc65585312)

[BAB I PENDAHULUAN 1](#_Toc65585313)

[1.1. Latar Belakang 1](#_Toc65585314)

[1.2. Rumusan Masalah 3](#_Toc65585315)

[1.3. Batasan Masalah 3](#_Toc65585316)

[1.4. Tujuan Penelitian 3](#_Toc65585317)

[1.5. Manfaat Penelitian 3](#_Toc65585318)

[1.6. Sistematika Penulisan 4](#_Toc65585319)

[1.7. Tinjauan Pustaka 4](#_Toc65585320)

[BAB II LANDASAN TEORI 7](#_Toc65585321)

[2.1. Tugas Akhir 7](#_Toc65585322)

[2.2. Algoritma Pencarian (*Search Algorithm*) 7](#_Toc65585323)

[2.3. Algoritma *Levenshtein Distance* 8](#_Toc65585324)

[2.4. Notasi *Big-O* 13](#_Toc65585325)

[2.5. *Hypertext Preprocessor* (PHP) 14](#_Toc65585326)

[2.6. *Framework CodeIgniter* 15](#_Toc65585327)

[2.7. *Database Management Sistem* 16](#_Toc65585328)

[2.8. MySQL 17](#_Toc65585329)

[2.9. XAMPP 19](#_Toc65585330)

[2.10. *Flowchart* 21](#_Toc65585331)

[2.11. Metode Perancangan *Unified Modeling Language* (UML) 22](#_Toc65585332)

[2.12. *Rational Unified Process* (RUP) 28](#_Toc65585333)

[BAB III METODOLOGI PENELITIAN 30](#_Toc65585334)

[3.1. Waktu dan Tempat Penelitian 30](#_Toc65585335)

[3.1.1. Waktu 30](#_Toc65585336)

[3.1.2. Tempat Penelitian 30](#_Toc65585337)

[3.2. Metode Pengumpulan Data 30](#_Toc65585338)

[3.3. Metode Pengembangan Sistem 31](#_Toc65585339)

[3.4. Analisis Sistem 32](#_Toc65585340)

[3.4.1. Analisis Kebutuhan Fungsional 32](#_Toc65585341)

[3.4.2. Analisis Kebutuhan Nonfungsional 32](#_Toc65585342)

[3.5. Analisis Perancangan Sistem 34](#_Toc65585343)

[3.5.1. Perancangan *Flowchart* 34](#_Toc65585344)

[3.5.2. *Unified Modeling Language* (UML) 37](#_Toc65585345)

[3.6. Perancangan Antarmuka (*Interface*) 49](#_Toc65585346)

[3.6.1. Menu *Login* 49](#_Toc65585347)

[3.6.2. Menu Beranda 50](#_Toc65585348)

[3.6.3. Menu *Dashboard* 51](#_Toc65585349)

[3.6.4. Menu Cari Skripsi 51](#_Toc65585350)

[3.6.5. Menu mengelola Skripsi 52](#_Toc65585351)

[3.6.6. Menu Tentang 53](#_Toc65585352)

[DAFTAR PUSTAKA 54](#_Toc65585353)

# **DAFTAR GAMBAR**

Gambar 3.1 *Flowchart* Sistem.…………… ………………...………………………35

Gambar 3.2 *Flowchart* Algoritma *Levenshtein Distance* …….…………......………36

Gambar 3.3 *Use Case Diagram* Mahasiswa…….…………...………………………37

[Gambar 3.4 *Use Case Diagram Admin*………….…………………………………..38](#_Toc58164189)

[Gambar 3.5 *Activity Diagram Login*…………….……………………….…………39](#_Toc58164190)

[Gambar 3.6 *Activity Diagram* Beranda………….…………………………………..40](#_Toc58164191)

Gambar 3.7 Activity Diagram *Dashboard*……….…………………………...……..40

Gambar 3.8 *Activity Diagram* Cari Skripsi……….…………………………............41

[Gambar 3.9 *Activity Diagram* Mengelola Skripsi……….…………………………..42](#_Toc58164192)

Gambar 3.10 *Activity Diagram Upload* Skripsi…...….…….……………………….42

[Gambar 3.11 *Activity Diagram* Tambah Data Mengelola Skripsi…….…………….](#_Toc58164192)43

Gambar 3.12 *Activity Diagram* Hapus Data Mengelola Skripsi.…..…….…..……...44

Gambar 3.13 *Activity Diagram Edit* Data Mengelola Skripsi ………….……...........45

[Gambar 3.14 *Activity Diagram* Petunjuk……………………………….…………...46](#_Toc58164191)

Gambar 3.15 *Activity Diagram* Tentang………………………………….…………46

Gambar 3.16 *Class Diagram*…....…………...…….……………………….……….47

Gambar 3.17 *Sequence Diagram Login*……………..…….………………….….….48

Gambar 3.18 *Sequence Diagram* Aplikasi….……………..…………...…….……...49

Gambar 3.19 Menu *Login*…….…………..………………………………….……...50

Gambar 3.20 Menu Beranda.....…….….…………………………………….……...50

Gambar 3.21 Menu *Dashboard*………..……………...…………………….………51

Gambar 3.22 Menu Cari Skripsi….……………………..…………………….…….52

Gambar 3.23 Menu Mengelola Skripsi.……..……………………………….……...52

Gambar 3.24 Menu Tentang……...…...……………………………………….……53

# **DAFTAR TABEL**

[Tabel 2.1 Inisiasi Awal *Levenshtein Distance*](#_Toc58164899) 10

[Tabel 2.2 Pencarian Baris Pertama *Levenshtein Distance* 11](#_Toc58164900)

[Tabel 2.3 Pencarian Baris Kedua *Levenshtein Distance* 11](#_Toc58164901)

[Tabel 2.4 Pencarian Baris Ketiga *Levenshtein Distance* 11](#_Toc58164900)

[Tabel 2.5 Pencarian Baris Keempat *Levenshtein Distance* 1](#_Toc58164901)2

[Tabel 2.6 Mendapatkan nilai *Levenshtein Distance* 12](#_Toc58164902)

[Tabel 2.7 Simbol-simbol *Flowchart* 21](#_Toc58164908)

[Tabel 2.8 Simbol *Use Case Diagram* 23](#_Toc58164908)

[Tabel 2.9 Simbol *Activity Diagram* …………..](#_Toc58164909) 24

[Tabel 2.10 Simbol *Class Diagram*……..………….……………………………….. 26](#_Toc58164910)

[Tabel 2.11 Simbol *Sequence Diagram*…………….…..…………………………… 27](#_Toc58164911)

Tabel 3.1 *Gannt Chart Waktu Penelitian*...………...…….………………………… 30

Tabel 3.2 Spesifikasi Perangkat Keras …....………………………………………....33

Tabel 3.3 Spesifikasi Perangkat Lunak…………...……….…………………………33

Tabel 3.4 Keterangan Use Case Diagram……………………....…………………...38

# **BAB I PENDAHULUAN**

* 1. **Latar Belakang**

Tugas akhir adalah sarana untuk mengetahui kemampuan seorang mahasiswa apakah sudah menguasai ilmu yang diberikan dan layak untuk mengabdi di masyarakat sesuai dengan kompetensi yang diajarkan oleh kampus. Tugas akhir atau sering disingkat TA merupakan langkah awal untuk dapat belajar dalam menghadapi dunia kerja yang akan dihadapi, dengan adanya tugas akhir mahasiswa dapat mempersiapkan diri untuk menyelesaikan proyek-proyek di masa kerja nanti. (Anggaeni and Sujatmiko, 2013).

Sebelum menyusun sebuah tugas akhir ada beberapa alur atau langkah yang harus di tempuh, antara lain: menentukan judul tugas akhir yang diketahui Ketua Jurusan, kemudian Mahasiswa diberikan Dosen Saran, setelah Dosen saran setuju maka selanjutnya akan diberikan Dosen pembimbing, selanjutnya mahasiswa dapat memulai penyusunan proposal, sidang proposal, penyusunan tugas akhir dan sidang tugas akhir.

Penyusunan laporan tugas akhir mahasiswa memerlukan banyak referensi baik berupa buku maupun laporan tugas akhir dari penelitian yang sebelumnya. Untuk referensi laporan tugas akhir mahasiswa dapat mencari di perpustakaan maupun *website*. Dalam melakukan pencarian mahasiswa dapat menggunakan sistem yang sudah tersedia, dengan cara memasukkan kata kunci yang ingin dicari maka secara otomatis sistem akan menampilkan judul tugas akhir yang sesuai dengan kata kunci yang dimasukkan. Seringkali, dalam melakukan pencarian mahasiswa menginputkan kata kunci yang bukan merupakan ejaan yang baku atau salah ketik. Tentu saja mahasiswa akan memperoleh informasi yang kurang lengkap dan bahkan mahasiswa gagal dalam mendapatkan informasi yang sesuai dengan kata kunci yang dimasukkan.

Saat ini Jurusan Teknik Informatika sudah memiliki sistem bank data tugas akhir dan KP mahasiswa, tetapi sistem ini masih menggunakan SQL untuk menyimpan dan memanipulasi data sehingga hasil pencarian dari sistem tersebut belum optimal. Oleh karena itu, perlu adanya implementasi algoritma pencarian yang akan membantu menghasilkan hasil pencarian yang tepat dan optimal (Sa’diah, 2017). Selain itu, sistem ini hanya bisa diakses oleh Mahasiswa yang sudah menyelesaikan skripsi untuk melakukan pengiriman file. Sementara yang membutuhkan sistem ini bukan hanya mahasiswa yang sudah melakukan skripsi tetapi mahasiswa semester akhir yang akan mulai mencari judul untuk Tugas Akhir pula. Hal ini menyebabkan proses pencarian menjadi kurang efektif serta membutuhkan waktu yang lama bagi mahasiswa yang ingin mengetahui tentang pembuatan tugas akhir dan skripsi bagi mereka.

Mengatasi hal ini, bank data yang memuat informasi mengenai TA dan skripsi akan dimuat dalam sebuah rancangan *web* dengan menerapkan algoritma pencarian. sehingga mahasiswa Jurusan Teknik Informatika dapat dengan mudah melakukan penelusuran dan mengaksesnya dimanapun tanpa harus menemui bagian akademik untuk mendapatkan informasi. Hal ini tentunya akan lebih efektif serta dapat mengefisienkan waktu sehingga mahasiswa mengetahui betul informasi mengenai tugas akhir dan skripsi yang ada di Jurusan Teknik Informatika.

Pada Pembuatan sistem informasi penelusuran tugas akhir dapat diimplementasikan dengan menggunakan algoritma *levenshtein distance*,dimana algoritma ini melakukan pendekatan perkiraan pada pencarian kalimat. Algoritma *Levenshtein* juga dapat digunakan untuk berbagai bidang, misalnya sebagai mesin pencarian, deteksi ejaan kata, pengenal kalimat pembicaraan, menganalisis DNA, deteksi pemalsuan kata pada sebuah kalimat, dan lain-lain.

Berdasarkan latar belakang tersebut penelitian yang akan dilakukan ini berjudul “Implementasi Algoritma *Levenshtein Distance* Pada Sistem Informasi Penelusuran Tugas Akhir Berbasis *Web*”.

* 1. **Rumusan Masalah**

Berdasarkan latar belakang masalah yang diuraikan sebelumnya maka rumusan masalah penelitian ini adalah:

1. Bagaimana mengembangkan sistem informasi penelurusan tugas akhir berbasis *web*?
2. Bagaimana menerapkan algoritma *levenshtein distance* beserta kompleksitasnya?
   1. **Batasan Masalah**

Berdasarkan rumusan masalah yang diuraikan diatas maka batasan masalah dalam tugas akhir ini adalah:

1. Penelitian ini dilakukan di Jurusan Teknik Informatika Universitas Halu Oleo.
2. Pembuatan sistem informasi penelurusan tugas akhir menggunakan algoritma *levenshtein distance*.
3. Dibangun berbasis *website*.
4. Data disimpan dalam *file* txt.
   1. **Tujuan Penelitian**

Tujuan dari penelitian ini adalah:

* + - 1. Mengembangkan sistem informasi penelurusan tugas akhir berbasis *web.*
      2. Menerapkan algoritma *levenshtein distance* beserta kompleksitasnyadalam sistem informasi penelurusan tugas akhir berbasis *web.*
  1. **Manfaat Penelitian**

Adapun manfaat dari penelitian yang akan dilakukan diharapkan dapat:

1. Memudahkan mahasiswa jurusan Teknik Informatika Universitas Halu Oleo dalam mencari tugas akhir.
2. Dapat menjadi acuan tambahan bagi penulis atau peneliti selanjutnya dalam menerapkan algoritma *levenshtein distance* terhadap pencarian kalimat.
   1. **Sistematika Penulisan**

Sistematika penulisan tugas akhir ini terdiri atas beberapa bagian utama 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 yang dilakukan oleh Khairun Nisa Meiah Ngafidin pada tahun 2015 dalam penelitiannya yang berjudul Implementasi *Fitur Autocomplete* dan Algoritma *Levenshtein Distance* untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia (KBBI). Hasil penelitian pada pengujian fitur *autocomplete* adalah muncul untuk setiap kata yang dimasukkan dan pada algoritma *levenshtein distance* saran yang muncul sesuai dengan kata masukkannya. Hasil pengujian efektivitas a*utocomplete* adalah sebesar 84.615 % yang berarti sangat efektif, dan *levenshtein distance* sebesar 76.04 % yang berarti efektif untuk digunakan di aplikasi KBBI.

Penelitian yang dilakukan oleh Ariyani, dkk pada tahun 2016 dalam penelitiannya yang berjudul Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode *Levenshtein Distance* menyimpulkan bahwa pengujian menggunakan data *real* yaitu data dokumen yang berplagiat yang diambil dari artikel/berita lewat internet, algoritma *levenshtein distance* menghasilkan nilai *Similarity* yang tinggi yaitu diatas 85% sampai 100% untuk dokumen yang tingkat kemiripannya tinggi. Sedangkan untuk dokumen dengan tingkat kemiripan yang rendah atau tidak berplagiat maka menghasilkan nilai *Similarity* dibawah 40%.

Penelitian yang dilakukan oleh Na’firul Hasna Ariyani pada tahun 2016 dalam penelitiannya yang berjudul Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode *Levenshtein Distance*. Penelitian ini bertujuan untuk mendeteksi kemiripan dokumen teks sehingga dapat digunakan untuk membantu menentukan plagiarisme. Tipe dokumen yang diuji adalah .pdf .docx dan .txt.

Penelitian yang dilakukan oleh Uli Fitrianti dan Mutammimul Ula pada tahun 2017 dalam penelitiannya yang berjudul Implementasi Algoritma *Levenshtein Distance* Dan Algoritma *Knuth Morris Pratt* Pada Aplikasi Asmaul Husna Berbasis Android. Hasil dari penelitian ini adalah penggunaan algoritma *levenshtein distance* untuk proses pengecekan ejaan jawaban yang diiputkan oleh pengguna dan algoritma *knuth morris pratt* untuk proses pencocokan jawaban berupa *string* yang sudah benar dengan *string* yang tersimpan dalam database yang dapat memudahkan para pengguna mengetahui hasil skor akhir jawaban yang benar dan jawaban yang salah.

Penelitian yang dilakukan oleh Nasib Marbun, dkk pada tahun 2019 dalam penelitiannya yang berjudul Penerapan Algoritma Levenshtein dalam Pencarian Arti Istilah Penelitian. Jumlah istilah penelitian sangat banyak membuat peneliti pemula kesulitan mencari informasi terkait dengan teori untuk memahami pengertian dari setiap istilah penelitian. Agar dapat mempermudah pencarian informasi terkait dengan teori untuk memahami pengertian dari setiap istilah-istilah penelitian diperlukan dilakukan penerapan algoritma *string matching*.

Berdasarkan penelitian-penelitian yang telah dipaparkan diatas, maka akan dilakukan penelitian lebih lanjut dengan judul **“Implementasi Algoritma *Levenshtein Distance* Pada Sistem Informasi Penelusuran Tugas Akhir Berbasis *Web*”**.

# **BAB II LANDASAN TEORI**

1. **Tugas Akhir**

Tugas akhir merupakan suatu karya ilmiah berdasarkan suatu kegiatan penelitian mandiri dari mahasiswa, disusun dalam jangka waktu maksimum dua semester, di bawah bimbingan seorang dosen pembimbing dan dapat dibantu seorang pembantu pembimbing. Dosen pembimbing di sini hanya sebagai fasilitator. Mandiri di sini diartikan bahwa perancangan penelitian, pelaksanaan penelitian, penulisan laporan hasil penelitian ada pada diri mahasiswa itu sendiri. Tim dosen penguji mengadakan penilaian dalam hal kemandirian dan cara mempertahankan tugas akhir (Anggaeni and Sujatmiko, 2013).

1. **Algoritma Pencarian (*Search Algorithm*)**

Algoritma adalah metode efektif diekspresikan sebagai rangkaian terbatas dari instruksi-instruksi yang telah didefinisikan dengan baik untuk menghitung sebuah fungsi. Dimulai dari sebuah kondisi awal dan input awal, instruksi-instruksi tersebut menjelaskan sebuah komputasi yang bila dieksekusi akan diproses lewat sejumlah urutan kondisi terbatas yang terdefinisikan dengan baik, yang pada akhirnya menghasilkan keluaran dan berhenti di kondisi akhir. Transisi dari satu kondisi ke kondisi lainnya tidak harus deterministik. Beberapa algoritma dikenal dengan algoritma pengacakan menggunakan masukan acak.

Pencarian atau *searching* merupakan upaya untuk mendapatkan data dalam kumpulan data. Pencarian seringkali ditemukan pada kehidupan sehari-hari, misalnya pencarian suatu istilah dalam kamus. Pada aplikasi komputer, pencarian dilakukan untuk menemukan data dari seorang pegawai, mendapat informasi kata yang tidak dimengerti dalam kamus digital, dan mendapat banyak informasi lainnya dari internet.

Algoritma pencarian atau *searching algorithm* adalah algoritma yang menerima sebuah argumen kunci dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (*successful*) atau tidak ditemukan (*unsuccessful*) (Abdurrahman, 2016).

1. **Algoritma *Levenshtein Distance***

Algoritma *levenshtein distance* atau bisa disebut juga dengan *edit distance*, diciptakan oleh Vladimir Levenshtein pada tahun 1965. Perhitungan *edit distance* merupakan suatu perhitungan dari matriks yang digunakan untuk menghitung jumlah perbedaan antara dua *string*. Perhitungan distance atau jarak antara kedua *string* ini ditentukan oleh jumlah minimum operasi perubahan untuk merubah dari *string* A menjadi *string* B, yang dihitung dengan menggunakan tabel perhitungan *levenshtein distance*, di mana nilai terakhir yang berada di paling pojok kanan bawah merupakan nilai akhir dari jarak kedua *string*.

*Levenshtein distance* adalah sebuah matriks *string* yang digunakan untuk mengukur perbedaan atau jarak (*distance*) antara dua *string*. Nilai *distance* antara dua *string* ini ditentukan oleh jumlah minimum dari operasi-operasi perubahan yang diperlukan untuk melakukan transformasi dari suatu *string* menjadi *string* lainnya. Operasi-operasi tersebut adalah penyisipan (*insertion*), penghapusan (*deletion*), atau penukaran (*subtition*). *Levenshtein distance* merupakan salah satu algoritma yang dapat digunakan dalam mendeteksi kemiripan antara dua *string* yang berpotensi melakukan tindak *plagiarisme* (Purba and Situmorang, 2017).

Terdapat 3 macam operasi utama yang dapat dilakukan oleh algoritma ini, yaitu:

1. Operasi pengubahan karakter

Operasi pengubahan karakter merupakan operasi untuk menukar suatu karakter dengan karakter lain. Seperti contohnya yaitu pengubahan karakter oleh *string* “jija” menjadi “jika”, di mana karakter j diubah menjadi karakter k.

1. Operasi penambahan karakter

Operasi penambahan karakter merupakan suatu operasi yang menambahkan karakter tertentu ke dalam suatu *string*. Misalnya yaitu penambahan karakter “a” ke dalam *string* “lemri”, sehingga dengan penambahan ini yang sebelumnya “lemri” menjadi *string* “lemari”. Penambahan karakter pun tidak hanya dilakukan diakhir kata, namun bisa ditambahkan diawal maupun disisipkan ditengah *string*.

1. Operasi penghapusan karakter

Operasi penghapusan karakter merupakan operasi untuk menghilangkan karakter tertentu pada suatu *string*. Misalkan saja pada kata “jumblah” akan dihapus satu karakter yaitu karakter “b” sehingga menjadi “jumlah”.

Untuk menghitung *levenshtein distance* dari dua buah *string* dapat dilakukan secara rekursif. Formula dari algoritma rekursif untuk *string* a dan *string* b diberikan oleh lev a, ( |a|, |b| ) dimana:

Lev a,b (i,j) =

Misalkan i adalah indeks dari *string* a pada suatu tahap dan j adalah indeks *string* b. formula yang diberikan diatas akan membandingkan *string* dari a dan *string* b dari indeks paling akhir. Untuk setiap tahap, akan diputuskan operasi mana yang lebih optimal. Untuk mengecek nilai solusi dari operasi penghapusan, maka status i akan dikurangi dengan sejumlah satu. Untuk mengecek nilai solusi dari operasi penyisipan, maka status j akan dikurangi dengan sejumlah satu. Untuk operasi penggantian, maka status i dan status j masing-masing akan dikurangi dengan sejumlah satu. Bila karakter ke i dari *string* a dan karakter ke j dari *string* b sama, maka operasi penggantian tidak perlu diperhitungkan, tetapi status i dan status j dapat langsung dikurangi dengan sejumlah satu.

Algoritma *levenshtein distance* berjalan mulai dari pojok kiri atas sebuah array dua dimensi yang telah diisi sejumlah karakter *string* awal dan *string* target dan diberikan nilai *cost*. Nilai *cost* pada ujung kanan bawah menjadi nilai *edit distance* yang menggambarkan jumlah perbedaan dua *string* (Fitrianti et al. n.d, 2017). Adapun contoh pencariannya sebagai berikut:

1. Melakukan *brute force* untuk setiap kata yang ada pada data.
2. Bila mendapatkan kalimat, maka kalimat akan dipisah pada setiap kalimatnya.
3. Mencocokkan kalimat yang ada dengan kata kunci yang telah dimasukkan oleh pengguna.
4. Melakukan perhitungan jarak kedua kata menggunakan algoritma *levenshtein distance*. Sebagai contoh perhitungan jarak antara kata “kipas” dan “kipam”.

**Tabel 2.1 Inisiasi Awal Levenshtein Distance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | - | K | I | P | A | S |
| - | 0 | 1 | 2 | 3 | 4 | 5 |
| K | 1 |  |  |  |  |  |
| I | 2 |  |  |  |  |  |
| P | 3 |  |  |  |  |  |
| A | 4 |  |  |  |  |  |
| M | 5 |  |  |  |  |  |

Berdasarkan Tabel 2.1 inisialisasi awal dilakukan dengan menyusun kata pertama secara vertikal dan kata kedua secara horizontal, lalu diberikan nilai secara berurutan pada setiap katanya.

**Tabel 2.2 Pencarian Baris Pertama Levenshtein Distance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | - | K | I | P | A | S |
| - | 0 | 1 | 2 | 3 | 4 | 5 |
| K | 1 | 0 | 1 | 2 | 3 | 4 |
| I | 2 |  |  |  |  |  |
| P | 3 |  |  |  |  |  |
| A | 4 |  |  |  |  |  |
| M | 5 |  |  |  |  |  |

Pada Tabel 2.2 terdapat nilai “0” yang telah didapat dari pencarian nilai minimum dari nilai kotak diatasnya yang telah ditambahkan 1 dan disampingnya yang telah ditambahkan 1 beserta nilai kotak diatasnya yang ditambahkan 1 bila huruf yang dicocokkan bernilai sama.

**Tabel 2.3 Pencarian Baris Kedua Levenshtein Distance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | - | K | I | P | A | S |
| - | 0 | 1 | 2 | 3 | 4 | 5 |
| K | 1 | 0 | 1 | 2 | 3 | 4 |
| I | 2 | 1 | 0 | 1 | 2 | 3 |
| P | 3 |  |  |  |  |  |
| A | 4 |  |  |  |  |  |
| M | 5 |  |  |  |  |  |

Pada Tabel 2.3 terdapat lanjutan baris yang dilakukan sama dengan cara sebelumnya. Dengan mencari nilai minimum dari beberapa nilai kotak.

**Tabel 2.4 Pencarian Baris Ketiga Levenshtein Distance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | - | K | I | P | A | S |
| - | 0 | 1 | 2 | 3 | 4 | 5 |
| K | 1 | 0 | 1 | 2 | 3 | 4 |
| I | 2 | 1 | 0 | 1 | 2 | 3 |
| P | 3 | 2 | 1 | 0 | 1 | 2 |
| A | 4 |  |  |  |  |  |
| M | 5 |  |  |  |  |  |

Pada Tabel 2.4 terdapat lanjutan baris yang dilakukan sama dengan cara sebelumnya. Dengan mencari nilai minimum dari beberapa nilai kotak.

**Tabel 2.5 Pencarian Baris Keempat Levenshtein Distance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | - | K | I | P | A | S |
| - | 0 | 1 | 2 | 3 | 4 | 5 |
| K | 1 | 0 | 1 | 2 | 3 | 4 |
| I | 2 | 1 | 0 | 1 | 2 | 3 |
| P | 3 | 2 | 1 | 0 | 1 | 2 |
| A | 4 | 3 | 2 | 1 | 0 | 1 |
| M | 5 |  |  |  |  |  |

Pada Tabel 2.5 terdapat lanjutan baris yang dilakukan sama dengan cara sebelumnya. Dengan mencari nilai minimum dari beberapa nilai kotak.

**Tabel 2.6 Mendapatkan Nilai Levenshtein Distance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | - | K | I | P | A | S |
| - | 0 | 1 | 2 | 3 | 4 | 5 |
| K | 1 | 0 | 1 | 2 | 3 | 4 |
| I | 2 | 1 | 0 | 1 | 2 | 3 |
| P | 3 | 2 | 1 | 0 | 1 | 2 |
| A | 4 | 3 | 2 | 1 | 0 | 1 |
| M | 5 | 4 | 3 | 2 | 1 | 1 |

Pada Tabel 2.6 terdapat hasil dari algoritma ini dengan mengisi semua kotak. Hasil dari algoritma ini dapat dilihat pada nilai akhir yang telah didapatkan yaitu “1”. Dengan ini menyatakan hasil dari algoritma *levenshtein distance* dari perbandingan kata “kipas” dan “kipam” bernilai “1”.

Tahap selanjutnya memasukkan hasil *levenshtein distance* ke dalam *similarity* sebagai berikut:

(2.2)

Keterangan:

*Similarity* = nilai kemiripan

Dis = nilai *Levenshtein Distance*

MaxLength = jumlah *string* terpanjang

= 0.75

Dengan menggunakan rumus *Similarity*, nilai kemiripan dari kata “kipas” dan “kipam” menghasilkan nilai 0.75.

1. **Notasi *Big-O***

Notasi menyatakan *running time* dari suatu algoritma untuk kemungkinan kasus terburuk. Notasi memiliki dari beberapa bentuk. Notasi dapat berupa salah satu bentuk maupun kombinasi dari bentuk-bentuk tersebut (Sarker and Mouftah 2011). :

1. Bentuk memiliki arti bahwa algoritma yang sedang dianalisis merupakan algoritma konstan. Hal ini mengindikasikan bahwa *running time* algoritma tersebut tetap, tidak bergantung pada .
2. Bentuk berarti bahwa algoritma tersebut merupakan algoritma linier. Artinya, bila menjadi maka *running time* algoritma akan menjadi dua kali *running time* semula.
3. Bentuk berarti bahwa algoritma tersebut merupakan algoritma kuadratik. Algoritma kuadratik biasanya hanya digunakan untuk kasus dengan yang berukuran kecil. Sebab, bila dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi empat kali semula.
4. Bentuk berarti bahwa algoritma tersebut merupakan algoritma kubik. pada algoritma kubik, bila dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi delapan kali semula.
5. Bentuk berarti bahwa algoritma tersebut merupakan algoritma eksponensial. Pada kasus ini, bila dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi kuadrat kali semula.
6. Bentuk berarti algoritma tersebut merupakan algoritma logaritmik. Pada kasus ini, laju pertumbuhan waktu lebih lambat dari pada pertumbuhan *n*. Algoritma yang termasuk algoritma logaritmik adalah algoritma yang memecahkan persoalan besar dengan mentransformasikannya menjadi beberapa persoalan yang lebih kecil dengan ukuran sama. Basis algoritma tidak terlalu penting, sebab bila misalkan dinaikkan menjadi dua kali semula, meningkat sebesar jumlah tetapan.
7. Bentuk , terdapat pada algoritma yang membagi persoalan menjadi beberapa persoalan yang lebih kecil, menyelesaikan setiap persoalan secara independen, kemudian menggabungkan solusi masing-masing persoalan.
8. Sedangkan bentuk berarti bahwa algoritma tersebut merupakan algoritma faktorial. Algoritma jenis ini akan memproses setiap masukan dan menghubungkannya dengan masukan lainnya. Bila menjadi dua kali semula, maka *running time* algoritma akan menjadi faktorial dari .
9. ***Hypertext Preprocessor* (PHP)**

*Hypertext Preprocessor***(**PHP) adalah bahasa scriptingweb *HTML-embedded*. Ini berarti kode PHP dapat disisipkan ke dalam HTML halaman Web. Ketika sebuah halaman PHP diakses, kode PHP dibaca atau"diurai" oleh *server*. *Output* dari fungsi PHP pada halaman biasanya dikembalikan sebagai kode HTML, yang dapat dibaca oleh *browser*. Karena kode PHP diubah menjadi HTML sebelum halaman dibuka, pengguna tidak dapat melihat kode PHP pada halaman. Ini membuat halaman PHP cukup aman untuk mengakses database dan informasi aman lainnya. (Ferdianto, 2013).

Banyak sintaks PHP yang hasil adaptasi dari bahasa lain seperti bahasa C, Java dan Perl. Namun, PHP memiliki sejumlah fitur unik dan fungsi tertentu juga. Tujuan dari bahasa pemrograman PHP adalah untuk memungkinkan pengemban web untuk menulis halaman yang dihasilkan secara dinamis dengan cepat dan mudah. PHP juga bagus untuk menciptakan situs Web *database-driven*. Jika Anda ingin mempelajari lebih lanjut tentang PHP, situs resminya yaitu PHP.net. (Ferdianto, 2013).

Beberapa kelebihan PHP dari bahasa pemrograman web, antara lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukansebuah kompilasi dalam penggunaanya.
2. *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulaiapache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudahkarena memiliki referensi yang banyak.
5. PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara *runtime*melalui *console* serta juga dapat menjalankan perintah-perintah system. (Ferdianto, 2013).
6. ***Framework CodeIgniter***

*Framework* merupakan perangkat lunak yang mulai menjadi pilihan untuk membuat suatu aplikasi. Kemudahan-kemudahan yang diberikan menarik orang-orang untuk menggunakannya. Hal ini tidak terlepas dari tingkat efektifitas dan efisiensinya yang lebih baik dalam proses pengembangan suatu perangkat lunak. *Framework* adalah suatu kumpulan kode berupa pustaka (*library*) dan alat (*tool*) yang dipadukan sedemikian rupa menjadi satu kerangka kerja (*framework*) guna memudahkan dan mempercepat proses pengembangan aplikasi web.

Proses pengembangan web dapat dilakukan dengan beragam bahasa pemrograman seperti *PHP, Python, Ruby, Perl, C++, JAVA* dan sebagainya. Saat ini, banyak bermuculan *framework* web yang dirancang untuk bahasa pemrograman tersebut. Salah satunya adalah *CodeIgniter*. *CodeIgniter* adalah kerangka kerja PHP yang kuat dengan tapak yang sangat kecil, dibangun untuk pengembang yang membuthkan *toolkit* sederhana dan elegan untuk membuat aplikasi web berfitur lengkap.

Beberapa keunggulan yang kemudian ditawarkan oleh *CodeIgniter* yaitu sebagai berikut:

1. *CodeIgniter* adalah *framework* yang bersifat *free* dan *open source*.
2. *CodeIgniter* memiliki ukuran yang kecil dibandingkan dengan *framework* lain. Setelah proses instalasi, *framework* *CodeIgniter* hanya berukuran kurang lebih 2 MB. Dokumentasi *CodeIgniter* memiliki ukuran sekitar 6 MB.
3. Aplikasi yang dibuat menggunakan *CodeIgniter* bisa berjalan cepat.
4. *CodeIgniter* menggunakan pola desain *Model-View-Controller* (MVC) sehingga satu file tidak terlalu berisi banyak kode. Hal ini menjadikan kode lebih mudah dibaca, dipahami, dan dipelihara dikemudian hari.
5. *CodeIgniter* dapat diperluas sesuai dengan kebutuhan.
6. *CodeIgniter* terdokumentasi dengan baik. Informasi tentang pustaka kelas dan fungsi yang disediakan oleh *CodeIgniter* dapat diperoleh melalui dokumentasi yang disertakan di dalam paket distribusinya.
7. ***Database Management Sistem***

*Database Management**Sistem* (DBMS) merupakan perangkat lunak untuk mengendalikan pembuatan, pemeliharaan, pengolahan, dan penggunaan data yang berskala besar. Penggunaan DBMS saat ini merupakan hal yang sangat penting dalam segala aspek, baik itu dalam skala yang besar atau kecil. Sebagai contoh media sosial facebook menggunakan DBMS untuk menyimpan data-data pengguna *facebook* yang sangat banyak kedalam DBMS MySQL. Beberapa DBMS yang digunakan adalah MySQL dan MariaDB. Berdasarkan survey yang dilakukan, MySQL dan MariaDB merupakan DBMS yang banyak digunakan sebagai contoh survey yang terdapat pada db-engines.com *DB-Engines Ranking* menempatkan MySQL pada posisi ke-2 sedangkan MariaDB pada posisi ke-20 namun pada *survey* yang terdapat di serverwatch.com *Top 10 Enterprise Database Sistem Of 2016*, MariaDB menempati posisi ke-6 dan MySQL menempati posisi ke-7 (Wulan Nafesa Septine, S.T., 2019).

1. **MySQL**

MySQL merupakan sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS *Multithread* dan *multi user*. MySQL sebenarnya merupakan turunan dari salah satu konsep utama dalam *database* untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasian data dikerjakan secara mudah dan otomatis. MySQL diciptakan oleh Michael "Monty" Widenius pada tahun 1979, seorang *programmer* komputer asal Swedia yang mengembangkan sebuah sistem *database* sederhana yang dinamakan UNIREG yang menggunakan koneksi *low-level ISAM database engine* dengan *indexing* 1.

1. Kelebihan Dan Kekurangan MYSQL

Adapun kelebihan dari MySQL dalam penggunaanya dalam *database* yakni sebagai berikut:

1. Gratis sehingga MySQL dapat dengan mudah untuk mendapatkannya.
2. MySQL stabil dalam pengoprasiannya.
3. MySQL mempunyai sistem keamanan yang cukup baik.
4. Sangat mendukung transaksi dan mempunyai banyak dukungan dari komunitas.
5. Sangat *fleksibel* dengan berbagai macam *program* Perkembangan dari *MySQL* sangat cepat.

Adapun kelemahan dari MySQL dalam penggunaanya dalam *database* yakni sebagai berikut:

* 1. Sulit untuk diaplikasikan pada intansi atau perusahan dengan *database* yang besar.
  2. *Support* yang kurang.
  3. Tidak populer untuk aplikasi *game* dan *mobile*.

2. Istilah-istilah dalam *Database*

Ada beberapa istilah dalam *database* yang nantinya akan sering disebut didalam tutorial, sebelum kita masuk ke tutorial selanjutnya maka harus dapat memahami istilah-istilah dibawah ini agar lebih mudah dalam mempelajarinya. Adapun istilah-istilah tersebut sebagai berikut:

1. Pengertian Tabel

Sebuah tabel merupakan kumpulan data (nilai) yang diorganisasikan ke dalam baris (*record*) dan kolom (*field*).

1. Pengertian *FieldField*

*FieldField* merupakan kolom dari sebuah *tabel*. *Field* memiliki ukuran *type* data tertentu yang menentukan bagaimana data nantinya tersimpan.

1. Pengertian *Record* atau Data

*Record* atau Data merupakan sebuah kumpulan nilai yang saling terkait.

1. Pengertian *Key*

*Key* merupakan suatu *field* yang dapat dijadikan kunci dalam operasi tabel. Dalam konsep database, key memiliki banyak jenis diantaranya *Primary Key, Foreign Key, Composite Key, dll*.

3. Fitur-fitur MYSQL

Adapun fitur-fitur serta kapabilitas yang dimiliki oleh MySQLdalam menjalankanya yakni sebagai berikut:

1. Unjuk kerja yang tinggi dalam memproses *query* sederhana, dalam arti dapat memproses lebih banyak SQL per satuan waktu.
2. Memiliki lebih banyak tipe data seperti : *signed*/*unsigned* *integer* yang memiliki panjang data sebesar 1,2,3,4 dan 8 *byte, Float, Double, Char, Varchar, Text, Blob, Date, Time, Datetime, Timestamp, Year, Set* dan *Tipe Enum.*
3. Mendukung *field* yang dijadikan *Index*, dengan maksimal 32 *index* dalam satu tabel.
4. MYSQL memiliki beberapa lapisan keamanan, seperti *subnetmask*, nama *host*, dan izin akses *user* dengan sistem perijinan yang mendetail serta sandi atau *password* terenkripsi.
5. Konektivitas, MySQL dapat melakukan koneksi dengan *client* menggunakan *protokol* TCP/IP, *Unix soket* (UNIX), atau *Named Pipes* (NT).
6. *Multi-user*. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik
7. *Command* and *function*, MySQL memiliki fungsi dan operator secara penuh yang mendukung perintah *select* dan *where* dalam *query*.
8. *Structure* *Tabel*, MySQL memiliki struktur tabel yang lebih *fleksibel* dalam menangani *Alter Tabel* dibandingkan DBMS lainnya.

Mendukung penuh terhadap kalimat *Sql Group By Dan Order By.* Mendukung terhadap fungsi penuh *(COUNT(), COUNT(), DISTINCT() AVG(), STD(), SUM(), MAX() dan MIN() ).*

1. **XAMPP**

XAMPP adalah sebuah paket perangkat lunak (software) komputer yang sistem penamaannya diambil dari akronim kata Apache, MySQL (dulu) / MariaDB (sekarang), PHP, dan *Perl*. Sementara imbuhan huruf “X” yang terdapat pada awal kata berasal dari istilah *cross platform* sebagai simbol bahwa aplikasi ini bisa dijalankan di empat sistem operasi berbeda, seperti OS *Linux*, OS *Windows*, Mac OS, dan juga Solaris. Sejarah mencatat, *software* XAMPP pertama kali dikembangkan oleh tim proyek bernama *Apache* *Friends* dan sampai saat ini sudah masuk dalam rilis versi 7.3.9yang bisa didapatkan secara gratis dengan label GNU (*General Public License*).

Jika dijabarkan secara gamblang, masing-masing huruf yang ada di dalam nama XAMPP memiliki arti sebagai berikut ini:

1. X artinya *Cross Platform*

Merupakan kode penanda untuk *software cross platform* atau yang bisa berjalan di banyak sistem operasi.

1. A artinya Apache

Apache adalah aplikasi web server yang bersifat gratis dan bisa dikembangkan oleh banyak orang (*open source*).

1. M artinya MySQL / MariaDB

MySQL atau MariaDB merupakan aplikasi database server yang dikembangkan oleh orang yang sama. MySQL berperan dalam mengolah, mengedit, dan menghapus daftar melalui database.

1. PHP

Huruf “P” yang pertama dari akronim kata XAMPP adalah inisial untuk menunjukkan eksistensi bahasa pemrograman PHP. Bahasa pemrograman ini biasanya digunakan untuk membuat *website* dinamis, contohnya dalam website berbasis CMS WordPress.

1. P artinya Perl

Sementara itu, untuk huruf P selanjutnya merupakan singkatan dari bahasa pemrograman Perl yang kerap digunakan untuk memenuhi berbagai macam kebutuhan. Perl ini bisa berjalan di dalam banyak sistem operasi sehingga sangat fleksibel dan banyak digunakan.

Program aplikasi XAMPP berfungsi sebagai server lokal untuk mengampu berbagai jenis data *website* yang sedang dalam proses pengembangan. Dalam prakteknya, XAMPP bisa digunakan untuk menguji kinerja fitur ataupun menampilkan konten yang ada didalam *website* kepada orang lain tanpa harus terkoneksi dengan internet, atau istilahnya *website offline*. XAMPP bekerja secara *offline* layaknya *web hosting* biasa namun tidak bisa diakses oleh banyak orang. Maka dari itu, XAMPP biasanya banyak digunakan oleh para mahasiswa maupun pelajar untuk melihat hasil desain *website* sebelum akhirnya dibuat *online* menggunakan *web hosting* yang biasa dijual dipasaran.

1. ***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 2.7 Simbol-Simbol Flowchart**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO.** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | Proses | Mempresentasikan operasi. |
| 2. |  | *Input / Output* | Mempresentasikan *Input* atau *Output* data yang diproses atau informasi. |
| 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.

* 1. **Metode Perancangan *Unified Modeling Language* (UML)**

*Unified Modeling Language* (UML) adalah bahasa permodelan untuk merancang sistem berbasis *object- oriented.* UML digunakan untuk pengembangan sebuah sistem yang dapat menyampaikan alur kerja sistem dan menjelaskan tugas setiap *user* dalam sebuah sistem. UML memiliki tiga kategori diagram, yaitu *strucrture diagrams, behavior diagram,* dan *interaction diagram.* Ada beberapa jenis diagram dalam UML:

1. *Use Case Diagram*

*Use case diagram* atau diagram *use case* merupakan permodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi- fungsi itu (A. S & Shalahuddin, 2013).

**Tabel 2.8 Simbol-Simbol Use Case**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 1. |  | *Use Case* | Fungsionalitasi yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau *actor*; biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case.* |
| 2. |  | Aktor */ actor* | Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan mengguakan kata benda diawal frase nama *actor*. |
| 3. |  | *Asosiasi / association* | Komuniksi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor*. |
| 4. | << extend >> | *Ekstensi / extend* | *Relasi* *use case* tambahan ke sebuah *use case* dimana *usecase* yang Ditambahkan dapat berdiri sendiri walau tanpa use case tambahan. |
| 5. |  | *Generalisasi*  */generalizati on* | Hubungan generalisasi dan spesialiasi (umum - khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya. |
| 6. | *<<Include>>* | *Include* | Relasi *use case* tambahan ke sebuah use case di mana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. |

Sumber: AS Rosa dan M. Shalahuddin, 2013.

1. *Activity Diagram*

*Activity diagram* atau diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau *menu* yang ada pada perangkat lunak (A. S & Shalahuddin, 2013). Yang perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor.

**Tabel 2.9 Simbol-Simbol Activity Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 1. |  | *Start* / Status awal | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal |
| 2. |  | Aktivitas | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja |
| 3. |  | Percabangan / *desicion* | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |
| 4. |  | Penggabungan / *join* | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
| 5. |  | Status akhir | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir. |
| 6. |  | *swimlane* | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |

Sumber: AS Rosa dan M. Shalahuddin, 2013.

1. *Class Diagram*

*Class diagram* atau diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Diagram kelas dibuat agar *programmer* dapat membuat kelas-kelas didalam *program* perangkat lunak sesuai dengan perancangan diagram kelas (A. S & Shalahuddin, 2013). Ada beberapa simbol-simbol yang terdapat dalam diagram kelas, yaitu:

**Tabel 2.10 Simbol-Simbol Class Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 1. | |  | | --- | | **nama\_kelas** | | + atribut | | + operasi() | | Kelas | Kelas pada struktur sistem |
| 2. |  | Antarmuka / *interface* | Semua dengan konsep *interface* dalam  pemrograman b*error* ientasi objek |
| 3. |  | Asosiasi / *association* | Relasi antarkelas dengan makna umum, asoasi biasanya juga disertai dengan *multiplicity* |
| 4. |  | Asosiasi berarah  /*directed asosiasi* | Relasi antarkelas dengan makna kelas yang satu  digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity* |
| 5. |  | Generalisasi | Relasi antarmuka dengan makna generalisasi spesialisasi (umum khusus) |
| 6. |  | Kebergantungan / *dependency* | Relasi antarmuka dengan makna kebergantungan antarkelas |
| 7. |  | Agregasi / *aggregation* | Relasi antarkelas dengan makna semua bagian (*whole- part*) |

Sumber: AS Rosa dan M. Shalahuddin, 2013.

1. *Sequence Diagram*

*Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan terima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang dimiliki kelas yang diinstasiasi menjadi objek itu. Menggambar diagram sekuen juga dibutuhkan untuk melihat scenario yang ada pada *use case* (A. S & Shalahuddin, 2013).

**Tabel 2.11 Simbol-Simbol Sequence Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 1. | atau  Nama aktor | Aktor | Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan mengguakan kata benda diawal frase nama aktor |
| 2. |  | Garis hidup / *lifeline* | Menyatakan kehidupan suatu objek |
| 3. | Nama objek : nama kelas | Objek | Menyatakan objek yang berinteraksi pesan |
| 4. |  | Waktu aktif | Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya. |
| 5. |  | Pesan tipe *create* | Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat |
| 6. |  | Pesan tipe *call* | Menyatakan suatu objek memanggil opeasi/metode yang ada pada objek  lain atau dirinya sendiri |
| 7. |  | Pesan tipe *send* | Menyatakan bahwa suatu objek mengirim data / masukkan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi |
| 8. |  | Pesan tipe *return* | Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian |
| 9. |  | Pesan *type* *distroy* | Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri,  sebaiknya jika ada create makan ada *destroy* |

Sumber: AS Rosa dan M. Shalahuddin, 2013.

## ***Rational Unified Process* (RUP)**

*Rational Unified Process* (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use-case driven* dan pendekatan iteratif untuk siklus pengembangan perankat lunak. Gambar dibawah menunjukkan secara keseluruhan arsitektur yang dimiliki RUP (Rahardjo, 2018).

Dalam metode ini, terdapat empat tahap pengembangan perangkat lunak yaitu:

1. *Inception*

Pada tahap ini pengembang mendefinisikan batasan kegiatan, melakukan analisis kebutuhan user, dan melakukan perancangan awal perangkat lunak (perancangan arsitektural dan *use case*). Pada akhir fase ini, prototipe perangkat lunak versi *Alpha* harus sudah dirilis

1. *Elaboration*

Pada tahap ini dilakukan perancangan perangkat lunak mulai dari menspesifikasikan fitur perangkat lunak hingga perilisan prototipe versi *Betha* dari perangkat lunak.

1. *Construction*

Pengimplementasian rancangan perangkat lunak yang telah dibuat dilakukan pada tahap ini. Pada akhir tahap ini, perangkat lunak versi akhir yang sudah disetujui administrator dirilis beserta  dokumentasi perangkat lunak.

1. *Transition*

Instalasi *deployment* dan sosialisasi perangkat lunak dilakukan pada tahap ini. Pada tahapan ini dilakukan tahap evaluasi dari sistem yang telah direncanakan, dianalisa dan dibangun. Dari tahap ini bisa dilihat kekurangan dari sistem yang telah dibangun.

# **BAB III METODOLOGI PENELITIAN**

1. **Waktu dan Tempat Penelitian**
   * 1. **Waktu**

Waktu pelaksaan penelitian tugas akhir dilaksanakan mulai dari Bulan Januari 2021 sampai dengan Mei 2021. Rincian Kegiatan dapat dilihat pada Tabel 3.1 berikut:

**Tabel 3.1 Gannt Chart Waktu Penelitian**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Uraian** | **Waktu (2021)** | | | | | | | | | | | | | | | | | | | |
| **Januari** | | | **Februari** | | | | **Maret** | | | | | **April** | | | | **Mei** | | | |
| **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| **1** | ***Inception*** |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **2** | ***Elaboration*** |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **3** | ***Construction*** |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| **4** | ***Transition*** |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |

* + 1. **Tempat Penelitian**

Adapun tempat penelitian tugas akhir yaitu akan dilakukan di Jurusan Teknik Informatika Universitas Halu Oleo Kota Kendari Sulawesi Tenggara.

1. **Metode Pengumpulan Data**

Dalam penelitian ini metode pengumpulan data yang digunakan pada perancangan aplikasi tersebut adalah studi literatur. Metode ini dilaksanakan dengan melakukan studi kepustakaan yang relevan. Metode ini dilakukan untuk mencari sumber pelengkap yang berhubungan dengan aplikasi yang akan dibangun, yaitu dengan mencari referensi yang berkaitan dengan tugas akhir, mulai dari buku-buku, jurnal maupun artikel dan sumber-sumber lain di *internet*. Serta mengumpulkan dokumen-dokumen yang berkaitan dengan metode *levenshtein distance* yang akan diterapkan pada penelitian ini.

1. **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 berkaitan dengan algoritma *levenshtein distance*.

* + 1. **Perluasan/Perencanaan (*Elaboration*)**

Setelah menentukan ruang lingkup penelitian, tahap ini akan dilakukan perancangan dan analisis sistem menggunakan *flowchart*,yaitu *flowchart* sistem dan *flowchart* algoritma *levenshtein distance.* Pada perancangan ini, digunakan juga *UML* (*Unified Modelling Language*) yang meliputi *use case diagram,* *activity diagram, class diagram* dan *sequence diagram* dari perangkat lunak yang akan dibuat, serta desain antarmuka aplikasi yang akan dibuat.

* + 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* algoritma *levenshtein distance* sebagai metode pencarian.

* + 1. **Transisi (*Transition*)**

Pada tahap *Transition* difokuskan untuk melakukan proses pengujian terhadap sistem. Memperbaiki masalah-masalah yang muncul saat pembuatan dan setelah pengujian aplikasi.

1. **Analisis Sistem**

Analisis sistem merupakan suatu tahapan yang bertujuan untuk mengetahui dan mengamati semua yang terlibat dalam suatu sistem. Pembahasan yang ada pada analisis sistem ini yaitu analisis kebutuhan fungsional meliputi perancangan sistem menggunakan bahasa pemodelan UML (*Unified Modeling Language*), perancangan tampilan *interface* serta analisis kebutuhan fungsional dan nonfungsional yang meliputi kebutuhan perangkat keras dan perangkat lunak yang akan digunakan.

### **3.4.1. Analisis Kebutuhan Fungsional**

Analisis kebutuhan fungsional adalah segala bentuk data yang dibutuhkan oleh sistem agar sistem dapat berjalan sesuai dengan prosedur yang dibangun melalui perancangan sistem. Adapun kebutuhan fungsional dari sistem yang akan dibangun yaitu Perancangan diagram sistem menggunakan bahasa pemodelan UML (*Unified Modeling Language*) yang meliputi *flowchart* sistem, *flowchart* metode, *use case diagram*, *activity diagram*, *class diagram*, serta *sequence diagram* dan Desain *Interface* sistem.

### **3.4.2. Analisis Kebutuhan Nonfungsional**

Analisis kebutuhan nonfungsional adalah sebuah langkah dimana pembangun aplikasi menganalisis sumber daya kebutuhan untuk membangun aplikasi yang akan dibangun. Analisis kebutuhan nonfungsional yang dilakukan dibagi dalam dua tahap, yaitu anlisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak. Kebutuhan perangkat keras yaitu kebutuhan perangkat atau komponen yang dibutuhkan pada sistem dan perangkat lunak yaitu kebutuhan perangkat lunak untuk membantu agar komponen perangkat keras dapat berfungsi dan dapat dijalankan pada sistem.

#### **3.4.2.1. Kebutuhan Perangkat Keras**

Untuk menerapkan rancangan yang telah dijelaskan sebelumnya, dibutuhkan beberapa perangkat keras sebagai sarana untuk mengimplementasikan aplikasi yang dibangun. Berikut ini spesifikasi perangkat keras yang dibutuhkan.

**Tabel 3.2 Spesifikasi Perangkat Keras**

|  |  |  |
| --- | --- | --- |
| No | Nama Perangkat | Spesifikasi |
| 1. | *PC* | *Toshiba l-40* |
| 2. | *Processor* | *Intel dual core* |
| 3. | *RAM* | *4 GB* |
| 4. | *Harddisk* | *128 GB SSD* |
| 5. | *Monitor* | *Monitor 14 Inch* |

#### **3.4.2.2. Kebutuhan Perangkat Lunak**

Perangkat lunak yang digunakan pada pembangunan aplikasi glosarium teknologi informasi. Adapun rincian kebutuhan perangkat lunak dapat dilihat pada Tabel 3.3 berikut ini.

**Tabel 3.3 Spesifikasi Perangkat Lunak**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Nama Perangkat | Fungsi | Spesifikasi |
| 1. | *Windows* | *Operating System* | *Windows 10 Home* |
| 2. | *Xampp* | Universal development environment | *Xampp v3.2.2* |
| 3. | *Mysql* | *Database Management* | *Mysql 5.7* |
| 4. | *PHP* | *Web development* | *PHP 7.3* |
| 5. | *Visual Studio Code* | *Code Editor* | *VS Code v1.38* |
| 6. | *Boostrap* | *Framework CSS* | *Version 1.41* |
| 7. | *CodeIgniter* | *Framework PHP* | *Version 3.1.11* |

1. **Analisis Perancangan Sistem**

Perancangan sistem yang akan dibangun terdiri atas perancangan *flowchar*t, perancangan UML serta perancangan *user interface.*

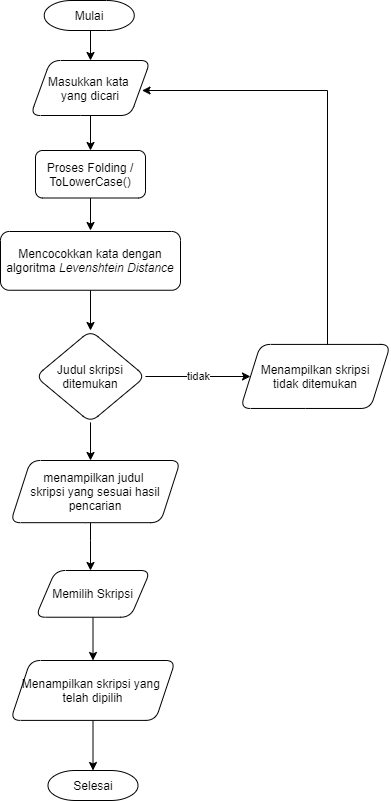
### **3.5.1. Perancangan *Flowchart***

*Flowchart* adalah simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang saling terhubung. Perancangan *flowchart* yang akan dibangun terdiri atas *flowchart* sistem dan *flowchart* Algoritma *Levenshtein Distance.*

**3.5.1.1. *Flowchart* Sistem**

Setelah menganalisis sistem, maka didapatkan *flowchart diagram* sistem informasi penelusuran tugas akhir berbasis *web* yang ditunjukkan oleh Gambar 3.1 Adapun alur kerja *flowchart diagram* sistem adalah sebagai berikut:

1. *User* memasukkan kata yang ingin dicari.
2. Kata yang dimasukkan akan melalui proses *case folding/toLowerCase* untuk mengubah kata menjadi huruf kecil.
3. Kata yang dicari akan dicocokkan menggunakan algoritma *Levenshtein Distance*.
4. Kemudian sistem akan menampilkan daftar judul skripsi yang berhubungan dengan kata kunci.
5. Jika judul tidak ditemukan, maka menampilkan pesan “skripsi tidak ditemukan” dan *User* dapat melakukan pencarian ulang. Tetapi, jika judul skripsi ditemukan, maka akan menampilkan hasil pencarian berdasarkan kata yang dimasukkan.
6. *User* memilih skripsi yang diinginkan.
7. Menampilkan halaman skripsi yang telah dipilih.

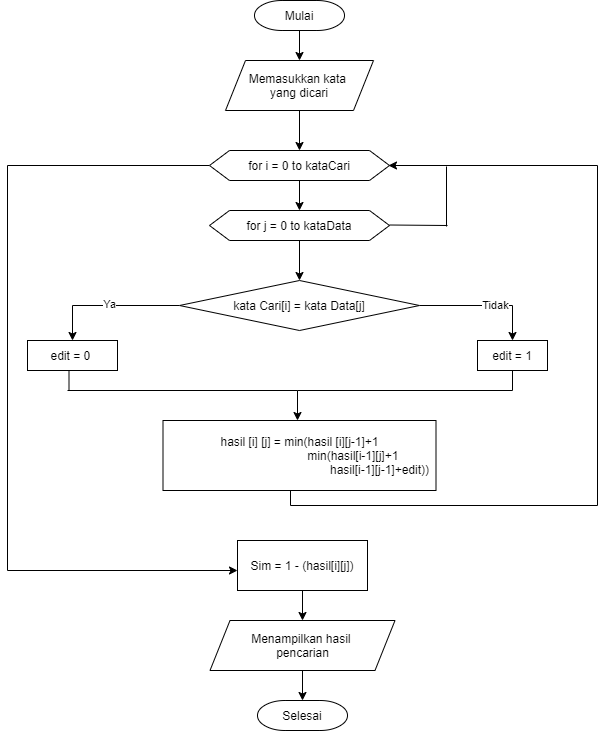
****

**Gambar 3.1 Flowchart Sistem**

**3.5.1.2. *Flowchart* Algoritma *Levenshtein Distance***

Setelah menganalisis sistem, maka didapatkan *flowchart diagram* algoritma *levenshtein distance* padasistem informasi penelusuran tugas akhir berbasis *web* yang ditunjukkan oleh Gambar 3.2 Adapun alur kerja *flowchart diagram* algoritma *levenshtein distance* adalah sebagai berikut:

* 1. Pertama *User* memasukkan kata yang ingin dicari.
  2. Sistem melakukan proses pencarian dengan syarat *for i* = panjang kata cari dimana *i=0.*
  3. Sistem melakukan proses pencarian dengan syarat *for j* = panjang kata data dimana *j=0.*
  4. Kemudian mengecek huruf pada kata cari dan kata data. Bila huruf tersebut berbeda, maka memberikan tambahan nilai 1 kata sebelumnya.
  5. Selanjutnya mencari nilai minimal dari kata cari, kata data dan kata sebelumnya.
  6. Setelah itu, huruf selanjutnya akan kembali diproses dengan cara yang sama. Apabila huruf yang akan diproses sudah tidak ada lagi, maka sistem akan menampilkan hasil pencarian.



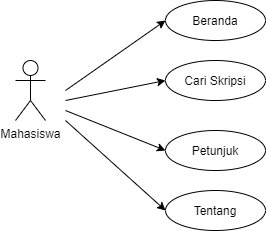
**Gambar 3.2 Flowchart Algoritma Levenshtein Distance**

### **3.5.2. *Unified Modeling Language* (UML)**

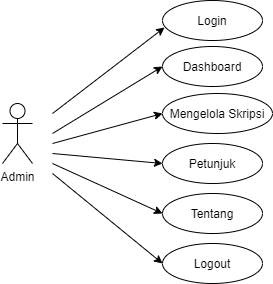
Aplikasi dibangun dengan menggunakan *Unified Modeling Language* (UML). UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram yang terdiri dari *use case diagram, activity diagram, class diagram* dan *sequence diagram.*

**3.5.2.1. *Use Case Diagram***

*Use Case Diagram* adalah sebuah diagram yang dapat merepresentasikan interaksi yang terjadi antara *user* dengan sistem. *Use Case Diagram* ini mendeskripsikan siapa saja yang menggunakan sistem dan bagaimana cara mereka berinteraksi dengan sistem. *Use Case Diagram* dari sistem yang akan dibangun adalah sebagai berikut.



**Gambar 3.3 *Use Case* *Diagram* Mahasiswa**



**Gambar 3.4 Use Case Diagram Admin**

**Tabel 3.4 Keterangan Use Case Diagram**

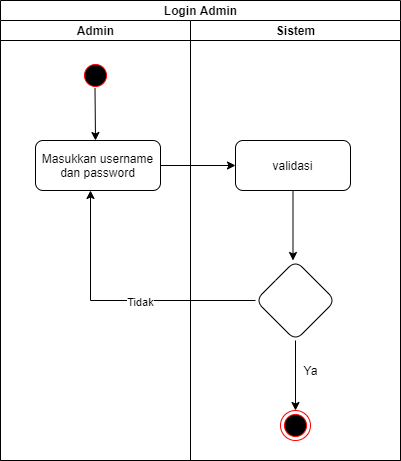
|  |  |
| --- | --- |
| **Aktor** | **Sistem** |
| *User* memilih beranda | Sistem akan menampilkan menu beranda |
| *User* memilih menu Cari Skripsi | Sistem akan menampilkan data skripsi hasil pencarian |
| *Admin* dan *user* memilih menu petunjuk | Sistem akan menampilkan menu petunjuk yang mendeskripsikan bagaimana cara pakai dari sistem. |
| *Admin* memilih menu *login* | Sistem akan menampilkan *form login* |
| *Admin* dan *user* memilih menu tentang | Sistem akan menampilkan deskripsi sistem dan menampilkan profil pembuat aplikasi. |
| *Admin* memilih menu Mengelola Skripsi | Sistem akan menyediakan menu untuk edit data, hapus data, tampilkan data dan tambah data skripsi |

**3.5.2.2. *Activity Diagram***

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

1. ***Activity Diagram Login***

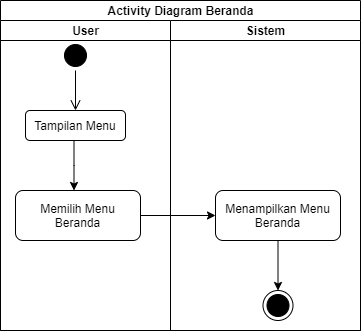
Pada Gambar 3.5 merupakan diagram aktivitas yang menunjukkan aktivitas admin ketika memasukkan username dan password, lalu sistem akan melakukan validasi.



**Gambar 3.5 Activity Diagram Login**

1. ***Activity Diagram* Beranda**

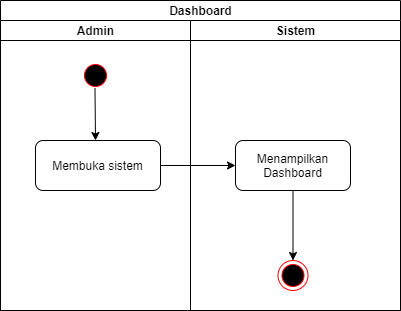
Gambar 3.6 merupakan gambar *Activity Diagram* yang menunjukkan aktivitas *user* ketika membuka aplikasi, lalu sistem akan menampilkan menu beranda yang berisikan tampilan menu beranda.

******

**Gambar 3.6 Activity Diagram Beranda**

1. ***Activity Diagram Dashboard***

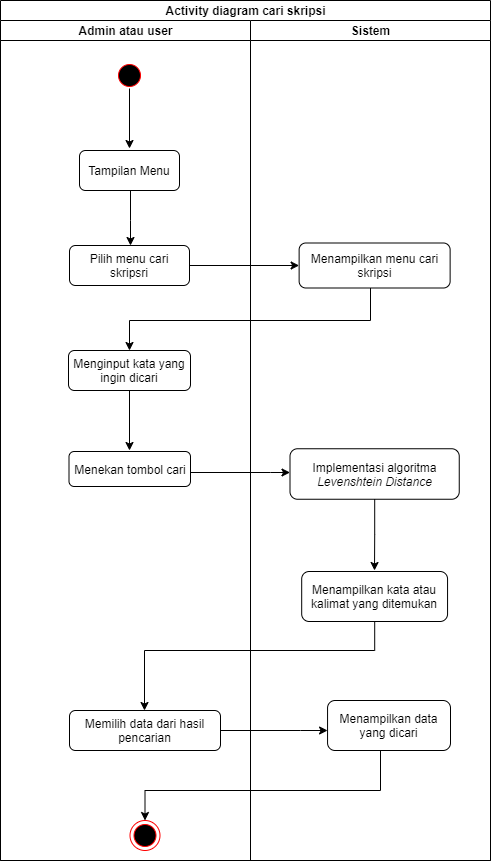
Gambar 3.7 merupakan gambar *Activity Diagram* yang menunjukkan aktivitas *admin* ketika membuka aplikasi, lalu sistem akan menampilkan menu *dashboard* yang berisikan tampilan menu *dashboard.*

******

**Gambar 3.7 Activity Diagram Dashboard**

1. ***Activity Diagram* Cari Skripsi**

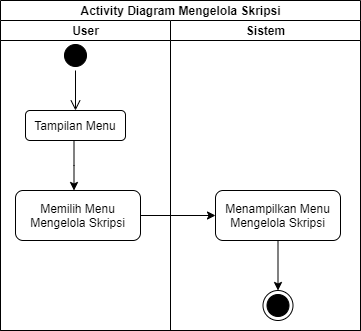
Pada Gambar 3.8 merupakan diagram aktivitas yang menunjukkan aktivitas admin atau user memilih menu Cari Skripsi lalu sistem menampilkan halaman cari Skripsi.

******

**Gambar 3.8 Activity Diagram Cari Skripsi**

1. ***Activity Diagram* Mengelola Skripsi**

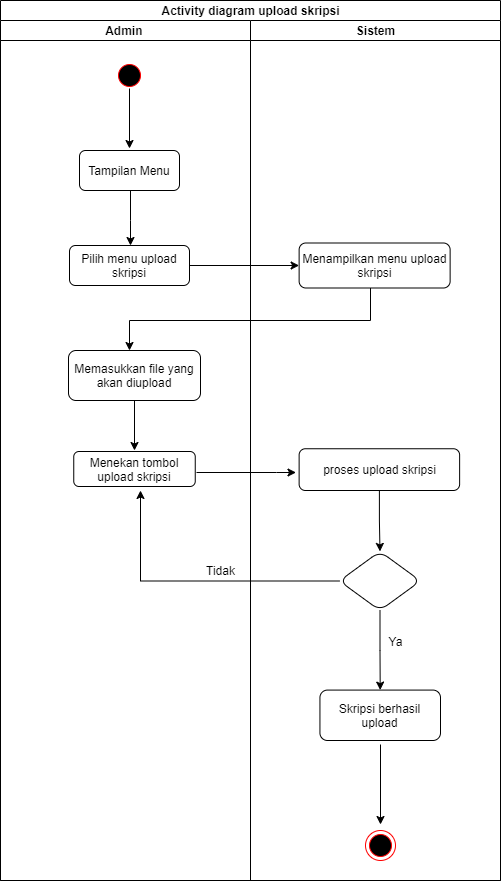
Pada Gambar 3.9 merupakan diagram aktivitas yang menunjukkan aktivitas admin memilih menu Mengelola Skripsi lalu sistem menampilkan halaman Mengelola Skripsi.

******

**Gambar 3.9 Activity Diagram Mengelola Skripsi**

1. ***Activity Diagram Upload* Skripsi**

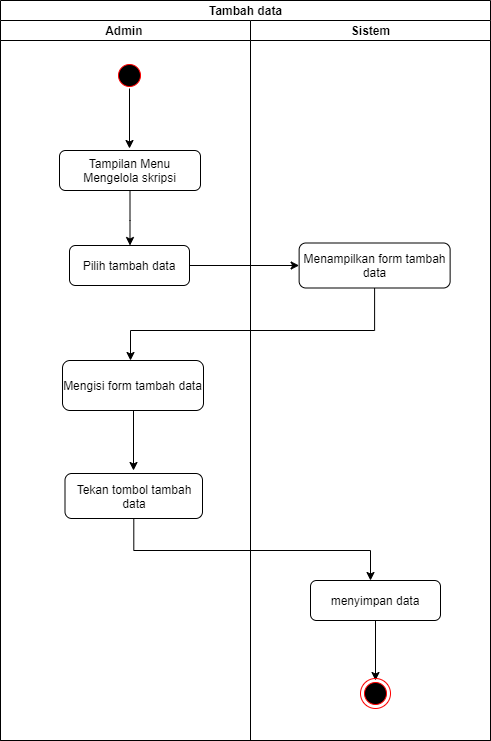
Pada Gambar 3.10 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* memilih menu *upload* Skripsi lalu sistem menampilkan halaman *upload* Skripsi.



**Gambar 3.10 Activity Diagram Upload Skripsi**

1. ***Activity Diagram* Tambah Data Megelola Skripsi**

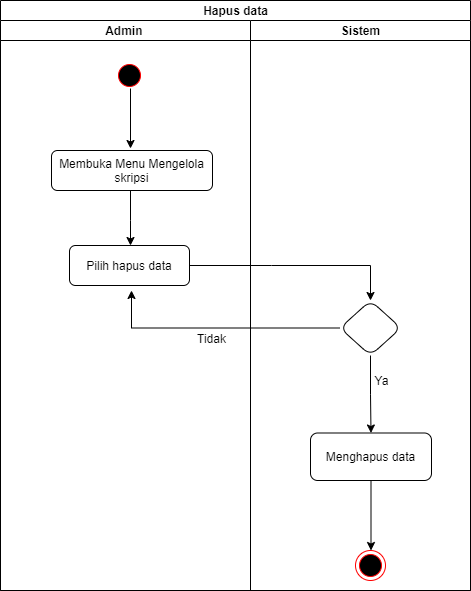
Pada Gambar 3.11 merupakan diagram aktivitas yang menunjukkan aktivitas admin memilih menu Mengelola Mahasiswa dan memilih untuk menambah data.

******

**Gambar 3.11 Activity Diagram Tambah**

1. ***Activity Diagram* Hapus Data Mengelola Skripsi**

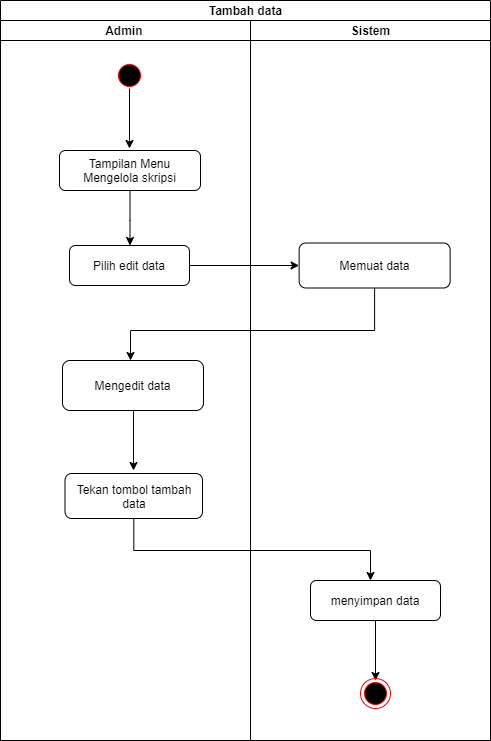
Pada Gambar 3.12 merupakan diagram aktivitas yang menunjukkan aktivitas admin memilih menu Mengelola Mahasiswa dan memilih untuk menghapus data.

****

**Gambar 3.12 Activity Diagram Hapus Data**

1. ***Activity Diagram Edit* Data Mengelola Skripsi**

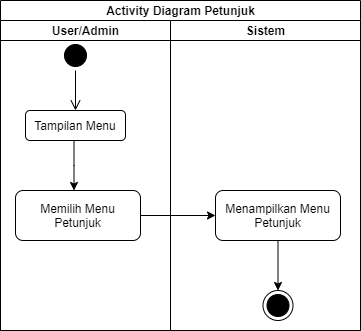
Pada Gambar 3.13 merupakan diagram aktivitas yang menunjukkan aktivitas admin memilih menu Mengelola Mahasiswa dan memilih untuk mengedit data.

****

**Gambar 3.13 Activity Diagram Edit Data**

1. ***Activity Diagram* Petunjuk**

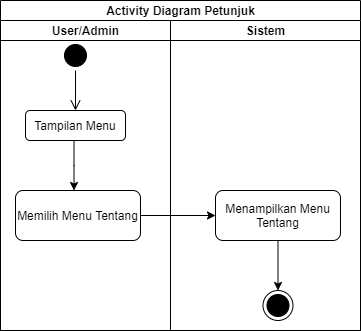
Pada Gambar 3.14 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* dan *user* memilih menu petunjuk, dimana berisi tentang deskripsi cara penggunaan sistem.

****

**Gambar 3.14 Activity Diagram Petunjuk**

1. ***Activity Diagram* Tentang**

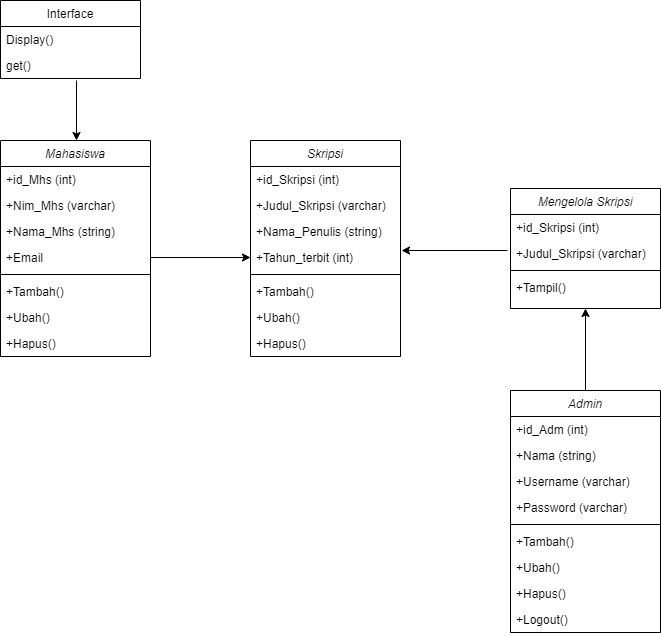
Pada Gambar 3.15 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* dan *user* memilih menu Tentang, dimana sistem akan menampilkan profil pembuat aplikasi.

****

**Gambar 3.15 Activity Diagram Tentang**

**3.5.2.3. *Class Diagram***

*Class diagram* merupakan diagram yang selalu ada dipemodelan sistem berorientasi objek. *Class diagram* menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Berikut ini adalah *class diagram* aplikasi yang akan dibangun adalah sebagai berikut.



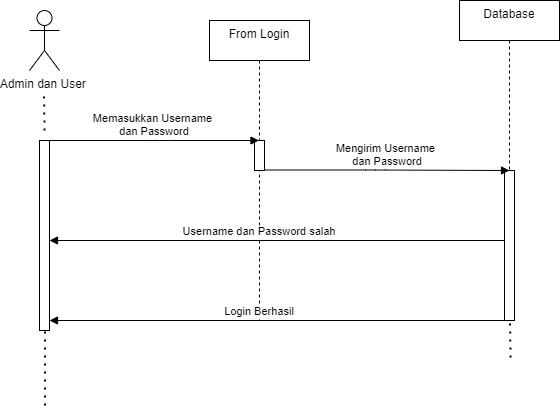
**Gambar 3.16 Class Diagram**

**3.5.2.4. *Sequence Diagram***

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem yang digambarkan terhadap waktu. Berikut ini adalah *sequence diagram* yang akan menggambarkan interkasi antar objek dan sistem.

1. ***Sequence Diagram Login***

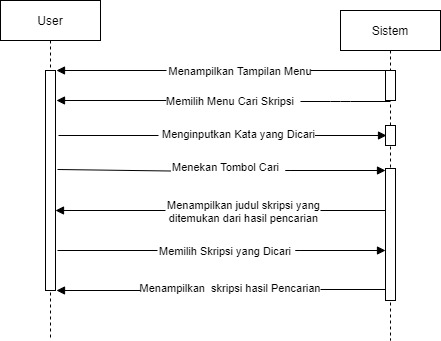
Gambar 3.17 adalah *sequence* yang menunjukkan proses *login* bagi *admin* dan *user* dimana aktor memasukkan *username* dan *password* lalu divalidasi oleh *database*.



**Gambar 3.17 Sequence Diagram Login**

1. ***Sequence Diagram* Aplikasi**

Gambar 3.18 adalah *sequence diagram* yang menjelaskan tentang korelasi antara sistem dan *user* pada Sistem Informasi Penelurusan Tugas Akhir. Dimulai dari *user* membuka sistem, lalu sistem menampilkan menu-menu, kemudian *user* memilih menu pencarian yang ingin digunakan hingga sistem menampilkan hasil dari pencarian.



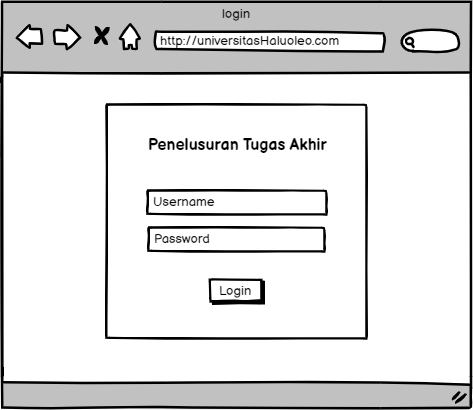
**Gambar 3.18 Sequence Diagram Aplikasi**

1. **Perancangan Antarmuka (*Interface*)**

Rancangan antarmuka pengguna atau *design user interface* merupakan penggambaran tampilan yang digunakan secara langsung oleh pengguna terhadap sistem. Rancangan *user* *interface* ini dibuat sederhana agar mudah dimengerti pengguna dan tidak ada kerumitan dalam menjalankannya sehingga mencapai tujuan perangkat lunak yang *user friendly*.

* + 1. **Menu *Login***

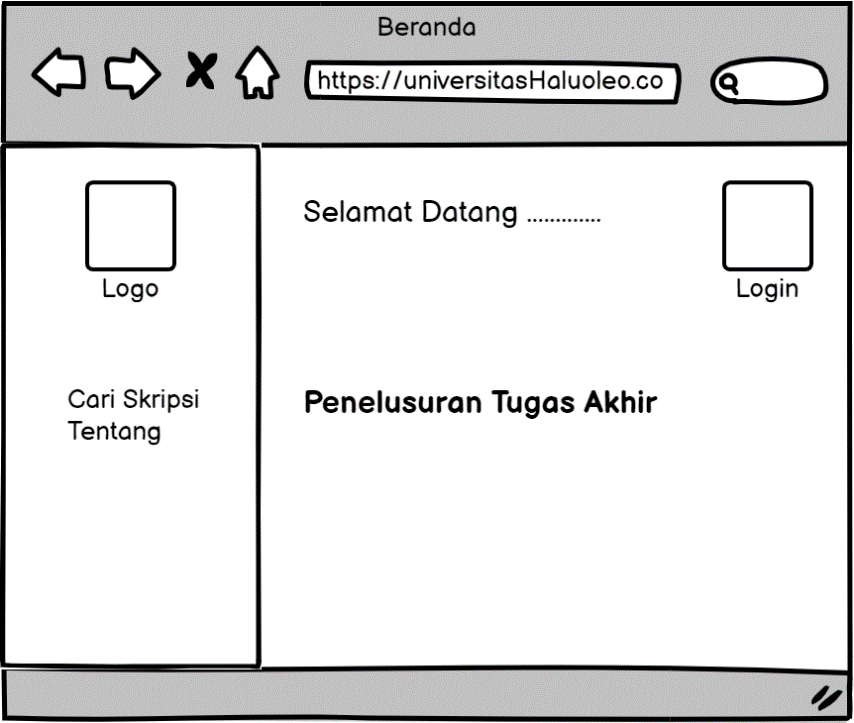
Halaman *login* pada *website* menampilkan *form login* yang terdiri dari *username* dan *password*. Setiap *field* harus diisi agar dapat melakukan *login*. Selain itu, *form* ini juga telah ditambahkan fungsi validasi agar dapat menyaring data *user* yang benar. Bentuk dari tampilan *login* dapat dilihat pada gambar berikut.



**Gambar 3.19 Menu Login**

* + 1. **Menu Beranda**

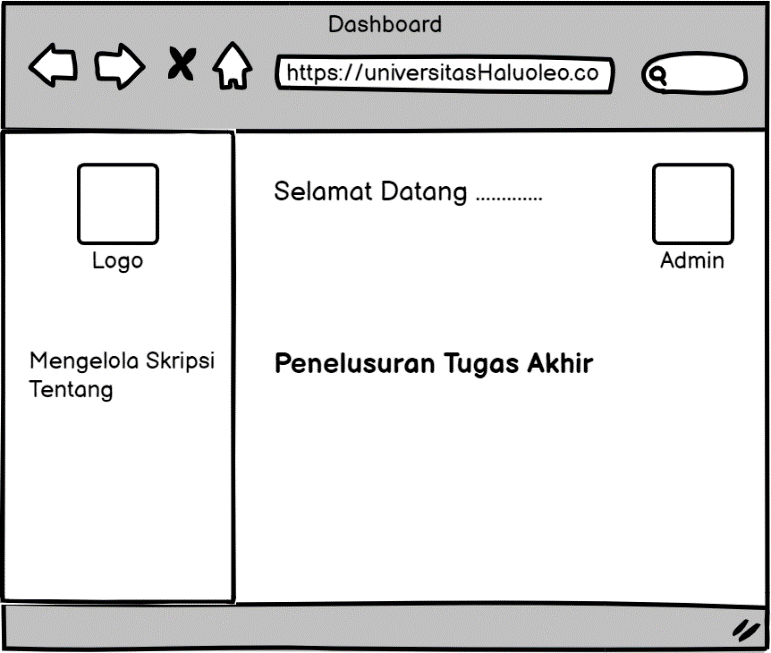
Tampilan Beranda adalah tampilan pertama yang akan digunakan pada sistem ini. Pada tampilan beranda berisi logo dan menu-menu pada sistem.



**Gambar 3.20 Menu Beranda**

* + 1. **Menu *Dashboard***

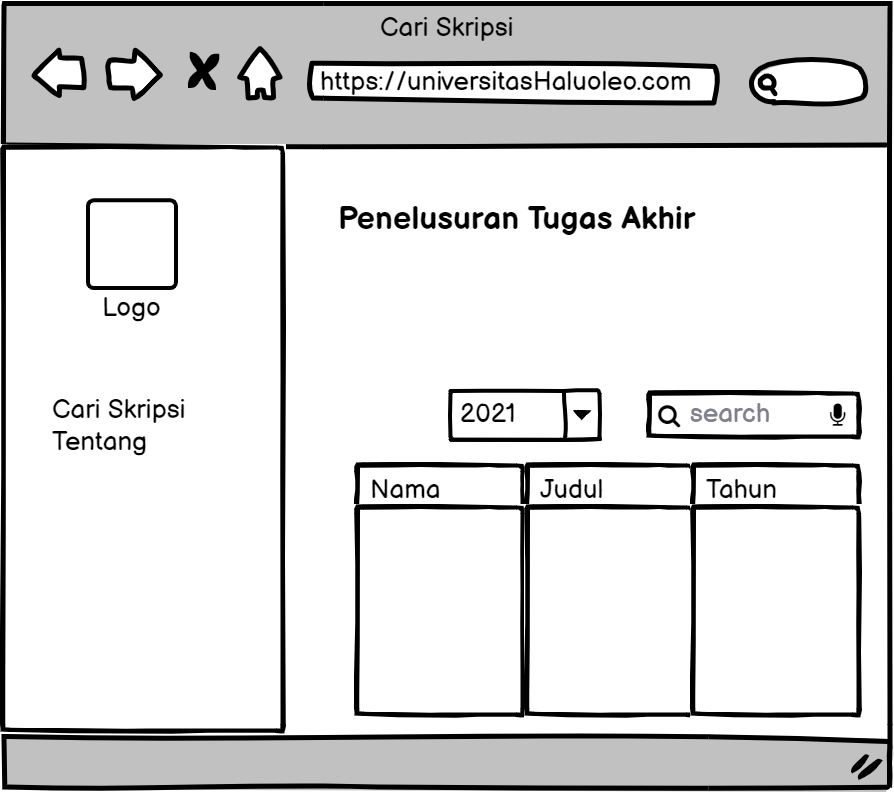
Menu *dashboard* menampilkan data sistem secara umum. Mulai dari mengelola skripsi, petunjuk, dan tentang. Bentuk dari tampilan menu *dashboard* dapat dilihat pada gambar berikut.

****

**Gambar 3.21 Menu Dashboard**

* + 1. **Menu Cari Skripsi**

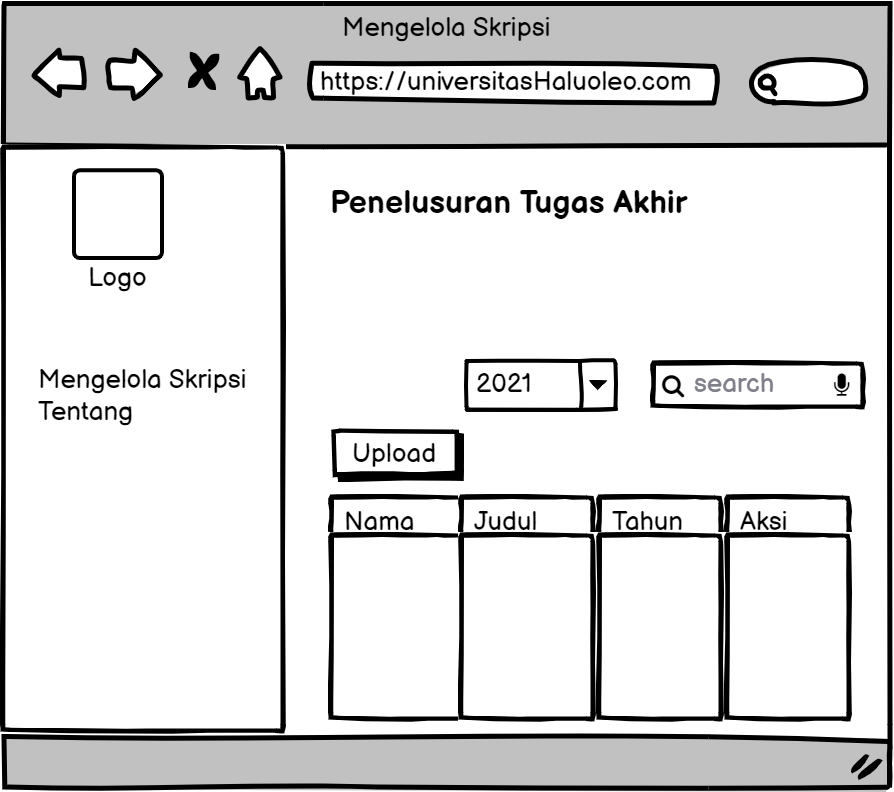
Menu cari skripsi menampilkan tombol *search* yang berguna untuk melakukan pencarian. Pada menu ini juga menampilkan nama, nim, dan tahun skripsi. Bentuk dari tampilan menu cari skripsi dapat dilihat pada gambar berikut.

****

**Gambar 3.22 Menu Cari Skripsi**

* + 1. **Menu mengelola Skripsi**

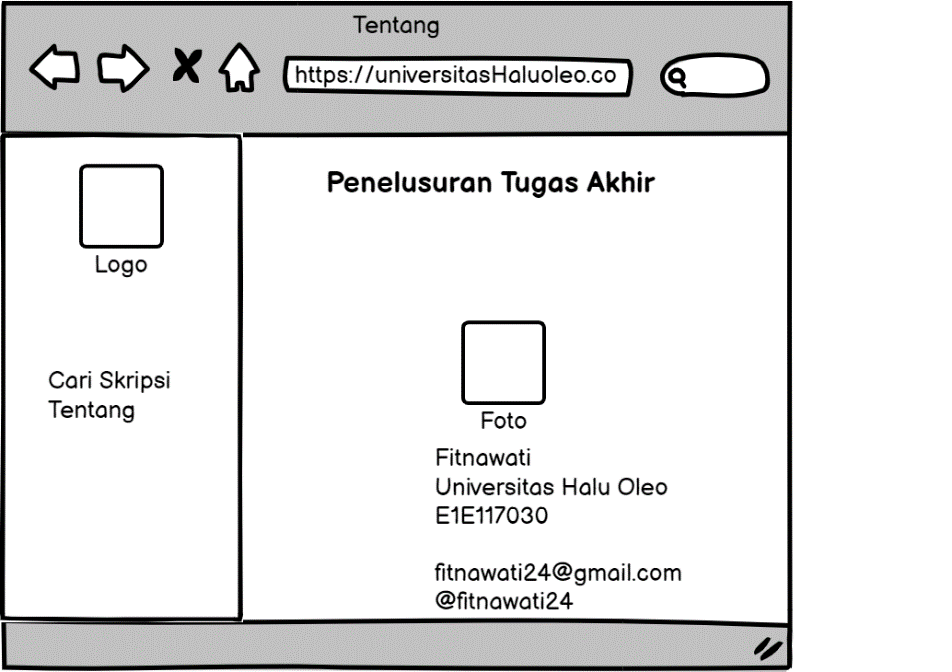
Menu Mengelola Skripsi menampilkan data skripsi dan juga admin bisa melakukan upload skripsi, edit dan hapus skripsi. Bentuk dari tampilan menu mengelola skripsi dapat dilihat pada gambar berikut.

****

**Gambar 3.23 Menu Mengelola Skripsi**

* + 1. **Menu Tentang**

Pada saat *user* memilih menu tentang aplikasi, sistem akan menampilkan deskripsi aplikasi. Disini akan ditampilkan biodata dari pembuat aplikasi.



**Gambar 3.24 Menu Tentang**

**DAFTAR PUSTAKA**

Abdurrahman, 2016. “APLIKASI KAMUS BAHASA INDONESIA – BUGIS BERBASIS WEB DENGAN METODE SEQUENTIAL SEARCH.” 3(September):246–58.

Andriansyah, Faiz Rizky. 2015. “IMPLEMENTASI WEB SEMANTIK UNTUK APLIKASI PENCARIAN.” 2(1):11–18.

Anggaeni, Puspita and Bambang Sujatmiko. 2013. “Sistem Informasi Tugas Akhir Berbasis Web (Studi Kasus D3 Manajemen Informatika Te Ft Unesa).” *Jurnal Manajemen Informatika* 2(2):37–45.

Ewolf, *Community*, 2011.  *Indeks Lengkap Syntax*, MediaKom, Yogyakarta.

Ferdianto, Y. 2013. *Pengertian PHP*. https://www.carawebs.info/2013/04/apa-itu-php.html.

Fitrianti, Uli, Mutammimul Ula, 2017. Teknik Informatika, Sistem Informasi, Universitas Malikussaleh, Aceh Utara, Levenshtein Distance, Knuth Morris Pratt, and Aplikasi Asmaul Husna. n.d. “ANDROID.” 25–60.

Purba, Andry Hery and Zakarias Situmorang. 2017. “Analisis Perbandingan Algoritma Rabin-Karp Dan Levenshtein Distance Dalam Menghitung Kemiripan Teks.” 02:24–32.

Rahardjo, A. 2018. *Pengertian RUP (Rational Unified Process)*. https://medium.com/@andrerahardjo/pengertian-rup-rational-unified-process-1bec9c664458

Sa’diah, Tus Halimah. 2017. “Implementasi Algoritma Knuth-Morris-Pratt Pada Fungsi Pencarian Judul Tugas Akhir Repository.” *Jurnal Komputasi* 14(1):115–25.

Sarker, Jahangir H. and Hussein T. Mouftah. 2011. “Secured Operating Regions of Slotted ALOHA in the Presence of Interfering Signals from Other Networks and DoS Attacking Signals.” *Journal of Advanced Research*.

Teknik, Pendidikan, Informatika Dan, Fakultas Teknik, and Universitas Negeri Semarang. 2015. “Implementasi Fitur Autocomplete Dan Algoritma Levenshtein Distance Untuk Meningkatkan Efektivitas Pencarian Kata Di Kamus Besar Bahasa Indonesia.”

Wulan Nafesa Septine, S.T., MMSI. 2019. “DATABASE MANAGEMENT SYSTEM (DBMS) MARIA DB. Retrieved (https://www.smktarunabangsa.sch.id/artikel/detail/database-management system-dbms-maria-db#:~:text=Berdasarkan survey yang dilakukan%2C MySQL,Top 10 Enterprise Database System).