

PROPOSAL

SISTEM PENDETEKSI PLAGIARISME PADA ABSTRAK SKRIPSI
DENGAN ALGORITMA *RABIN KARP*.

(Studi Kasus : Jurusan Teknik Informatika Universitas Halu Oleo)

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



NUR'AZIZA TADJUDDIN
E1E117023

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HALU OLEO
2021

Halaman Pengesahan

DAFTAR ISI

HALAMAN SAMPUL.....	i
HALAMAN PENGESAHAN.....	ii
DAFTAR ISI.....	iii
DAFTAR TABEL	v
DAFTAR GAMBAR.....	vi
BAB I PENDAHULUAN.....	1
1.1 Latar belakang.....	1
1.2 Rumusan masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan penelitian	3
1.5 Manfaat penelitian	3
1.6 Sistematika penulisan.....	3
1.7 Tinjauan pustaka	4
BAB II LANDASAN TEORI	7
2.1 Pengertian plagiarisme	7
2.1.1 Tipe-tipe Plagiarisme	8
2.1.2 Metode Pendeteksi Plagiarisme	8
2.2 Pengertian <i>Information Retrieval</i> (IR)	9
2.2.1 <i>Proses Indexing</i>	9
2.2.2 <i>Proses Searching</i>	10
2.2.3 <i>Model Information Retrieval</i> (IR)	11
2.3 <i>American Standard Code for Information Interchange</i> (ASCII).....	12
2.4 Algoritma <i>Rabin Karp</i>	13
2.5 <i>Hyper Text Markup Language</i> (HTML)	20
2.6 <i>Hypertext Preprocessor</i> (PHP).....	21
2.7 MySQL	22
2.8 <i>Database Management System</i> (DBMS)	22
2.9 <i>Unified Modeling Language</i> (UML).....	23

BAB III METODOLOGI PENELITIAN	26
3.1 Waktu dan Tempat Penelitian	26
3.1.1 Waktu	26
3.1.2 Tempat Penelitian.....	26
3.2 Metode Pengumpulan Data.....	26
3.3 Metode Pengembangan Sistem	30
3.3.1 Permulaan (<i>Inception</i>).....	30
3.3.2 Perluasaan / Perencanaan (<i>Elaboration</i>)	30
3.3.3 Konstruksi (<i>Construction</i>).....	30
3.3.4 Transisi (<i>Transition</i>).....	30
3.4 Analisis Kebutuhan Sistem	31
3.4.1 Kebutuhan Fungsional	31
3.4.2 Kebutuhan Nonfungsional	31
3.5 Analisis Perancangan Sistem	32
3.5.1 <i>Unified Modeling Language</i> (UML)	32
3.5.2 Perancangan Antarmuka (<i>Interface</i>)	45
DAFTAR PUSTAKA	49

DAFTAR TABEL

Tabel 2.1 ASCII	13
Tabel 2.2 Komponen-komponen <i>use case diagram</i>	24
Tabel 2.3 Komponen-komponen <i>activity diagram</i>	25
Tabel 2.4 Simbol-Simbol <i>Class Diagram</i>	26
Tabel 2.5 Simbol-Simbol <i>Sequence Diagram</i>	27
Tabel 3.1 <i>Gannt Chart Waktu Penelitian</i>	26
Tabel 3.2 Spesifikasi Perangkat	31
Tabel 3.3 Spesifikasi Perangkat Lunak	32
Tabel 3.4 Deskripsi <i>Case Login (Admin)</i>	33
Tabel 3.5 Deskripsi <i>Case Mengolah Abstrak (Admin)</i>	34
Tabel 3.6 Deskripsi <i>Case Lihat daftar abstrak</i>	35
Tabel 3.7 Deskripsi <i>Case Cek plagiarisme</i>	35

DAFTAR GAMBAR

Gambar 3.1 <i>Use Case Diagram</i>	33
Gambar 3.2 <i>Activity diagram login</i>	36
Gambar 3.3 <i>Activity Diagram</i> Daftar abstrak	37
Gambar 3.4 <i>Activity Diagram</i> edit atau hapus abstrak.....	38
Gambar 3.5 <i>Activity Diagram</i> Tambah Akun	39
Gambar 3.6 <i>Activity Diagram</i> Cek plagiarisme abstrak.....	40
Gambar 3.7 <i>Class diagram</i>	41
Gambar 3.8 <i>Sequence Diagram Login</i>	42
Gambar 3.9 <i>Sequence Diagram</i> Edit dan hapus	43
Gambar 3.10 <i>Sequence Diagram</i> Tambah Akun	44
Gambar 3.11 <i>Sequence Diagram</i> Cek plagiarisme	45
Gambar 3.12 Menu <i>Login</i>	46
Gambar 3.13 Daftar abstrak	46
Gambar 3.14 Menu Kelola Abstrak	47
Gambar 3.15 Menu Cek plagiarisme Abstrak.....	48
Gambar 3.16 Menu Tentang	48

BAB I

PENDAHULUAN

1.1 Latar belakang

Informasi dapat beredar dengan cepat dan luas dari berbagai belahan dunia melalui Internet. Kemudahan akses internet yang sudah bisa dinikmati di mana saja merupakan salah satu faktor utama yang menyebabkan informasi dapat dengan mudah diperoleh. Namun, dikarenakan kemudahan informasi yang didapat tersebut, tidak sedikit tindak kejahatan yang terjadi di dunia maya contohnya adalah plagiarisme.

Plagiarisme sering di jumpai dalam sektor akademis maupun non akademis. Dalam sektor akademis, plagiarisme di anggap sebagai tindak pidana serius karena di anggap pengambilan karangan, pendapat, ide dan gagasan orang lain. Plagiarisme atau penjiplakan karya tulis orang lain dalam dunia akademik dipicu oleh banyak faktor. Salah satu faktor pemicunya adalah penulis (mahasiswa) ingin segera menyelesaikan skripsinya agar bisa meraih gelar akademik secepatnya tanpa harus bekerja keras sesuai proses riset dan penulisan ilmiah yang benar. Salah satu plagiarisme yang sering terjadi di kalangan mahasiswa adalah plagiarisme di dalam penulisan skripsi, yaitu pada Abstrak Skripsi.

Abstrak adalah suatu ringkasan atau rangkuman isi karya tulis ilmiah, bertujuan untuk memudahkan pembaca dalam memahami maksud penulisan karya ilmiah tersebut. Di dalam dunia pendidikan tulisan abstrak selalu digunakan oleh lembaga sebagai informasi pada halaman awal saat penulisan karya ilmiah mengenai suatu penelitian, seperti pada skripsi, tesis, jurnal, dan sebagainya. Struktur penulisan abstrak saat ini berbeda-beda, misalnya setiap lembaga universitas memiliki standar penulisan abstrak masing-masing. Dalam skripsi dan tesis abstrak di tulis dengan bahasa Indonesia, dan dalam bahasa Inggris.

Tindakan plagiarisme secara perlahan harus di cegah dan dihilangkan dengan melakukan pendeteksian plagiat secara manual maupun dengan memanfaatkan

metode pencocokan *string*. Namun pendeteksian secara manual memiliki masalah yang cukup besar yaitu sangat tidak memungkinkan melakukan pendeteksian dokumen lain yang berjumlah ratusan bahkan sampai ribuan. Dengan demikian melakukan pendeteksian plagiarisme secara manual sangat tidak efektif sehingga Algoritma *Rabin Karp* dapat menjadi solusi untuk menyelesaikan hal tersebut.

Algoritma *Rabin Karp* merupakan algoritma pencarian *string* yang ditemukan oleh Michael Rabin dan Richard Karp. Algoritma *Rabin Karp* merupakan algoritma pencarian *string* yang menggunakan fungsi hashing untuk membandingkan *string* yang dicari (m) dengan *string* yang dibandingkan (n). Fungsi hash adalah fungsi matematis yang digunakan untuk mengubah data menjadi bilangan bulat yang relatif kecil yang dapat berfungsi sebagai indeks pada *array*. Fungsi *hashing* merupakan proses inti pada perhitungan algoritma *Rabin Karp*. *Hashing* merupakan representasi ASCII yang menggantikan atau mentransformasikan karakter atau tanda baca menjadi sebuah nilai atau angka (Suryati, Wibisono, & Wihardi, 2018).

Berdasarkan uraian diatas, peneliti akan melakukan implementasi algoritma *Rabin Karp* ke dalam sistem berbasis web agar dapat mendeteksi seberapa besar persentase plagiarisme atau kemiripan Abstrak skripsi mahasiswa.

1.2 Rumusan masalah

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

1. Bagaimana mendesain dan mengembangkan Sistem pendeteksian plagiarisme?
2. Bagaimana menerapkan algoritma *Rabin Karp* yang akan digunakan dalam pengembangan sistem pendeteksian plagiarisme?

1.3 Batasan Masalah

Adapun batasan masalah dalam sistem ini adalah:

1. Pembuatan sistem pendeteksi plagiarisme abstrak menggunakan Algoritma *Rabin-Karp*.

2. Data yang di uji merupakan abstrak skripsi yang menggunakan bahasa indonesia.
3. Tidak memperhatikan kesalahan ejaan/penulisan pada dokumen.
4. Tidak memperhatikan sinonim/persamaan kata.

1.4 Tujuan penelitian

Adapun Tujuan dari penelitian ini adalah:

1. Mendesain dan membangun sistem pendeteksi plagiarisme dalam abstrak skripsi dengan algoritma *Rabin Karp*.
2. Mengurangi tingkat plagiarisme pada abstrak
3. Mendeteksi tingkat kesamaan Abstrak dengan *algoritma Rabin Karp*.

1.5 Manfaat penelitian

Adapun manfaat dari penelitian ini adalah:

1. Memudahkan untuk mendeteksi plagiarisme abstrak dengan algoritma *Rabin Karp*.
2. Dapat membantu sebagai bahan pertimbangan dalam menentukan plagiarisme.
3. Dapat menentukan persentase kemiripan (*similarity*) antara dokumen yang diuji dengan dokumen asli oleh sistem.

1.6 Sistematika penulisan

Sistematika penulisan tugas akhir ini terdiri atas beberapa bagian utama sebagai berikut:

BAB I Pendahuluan

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan dan tinjauan pustaka.

BAB II Landasan teori

Pada bab ini membahas teori-teori pendukung yang berkaitan dengan skripsi yang akan di buat. Teori yang akan di angkat yaitu mengenai deteksi plagiarisme dengan algoritma *Rabin Karp*.

BAB III Metodologi penelitian

Pada bab ini membahas tentang tahapan penelitian, tahapan pengumpulan data, analisa kebutuhan sistem, perancangan perangkat lunak, implementasi pengujian sistem dan analisa akhir.

1.7 Tinjauan pustaka

Menurut penelitian (Filcha & Hayaty, 2019) yang berjudul “Implementasi Algoritma *Rabin-Karp* untuk pendeteksi pada dokumen tugas Mahasiswa” menjelaskan Berdasarkan hasil dan pembahasan maka peneliti dapat mengambil kesimpulan diantaranya Algoritma *Rabin-Karp* berhasil diimplementasikan pada sistem pendeteksi plagiarisme dokumen tugas mahasiswa. Sistem ini berhasil menampilkan persentase kemiripan dokumen tugas antar mahasiswa. Hasil perhitungan akurasi dengan *confusion matrix* pada sistem pendeteksi plagiarisme dokumen tugas mahasiswa adalah 90% yang diperoleh dari 20 perbandingan dokumen tugas mahasiswa. Algoritma yang digunakan pada sistem pendeteksi plagiarisme dokumen tugas mahasiswa tidak memiliki perbedaan persentase saat urutan perbandingan diubah.

Menurut penelitian (Putra & Sularno, 2019) yang berjudul “Penerapan Algoritma *Rabin Karp* Dengan Pendekatan *Synonym Recognition* Sebagai Antisipasi Plagiarisme Pada Penulisan Skripsi” menjelaskan, Berdasarkan hasil analisa dan penerapan yang sudah dilakukan pada bab sebelumnya, maka dapat disimpulkan bahwa Telah dibuat suatu sistem berbasis web yang dapat digunakan untuk pendeteksian plagiarisme terhadap suatu dokumen teks dengan menggunakan algoritma *Rabin Karp*. Tindakan plagiarisme dapat dilakukan dengan mengubah beberapa bagian bahkan mungkin secara keseluruhan dengan cara mengubah kata-kata dengan sinonim daripada kata-kata tersebut. Algoritma *Rabin Karp* yang menggunakan metode *Synonym Recognition* memiliki akurasi penentuan nilai persentase *Similarity* yang lebih akurat bila dibandingkan dengan algoritma *Rabin Karp* tanpa *Synonym Recognition*. Karena walaupun kata pada dokumen uji diganti sedemikian rupa namun memiliki makna yang sama akan terdeteksi oleh sistem ini. Hasil dari *Synonym Recognition* sangat bergantung pada

banyaknya sinonim yang diinputkan ke dalam kamus sinonim pada database. Nilai *K-gram* yang lebih kecil akan menghasilkan akurasi nilai similarity yang lebih baik dibandingkan dengan nilai *K-gram* yang lebih besar. Untuk nilai basis, tidak semua angka bisa digunakan. Karena di beberapa kasus nilai basis yang salah akan mengakibatkan nilai hash yang dihasilkan akan sama dengan nilai hash lain yang memiliki kata berbeda. Dokumen uji dinyatakan terduplikasi apabila nilai persentase *Similarity* nya diatas 50%.

Menurut penelitian (Pamungkas, Teknologi, Universitas, & Barat, n.d.) yang berjudul “Deteksi Similaritas Dokumen Ilmiah Menggunakan Algoritma *Rabin-Karp*” menjelaskan, Beberapa kesimpulan yang dapat diambil pada penelitian ini antara lain : Pemisahan kalimat pada dokumen input menggunakan *pattern regex* telah berhasil dilakukan dengan baik. Pencarian kalimat pada dokumen input pada mesin pencari *google* dan *crawling* data dari hasil pencarian tersebut telah berhasil dilakukan. Proses Stemming untuk mendapatkan kata dasar dari sebuah kata dengan menggunakan algoritma stemming untuk dokumen berbahasa Indoneisa Nazief–Adriani pada sebuah kalimat input maupun konten hasil pencarian kalimat tersebut telah berhasil dilakukan dengan baik. Penelitian ini berhasil melakukan pembentukan *K-gram* dan *hashing* dari *string output* tahap *preprocessing* telah berhasil dilakukan dengan baik. Proses perhitungan jumlah hash yang sama pada kalimat *query* dan 10 konten hasil pencarian pada mesin pencari *google* telah berhasil dilakukan dengan baik. Penelitian ini berhasil membentuk aplikasi pendeteksi similaritas tulisan ilmiah menggunakan sumber hasil pencarian mesin pencari *google*. Pengembangan lebih lanjut yang dapat dilakukan pada penelitian selanjutnya antara lain dengan mengimplementasikan algoritma yang dapat mempertimbangkan posisi dan urutan kata, sinonim kata dari sebuah kalimat, dan kemiripan semantik dari sebuah dokumen input sehingga menambah nilai akurasi dari similaritas dokumen dengan dokumen pembanding. Selain itu, dalam teknik *crawling* dapat dikembangkan dengan melakukan *crawling* dari tiap website yang diketahui memiliki similaritas dengan kalimat input secara utuh, tidak hanya dari hasil pencarian *google search*.

Menurut penelitian (Adi, 2018) yang berjudul “Penerapan Algoritma *Rabin Karp* Untuk Mendeteksi Kemiripan Judul Skripsi” menjelaskan, Beberapa kesimpulan yang dapat diambil pada penelitian ini antara lain : Sistem dalam membandingkan file memberikan hasil berupa prosentase *similarity*. Preprocessing sangat membantu sekali dalam proses *Rabin Karp*. Karena dengan melakukan preprocessing maka dapat mengurangi volume dataset tanpa mengurangi esensi nilai dataset tersebut, sehingga proses parsing jumlah k-nya menjadi lebih sedikit. Jika file tidak mengalami proses *preprocessing*, maka waktu yang diperlukan semakin kecil namun nilai similaritas-nya berkurang.

Menurut penelitian (Suryati et al., 2018) yang berjudul “Aplikasi Deteksi Plagiarisme Dokumen Skripsi Dengan Algoritma *Rabin Karp*” menjelaskan beberapa kesimpulan yang dapat di ambil pada penelitian ini antara lain : Algoritma *Rabin-Karp* dapat digunakan untuk menghitung kemiripan antara dokumen input dengan setiap dokumen pada database. Dengan melihat perbandingan hasil evaluasi dari berbagai skenario eksperimen yang telah dilakukan, dapat disimpulkan bahwa semakin besar gram yang digunakan maka semakin tinggi nilai error persentase kemiripannya. Dari hasil evaluasi juga dapat disimpulkan bahwa algoritma *Rabin-Karp* tidak dapat mengatasi plagiat dengan teknik modifikasi kata dengan kata sinonimnya, karena pada algoritma *Rabin-Karp* nilai hash akan berbeda jika kata tersebut dimodifikasi. Perlu penelitian lanjutan untuk mengatasi plagiarisme yang menggunakan modifikasi kata dan sinonim dan dengan jumlah data yang lebih banyak.

BAB II

LANDASAN TEORI

2.1 Pengertian plagiarisme

Plagiarisme atau sering disebut plagiat adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri. Plagiat dapat dianggap sebagai tindak pidana karena mencuri hak cipta orang lain tanpa meminta izin dan menyertakan sumber yang dicatutnya. Di dunia pendidikan, pelaku plagiarisme dapat mendapat hukuman berat seperti dikeluarkan dari sekolah / universitas, denda berupa uang dan bahkan hukuman penjara. Pelaku plagiat disebut sebagai plagiator. Sejak abad ke-19, plagiat atau plagiarisme telah menjadi masalah serius dalam dunia pendidikan yang masih berlangsung sampai saat ini. Ini tentu memerlukan pertimbangan khusus karena memiliki dampak yang tidak sehat dalam dunia pendidikan. (Aziz, Lulu A., Ana I., 2015)

Berikut ini adalah hal-hal yang tergolong kedalam tindakan plagiarisme, antara lain:

1. Mengakui tulisan orang lain sebagai tulisan sendiri.
2. Mengakui gagasan orang lain sebagai pemikiran sendiri.
3. Mengakui temuan orang lain sebagai kepunyaan sendiri.
4. Mengakui karya kelompok sebagai kepunyaan atau hasil sendiri.
5. Menyajikan tulisan yang sama dalam kesempatan yang berbeda tanpa menyebutkan asal-usulnya.
6. Meringkas dan memparafrasekan (mengutip tak langsung) tanpa menyebutkan sumbernya.
7. Meringkas dan memparafrasekan dengan menyebut sumbernya, tetapi rangkaian kalimat dan pilihan katanya masih terlalu sama dengan sumbernya.

2.1.1 Tipe-tipe Plagiarisme

Plagiarisme tidak selalu disengaja atau mencuri sesuatu dari orang lain. Praktik ini dapat bersifat tidak disengaja, kebetulan, dan dapat mencakup pencurian sendiri (*self stealing*). Berikut ini beberapa tipe plagiarisme.

a. Kebetulan (*accidental*)

Praktik plagiarisme ini dapat terjadi karena kurangnya pengetahuan akan plagiarisme dan pemahaman mengenai penulisan referensi.

b. Tidak disengaja (*unintentional*)

Ketersediaan informasi dalam jumlah yang sangat besar mempengaruhi pemikiran sehingga ide yang sama dapat dihasilkan secara tertulis maupun lisan sebagai milik pribadi.

c. Disengaja (*intentional*)

Tindakan menyalin sebagian atau keseluruhan hasil karya orang lain secara sengaja tanpa mengikutsertakan nama pemilik hasil karya.

d. Diri sendiri (*self plagiarism*)

Penggunaan hasil karya yang dibuat diri sendiri dalam bentuk lain tanpa menunjuk hasil karya asli.

2.1.2 Metode Pendeteksi Plagiarisme

Metode Pendeteksi Plagiarisme dibagi menjadi tiga bagian yaitu metode perbandingan teks lengkap, metode dokumen fingerprinting, dan metode kesamaan kata kunci. Metode pendeteksi plagiarisme dapat digambarkan sebagai berikut :

a. Perbandingan Teks Lengkap

Metode ini diterapkan dengan membandingkan semua isi dokumen. Dapat diterapkan untuk dokumen yang besar. Pendekatan ini membutuhkan waktu yang lama tetapi cukup efektif, karena kumpulan dokumen yang diperbandingkan adalah dokumen yang disimpan pada penyimpanan lokal. Metode perbandingan teks lengkap tidak dapat diterapkan untuk kumpulan dokumen yang tidak terdapat pada dokumen lokal. Algoritma yang digunakan pada metode ini adalah algoritma *Brute-Force*, algoritma *edit distance*, algoritma *Boyer Moore* dan algoritma *lavenshtein distance*.

b. Dokumen *Fingerprinting*

Dokumen *fingerprinting* merupakan metode yang digunakan untuk mendeteksi keakuratan salinan antar dokumen, baik semua teks yang terdapat di dalam dokumen atau hanya sebagian teks saja. Prinsip kerja dari metode dokumen *fingerprinting* ini adalah dengan menggunakan teknik hashing. Teknik hashing adalah sebuah fungsi yang mengkonversi setiap *string* menjadi bilangan. Misalnya *Rabin Karp*, *Winnowing* dan *Manber*.

c. Kesamaan Kata Kunci

Prinsip dari metode ini adalah mengekstrak kata kunci dari dokumen dan kemudian dibandingkan dengan kata kunci pada dokumen yang lain. Pendekatan yang digunakan pada metode ini adalah teknik dot.

2.2 Pengertian *Information Retrieval* (IR)

Information Retrieval merupakan bagian dari *computer science* yang berhubungan dengan pengambilan informasi dari dokumen-dokumen yang didasarkan pada isi dan konteks dari dokumen-dokumen itu sendiri. Berdasarkan referensi dijelaskan bahwa *Information Retrieval* merupakan suatu pencarian informasi yang didasarkan pada suatu *query* yang diharapkan dapat memenuhi keinginan user dari kumpulan dokumen yang ada.

Beberapa contoh aplikasi dari *Information Retrival* adalah search engine atau mesin pencarian yang terdapat pada jaringan internet. Pengguna dapat mencari halamanhalaman web yang dibutuhkannya melalui *search engine*. Contoh lain penerapan dari sistem temu kembali informasi adalah text mining, dokumen clustering. Proses yang terjadi di dalam *Information Retrieval System* terdiri dari 2 bagian utama, yaitu *Indexing subsystem*, dan *Searching subsystem (matching system)*.

2.2.1 Proses *Indexing*

Indexing subsystem adalah proses *subsystem* yang merepresentasikan koleksi dokumen kedalam bentuk tertentu untuk memudahkan dan mempercepat proses pencarian dan penemuan kembali dokumen yang relevan.

Pembangunan index dari koleksi dokumen merupakan tugas pokok pada tahapan preprocessing di dalam IR. Kualitas index mempengaruhi efektifitas dan efisiensi sistem IR. *Index* dokumen adalah himpunan term yang menunjukkan isi atau topik yang dikandung oleh dokumen. Index akan membedakan suatu dokumen dari dokumen lain yang berada di dalam koleksi. Ukuran *index* yang kecil dapat memberikan hasil buruk dan mungkin beberapa item yang relevan terabaikan. *Index* yang besar memungkinkan ditemukan banyak dokumen yang relevan tetapi sekaligus dapat menaikkan jumlah dokumen yang tidak relevan dan menurunkan kecepatan pencarian (searching).

Tahap-tahap yang terjadi pada proses *indexing* adalah sebagai berikut :

1. *Word Token*

Yaitu mengubah dokumen menjadi kumpulan term dengan cara menghapus semua karakter dalam tanda baca yang terdapat pada dokumen dan mengubah kumpulan term menjadi lowercase.

2. *Stopword Removal*

Proses penghapusan kata-kata yang sering ditampilkan dalam dokumen seperti: and, or, not dan sebagainya.

3. *Stemming*

Proses mengubah suatu kata bentukan menjadi kata dasar.

4. *Term Weighting*

Proses pembobotan setiap term di dalam dokumen.

2.2.2 Proses *Searching*

Ada beberapa proses yang terjadi saat melakukan *search*, antara lain adalah sebagai berikut:

1. Parse *query* yaitu memecah *query* menjadi bentuk token.

2. Proses *Stopword filtration*.

Token-token *query* yang telah dihasilkan pada proses parse *query* kemudian di filter melalui proses pembuangan token yang termasuk Stopword.

3. Proses *Stemming*

Stopword tokens dari proses stopwords sebelumnya kemudian di filter kembali melalui proses *Stemming* sehingga menghasilkan *stemmed term query*.

4. Transformasi *Query*

Stemmed term query yang dihasilkan kemudian ditransformasikan apabila memerlukan. Artinya, apabila *query* yang diinputkan membutuhkan terjemahan ke dalam bentuk *query* bahasa lain maka sebelum mencari dokumen pada koleksi dokumen, *query* tersebut diterjemahkan dahulu melalui proses penerjemahan *query*. Sistem akan membandingkan *query* tersebut dengan koleksi dokumen sehingga mengembalikan dokumen- dokumen yang relevan dalam suatu bahasa yang berbeda dengan bahasa *query*.

5. Pemodelan dalam model ruang vektor

Tiap term atau kata yang ditemukan pada dokumen dan *query* diberi bobot dan disimpan sebagai salah satu elemen vektor dan dihitung nilai kemiripan antara *query* dan dokumen.

6. Perangkingan dokumen atau konten berdasarkan nilai kemiripan antara *query* dan dokumen.

2.2.3 *Model Information Retrieval (IR)*

Model yang terdapat dalam *Information Retrieval* terbagi dalam 3 model besar, yaitu:

1. *Set-theoretic* model, model merepresentasikan dokumen sebagai himpunan kata atau frase. Contoh model ini ialah standard Boolean model dan *extended Boolean* model.
2. *Algebraic* model, model merepresentasikan dokumen dan *query* sebagai vektor atau matriks *similarity* antara vektor dokumen dan vektor *query* II-7 yang direpresentasikan sebagai sebuah nilai skalar. Contoh model ini ialah *vector space* model dan *Latent Semantic Indexing (LSI)*.
3. Probabilistic model, model memperlakukan proses pengembalian dokumen sebagai sebuah probabilistic inference. Contoh model ini ialah penerapan teorema bayes dalam model probabilistik.

2.3 *American Standard Code for Information Interchange (ASCII)*

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (American Standard Code for Information Interchange) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 7 bit. Namun, ASCII disimpan sebagai sandi 8 bit dengan menambahkan satu angka 0 sebagai bit significant paling tinggi. Bit tambahan ini sering digunakan untuk uji prioritas. Karakter control pada ASCII dibedakan menjadi 5 kelompok sesuai dengan penggunaan yaitu berturut-turut meliputi logical communication, Device control, Information separator, Code extention, dan physical communication. Code ASCII ini banyak dijumpai pada papan ketik (keyboard) komputer atau instrument-instrument digital. (Anonymous, 2011) Jumlah kode ASCII adalah 255 kode. Kode ASCII 0..127 merupakan kode ASCII untuk manipulasi teks; sedangkan kode ASCII 128..255 merupakan kode ASCII untuk manipulasi grafik (Tantoni & Zaen, 2018).

Tabel 2.1 ASCII

Symbol	Decimal	Binary	Symbol	Decimal	Binary
A	65	01000001	a	97	01100001
B	66	01000010	b	98	01100010
C	67	01000011	c	99	01100011
D	68	01000100	d	100	01100100
E	69	01000101	e	101	01100101
F	70	01000110	f	102	01100110
G	71	01000111	g	103	01100111
H	72	01001000	h	104	01101000
I	73	01001001	i	105	01101001
J	74	01001010	j	106	01101010
K	75	01001011	k	107	01101011
L	76	01001100	l	108	01101100
M	77	01001101	m	109	01101101
N	78	01001110	n	110	01101110
O	79	01001111	o	111	01101111
P	80	01010000	p	112	01110000
Q	81	01010001	q	113	01110001
R	82	01010010	r	114	01110010
S	83	01010011	s	115	01110011
T	84	01010100	t	116	01110100
U	85	01010101	u	117	01110101
V	86	01010110	v	118	01110110
W	87	01010111	w	119	01110111
X	88	01011000	x	120	01111000
Y	89	01011001	y	121	01111001
Z	90	01011010	z	122	01111010

2.4 Algoritma Rabin Karp

Algoritma *Rabin Karp* adalah algoritma pencarian kata yang mencari sebuah pola berupa *substring* dalam sebuah teks menggunakan hashing. Algoritma ini sangat efektif untuk pencocokan kata dengan pola banyak. Salah satu aplikasi praktis dari algoritma *Rabin Karp* adalah dalam pendeteksian plagiarisme. Dalam ilmu komputer, algoritma *Rabin-Karp* adalah algoritma pencarian *string* yang dibuat oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 yang menggunakan hashing untuk menemukan salah satu dari satu set *string* pola dalam teks.

Algoritma *Rabin Karp* merupakan algoritma pencarian *string* yang ditemukan oleh Michael Rabin dan Richard Karp. Algoritma *Rabin Karp* merupakan algoritma pencarian *string* yang menggunakan fungsi hashing untuk membandingkan *string* yang dicari (m) dengan *string* yang dibandingkan (n). Fungsi hash adalah fungsi matematis yang digunakan untuk mengubah data menjadi bilangan bulat yang relatif kecil yang dapat berfungsi sebagai indeks pada array. Fungsi hashing merupakan proses inti pada perhitungan algoritma *Rabin Karp*. *Hashing* merupakan representasi ASCII yang menggantikan atau mentransformasikan karakter atau tanda baca menjadi sebuah nilai atau angka (Suryati et al., 2018).

Langkah-langkah dalam algoritma *Rabin Karp*:

1. Menghilangkan tanda baca dan mengubah ke teks sumber dan kata yang ingin dicari menjadi kata-kata tanpa huruf.
2. Membagi teks kedalam gram-gram yang ditentukan nilai k-gram nya
3. Mencari nilai hash dengan fungsi hash dari tiap kata yang terbentuk
4. Mencari nilai hash yang sama antara dua teks

Adapun tahapan proses sistem deteksi kesamaan dengan metode *rabin karp* adalah sebagai berikut :

1. Tahap Input Parameter

Proses sistem yang akan dibuat dimulai dari memasukkan dokumen yaitu dokumen asli dan dokumen uji. Kemudian akan dilakukan pengujian melalui tahapan-tahapan *preprocessing*, yaitu menghilangkan tanda baca, merubah huruf kehuruf kecil, menghilangkan tanda baca, menghilangkan kata kata yang kurang penting, kemudian pembentukan rangkaian gram panjang k, lalu menghitung nilai hashing. Dari pengujian tersebut akan didapatkan hasil akhirnya berupa persentase kemiripan (*similarity*).

Berikut ini adalah contoh perhitungan kesamaan dokumen yang akan di proses dengan menggunakan metode *Rabin Karp*.

Dokumen asli yang di input

Salah satu dampak negatif dari perkembangan ilmu, teknologi dan informasi adalah maraknya tindakan “Plagiarisme”.

Dokumen uji yang di input

"Plagiarisme" merupakan dampak negatif dari perkembangan teknologi informasi.

Dengan *K-gram* $k=5$

Basis bilangan $b=7$

2. Tahapan *Preprocessing*

Tahap *Preprocessing* harus dilalui untuk menentukan *keyword* pada kedua dokumen yang akan dilakukan pengujian, yaitu dokumen asli dan dokumen uji. Pada tahap *preprocessing* terdapat beberapa proses yang dilakukan oleh sistem terhadap dokumen yang diinputkan. Proses-proses tersebut antara lain *case folding*, *filtering*, *Tokenizing*.

a. Sub Proses *Case Folding*

Proses *case folding* (mengubah huruf kapital menjadi huruf kecil) merupakan tahap pertama yang akan dilakukan dari rangkaian tahapan yang terdapat pada *preprocessing*. Berikut adalah contoh proses *case folding* :

Dokumen asli setelah melalui proses *case folding* (proses 1.1)

salah satu dampak negatif dari perkembangan ilmu, teknologi dan informasi adalah maraknya tindakan “plagiarisme”.

dokumen uji setelah melalui proses *case folding* (proses 1.1)

"plagiarisme" merupakan dampak negatif dari perkembangan teknologi informasi.

Pada contoh proses *case folding* diatas, kalimat yang terdapat huruf kapitalnya adalah pada kata “Salah” dan pada kata “Plagiarisme”. Dimana pada

kedua kata tersebut setelah dilakukan proses *case folding* akan diubah menjadi huruf kecil semua. Pada kata “Salah” akan berubah menjadi “salah” dan pada kata “Plagiarisme” akan berubah menjadi “plagiarisme”. Sehingga secara keseluruhan kata-kata pada dokumen yang terdapat huruf kapitalnya akan berubah menjadi huruf kecil semua.

b. Sub Proses *Filtering*

Proses *filtering* dilakukan untuk mengambil kata-kata penting dari hasil *Tokenizing*. Bisa menggunakan algoritma stop word (membuang kata yang kurang penting) atau word list (menyimpan kata penting). Setelah proses *case folding*, dokumen selanjutnya masuk ketahap *filtering*, yaitu proses membuang kata yang tidak penting dan membuang tanda baca. Berikut ini contoh proses *filtering*: Dokumen asli setelah melalui proses *filtering* (proses 1.2)

salah satu dampak negatif perkembangan ilmu teknologi informasi maraknya tindakan plagiarisme

Dokumen uji setelah melalui proses *filtering* (proses 1.2)

Plagiarisme merupakan dampak negatif perkembangan teknologi informasi

Pada contoh *filtering* diatas yaitu proses menghilangkan tanda baca dan menghilangkan stop word. Pada contoh dokumen asli tanda baca yang di hilangkan adalah titik (.), koma (,) dan tanda petik (“ ”), sedangkan kata yang di anggap tidak penting dan dihilangkan adalah kata “dari”, kata “dan” dan kata “adalah”. Sedangkan pada dokumen uji tanda baca yang di hilangkan adalah titik (.) dan tanda petik (“ ”), sedangkan dokumen uji kata yang dihilangkan adalah kata “dari”.

c. Sub Proses *Tokenizing*

Setelah dilakukan proses *filtering*, selanjutnya adalah proses *Tokenizing*, proses ini merupakan proses pembentukan pola kata, dimana pola katanya dalam bentuk gram dengan panjang k. Pada proses *Tokenizing* di bagi menjadi dua sub proses yaitu proses parsing *K-gram* dan proses *hashing*.

1. Proses Parsing *K-gram*

Langkah selanjutnya adalah parsing *k-gram*, dimana pada proses ini kata dipecah menjadi potongan-potongan dimana setiap potongan mengandung karakter sebanyak *k*. Berikut ini adalah contoh proses parsing *K-gram* $k = 5$:

Hasil *K-gram* dokumen asli yang telah melalui proses tokenisasi (proses 2.1)

```
{ salah } { alah } { lah s } { ah sa } { h sat } { satu } { satu } { atu d } { tu da } { u
dam } { damp } { dampa } { ampak } { mpak } { pak n } { ak ne } { k neg } { nega
} { negat } { egati } { gatif } { atif } { tif p } { if pe } { f per } { perk } { perke } {
erkem } { rkemb } { kemba } { emban } { mbang } { banga } { angan } { ngan } {
gan i } { an il } { n ilm } { ilmu } { ilmu } { lmu t } { mu te } { u tek } { tekn } {
tekno } { eknol } { knolo } { nolog } { ologi } { logi } { ogi i } { gi in } { i inf } {
info } { infor } { nform } { forma } { ormas } { rmasi } { masi } { asi m } { si ma } {
i mar } { mara } { marak } { arakn } { rakny } { aknya } { knya } { nya t } { ya ti } {
a tin } { tind } { tinda } { indak } { ndaka } { dakan } { akan } { kan p } { an pl } { n
pla } { plag } { plagi } { lagia } { agiar } { giari } { iaris } { arism } { risme } { isme }
```

Hasil *K-gram* dokumen uji yang telah melalui proses tokenisasi (proses 2.1)

```
{ plag } { plagi } { lagia } { agiar } { giari } { iaris } { arism } { risme } { isme } { sme
m } { me me } { e mer } { meru } { merup } { erupa } { rupak } { upaka } {
pakan } { akan } { kan d } { an da } { n dam } { damp } { dampa } { ampak } {
mpak } { pak n } { akne } { kneg } { nega } { negat } { egati } { gatif } { atif } { tifp }
{ ifpe } { fper } { perk } { pe rke } { erkem } { rkemb } { kemba } { emban } { mbang }
{ banga } { angan } { ngan } { gant } { ante } { ntek } { tekn } { tekno } { eknol } { knolo }
{ nolog } { ologi } { logi } { ogi i } { gi in } { iinf } { info } { infor } { nform } { forma }
{ ormas } { rmasi } { masi }
```

Pada contoh parsing *K-gram* diatas adalah proses pembentukan pola kata dalam bentuk gram dengan panjang karakter $k=5$, sehingga isi kalimat pada dokumen asli dan dokumen uji dirubah dalam bentuk kgram.

2. Proses *Hashing*

Kemudian dilakukan proses *hashing*, dimana pada proses ini hash berfungsi untuk mengkonvert setiap *string* menjadi bilangan. Dengan cara mengalikan nilai ASCII hasil huruf hasil *K-gram* dengan basis bilangan tertentu, dimana basis bilangan merupakan bilangan prima. Dengan menggunakan persamaan (2.1), maka dapat ditung hasil hashnya. Algoritma *Rabin Karp* didasarkan pada fakta jika dua buah *string* sama maka harga hash valuenya pasti sama. Sebagai contoh kita ambil kata dari hasil *K-gram* yang pertama pada dokumen asli, yaitu kata {salah}.

Contoh proses hashing untuk menghitung nilai hash dari kata {salah}, dengan nilai $k=5$ dan $b=7$

Nilai ASCII dari kata {salah}

ASCII(s)= 115

ASCII(a)= 97

ASCII(l)= 108

ASCII(h)= 104

$$H(C_1 \dots C_K) = C_1 \cdot b^{(K-1)} + C_2 \cdot b^{(K-2)} + \dots + C_{(k-1)} \cdot b^{(k)} + C_K$$

$$H = (115 \cdot 7^4) + (97 \cdot 7^3) + (108 \cdot 7^2) + (97 \cdot 7^1) + (104 \cdot 7^0)$$

$$H = (115 \cdot 2401) + (97 \cdot 343) + (108 \cdot 49) + (97 \cdot 7) + (104 \cdot 1)$$

$$H = 276115 + 33271 + 5292 + 679 + 104$$

$$H = 315461$$

Jadi nilai hash pada dokumen asli *K-gram* yang pertama adalah 315461, proses perhitungan nilai hash di ulang kembali hingga *K-gram* keseluruhan dihitung. Berikut ini adalah nilai hasil hash dari dokomen asli dan hash dokumen uji:

Hasil *hashing* dokumen asli:

{315461}	{275454}	{298014}	{271039}	{267110}	{121959}	{315921}
{278742}	{321012}	{297581}	{116760}	{279593}	{276558}	{305659}
{307760}	{272037}	{274083}	{120329}	{304595}	{283500}	{287095}
{278576}	{319865}	{289544}	{262187}	{121102}	{309991}	{287662}
{316225}	{297674}	{285479}	{300949}	{274777}	{276463}	{304994}

{286293}	{273038}	{281096}	{119019}	{295341}	{302768}	{304321}
{298391}	{122428}	{319283}	{285477}	{300943}	{308355}	{309820}
{303195}	{307314}	{285731}	{269098}	{119062}	{295724}	{305442}
{289421}	{311748}	{316764}	{301382}	{277820}	{314558}	{269215}
{119867}	{301352}	{277611}	{313119}	{275932}	{301277}	{310706}
{326277}	{250402}	{122635}	{320718}	{295521}	{304009}	{279403}
{275153}	{295904}	{273087}	{281427}	{121322}	{311535}	{298458}
{274164}	{288974}	{291812}	{278058}	{316228}	{297630}	

Jumlah hash pada dokumen asli adalah= $\sum H_{\text{asli}} = 90$

Hasil *hashing* dokumen uji:

{6982}	{13275}	{12706}	{11988}	{12498}	{12580}	{12334}	{13532}
{12926}	{13372}	{12272}	{10443}	{6903}	{13045}	{12745}	{13799}
{13792}	{13055}	{11981}	{12472}	{11512}	{11074}	{6604}	{12133}
{12206}	{13079}	{12860}	{11465}	{10927}	{6877}	{12971}	{12288}
{12423}	{12272}	{13357}	{11984}	{10551}	{6974}	{13247}	{12634}
{13457}	{12766}	{12407}	{12781}	{11953}	{12155}	{12926}	{12164}
{11564}	{11228}	{7064}	{13527}	{12501}	{13071}	{13315}	{13320}
{13019}	{12918}	{11891}	{10746}	{6834}	{12840}	{13114}	{12709}
{13456}	{13500}	{12830}					

Jumlah hash pada dokumen uji adalah= $\sum H_{\text{uji}} = 67$

Hash yang sama:

{116760}	{279593}	{276558}	{305659}	{307760}	{272037}	{274083}
{120329}	{304595}	{283500}	{287095}	{278576}	{319865}	{289544}
{262187}	{121102}	{309991}	{287662}	{316225}	{297674}	{285479}
{300949}	{274777}	{276463}	{304994}	{122428}	{319283}	{285477}
{300943}	{308355}	{309820}	{303195}	{307314}	{285731}	{269098}
{119062}	{295724}	{305442}	{289421}	{311748}	{316764}	{301382}

{275153} {121322} {311535} {298458} {274164} {288974} {291812}
 {278058} {316228} {297630}

Jumlah hash yang sama dari dokumen asli dan dokumen uji adalah $= \sum H_{asli} \cap H_{uji}$
 $= 52$

Setelah jumlah hash diketahui, yaitu hash pada dokumen asli sebanyak 90, hash dokumen uji sebanyak 67 dan hash yang sama sebanyak 52, maka proses selanjutnya adalah menghitung *similarity*.

3. Deteksi Kesamaan

Setelah diketahui nilai hashnya, jumlah hash pada dokumen asli 90, jumlah hash pada dokumen uji 67 dan hash yang sama pada kedua dokumen 52. Proses selanjutnya adalah menghitung *similarity* yaitu tingkat kesamaan dua dokumen dengan menggunakan persamaan (2.2), yaitu berapa persen tingkat kesamaannya. *Similarity* didapat dari hasil hash yang sama di bagi dengan jumlah hash kedua dokumen dikali dengan seratus persen. Berikut ini adalah proses menghitung *similarity* dua dokumen di atas:

$$\begin{aligned}
 \text{Similarity (asli, uji)} &= \frac{\sum H_{asli} \cap \sum H(uji)}{\sum H_{asli} \cup \sum H(uji)} \times 100 \% \\
 &= \frac{\sum H_{asli} \cap H_{uji}}{\sum H_{asli} + \sum H_{uji} - \sum H_{asli} \cap H_{uji}} \times 100 \% \\
 &= \frac{52}{90 + 67 - 52} \times 100 \% \\
 &= \frac{52}{157 - 52} \times 100 \% \\
 &= \frac{52}{105} \times 100 \% \\
 &= 49,52 \%
 \end{aligned}$$

Hasil perhitungan *similarity* dari kedua dokumen diatas adalah 49,52%, dokumen yang diuji bias dikatakan plagiat, karena tingkat kesamaannya sangat besar.

2.5 Hyper Text Markup Language (HTML)

HTML yang merupakan singkatan dari *Hyper Text Markup Language* adalah serangkaian kode program yang merupakan dasar dari representasi visual sebuah halaman Web. Didalamnya berisi kumpulan informasi yang disimpan dalam tag-

tag tertentu, dimana tag-tag tersebut digunakan untuk melakukan format terhadap informasi yang dimaksud. Berbagai pengembangan telah dilakukan terhadap kode HTML dan telah melahirkan teknologi-teknologi baru di dalam dunia pemrograman web. Kendati demikian, sampai sekarang HTML tetap berdiri kokoh sebagai dasar dari bahasa web seperti PHP, ASP, JSP dan lainnya. Bahkan secara umum, mayoritas situs web yang ada di Internet pun masih tetap menggunakan HTML sebagai teknologi utama.

2.6 *Hypertext Preprocessor (PHP)*

PHP Kepanjangan dari PHP adalah "*Hypertext Preprocessor*" (ini merupakan singkatan rekursif). PHP adalah bahasa *scripting web 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. 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 pengembangan 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 melakukan sebuah kompilasi dalam penggunaannya.
2. Web Server yang mendukung PHP dapat ditemukan dimana - mana dari mulai *apache*, *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 pemahaman, PHP adalah bahasa scripting yang paling mudah karena 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.

2.7 MySQL

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

2.8 Database Management System (DBMS)

Database Management System (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 ke dalam 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. *Table*

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

2. *Record*

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

3. *Field*

Field adalah atribut dari objek yang memiliki tipe data tertentu.




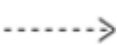



2.9 *Unified Modeling Language (UML)*




Unified Modeling Language (UML) adalah bahasa permodelan untuk menentukan visualisasi, konstruksi, dan merancang sistem berbasis *object-oriented*. *Unified Modeling Language* juga digunakan untuk pengembangan sebuah sistem yang dapat menyampaikan alur kerja sistem dan menjelaskan tugas setiap *user* dalam sebuah sistem. Ada beberapa jenis diagram dalam UML, yaitu:

1. *Use Case Diagram*

Use case adalah kegiatan atau urutan interaksi yang saling berkaitan antara sistem dan actor, *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Adapun komponen-komponen *use case* diagram adalah sebagai berikut :

Tabel 2.2 Komponen-komponen *use case diagram*

Simbol	Nama	Keterangan
	<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
	<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>dependent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (<i>independent</i>).
	<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
	<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
	<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

	<i>Use case</i>	Deskripsi dari uraian aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu <i>actor</i> .
	<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
	<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.


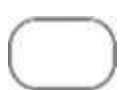

Sumber: (Kiswanto, 2016).



2. *Activity Diagram*

Activity diagram ini menggambarkan tentang aktifitas yang terjadi pada sistem. Dari mulai sampai selesai, diagram ini menunjukkan langkah-langkah dalam proses kerja sistem yang dibuat (Shasi Aprilia Windiyani, 2020).

Adapun komponen-komponen yang terdapat pada diagram ini adalah sebagai berikut :

Tabel 2.3 Komponen-komponen *activity diagram*

Simbol	Nama	Keterangan
	<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
	<i>Action</i>	<i>State</i> dari sistem yang mencerminkan eksekusi dari suatu aksi.
	<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali (<i>start flow</i>).




	<i>Activity Final Node</i>	Bagaimana objek dibentuk dan diakhiri (<i>end flow</i>).
	<i>Fork Node</i>	Satu aliran yang pada tahap tertentu berubah menjadi beberapa alur.

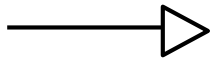

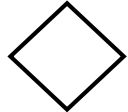
Sumber: (Kiswanto, 2016).

3. *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 (Kiswanto, 2016). Ada beberapa simbol-simbol yang terdapat dalam diagram kelas, yaitu:

Tabel 2.4 Simbol-Simbol *Class Diagram*

No	Simbol	Nama	Keterangan			
1	<table border="1"><tr><td>nama_kelas</td></tr><tr><td>+ atribut</td></tr><tr><td>+ operasi()</td></tr></table>	nama_kelas	+ atribut	+ operasi()	Kelas	Kelas pada struktur sistem
nama_kelas						
+ atribut						
+ operasi()						
2.		Antarmuka / <i>interface</i>	Semua dengan konsep <i>interface</i> dalam pemrograman berorientasi objek			
3.		Asosiasi / <i>association</i>	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i>			
4.		Asosiasi berarah / <i>directed asosiasi</i>	Relasi antarkelas dengan makna kelas yang satu			



			digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
5.		Generalisasi	Relasi antarmuka dengan makna generalisasi spesialisasi (umum khusus)
6.		Kebergantungan / <i>dependency</i>	Relasi antarmuka dengan makna kebergantungan antarkelas
7.		Agregasi / <i>aggregation</i>	Relasi antarkelas dengan makna semua bagian (<i>whole- part</i>)


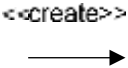




Sumber: (Kiswanto, 2016).

4. *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 *sequence* maka harus diketahui objek-objek yang dimiliki kelas yang diinstansiasi menjadi objek itu. Menggambar diagram *sequence* juga dibutuhkan untuk melihat skenario yang ada pada *use case* (Kiswanto, 2016).

Tabel 2. 5 Simbol-Simbol Sequence Diagram

No	Simbol	Nama	Keterangan
1.		Aktor	Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri.
2.		Garis hidup / <i>lifeline</i>	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 <i>create</i>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6.		Pesan tipe <i>call</i>	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri
7.		Pesan tipe <i>send</i>	Menyatakan bahwa suatu objek mengirim data / masukkan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
8.		Pesan tipe <i>return</i>	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 tipe <i>destroy</i>	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada <i>destroy</i>

Sumber: (Kiswanto, 2016).

BAB III

METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

3.1.1 Waktu

Waktu pelaksanaan penelitian tugas akhir dilaksanakan mulai dari bulan maret 2021 sampai dengan Juni 2021. Rincian kegiatan dapat dilihat pada Tabel 3.1 berikut:

Tabel 3.1 Gantt Chart Waktu Penelitian

No	Uraian	Waktu (Tahun 2021)															
		Maret				April				Mei				Juni			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.	<i>Inception</i>																
2.	<i>Elaboration</i>																
3.	<i>Construction</i>																
4.	<i>Transition</i>																

3.1.2 Tempat Penelitian

Adapun tempat penelitian tugas akhir yang akan dilakukan yaitu di Jurusan Teknik Informatika Fakultas Teknik Universitas Halu Oleo.

3.2 Metode Pengumpulan Data

Dalam penelitian ini metode pengumpulan data yang digunakan pada perancangan sistem tersebut adalah studi literatur, mulai dari buku-buku, jurnal, data dari Jurusan teknik informatika maupun artikel, dan sumber-sumber lain. Metode ini dilaksanakan dengan melakukan studi kepustakaan yang relevan. Studi kepustakaan ini dilakukan untuk mencari sumber pelengkap yang berhubungan

dengan sistem yang akan dibangun, yaitu dengan mencari referensi yang membahas tentang penggunaan metode *Rabin Karp*, sehingga metode *Rabin Karp* dapat diimplementasikan dalam Sistem pendeteksi plagiarisme Abstrak Skripsi di jurusan Teknik Informarika Universitas Halu Oleo.

3.3 Metode Pengembangan Sistem

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

3.3.1 Permulaan (*Inception*)

Pada fase ini dilakukan proses pengidentifikasian sistem, dilakukan dengan analisis kebutuhan akan sistem, melakukan kajian terhadap penelitian yang terkait dengan metode *Rabin Karp*.

3.3.2 Perluasaan / Perencanaan (*Elaboration*)

Setelah menentukan ruang lingkup penelitian, tahap ini akan dilakukan perancangan dan analisis sistem menggunakan flowchart meliputi flowchart Sistem pendeteksi plagiarisme, dan flowchart metode *Rabin Karp*. Pada perancangan ini, digunakan juga *UML (Unified Modelling Language)* yang meliputi use case diagram, activity diagram, class diagram dan sequence diagram.

3.3.3 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.

3.3.4 Transisi (*Transition*)

Pada tahap ini difokuskan untuk melakukan pengujian terhadap sistem pencarian. Memperbaiki masalah yang muncul saat pembuatan dan setelah pengujian aplikasi.

3.4 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan untuk mengidentifikasi permasalahan-permasalahan yang ada pada sistem dimana sistem dibangun, meliputi perangkat lunak, dan hasil analisis terhadap sistem serta elemen-elemen sistem. Pembahasan yang ada pada analisis sistem ini yaitu analisis kebutuhan fungsional dan analisis kebutuhan nonfungsional.

3.4.1 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. Informasi data mahasiswa Universitas Halu Oleo.

3.4.2 Kebutuhan Nonfungsional

Analisis kebutuhan nonfungsional adalah sebuah langkah dimana seorang pembangun aplikasi menganalisis sumber daya yang dibutuhkan untuk membangun aplikasi yang akan dibangun. Analisis kebutuhan nonfungsional yang dilakukan dibagi dalam dua tahap, yaitu analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

1. Kebutuhan Perangkat Keras

Perangkat keras yang digunakan pada pembangunan sistem ini, sebagai berikut:

Tabel 3.2 Spesifikasi Perangkat

No	Nama Perangkat	Spesifikasi
1.	<i>PC</i>	<i>Acer</i>
2.	<i>Processor</i>	<i>Intel dual core</i>
3.	<i>RAM</i>	<i>4 GB</i>

4.	<i>Harddisk</i>	<i>128 GB SSD</i>
5.	<i>Monitor</i>	<i>Monitor 14 Inch</i>

2. Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan pada pembangunan sistem ini, sebagai berikut:

Tabel 3.3 Spesifikasi Perangkat Lunak

No.	Nama Perangkat	Fungsi	Spesifikasi
1.	<i>Windows</i>	<i>Operating System</i>	<i>Windows 10 Home</i>
2.	<i>Xampp</i>	<i>Universal development environment</i>	<i>Xampp v3.2.2</i>
3.	<i>Apache</i>	<i>Web Server</i>	<i>Apache 2.4 win64-VC14</i>
4.	<i>Mysql</i>	<i>Database Management</i>	<i>Mysql 5.7</i>
5.	<i>PHP</i>	<i>Web development</i>	<i>PHP 7.3</i>
6.	<i>Chrome</i>	<i>Web browser</i>	<i>Chrome version 76.0</i>
7.	<i>Visual Studio Code</i>	<i>Code Editor</i>	<i>VS Code v1.38</i>

3.5 Analisis Perancangan Sistem

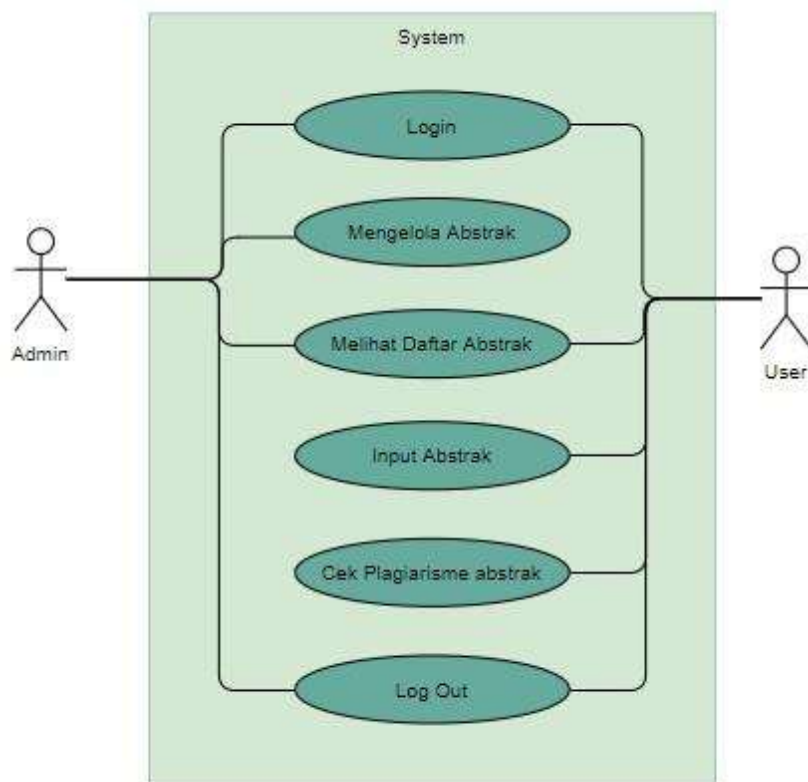
Perancangan sistem yang akan dibangun yaitu perancangan UML dan perancangan *user interface* sistem.

3.5.1 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*.

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 3.1 *Use Case Diagram*

Tabel 3.4 Deskripsi Case Login (Admin dan Mahasiswa)

Nama	<i>Login</i>
Deskripsi Singkat:	Proses Keamanan login untuk <i>Admin</i> dan Mahasiswa
Persyaratan	<i>Admin</i> dan Mahasiswa harus memasukkan <i>username</i> dan <i>password</i>
Kondisi Akhir	Masuk untuk manage sistem

Situasi Kesalahan	i. Yang masuk bukan <i>Admin</i> dan Mahasiswa ii. Salah memasukkan <i>username</i> dan <i>password</i>
Status sistem dalam saat terjadi kesalahan	1. Menampilkan pesan <i>error</i> 2. Tidak dapat masuk kedalam sistem
Aktor	<i>Admin</i> dan Mahasiswa
Trigger	Keamanan <i>login</i> untuk <i>Admin</i>
Proses Standar	1. <i>Admin</i> dan Mahasiswa masuk ke sistem 2. <i>Admin</i> dan Mahasiswa memasukkan <i>username</i> 3. <i>Admin</i> dan Mahasiswa memasukkan <i>password</i>
Proses Alternatif	Mengecek <i>username</i> dan <i>password</i>

Tabel 3.5 Deskripsi Case Mengolah Abstrak (Admin)

Nama	Mengelola Abstrak
Deskripsi Singkat:	<i>Admin</i> ingin melihat atau mengelolah abstrak
Persyaratan	<i>Admin</i> harus melakukan <i>login</i> terlebih dahulu
Kondisi Akhir	Data Abstrak di tampilkan
Situasi Kesalahan	-
Status sistem dalam saat terjadi kesalahan	-
Aktor	<i>Admin</i>
Trigger	<i>Admin</i> ingin menambahkan, menghapus atau memperbarui abstrak
Proses Standar	1. <i>Admin</i> masuk ke sistem dan telah melakukan proses <i>login</i> 2. <i>Admin</i> melihat data Abstrak 3. <i>Admin</i> menambahkan akun mahasiswa 4. <i>Admin</i> menghapus data abstrak 5. <i>Admin</i> mengedit data abstrak
Proses Alternatif	-

Tabel 3.6 Deskripsi Case Lihat daftar abstrak

Nama	Melihat daftar abstrak
Deskripsi Singkat:	<i>Admin</i> dan Mahasiswa ingin melihat daftar abstrak
Persyaratan	<i>Admin</i> dan Mahasiswa harus masuk kesistem terlebih dahulu
Kondisi Akhir	Data Abstrak di tampilkan
Situasi Kesalahan	-
Status sistem dalam saat terjadi kesalahan	-
Aktor	<i>Admin</i> dan Mahasiswa
Trigger	
Proses Standar	<ol style="list-style-type: none"> 1. <i>Admin</i> dan Mahasiswa masuk kesistem 2. <i>Admin</i> dan Mahasiswa melakukan pencarian 3. Data data abstrak ditampilkan
Proses Alternatif	-

Tabel 3.7 Deskripsi Case Cek plagiarisme

Nama	Cek Plagiarisme
Deskripsi Singkat:	Mahasiswa ingin mengecek plagiarisme
Persyaratan	Mahasiswa harus masuk kesistem terlebih dahulu
Kondisi Akhir	Data Abstrak hasil persentase harus di tampilkan
Situasi Kesalahan	-
Status sistem dalam saat terjadi kesalahan	-
Aktor	Mahasiswa
Trigger	

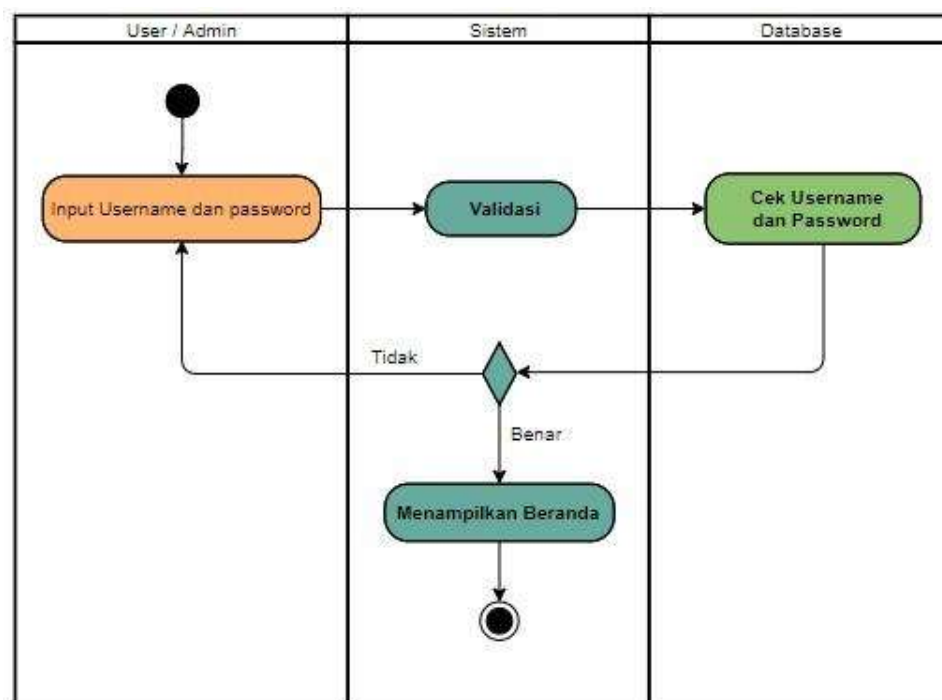
Proses Standar	<ol style="list-style-type: none"> 1. Mahasiswa masuk kesistem 2. Mahasiswa input data abstrak dan menentukan k-gramnya 3. Data data abstrak dan persentase plagiarismenya ditampilkan
Proses Alternatif	-

2. Activity diagram

Activity Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang di rancang, 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.

a. Activity Diagram Login

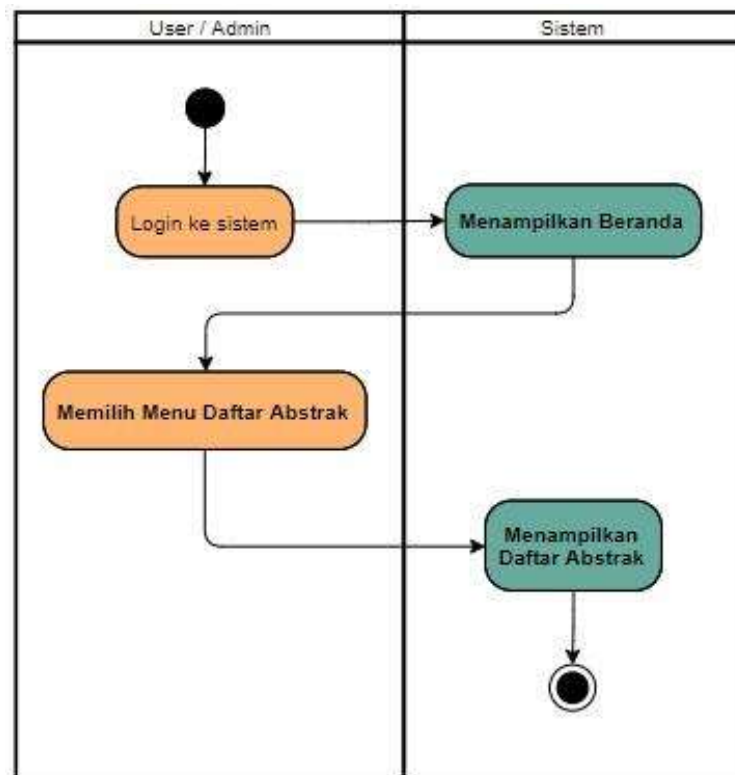
Pada Gambar 3.2 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* dan *user* ketika memasukkan *username* dan *password*, lalu sistem akan menampilkan melakukan validasi.



Gambar 3.2 Activity diagram login

b. *Activity diagram* Daftar abstrak skripsi

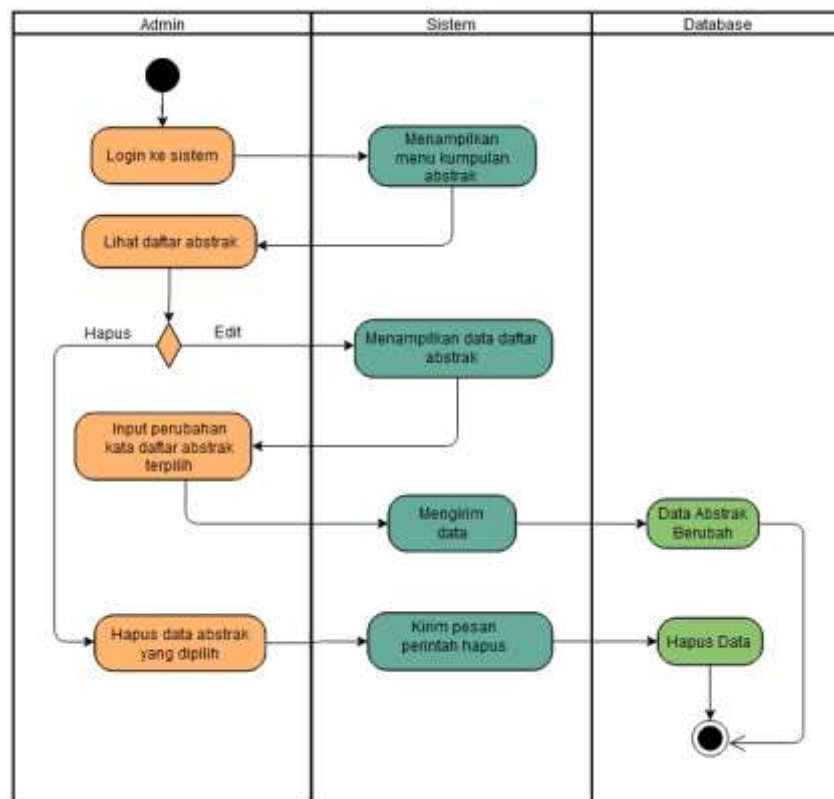
Pada Gambar 3.3 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* dan User ketika telah login, lalu sistem akan menampilkan beranda yang berisikan tampilan Daftar abstrak.



Gambar 3.3 Activity Diagram Daftar abstrak

c. *Activity Diagram* Mengelola Abstrak skripsi (Edit dan Hapus)

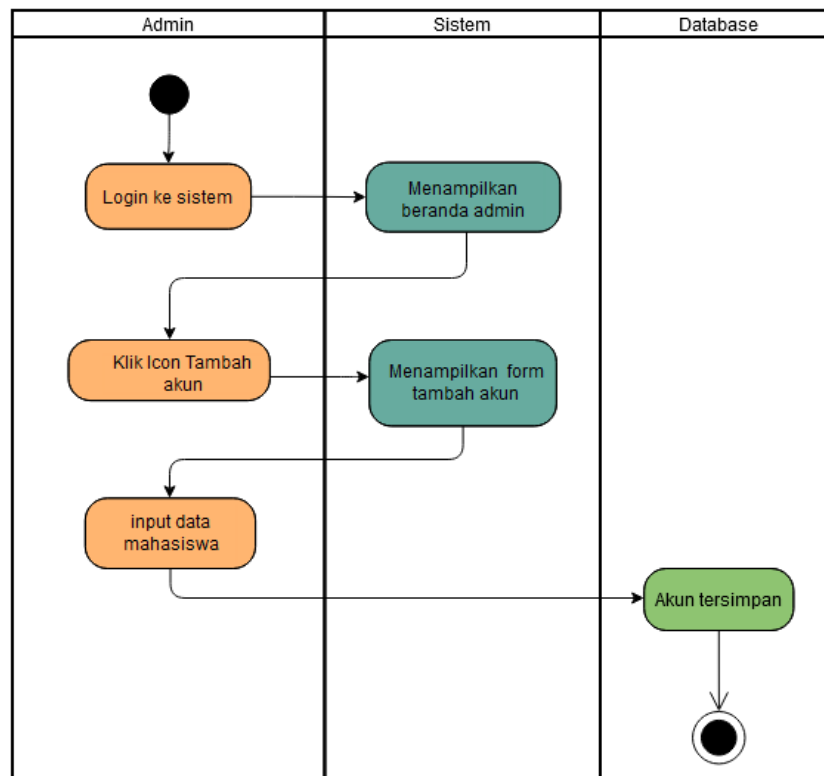
Pada Gambar 3.4 merupakan diagram aktivitas yang menunjukkan aktivitas admin memilih menu Mengelola Abstrak dan memilih untuk edit atau hapus dari salah satu daftar abstrak.



Gambar 3.4 Activity Diagram edit atau hapus abstrak

d. *Activity Diagram* Mengelola Abstrak skripsi (Tambah akun)

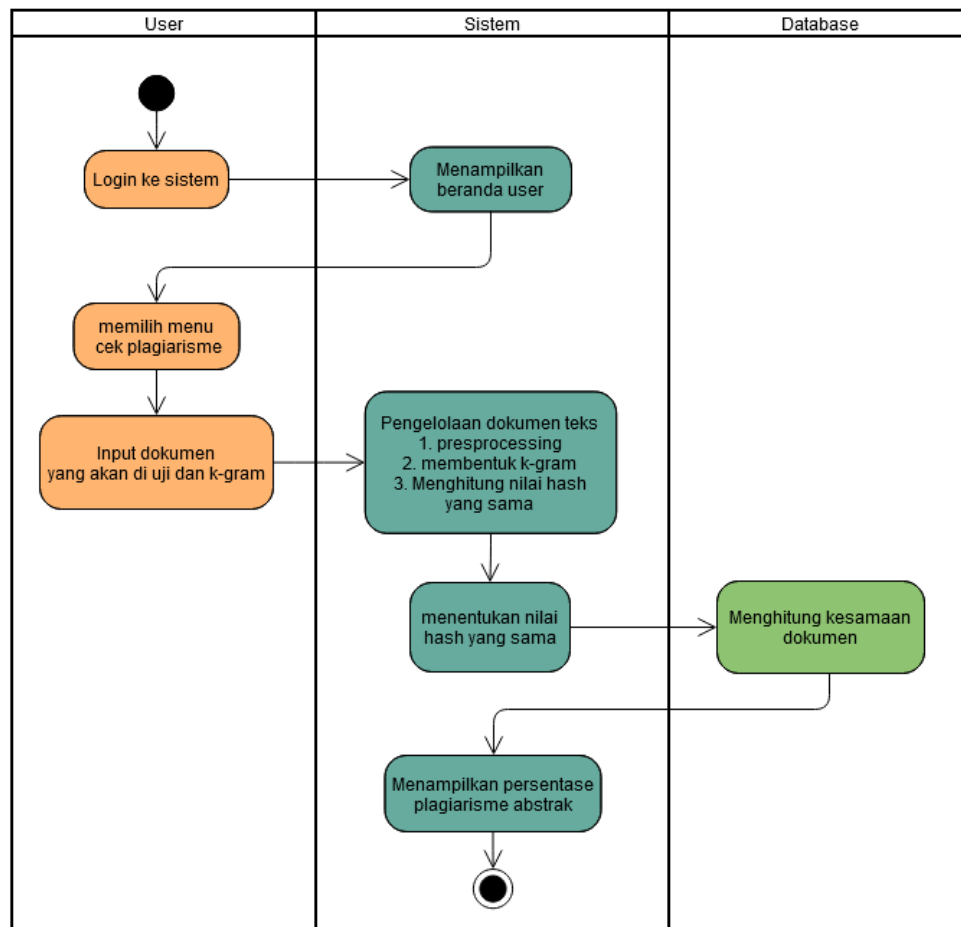
Pada Gambar 3.5 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* memilih menu Tambah akun untuk memberikan akun kepada mahasiswa agar dapat menginput abstrak.



Gambar 3.5 Activity Diagram Tambah Akun

e. *Activity Diagram Cek Plagiarisme Abstrak*

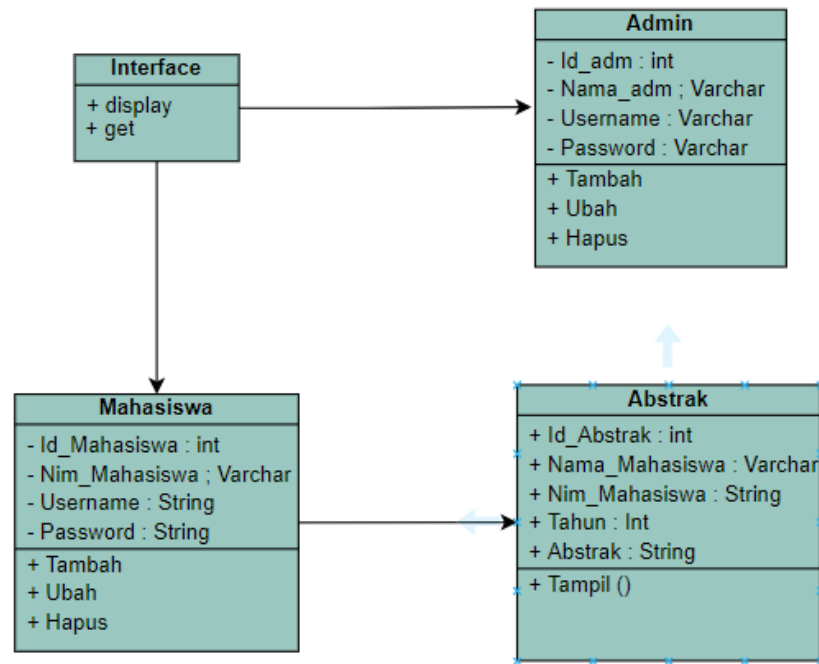
Pada Gambar 3.6 merupakan diagram aktivitas yang menunjukkan aktivitas *User* memilih menu *Cek plagiarisme Abstrak* lalu sistem menampilkan halaman *Input abstrak*, setelah itu upload abstrak yang telah di input lalu user melakukan cek plagiarisme pada abstrak.



Gambar 3.6 Activity Diagram Cek plagiarisme abstrak

3. *Class diagram*

Class diagram adalah diagram yang menjelaskan tentang pemodelan sistem berorientasi objek. *Class diagram* juga menunjukkan hubungan *Class* dengan sistem yang sedang dibangun dan bagaimana mereka saling berinteraksi untuk mencapai suatu tujuan. Berikut ini *Class diagram* sistem yang akan dibangun.



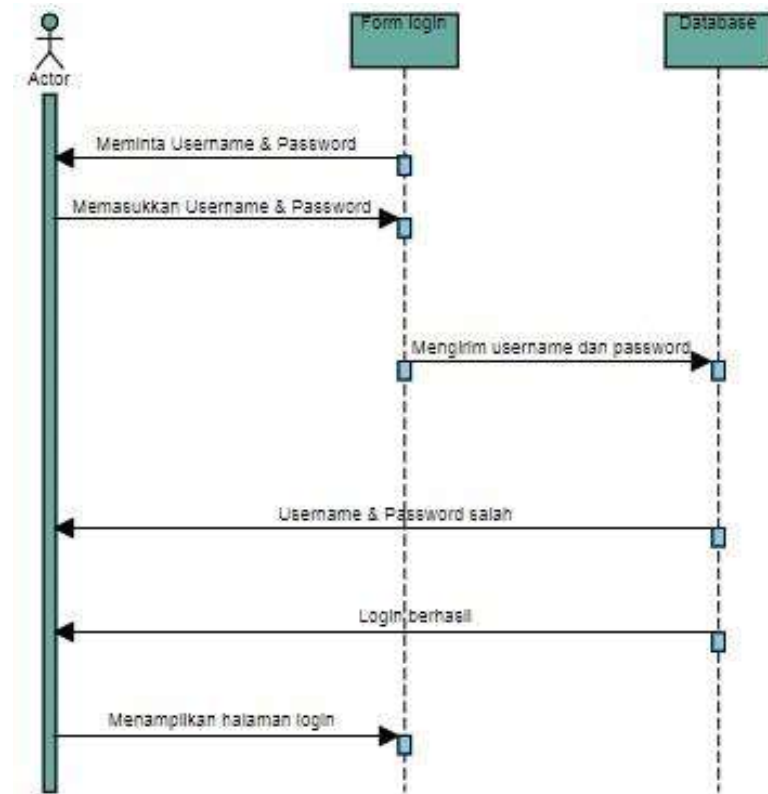
Gambar 3.7 Class diagram

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 interaksi antar objek dan sistem.

a. Sequence Diagram Login

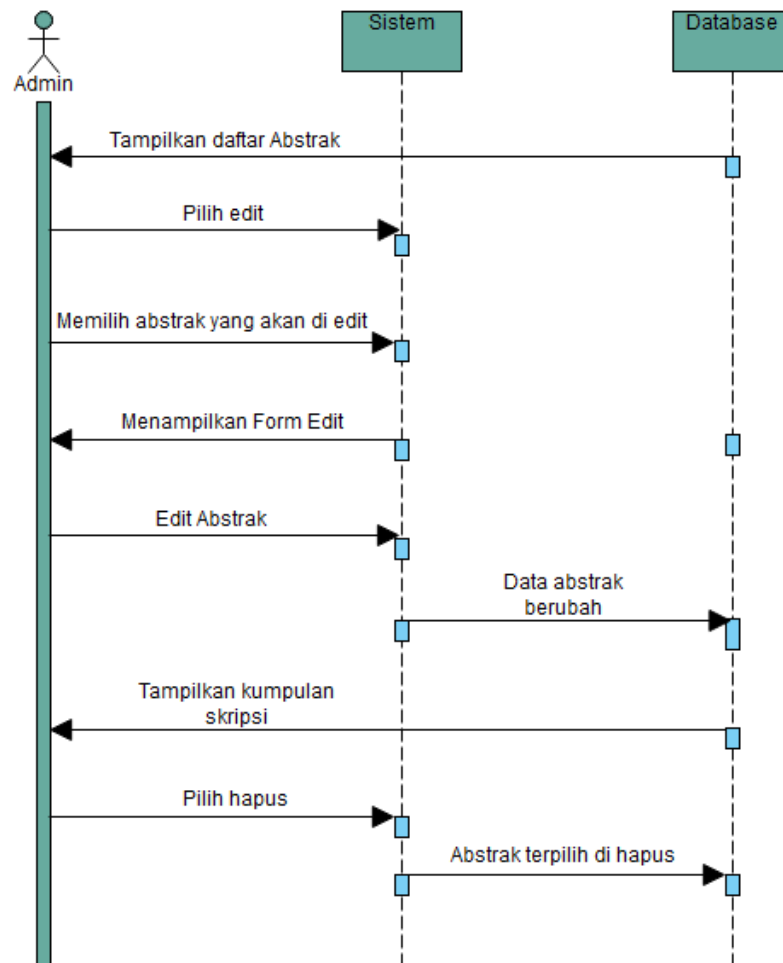
Pada Gambar 3.8 adalah *sequence* yang menunjukkan proses login bagi admin dan user dimana *actor* memasukkan *username* dan *password* lalu divalidasi oleh *database*.



Gambar 3.8 Sequence Diagram Login

- b. *Sequence Diagram* mengelola abstrak (edit dan hapus)

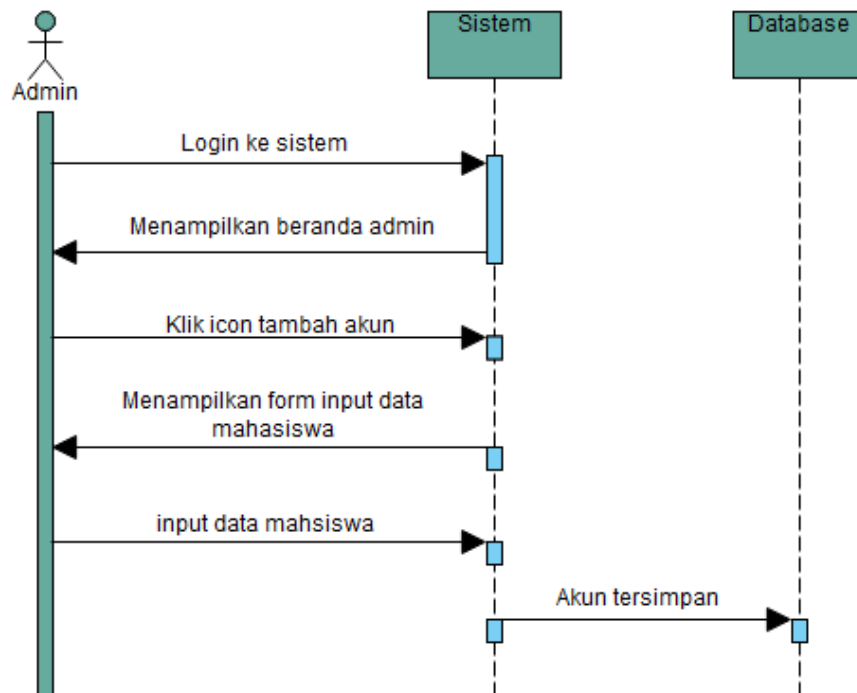
Gambar 3.9 adalah *sequence diagram* yang menjelaskan tentang korelasi antara sistem dan *user* pada Sistem Informasi Penelusuran Tugas Akhir. Dimulai dari *user* membuka sistem, lalu sistem menampilkan menu-menu, kemudian *user* memilih menu pencarian yang ingin digunakan hingga sistem menampilkan hasil dari pencarian.



Gambar 3.9 Sequence Diagram Edit dan hapus

c. *Sequence Diagram* mengelola abstrak (Tambah Akun)

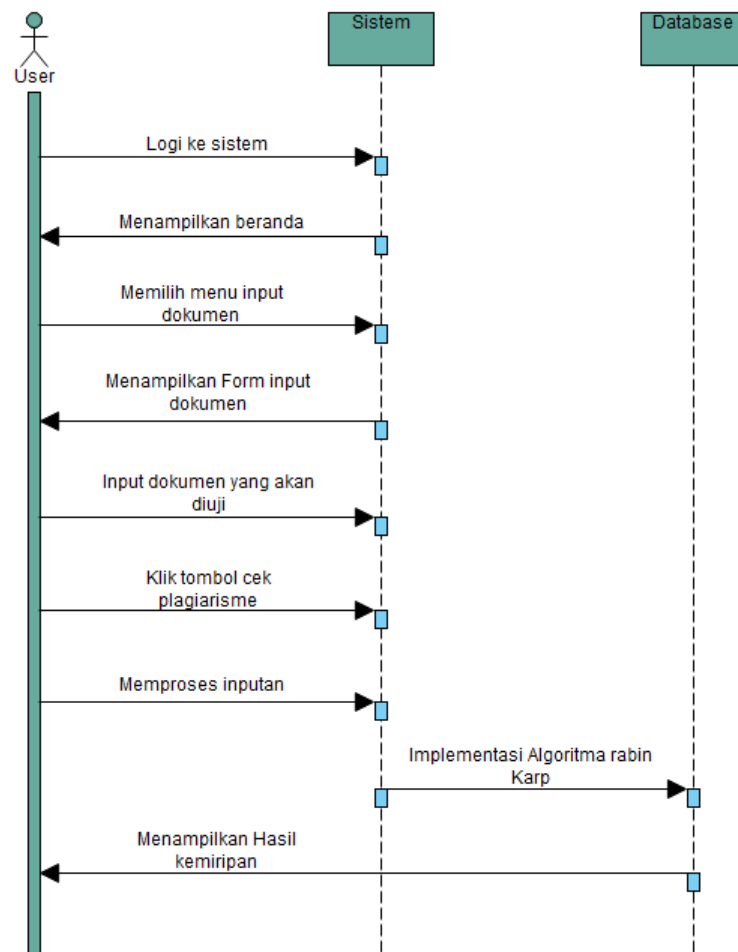
Gambar 3.10 merupakan *Sequence Diagram* yang menunjukkan aktivitas *admin* memilih menu Tambah akun untuk memberikan akun kepada mahasiswa agar dapat menginput abstrak.



Gambar 3.10 Sequence Diagram Tambah Akun

d. *Sequence Diagram* Cek plagiarisme

Pada Gambar 3.11 merupakan *Sequence Diagram* yang menunjukkan Proses *User* memilih menu *Cek plagiarisme* Abstrak lalu sistem menampilkan halaman *Input abstrak*, setelah itu upload abstrak yang telah di input lalu user melakukan cek plagiarisme pada abstrak.



Gambar 3.11 Sequence Diagram Cek plagiarisme

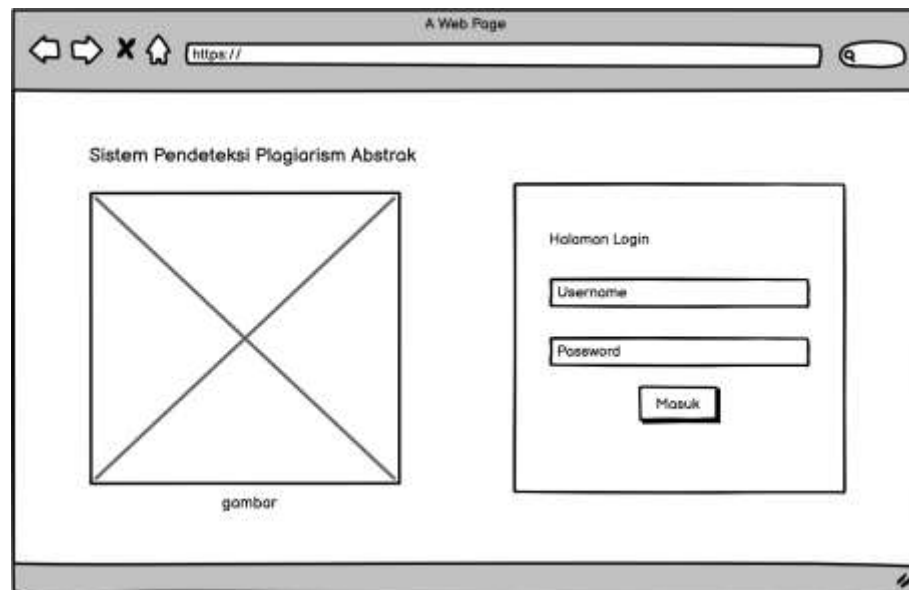
3.5.2 Perancangan Antarmuka (*Interface*)

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

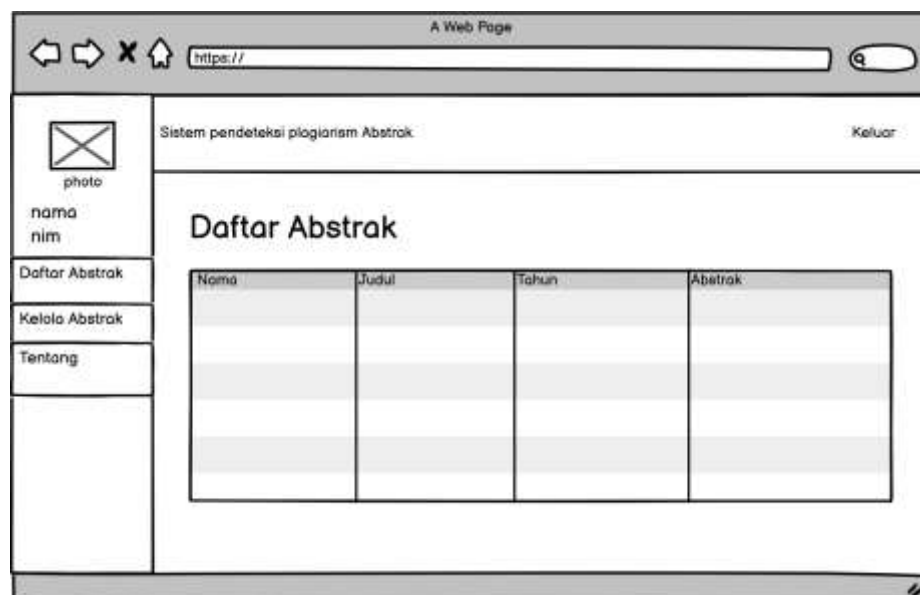
itu, *form* ini juga telah ditambahkan fungsi validasi agar dapat menyaring data *user* yang benar. Bentuk dari tampilan *login* dapat dilihat pada gambar berikut.



Gambar 3.12 Menu *Login*

2. Menu Daftar abstrak

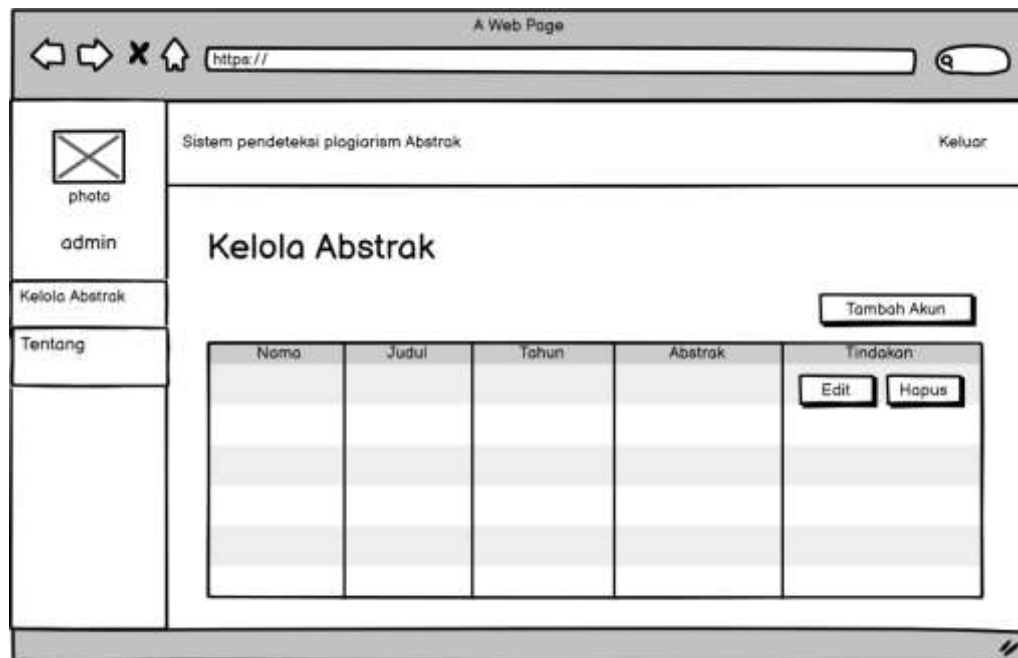
Tampilan Daftar abstrak adalah tampilan pertama yang akan dilihat pada sistem ini. Pada tampilan daftar abstrak berisi logo dan menu-menu pada sistem.



Gambar 3.13 Daftar abstrak

3. Menu mengelola abstrak

Menu Mengelola Abstrak menampilkan data Abstrak dan juga admin bisa melakukan Tambah abstrak, edit dan hapus Abstrak. Bentuk dari tampilan menu mengelola Abstrak dapat dilihat pada gambar berikut.



Gambar 3.14 Menu Kelola Abstrak

4. Menu Mengecek plagiarisme abstrak

Menu Mengecek plagiarisme Abstrak menampilkan data form input abstrak, selanjutnya data tersebut di upload dan akan di cek plagiarismenya. Setelah proses itu selesai maka akan tampil tabel yang berisi daftar abstrak beserta dengan persentase keasamaannya dengan abstrak lainnya. Bentuk dari tampilan menu mengecek plagiarisme Abstrak dapat dilihat pada gambar berikut.

Gambar 3.15 Menu Cek plagiarisme Abstrak

5. Menu Mengecek plagiarisme abstrak

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

Gambar 3.16 Menu Tentang

Daftar Pustaka

- Adi, S. (2018). Penerapan Algoritma Rabin Karp Untuk Mendeteksi Kemiripan Judul Skripsi. *Jurnal Mantik Penuasa*, 22(1), 125–130.
- Aziz, Lulu A., Ana I., & A. T. K. (2015). Upaya Perpustakaan Dalam Mengurangi Plagiarisme Pada Karya Ilmiah Mahasiswa (Studi Kasus Di Upt Perpustakaan Unika Soegijapranata). *Jurnal Ilmu Perpustakaan*, 4(3), 1–13. Retrieved from <https://www.neliti.com/id/publications/137458/upaya-perpustakaan-dalam-mengurangi-plagiarisme-pada-karya-ilmiah-mahasiswa-stud>
- Filcha, A., & Hayaty, M. (2019). Implementasi Algoritma Rabin-Karp untuk Pendeteksi Plagiarisme pada Dokumen Tugas Mahasiswa. *JUITA : Jurnal Informatika*, 7(1), 25. <https://doi.org/10.30595/juita.v7i1.4063>
- Kiswanto, C. F. (2016). “*Sistem Informasi Akademik Sub-Sistem : Utility Dan Epsbed.*”
- Pamungkas, H. Y., Teknologi, F., Universitas, I., & Barat, J. (n.d.). *ALGORITMA RABIN-KARP*. (100), 209–219.
- Putra, N. P., & Sularno. (2019). Penerapan Algoritma Rabin-Karp Dengan Pendekatan Synonym Recognition Sebagai Antisipasi Plagiarisme Pada Penulisan Skripsi. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, 1(2), 49–58.
- Suryati, T., Wibisono, Y., & Wihardi, Y. (2018). Aplikasi Deteksi Plagiarisme Dokumen Skripsi dengan Algoritma Rabin-Karp. *JATIKOM: Jurnal Teori Dan Aplikasi Ilmu Komputer*, 1(2), 91–95.
- Ferdianto, Yosefa. 2013. “Pengertian PHP.”
- Tantoni, A., & Zaen, M. T. A. (2018). Implementasi Double Caesar Cipher Menggunakan Ascii. *Jurnal Informatika Dan Rekayasa Elektronik*, 1(2), 24. <https://doi.org/10.36595/jire.v1i2.56>

Wulan Nafesa Septine, S.T., MMSI. 2019. "DATABASE MANAGEMENT SYSTEM (DBMS) MARIA DB." *06 Nov 2019*. Retrieved (<https://www.smktarunabangsa.sch.id/artikel/detail/database-management-system-dbms-maria-db#:~:text=Berdasarkan survey yang dilakukan%2CMySQL,Top 10 Enterprise Database System>).