**PROPOSAL**

**SISTEM PENDETEKSI PLAGIARISME PADA PROPOSAL TUGAS AKHIR MAHASISWA 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**

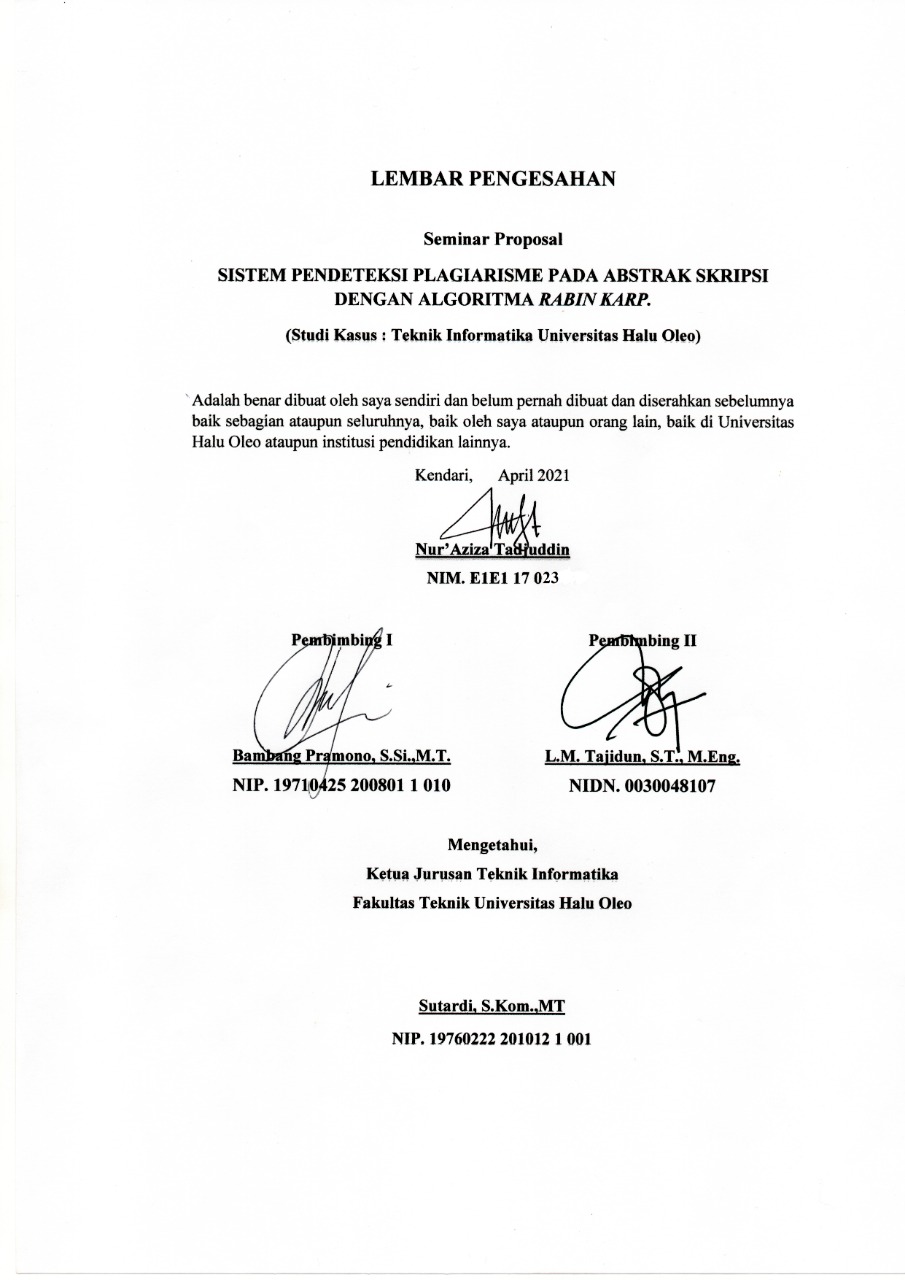
**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS HALU OLEO**

**2021**

# LEMBAR PENGESAHAN



# DAFTAR ISI

[HALAMAN SAMPUL i](#_Toc67579491)

[HALAMAN PENGESAHAN](#_Toc67579492) ii

[DAFTAR ISI iii](#_Toc67579493)

[DAFTAR TABEL v](#_Toc67579494)

[DAFTAR GAMBAR vi](#_Toc67579495)

[BAB I PENDAHULUAN 1](#_Toc67579496)

[1.1 Latar belakang 1](#_Toc67579497)

[1.2 Rumusan masalah 2](#_Toc67579498)

[1.3 Batasan Masalah 2](#_Toc67579499)

[1.4 Tujuan penelitian 3](#_Toc67579500)

[1.5 Manfaat penelitian 3](#_Toc67579501)

[1.6 Sistematika penulisan 3](#_Toc67579502)

[1.7 Tinjauan pustaka 4](#_Toc67579503)

[BAB II LANDASAN TEORI 7](#_Toc67579504)

[2.1 Pengertian plagiarisme 7](#_Toc67579505)

[2.1.1 Tipe-tipe Plagiarisme 8](#_Toc67579506)

[2.1.2 Metode Pendeteksi Plagiarisme 8](#_Toc67579507)

[2.2 Pengertian *Information Retrieval* (IR) 9](#_Toc67579508)

[2.2.1 *Proses Indexing* 9](#_Toc67579509)

[2.2.2 Proses *Searching* 10](#_Toc67579510)

[2.2.3 *Model Information Retrieval* (IR) 11](#_Toc67579511)

[2.3 *American Standard Code for Information Interchange* (ASCII ) 12](#_Toc67579512)

[2.4 Algoritma *Rabin Karp* 13](#_Toc67579513)

[2.5 *Hyper Text Markup Language* (HTML) 20](#_Toc67579515)

[2.6 *Hypertext Preprocessor (PHP)* 21](#_Toc67579516)

[2.7 MySQL 22](#_Toc67579517)

[2.8 *Database Management System* (DBMS) 22](#_Toc67579518)

[2.9 *Unified Modeling Language* (UML) 23](#_Toc67579519)

[BAB III METODOLOGI PENELITIAN 26](#_Toc67579520)

[3.1 Waktu dan Tempat Penelitian 26](#_Toc67579521)

[3.1.1 Waktu 26](#_Toc67579522)

[3.1.2 Tempat Penelitian 26](#_Toc67579523)

[3.2 Metode Pengumpulan Data 26](#_Toc67579524)

[3.3 Metode Pengembangan Sistem 30](#_Toc67579525)

[3.3.1 Permulaan *(Inception)* 30](#_Toc67579526)

[3.3.2 Perluasaan / Perencanaan *(Elaboration)* 30](#_Toc67579527)

[3.3.3 Konstruksi *(Construction)* 30](#_Toc67579528)

[3.3.4 Transisi *(Transition)* 30](#_Toc67579529)

[3.4 Analisis Kebutuhan Sistem 31](#_Toc67579530)

[3.4.1 Kebutuhan Fungsional 31](#_Toc67579531)

[3.4.2 Kebutuhan Nonfungsional 31](#_Toc67579532)

[3.5 Analisis Perancangan Sistem 32](#_Toc67579533)

[3.5.1 *Unified Modeling Language* (UML) 32](#_Toc67579535)

[3.5.2 Perancangan Antarmuka (*Interface*) 46](#_Toc67579536)

[DAFTAR PUSTAKA 52](#_Toc67579537)

# DAFTAR TABEL

[Tabel 2.1 ASCII 12](#_Toc67498181)

[Tabel 2.2 Komponen-komponen *use case diagram* 23](#_Toc67498182)

[Tabel 2.3 Komponen-komponen *activity diagram* 25](#_Toc67498183)

[Tabel 2.4 Simbol-Simbol *Class Diagram* 26](#_Toc67498184)

[Tabel 2.5 Simbol-Simbol *Sequence Diagram* 27](#_Toc67498185)

[Tabel 3.1 *Gannt Chart Waktu Penelitian* 26](#_Toc67498188)

[Tabel 3.2 Spesifikasi Perangkat 31](#_Toc67498189)

[Tabel 3.3 Spesifikasi Perangkat Lunak 32](#_Toc67498190)

[Tabel 3.4 Deskripsi *Case Login (Admin*) 33](#_Toc67498191)

[Tabel 3.5 Deskripsi *Case* Mengolah Abstrak (Admin) 34](#_Toc67498192)

[Tabel 3.6 Deskripsi *Case* Lihat daftar abstrak 35](#_Toc67498193)

[Tabel 3.7 Deskripsi *Case* Cek plagiarisme 35](#_Toc67498194)

# DAFTAR GAMBAR

[Gambar 3.1 *Use Case Diagram* 33](#_Toc68323168)

[Gambar 3.2 *Activity diagram* *login* 36](#_Toc68323169)

Gambar [3.3 A*ctivity Diagram* Tambah Akun 37](#_Toc68323170)

[Gambar 3.4 *Activity Diagram* edit atau hapus abstrak 38](#_Toc68323171)

[Gambar 3.5 *Activity Diagram* Tambah Abstrak 39](#_Toc68323172)

Gambar [3.6 *Activity Diagram* Daftar abstrak 40](#_Toc68323173)

[Gambar 3.7 *Activity Diagram* Cek plagiarisme abstrak 41](#_Toc68323174)

[Gambar 3. 8 *Class diagram* 42](#_Toc68323175)

[Gambar 3. 9 Sequence Diagram Login 43](#_Toc68323176)

[Gambar 3. 10 Sequence Diagram Edit dan hapus 44](#_Toc68323177)

[Gambar 3. 11 Sequence Diagram Tambah Akun 45](#_Toc68323178)

[Gambar 3. 12 Sequence Diagram Cek plagiarisme 46](#_Toc68323179)

[Gambar 3. 13 Menu Login 47](#_Toc68323180)

[Gambar 3. 14 Tambah akun 47](#_Toc68323181)

[Gambar 3. 15 Menu Kelola Abstrak 48](#_Toc68323182)

[Gambar 3. 16 Tambah abstrak 49](#_Toc68323183)

[Gambar 3. 17 Daftar abstrak 49](#_Toc68323184)

[Gambar 3. 18 Menu Cek plagiarisme Abstrak 50](#_Toc68323185)

[Gambar 3. 19 Menu Tentang 51](#_Toc68323186)

# BAB I PENDAHULUAN

## **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 Proposal tugas akhir.

Proposal merupakan sebuah rancangan dan rencana kerja yang sistematis dan terperinci. Pada umumnya sebuah proposal bertujuan untuk menyampaikan rencana kegiatan pada pihak terkait, sehingga kegiatan tersebut dapat diterima dengan tujuan mendapatkan dukungan, mendapatkan izin, memperoleh dana dan sponsor, dan sebagainya. Sama halnya dengan proposal skirpsi, yang mana mahasiswa bertujuan untuk menyampaikan rancangan penelitiannya kepada dosen pembimbing agar disetujui sebelum membuat skripsi sehingga sistem ini dapat di gunakan oleh dosen pembimbing untuk mendeteksi tingkat plagiarisme proposal mahasiswa yang melakukan bimbingan sebelum melakukan ujian seminar proposal.

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

metode pencocokan *string*. 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 proposal tugas akhir mahasiswa.

## **Rumusan masalah**

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

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

## **Batasan Masalah**

Adapun batasan masalah dalam sistem ini adalah:

1. Pembuatan sistem pendeteksi plagiarisme proposal skripsi menggunakan Algoritma *Rabin-Karp*.
2. Data yang di uji merupakan proposal tugas akhir yang menggunakan bahasa indonesia.
3. Dokumen yang di uji pada proposal skripsi yaitu pada bagian bab pertama (latar belakang).
4. Tidak memperhatikan sinonim kata.
5. Hanya mendeteksi file teks yang bersifat plain teks seperti: huruf dan angka, tidak berupa gambar dan diagram.

## **Tujuan penelitian**

Adapun Tujuan dari penelitian ini adalah:

1. Mendesain dan membangun sistem pendeteksi plagiarisme dalam proposal tugas akhir mahasiswa dengan algoritma *Rabin Karp*.
2. Mengurangi tingkat plagiarisme pada Proposal Tugas akhir mahasiswa.
3. Mendeteksi tingkat plagiarisme pada Proposal Tugas akhir mahasiswa.

dengan *algoritma* *Rabin Karp*.

## **Manfaat penelitian**

Adapun manfaat dari penelitian ini adalah:

1. Memudahkan dosesn pembimbing untuk mendeteksi plagiarisme proposal mahasiswa yang melakukan bimbingan.
2. Dapat membantu sebagai bahan pertimbangan dalam menentukan plagiarisme.
3. Dapat menentukan persentase kemiripan (*similarity*) antara dokumen yang diuji dengan dokumen lainnya oleh sistem.

## **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.

## **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 plagarisme 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 berbasiskan 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* memilki 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 memilki 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

## **Proposal tugas akhir**

Proposal adalah rencana yang dituangkan dalam bentuk rancangan kerja, perencanaan sistematis, matang, dan teliti yang dibuat oleh peneliti sebelum melaksanakan penelitian. dari sudut pandang dunia ilmiah, pengertian proposal adalah rancangan dari suatu usulan sebuah penelitian yang kemudian dilaksanakan oleh peneliti terhadap bahan penelitiannya. Sehingga proposal sama halnya dengan sebuah usulan kegiatan yang memerlukan persetujuan dari pihak lain. Tujuan proposal adalah untuk mendapatkan persetujuan dari suatu pihak mengenai rencana atau rancangan yang akan dilakukan. Di dalam proposal berisi kelayakan dari proposal untuk di lanjutkan menjadi tugas akhir. Adapun isi dari proposal tugas akhir di sesuaikan dengan jenis tugas akhir, yaitu tugas akhir penelitian dan tugas akhir rekayasa sistem. Salah satu isi dari proposal tugas akhir penelitian adalah latar belakang. Pada latar belakang berisi alasan dipilihnya judul penelitian, yang biasanya berasal dari fenomena atau kejadian empirris yang akan diangkat sebagai topik penelitian. Kemukaan uraian tentang masalah yang menarik minat untuk diteliti. Selain itu memuat uraian atau penjelasan mengenai alasan-alasan sehingga permasalahan yang di ajukan merupakan permasalahan yang layak untuk diteliti dan di temukan cara untuk menyelesaikannya. Penelitian yang di lakukan hendaknya asli, dalam arti bahwa masalah yang dipilih belum pernah di teliti oleh peneliti sebelumnya atau pengambilan karya orang lain (plagiarisme) sehingga harus di nyatakan dengan tegas perbedaan penelitian yang di lakukan dengan penelitian yang sudah pernah di lakukan sebelumnya.

## **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.

### **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.

1. Kebetulan (*accidental*)

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

1. 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.

1. Disengaja (*intentional*)

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

1. Diri sendiri (*self* *plagiarism*)

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

### **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 :

1. 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*.

1. 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*.

1. 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.

## **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.

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

### ***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.

1. *Stopword Removal*

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

1. *Stemming*

Proses mengubah suatu kata bentukan menjadi kata dasar.

1. *Term Weighting*

Proses pembobotan setiap term di dalam dokumen.

### **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.

1. Proses *Stemming*

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

1. 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 duhulu 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*.

1. 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.

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

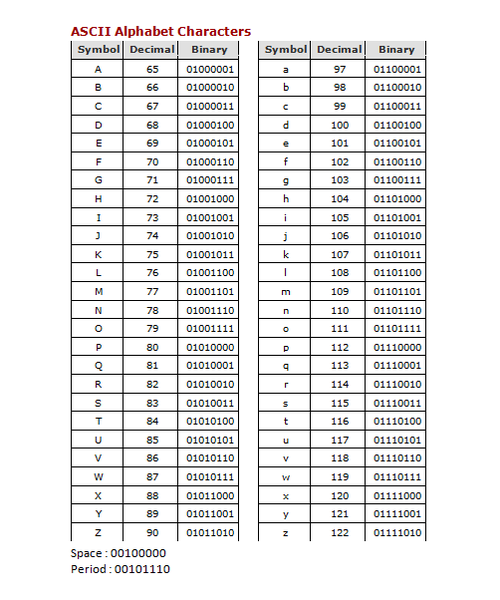
### ***Model Information Retrieval* (IR)**

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. *Algebratic* 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.
   1. ***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 "|". Itu 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 menambakan 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. Jumlah kode ASCII adalah 255 kode. (Tantoni & Zaen, 2018).

Tabel 2.1 ASCII



## **Algoritma *Rabin Karp***

Algoritma *Rabin Karp* adalah algoritma pencarian kata yang mencari sebuah pola berupa sub*string* 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 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 algortima *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 dokumen 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 ke huruf kecil, 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*). Dalam menentukan standar persentase kemiripan terdapat 3 klasifikasi yang digunakan dalam sistem pendeteksi plagiarism tugas akhir , yaitu:

* + - 1. Plagiarisme ringan : < 30%
      2. Plagiarisme sedang : < 30% - 70%
      3. Plagiarism berat : > 70%

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

1. **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.*

1. 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.

1. 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 plagiarism |

Dokumen uji setelah melalui proses *filtering* (proses1.2)

|  |
| --- |
| Plagiarisme merupakan dampak negatif perkembangan teknologi informasi |

Pada contoh filtering diatas yaitu proses menghialngkan 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 hilanghan adalan titik (.) dan tanda petik (“ ”), sedangkan dokumen uji kata yang dihilangkan adalah kata “dari”.

1. 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 proseses *hashing*.

1. Proses Parsing *K-gram*

Langkah selanjutnya adalah parsing *k-gram*, K-gram adalah rangkaian terms dengan panjang K. Kebanyakan yang digunakan sebagai terms adalah kata, dimana pada proses ini kata dipecah menjadi potongan-potongan dimana setiap potongan mengandung karakter sebanyak k. Nilai k-gram sangat berpengaruh terhadap nilai persentase suatu kemiripan dokumen. pemilihan k-gram yang semakin kecil akan memperoleh nilai persentase kemiripan yang besar. Hal ini terjadi karena pada k-gram yang lebih sedikit, string yang dipotong lebih kecil sehingga kemungkinan untuk ditemukannya rangkaian karakter yang sama semakin besar. Semakin besarnya k-gram, maka mengandung karakter yang lebih banyak dibandingkan dengan k-gram yang lebih kecil sehingga menyebabkan rangkaian karakter yang ditemukan semakin berkurang sehingga menurunkan nilai similaritas (masukkan sitasi link jurnal nama filenya yaitu juranal plagiarisme proposal). Nilai k-gram bisa kita tetapkan sesuai kebutuhan, pada sistem ini nilai k-gram nya telah di tetapkan yaitu nilai k=3. Berikut ini adalah contoh proses parsing *K-gram k* = 3:

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}{ampak}{ mpak}{ pak n}{ak ne}{k neg}{ nega} {negat}{egati}{gatif}{atif }{tif p}{if pe}{f per}{ perk}{perke}{erkem} {rkemb}{kemba}{emba}{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 d}{gi da}{i dan}{ dan }{dan i}{an in}{n inf}{ info}{info }{nfo r}{fo rm}{o rma}{ rmas}{rmasi}{masi }{asi a}{si ad}{i ada}{ adal}{adala}{dalah}{alah }{lah m}{ah ma}{h 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} {agair} {giari}{iaris} {arism}{risme}{isme } |

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

|  |
| --- |
| {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}{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 t}{an te}{n tek}{ tekn}{tekno}{eknol}{knolo}{nolog}{ologi}{logi }{ogi i}{gi in} {i inf} { 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=3, sehinga isi kalimat pada dokumen asli dan dokumen uji dirubah dalam bentuk k-gram.

1. 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 Dalam prosesnya digunakan basis yang biasanya adalah bilangan prima yang cukup besar, dengan tujuan agar meminimalkan terjadinya tabrakan (tambahkan materi tentang basis bilangan). Dengan menggunakan persamaan, 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 manuual yang di laporan mau di sesuaikan dengan yang di sistem)**

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(a)= 97

ASCII(h)= 104

H(C1 …. CK) =C1.b(K-1)+C2.b(K-2)+ C3.b(k-1)+ C4.b(k-1)+( C5.b(k-1))

H = (115.74 )+(97.73 )+(108.72 )+( 97.71 )+( 104.70 )

H = (115.2401)+(97.343)+(108.49)+(97.7)+(104.1)

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= ∑ ***H***asli= 90

Hasil *hashing* dokumen uji:

|  |
| --- |
| {121322} {311535} {298458} {274164} {288974} {291812} {278058} {316228} {297630} {318784} {298784} {259639} {120083} {302869} {283758} {320119} {324932} {308215} {275153} {295892} {272992} {280774} {116760} {279593} {276558} {305659} {307760} {272037} {274083} {120329} {304595} {283500} {287095} {278576} {319865} {289544} {262187} {121102} {309991} {287662} {316225} {297674} {285479} {300949} {274777} {276463} {286304} {273108} {281584} {122428} {319283} {285477} {300943} {308355} {309820} {303195} {307314} {285731} {269098} {119062} {295724} {305442} {289421} {311748} {316764} {301382} |

Jumlah hash pada dokumen uji adalah= ∑ ***H***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 = ∑ ***H***asliᴖ***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.*

1. Deteksi Kesamaan

Setelah diketahui nilai hashnya, jumlah hash pada dokumen asli 90, jumlah hash pada dokumen uji 67 dan hash yang sanma pada kedua dokumen 52. Proses selanjutnya adalah menghitung *similarity* yaitu tingkat kesamaan dua dokumen dengan menggunakan persamaan (2.2), yaitu barapa persen tingkat kasamaannya. *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:

*Similarity* (asli, uji) = x 100 %

= x 100 %

= x 100 %

= x 100 %

= x 100 %

= 49,52 %

Hasil perhitungan *similaritry* dari kedua dokumen diatas adalah 49,52%, dokumen yang diuji dapat dikatakn plagiat, karena tingkat kesamaannya sangat besar.

Tambahkan materi standar persentasi plagiarism (plagiarism ringan, sedang, berat).

## ***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.

## ***Hypertext Preprocessor (PHP)***

PHP Kepanjangan dari PHP adalah "*Hypertext Preprocessor*" (ini merupakan singkatan rekursif).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. 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 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 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 runtimemelalui console serta juga dapat menjalankan perintah-perintah system.

## **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 menghandle permintaan layanan secara stimultan. Model koneksi dengan protocol TCP-IP membuat akses ke server *database* lebih cepat jika dibandingkan dengan menggunakan *mapping drive*.

* 1. ***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 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. *Table*

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

1. *Record*

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

1. *Field*

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

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

[*Unified Modeling Language*](http://fadhlyashary.blogspot.com/2012/06/pengertian-uml-unified-modeling.html) (UML) adalah bahasa permodelan untuk menentukan visualisasi, kontruksi, dan merancang sistem berbasis *object-oriented.* [*Unified Modeling Language*](http://fadhlyashary.blogspot.com/2012/06/pengertian-uml-unified-modeling.html) 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 |
|  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case.* |
|  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*dependent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). |
|  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada  diatasnya objek induk (*ancestor*). |
|  | *Include* | Menspesifikasikan bahwa *use case* sumber secara *eksplisit*. |
|  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang  diberikan. |
|  | *Association* | Apa yang menghubungkan antara objek  satu dengan objek lainnya. |
|  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
|  | *Use case* | Deskripsi dari uraian aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor.* |
|  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah  dan elemen-elemennya (sinergi). |
|  | *Note* | Elemen fisik yang eksis saat aplikasi  dijalankan dan mencerminkan suatu  sumber daya komputasi. |

Sumber: (Kiswanto, 2016).

1. ***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** | **Keterrangan** |
|  | *Activity* | Memperlihatkan bagaimana masing- masing kelas antarmuka saling berinteraksi satu sama lain. |
|  | *Action* | *State* dari sistem yang mencerminkan  eksekusi dari suatu aksi. |
|  | *Initial Node* | Bagaimana objek dibentuk atau diawali  (*start flow*). |
|  | *Activity Final*  *Node* | Bagaimana objek dibentuk dan diakhiri  (*end flow*). |
|  | *Fork Node* | Satu aliran yang pada tahap tertentu  berubah menjadi beberapa alur. |

Sumber: (Kiswanto, 2016).

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

Tabel 2.4 Simbol-Simbol *Class Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Keterangan** |
| 1 | |  | | --- | | **nama\_kelas** | | + atribut | | + operasi() | | Kelas | Kelas pada struktur sistem |
| 2. |  | *Interface* | Semua dengan konsep *interface* dalam  pemrograman b*error* ientasi objek |
| 3. |  | *Association* | Relasi antarkelas dengan makna umum, asoasi biasanya juga disertai dengan *multiplicity* |
| 4. |  | *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. |  | *Dependency* | Relasi antarmuka dengan makna kebergantungan antarkelas |
| 7. |  | *Aggregation* | Relasi antarkelas dengan makna semua bagian (*whole- part*) |

Sumber: (Kiswanto, 2016).

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 *sequence* maka harus diketahui objek-objek yang dimiliki kelas yang diinstasiasi 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 / *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: (Kiswanto, 2016).

# BAB III METODOLOGI PENELITIAN

## **Waktu dan Tempat Penelitian**

### **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 *Gannt 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. | *Inception* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. | *Elaboration* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. | *Construction* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4. | *Transition* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## **Tempat Penelitian**

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

## **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 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 proposal tugas akhir mahasiswa di jurusan Teknik Informarika Universitas Halu Oleo.

## **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:

### **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.*

### **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.*

### **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.

### **Transisi *(Transition)***

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

## **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.

### **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 *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. | *PC* | *Acer* |
| 2. | *Processor* | *Intel dual core* |
| 3. | *RAM* | *4 GB* |
| 4. | *Harddisk* | *128 GB SSD* |
| 5. | *Monitor* | *Monitor 14 Inch* |

1. **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. | *Windows* | *Operating System* | *Windows 10 Home* |
| 2. | *Xampp* | *Universal development environment* | *Xampp v3.2.2* |
| 3. | *Apache* | *Web Server* | *Apache 2.4 win64-VC14* |
| 4. | *Mysql* | *Database Management* | *Mysql 5.7* |
| 5. | *PHP* | *Web development* | *PHP 7.3* |
| 6. | *Chrome* | *Web browser* | *Chrome version 76.0* |
| 7. | *Visual Studio Code* | *Code Editor* | *VS Code v1.38* |

* 1. **Analisis Perancangan Sistem**

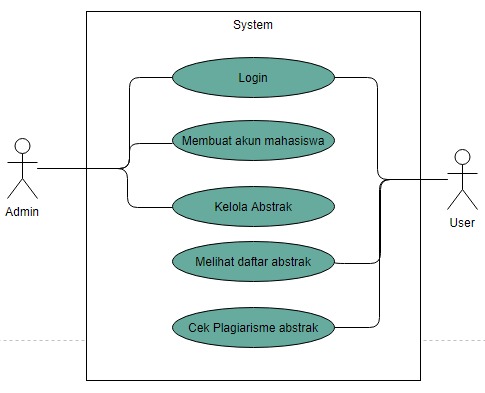
Perancangan sistem yang akan dibangun yaitu perancangan UML dan perancangan *user interface* sistem*.*

### ***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** | ***Login*** |
| Deskripsi Singkat: | Proses Keamanan login untuk *Admin* dan Mahasiswa |
| Persyaratan | *Admin* dan Mahasiswa harus memasukkan *username* dan *password* |
| Kondisi Akhir | Masuk untuk manage sistem |
| Situasi Kesalahan | * + 1. Yang masuk bukan *Admin* dan Mahasiswa     2. Salah memasukkan *username* dan *password* |
| Status sistem dalam  saat terjadi kesalahan | 1. Menampilkan pesan *error* 2. Tidak dapat masuk kedalam sistem |
| Aktor | *Admin* dan Mahasiswa |
| Trigger | Keamanan *login* untuk *Admin* |
| Proses Standar | 1. *Admin* dan Mahasiswa masuk ke sistem 2. *Admin* dan Mahasiswa memasukkan *username* 3. *Admin* dan Mahasiswa memasukkan *password* |
| Proses Alternatif | Mengecek *username* dan *password* |

Tabel 3.5 Deskripsi *Case* Mengolah Abstrak (Admin)

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

Tabel 3.6 Deskripsi *Case* Lihat daftar abstrak

|  |  |
| --- | --- |
| **Nama** | **Melihat daftar abstrak** |
| Deskripsi Singkat: | *Admin* dan Mahasiswa ingin melihat daftar abstrak |
| Persyaratan | *Admin* dan Mahasiswa harus masuk kesistem terlebih dahulu |
| Kondii Akhir | Data Abstrak di tampilkan |
| Situasi Kesalahan | - |
| Status sistem dalam saat terjadi kesalahan | - |
| Aktor | *Admin* dan Mahasiswa |
| Trigger |  |
| Proses Standar | 1. *Admin* dan Mahasiswa masuk kesistem 2. *Admin* dan Mahasiswa melakukan pencarian 3. Data data abstrak ditampilkan |
| Proses Alternatif | *-* |

Tabel 3.7 Deskripsi *Case* Cek plagiarisme

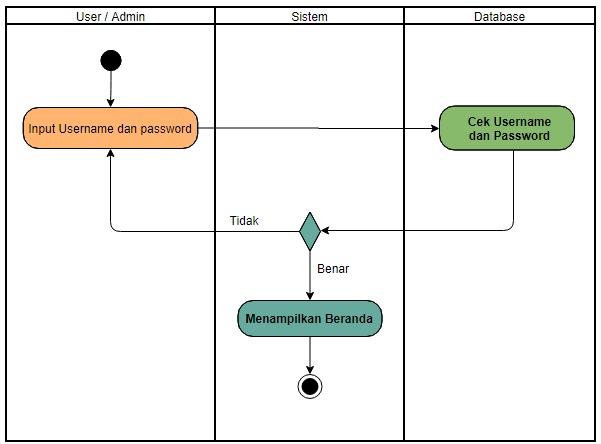
|  |  |
| --- | --- |
| **Nama** | **Cek Plagiarisme** |
| Deskripsi Singkat: | Mahasiswa ingin mengecek plagiarisme |
| Persyaratan | Mahasiswa harus masuk kesistem terlebih dahulu |
| Kondisi Akhir | Data Abstrak hasil persentase harus di tampilkan |
| Situasi Kesalahan | - |
| Status sistem dalam saat terjadi kesalahan | - |
| Aktor | Mahasiswa |
| Trigger |  |
| Proses Standar | 1. Mahasiswa masuk kesistem 2. Mahasiswa input data abstrak dan menentukan k-gramnya 3. Data data abstrak dan persentase plagiarismenya ditampilkan |
| Proses Alternatif | *-* |

1. ***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.

1. *Activity Diagram Login*

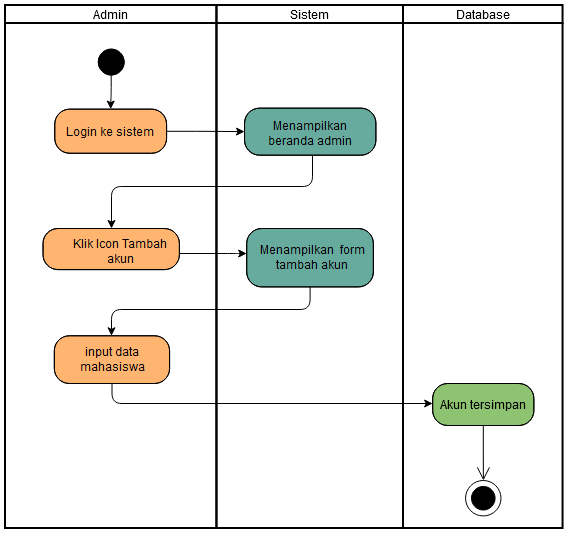
Pada Gambar 3.2 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* dan *user* ketika memasukkan *username* dan *password*, lalu sistem akan menampilkan melakukan validasi.



Gambar 3.2 *Activity diagram* *login*

1. *Activity Diagram* Tambah akun mahasiswa

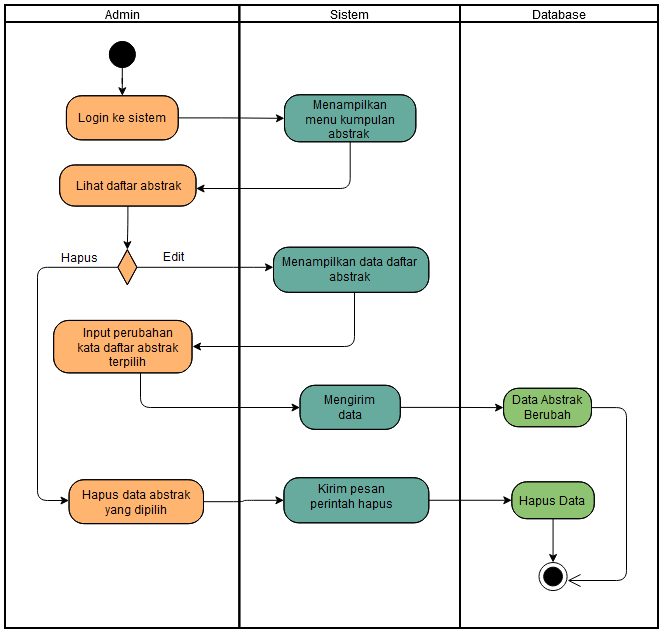
Pada Gambar 3.3 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* memilih menu Tambah akun untuk memberikan akun kepada mahasiswa agar dapat melakukan pengecekan plagiarisme abstrak.



Gambar 3. 3 A*ctivity Diagram* Tambah Akun

1. *Activity Diagram* Mengelola Abstrak (Edit dan Hapus)

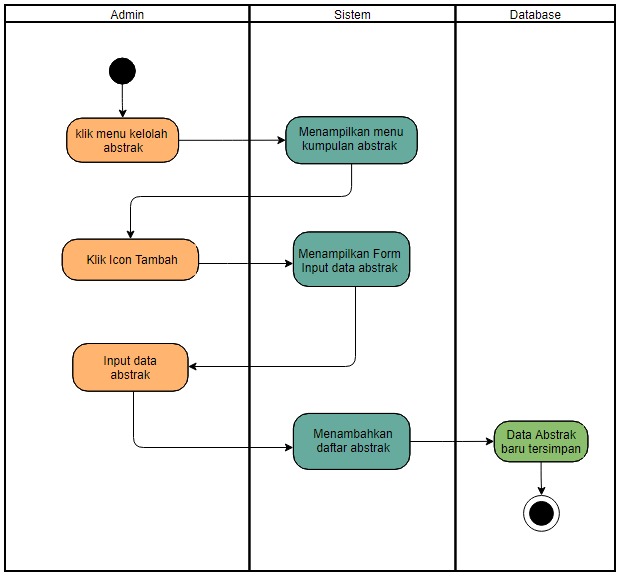
Pada Gambar 3.4 merupakan diagram aktivitas yang menunjukkan aktivitas admin memilih menu Mengelola Abstrak dan memilih untuk edit atau hapus dari salah satu daftar abstrak.



Gambar 3.4 *Activity Diagram* edit atau hapus abstrak

1. *Activity Diagram* Mengelola Abstrak ( Tambah abstrak )

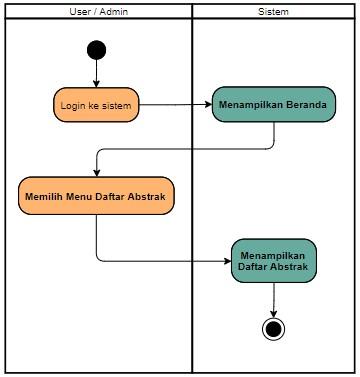
Pada Gambar 3.5 merupakan diagram aktivitas yang menunjukkan aktivitas *admin* memilih menu Tambahabstrak untuk menambahkan abstrak baru ketika ada jurnal yang telah di *upload*.



Gambar 3.5 *Activity Diagram* Tambah Abstrak

1. *Activity diagram* Daftar abstrak

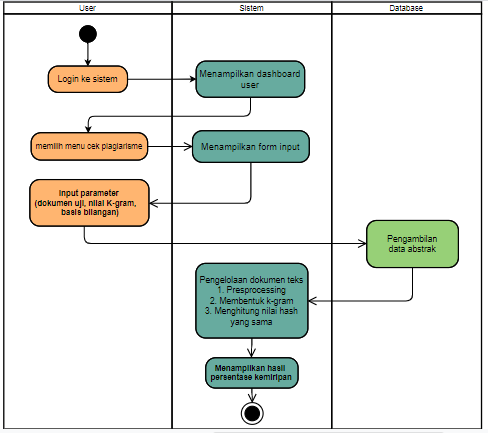
Pada Gambar 3.6 merupakan diagram aktivitas yang menunjukkan aktivitas User ketika telah login, sistem akan menampilkan menu Daftar abstrak yang di dalamnya berisi kumpulan data abstrak.



Gambar 3. 6 *Activity Diagram* Daftar abstrak

1. *Activity Diagram* Cek Plagiarisme Abstrak

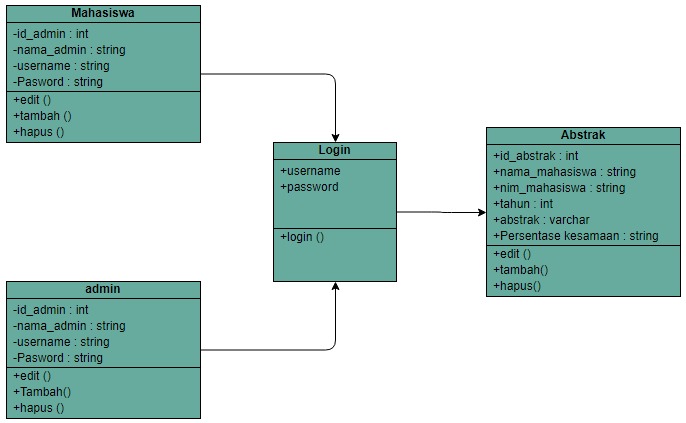
Pada Gambar 3.7 merupakan diagram aktivitas yang menunjukkan aktivitas *User* memilih menu *Cek plagiarisme* Abstrak lalu sistem menampilkan halaman *Input abstrak*, setelah itu upload abstrak yang telah di input lalu user melakukan cek plagiarisme pada abstrak.



Gambar 3. 7 *Activity Diagram* Cek plagiarisme abstrak

1. ***Class diagram***

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

**

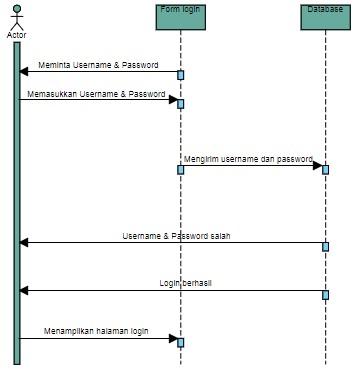
Gambar 3. 8 *Class diagram*

1. ***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 interakasi antar objek dan sistem.

1. *Sequence Diagram Login*

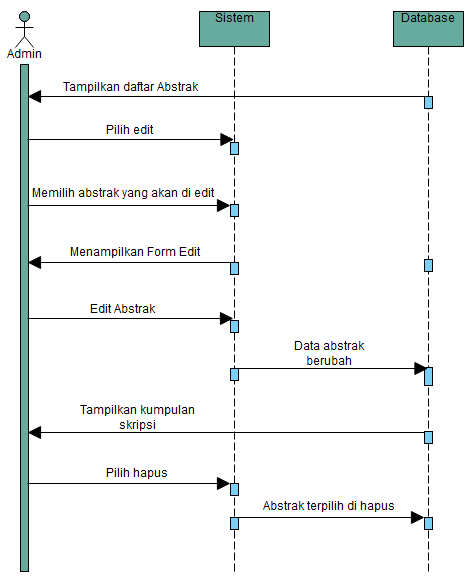
Pada Gambar 3.9 adalah *sequence* yang menunjukkan proses *login* bagi admin dan user dimana *actor* memasukkan *username* dan *password* lalu divalidasi oleh *database*.

******

Gambar 3. 9 Sequence Diagram Login

1. *Sequence Diagram* mengelola abstrak (edit dan hapus)

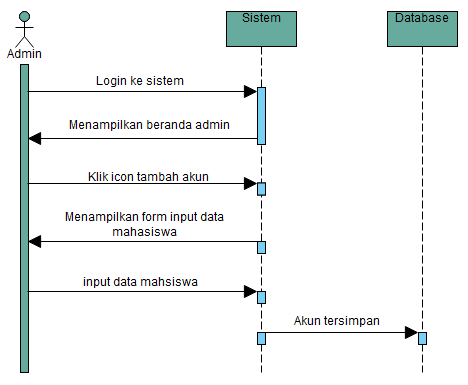
Gambar 3.10 adalah *sequence diagram* yang menjelaskan tentang korelasi antara sistem dan *user* pada Sistem Informasi Penelurusan Tugas Akhir. Dimulai dari *user* membuka sistem, lalu sistem menampilkan menu-menu, kemudian *user* memilih menu pencarian yang ingin digunakan hingga sistem menampilkan hasil dari pencarian.



Gambar 3. 10 *Sequence Diagram* Edit dan hapus

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

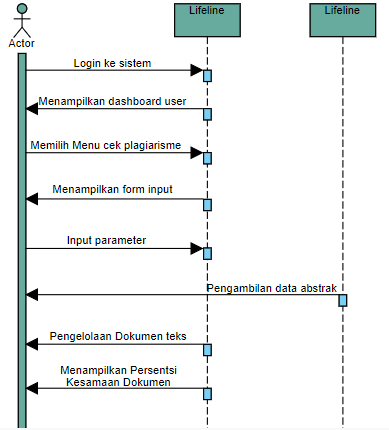
Gambar 3.11 merupakan *Sequence Diagram* yang menunjukkan aktivitas *admin* memilih menu Tambahakun untuk memberikan akun kepada mahasiswa agar melakukan pengecekan plagiarisme.



Gambar 3. 11 *Sequence Diagram* Tambah Akun

1. *Sequence Diagram* Cek plagiarisme

Pada Gambar 3.12 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.



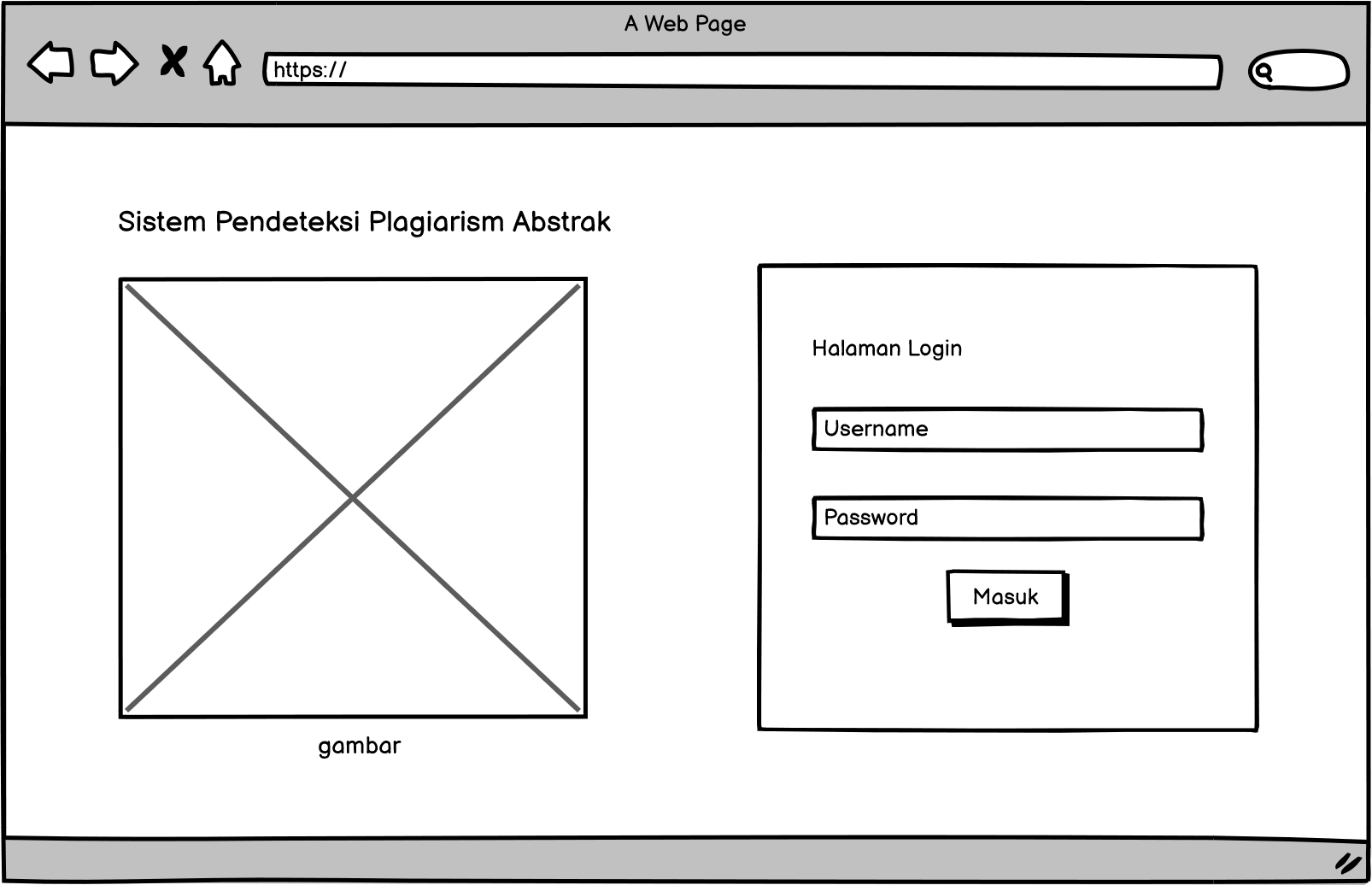
Gambar 3. 12 *Sequence Diagram* Cek plagiarisme

### **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***

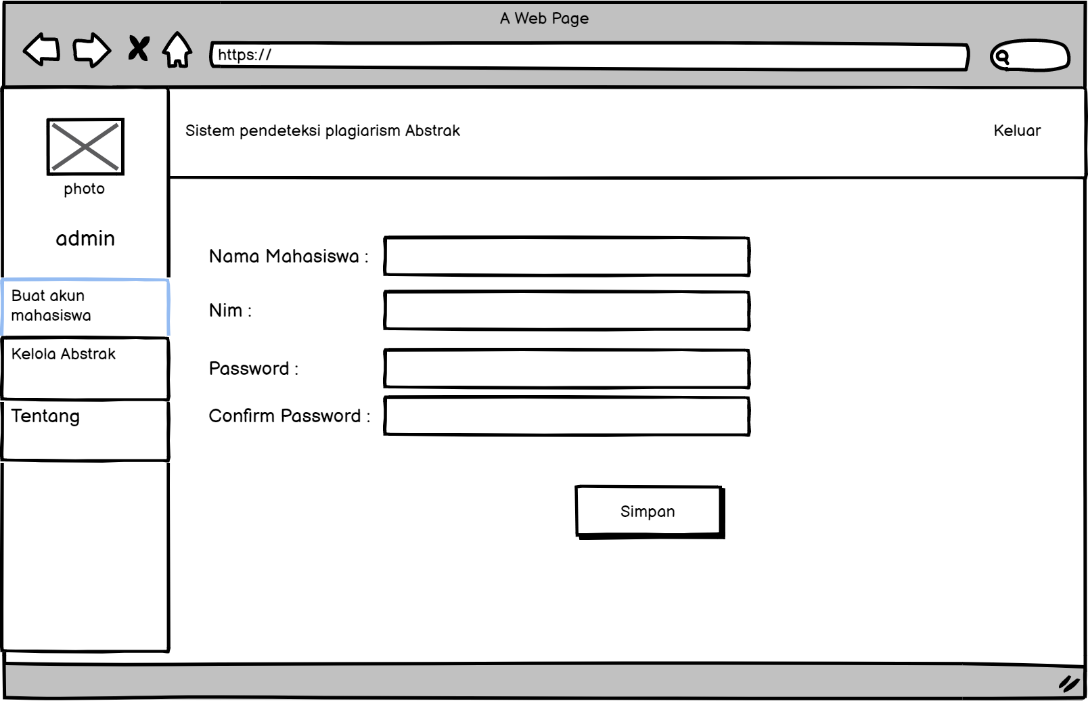
Halaman *login* pada *website* menampilkan *form login* yang terdiri dari *username* dan *password*. Setiap *field* harus diisi agar dapat melakukan *login*. Selain itu, *form* ini juga telah ditambahkan fungsi validasi agar dapat menyaring data *user* yang benar. Bentuk dari tampilan *login* dapat dilihat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

****

Gambar 3. 13 Menu *Login*

* + - 1. **Tambah akun mahasiswa**

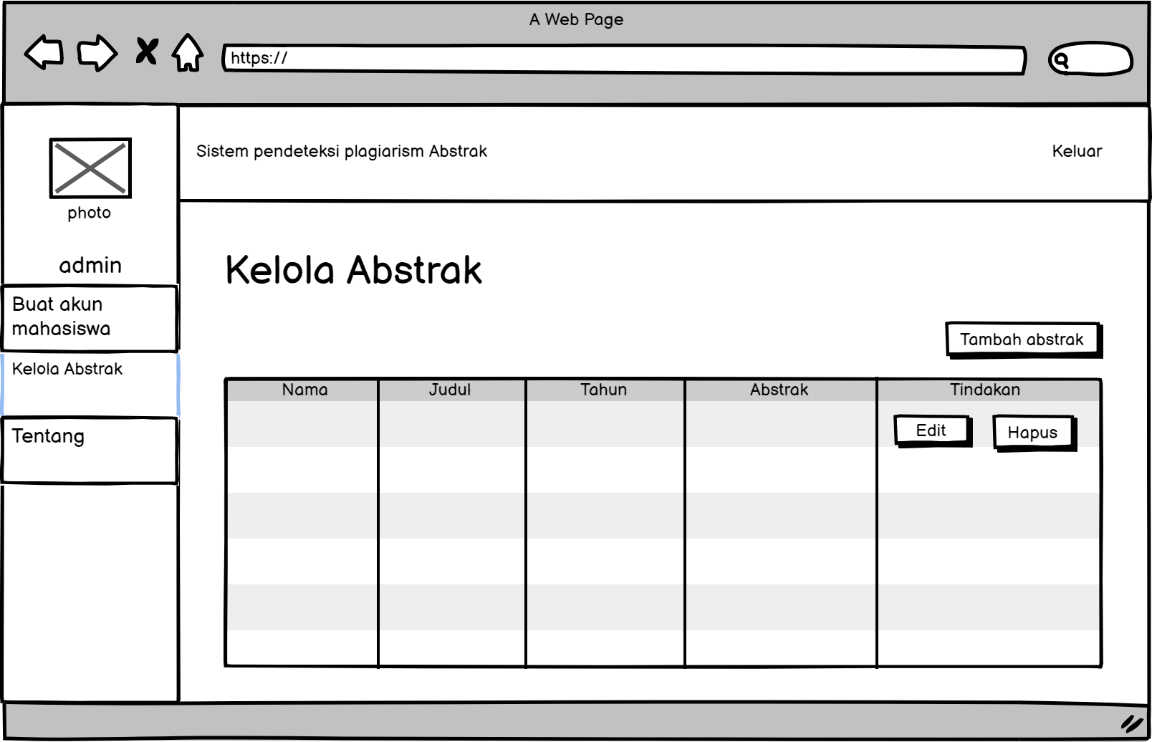
Menu Tambah akun mahasiswa menampilkan form tambah akun mahasiswa yang akan memasukkan data mahasiswa untuk di jadikan sebagai akun untuk dapat mengecek plagiarisme abstrak. Bentuk dari tampilan menu tambah akun mahasiswa dapat dilihat pada gambar berikut.



Gambar 3. 14 Tambah akun

* + - 1. **Menu mengelola abstrak**

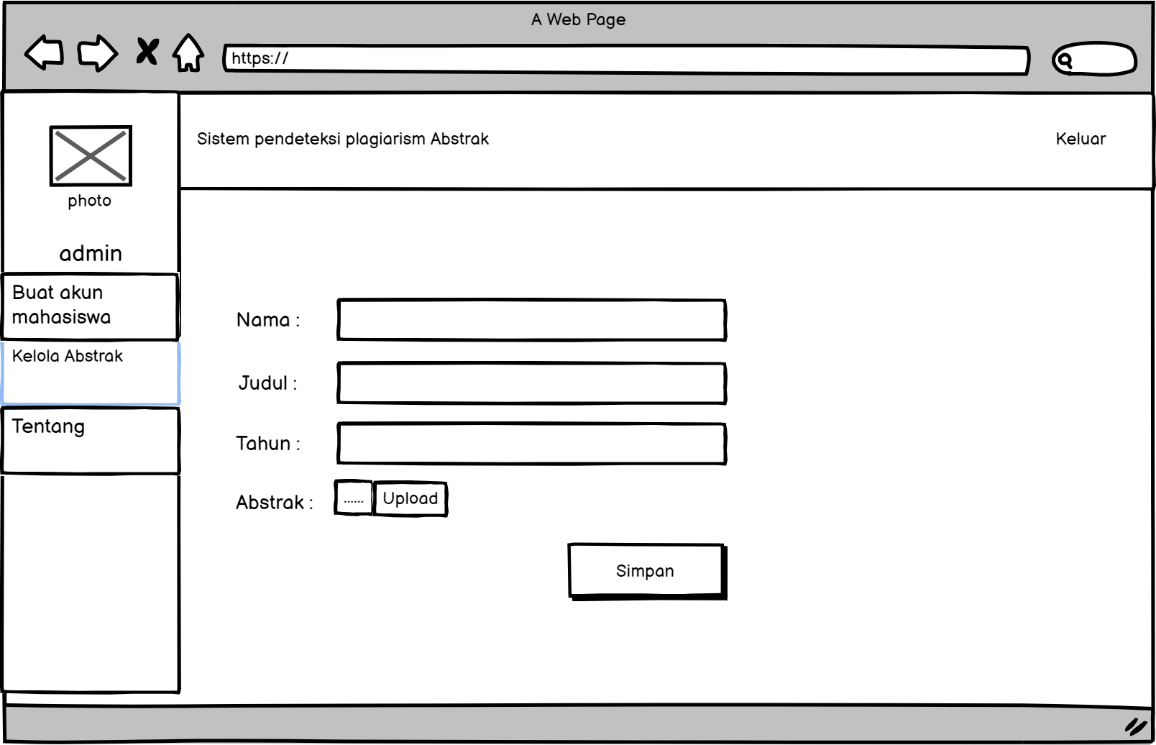
Menu Mengelola Abstrak menampilkan data Abstrak dan juga admin bisa melakukan Tambah abstrak, edit dan hapus Abstrak. Bentuk dari tampilan menu mengelola Abstrak dapat dilihat pada gambar berikut.



Gambar 3. 15 Menu Kelola Abstrak

* + - 1. **Menu tambah abstrak**

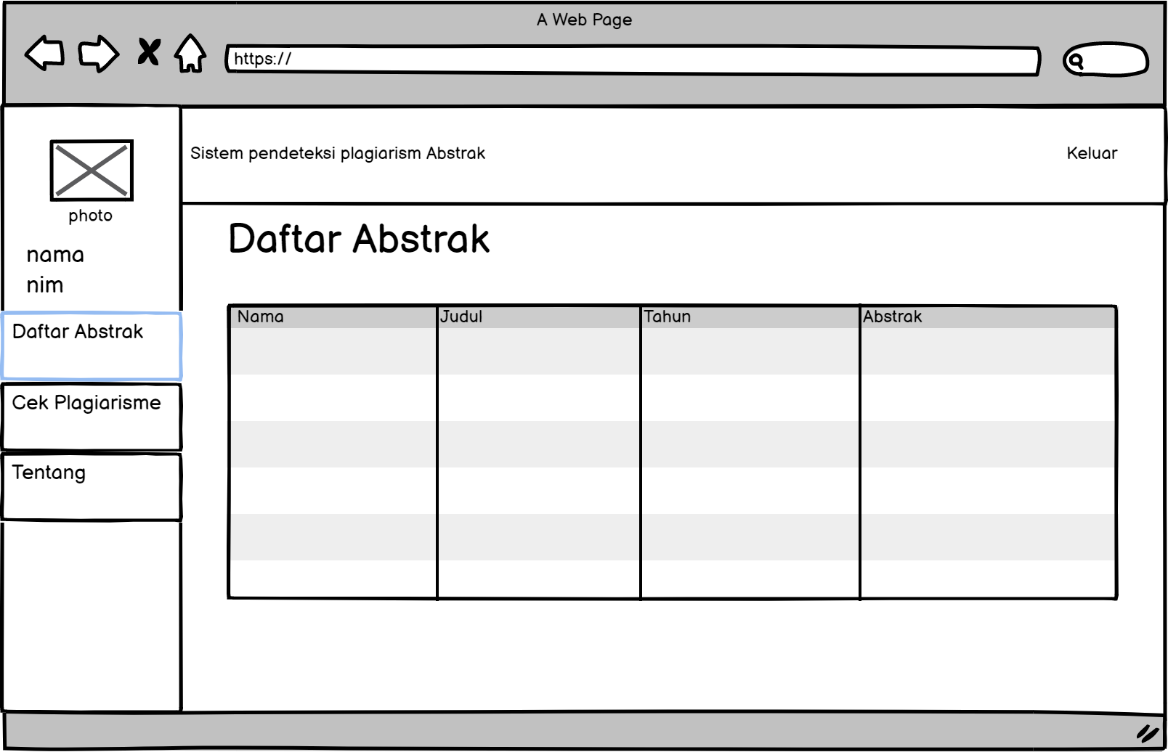
Menu tambah Abstrak menampilkan form untuk admin agar bisa melakukan Tambah. Bentuk dari tampilan menu tambah Abstrak dapat dilihat pada gambar berikut.



Gambar 3. 16 Tambah abstrak

* + - 1. **Menu Daftar abstrak**

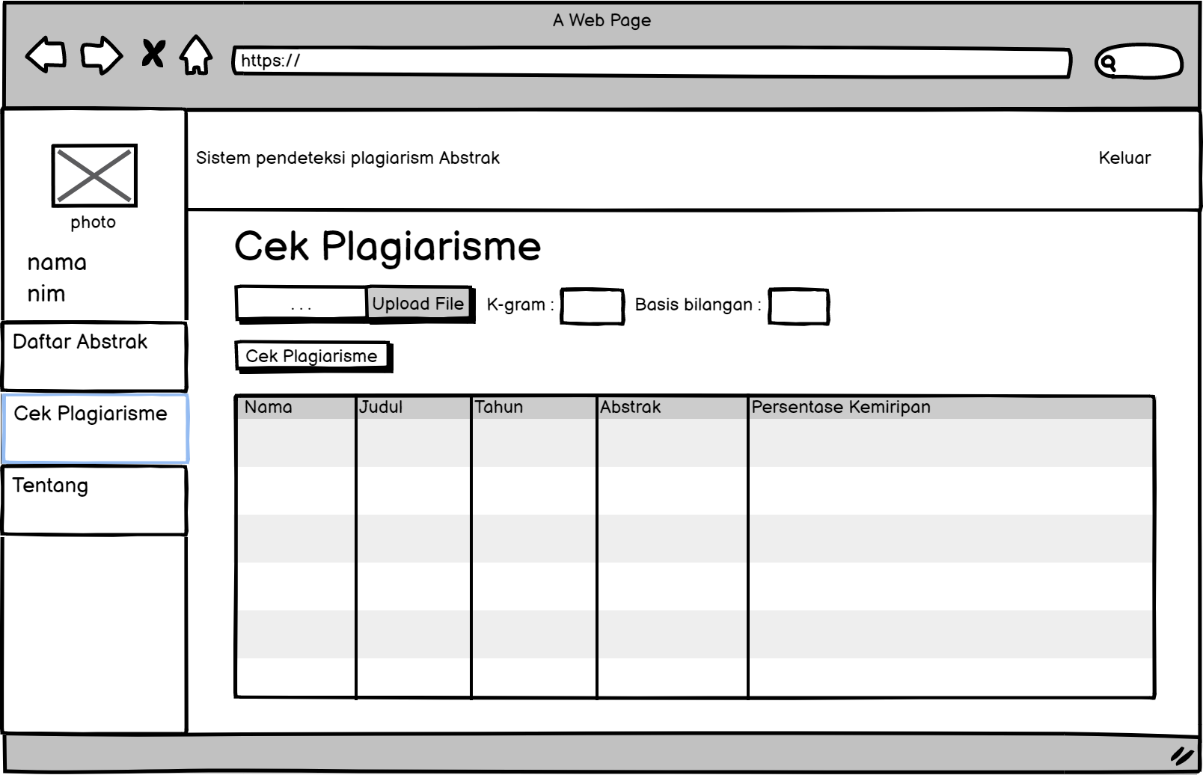
Tampilan Daftar abstrakadalah menu kumpulan abstrak yang akan dilihat pada sistem ini.



Gambar 3. 17 Daftar abstrak

* + - 1. **Menu Mengecek plagiarisme** **abstrak**

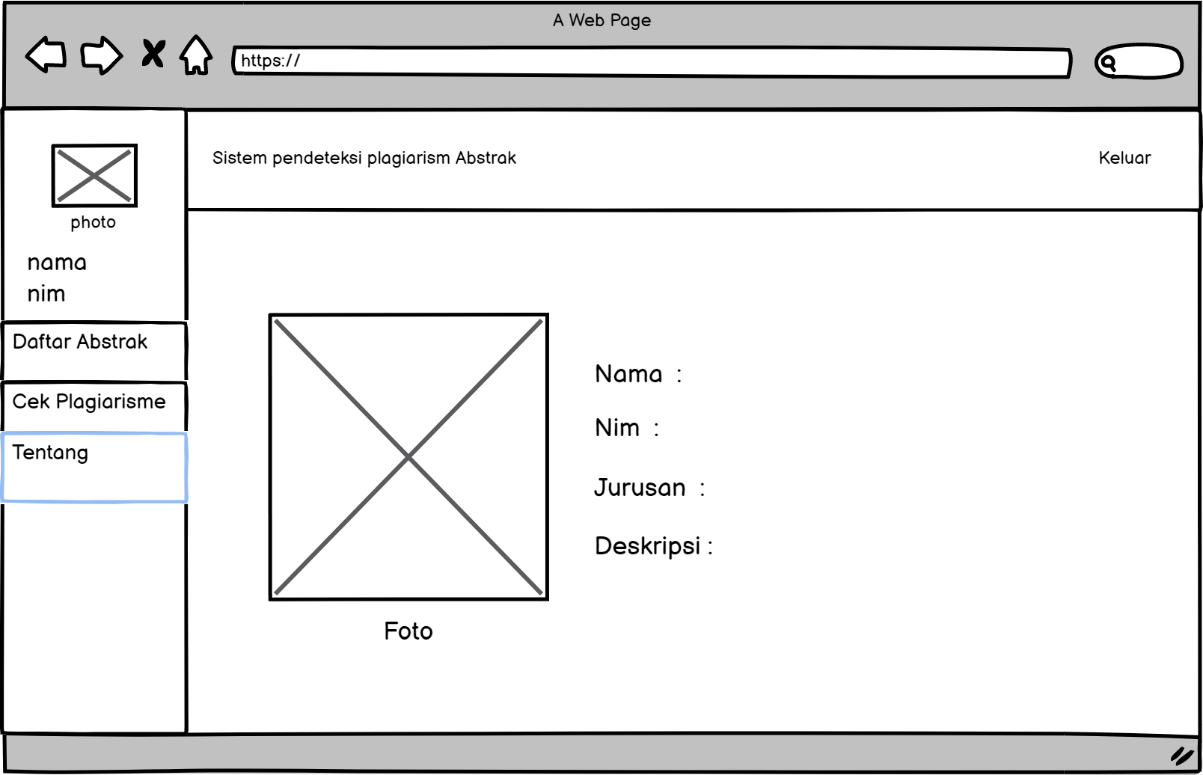
Menu Mengecek plagiarisme Abstrak menampilkan data form input abstrak, selanjutnya data tersebut di upload dan akan di cek plagiarismenya. Setelah proses itu selesai maka akan tampil tabel yang berisi daftar abstrak beserta dengan persentase keasamaannya dengan abstrak lainnaya. Bentuk dari tampilan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



Gambar 3. 18 Menu Cek plagiarisme Abstrak

* + - 1. **Menu tentang**

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



Gambar 3. 19 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%2C MySQL,Top 10 Enterprise Database System).